TMS320C6472/TMS320TCI6486 DSP Universal Test and Operations PHY Interface for ATM 2 (UTOPIA2)

User's Guide

Literature Number: SPRUEG2E
March 2006–Revised October 2009

List of Figures

List of Tables

Read This First

About This Manual

This document describes the universal test and operations PHY interface for asynchronous transfer mode (ATM) 2 (UTOPIA2) in the TMS320TCI6486/TMS320C6472 digital signal processors (DSPs).

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.
- Registers in this document are shown in figures and described in tables.
  - Each register figure shows a rectangle divided into fields that represent the fields of the register. Each field is labeled with its bit name, its beginning and ending bit numbers above, and its read/write properties below. A legend explains the notation used for the properties.
  - Reserved bits in a register figure designate a bit that is used for future device expansion.

Related Documentation From Texas Instruments

The following documents describe the C6000™ devices and related support tools. Copies of these documents are available on the Internet. *Tip:* Enter the literature number in the search box provided at www.ti.com.

SPRU189 — *TMS320C6000 DSP CPU and Instruction Set Reference Guide.* Describes the CPU architecture, pipeline, instruction set, and interrupts for the TMS320C6000 digital signal processors (DSPs).

SPRU198 — *TMS320C6000 Programmer's Guide.* Describes ways to optimize C and assembly code for the TMS320C6000™ DSPs and includes application program examples.

SPRU301 — *TMS320C6000 Code Composer Studio Tutorial.* Introduces the Code Composer Studio™ integrated development environment and software tools.

SPRU321 — *Code Composer Studio Application Programming Interface Reference Guide.* Describes the Code Composer Studio™ application programming interface (API), which allows you to program custom plug-ins for Code Composer.

SPRU871 — *TMS320C64x+ Megamodule Reference Guide.* Describes the TMS320C64x+ digital signal processor (DSP) megamodule. Included is a discussion on the internal direct memory access (IDMA) controller, the interrupt controller, the power-down controller, memory protection, bandwidth management, and the memory and cache.

C6000, TMS320C6000, Code Composer Studio are trademarks of Texas Instruments.
Motorola MPC8260 Power QUICC IIE is a trademark of Motorola, Inc.
All other trademarks are the property of their respective owners.

User's Guide
SPRUEG2E – March 2006 – Revised October 2009

C6472/TCI6486 UTOPIA2

1 Overview

The UTOPIA is an ATM controller (ATMC) slave device that interfaces to a master ATM controller. The UTOPIA port conforms to the ATM Forum standard specification af-phy-0039.000. Specifically, this interface supports the UTOPIA level 2 interface that allows 16-/8-bit slave operations of up to 50 MHz for both transmit and receive operations.

The UTOPIA slave interface relies on the master ATM controller to provide the necessary control signals such as the clock, enable, and address values. Only cell-level handshaking is supported.

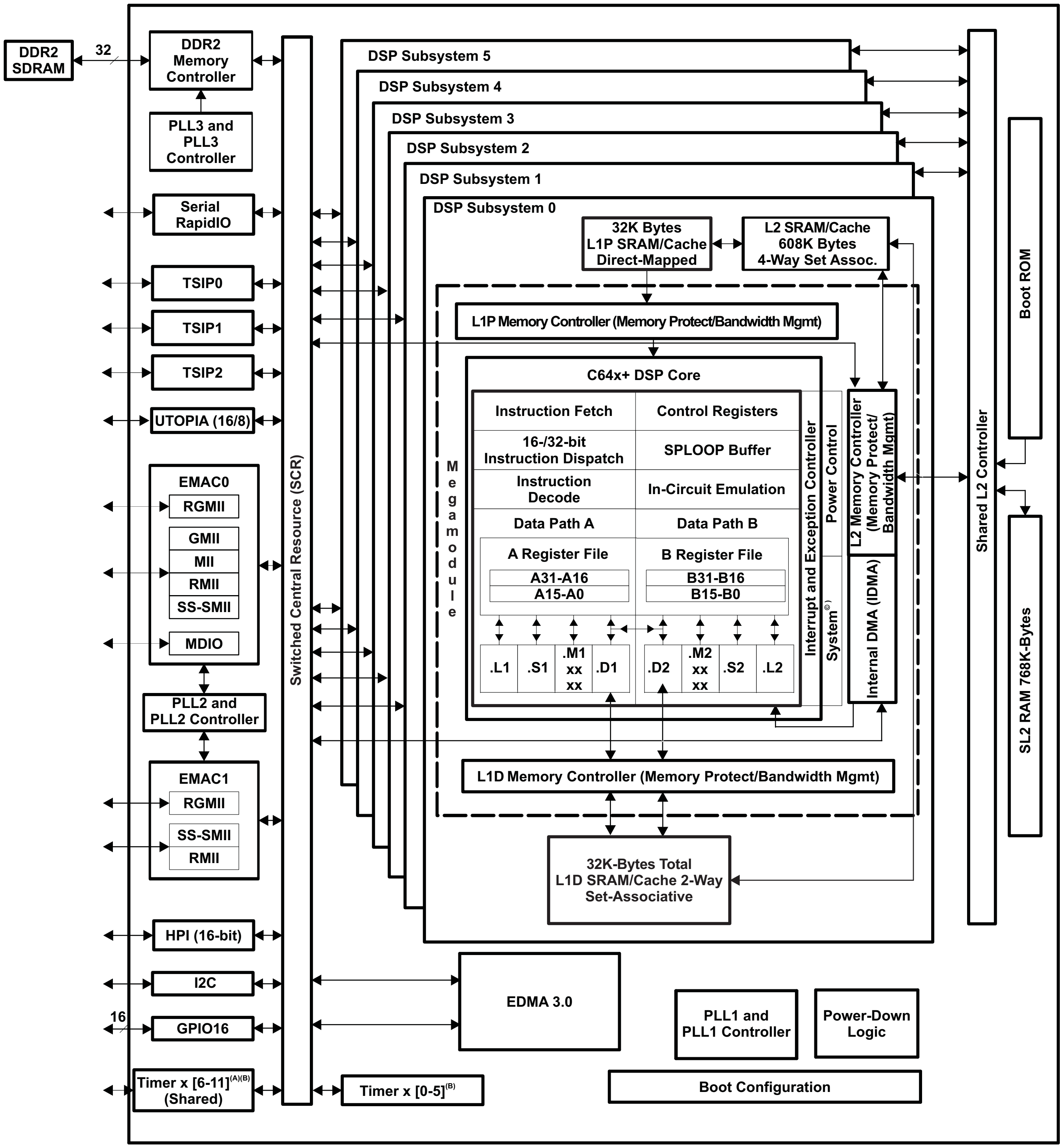
The peripheral direct memory access (PDMA) controller services the UTOPIA. The ATM adaptation layer (AAL) is commonly called, as segmentation and reassembly (SAR) functions should be performed in software.

All references to the term slave devices are analogous to multi-PHYs (MPHYs) as referenced in the ATM Forum specification. For MPHY systems, refer to the ATM Forum standard specification af-phy-0039.000. For single-PHY (single device) systems, refer to the ATM Forum standard specification af-phy-0017.000.

Figure 1 shows the TMS320TCI6486/TMS320C6472 block diagram. The UTOPIA port is connected to the TMS320TCI6486/TMS320C6472 subsystems through the PDMA and the switched central resource (SCR).

The UTOPIA slave consists of the transmit interface and the receive interface as shown in Figure 2. The interface signals are described in Section 3.1. The UTOPIA sends synchronization events to the PDMA controller through the UXEVT and UREVT signals. An interrupt signal (UINT) is also generated to the CPU to communicate error conditions (see Section 9).

Figure 1. TMS320TCI6486/TMS320C6472 DSP Block Diagram



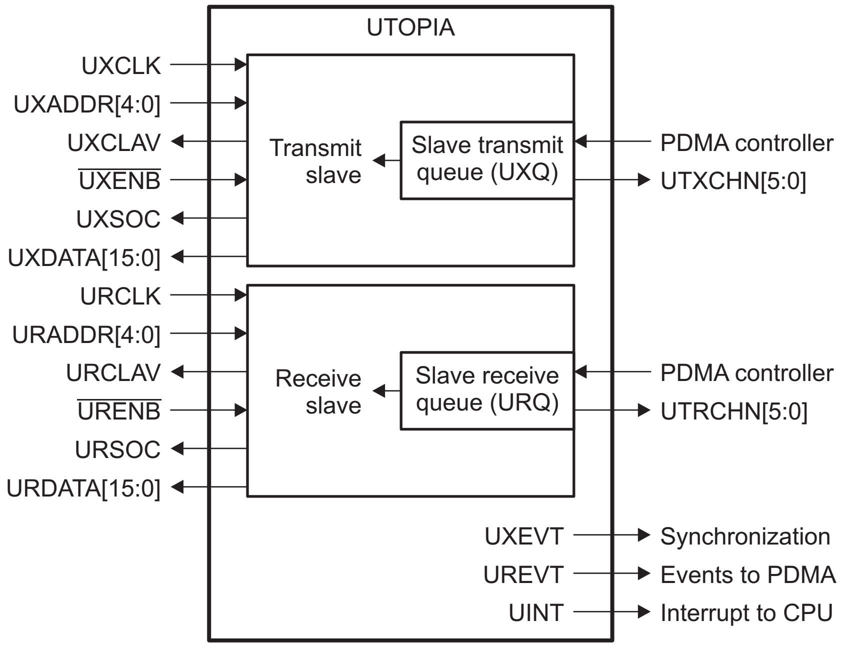
A Timers 6-11 are shared.

B Each of the Timer peripherals are configurable as either one 64-bit general-purpose timer *or* two 32-bit general-purpose timers *or* a watchdog timer.

C System consists of Test, Emulation, Power Down, and Interrupt Controller.

NOTE: For your peripheral set, see the *TMS320TCI6486 Communications Infrastructure Digital Signal Processor* data manual (SPRS300) or the *TMS320C6472 Fixed-Point Digital Signal Processor* data manual (SPRS612).

Figure 2. UTOPIA Block Diagram

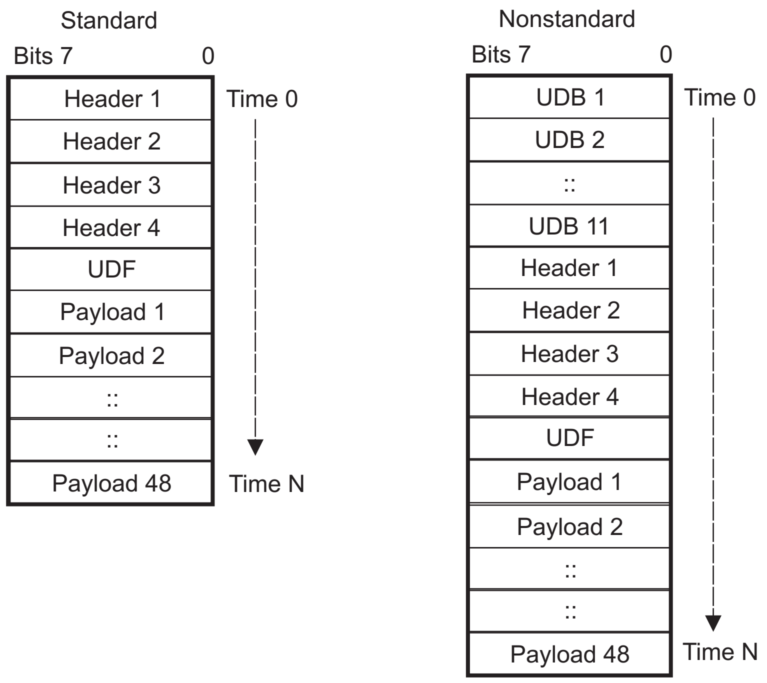


2 Cell Transfer Format

The ATM Forum specification for UTOPIA level 2 specifies the order in which header and payload information is sent across the ATM-PHY interface. The header information is sent first, followed by the 48-byte payload. A standard ATM cell is 53 bytes (5-byte header + 48-byte payload), as shown in Figure 3. The UTOPIA also supports a nonstandard ATM cell of size 54 to 64 bytes (R/XUDC = 1 to 11 + 5-byte header + 48-byte payload), as shown in Figure 3. The UTOPIA transmit queue and receive queue each accommodate two cells. The number of cells each queue accommodates is not dependent upon cell size.

For the TMS320TCI6486/TMS320C6472 DSP, each ATM cell must be aligned on a word-boundary. Therefore, each ATM cell (53 bytes) in the DSP memory (internal or external) and in the UTOPIA transmit/receive queues is padded with dummy bytes, as necessary, before the ATM header. The standard 56-byte cell-packet consists of the 53-byte ATM cell, plus 3 bytes of dummy data before the ATM header. This 56-byte packet is referred to as a cell packet. For more details, see Section 10.

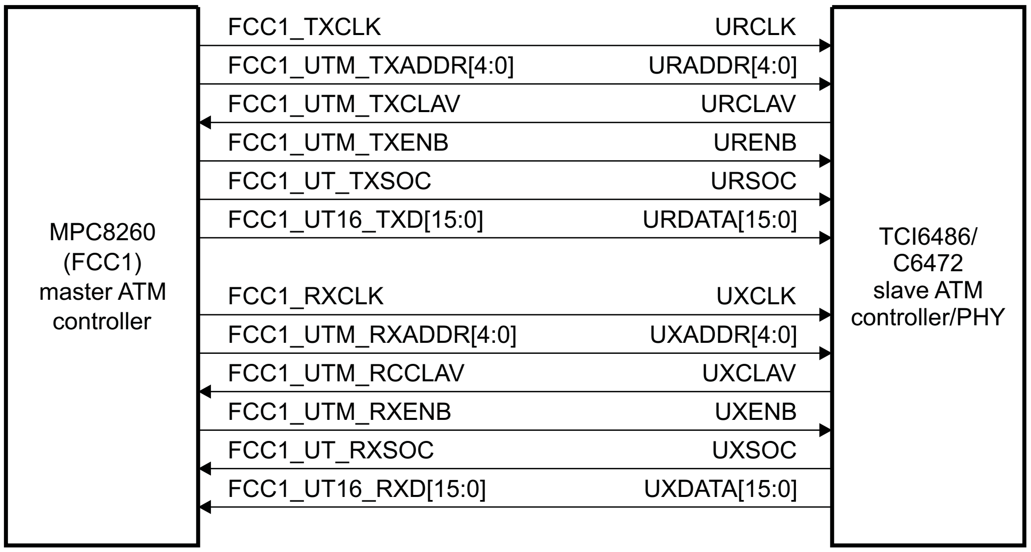
Figure 3. Cell Transfer Formats for 8-Bit Mode



3 UTOPIA Slave as ATM Controller

The UTOPIA interface is used as an ATM controller slave in either a single-PHY or multi-PHY (MPHY) configuration. As a slave, the clock, address, and enable signals of the transmit and receive interfaces are driven by the master. An example configuration is shown in Figure 4.

Figure 4. UTOPIA Slave Interfaced to Motorola MPC8260 Power QUICC IIE™ Master in 8-Bit Mode



3.1 UTOPIA Slave Pins

As a slave device in an ATM system, the UTOPIA performs all ATM cell transfers when directed by the master. The clock, address, and enable signals are inputs. The master can configure the slave's address in the UTOPIA control register (UCR) through the HPI/PCI interface. Table 1 and Table 2 show the pins and their direction in relation to the UTOPIA slave interface.

The slave responds when it detects its assigned address on the address bus by asserting its UXCLAV or URCLAV signal, if a cell is available for transmit or receive, respectively. If the slave does not have a cell to transmit or cell space to receive, it does not assert the relevant CLAV signal, but the master continues to poll the remaining slaves/PHYs in the system on the address bus.

Table 1. UTOPIA Transmit Interface Slave Pin Descriptions

Pin	Direction	Value	Description
UXCLK	In		UTOPIA Transmit Clock. An input driven by the master in the system. Transmit data and transmit control signals are synchronous to this clock.
UXADDR[4:0]	In		5-bit address input driven by the master ATM controller to identify each of the slave devices (up to 31) in the ATM system.
UXCLAV	Out		Transmit Cell Available status output signal of the slave. For cell-level handshake, the following is true:
		0	Indicates that the slave does not have a complete cell available for transmit.
		1	Indicates that the slave has a complete cell available to transmit.
$\overline{\text{UXENB}}$	In		UTOPIA Transmit Interface Enable input signal. Asserted active low by the master to indicate that the slave should put first byte of valid data and assert SoC signal in the next clock cycle.
UXSOC	Out		Transmit Start-Of-Cell signal (active high) output by the slave on rising edge of UXCLK to indicate that the first valid byte of the cell is available on the Transmit Data Bus UXDATA[15:0].
UXDATA[15:0]	Out		16-bit Transmit Data Bus. Slave transmits ATM cells to the master using this bus on rising edge of UXCLK.

Table 2. UTOPIA Receive Interface Slave Pin Descriptions

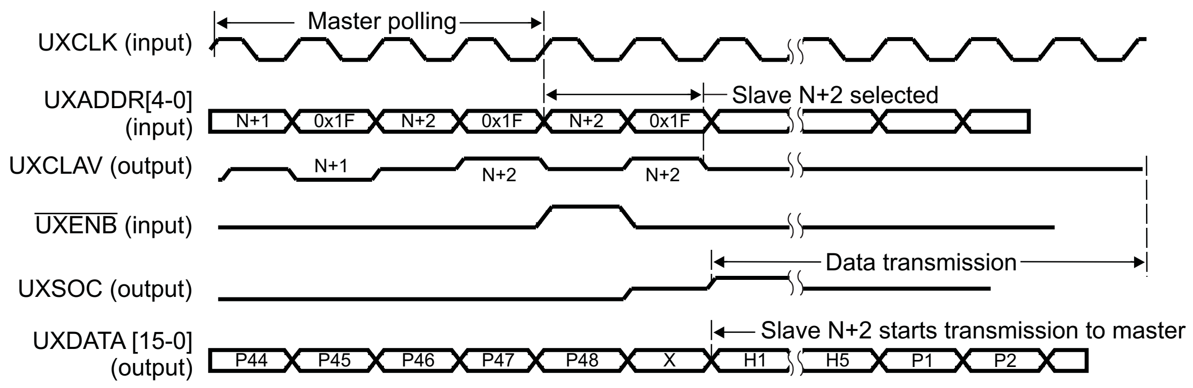
Pin	Direction	Value	Description
URCLK	In		UTOPIA Receive Clock is an input signal driven by the ATM master. Receive data and control signals are sampled and synchronous to this clock.
URADDR[4:0]	In		5-bit address bus input driven by the master to select a slave.
URCLAV	Out		Receive Cell Available status signal is an output from the slave to indicate that it has space available to receive a cell from the master. For cell-level handshake, the following is true:
		0	No space is available to receive a cell from the master.
		1	Space is available to receive a cell from the master.
$\overline{\text{URENB}}$	In		UTOPIA Receive Interface Enable. An active-low signal driven by the master to enable the receive interface of the slave. It tells the slave to sample Receive Data and SoC signals in the next clock cycle or thereafter.
URSOC	In		Receive Start-Of-Cell signal driven by the master to indicate that the first valid byte of the cell is available on the Receive Data Bus for the slave to sample.
URDATA[15:0]	In		16-bit UTOPIA Receive Data Bus. Data from the master is received on this bus. Data is sampled on the rising edge of URCLK.

3.2 Slave-Transmit Operation

The UTOPIA slave-transmit block consists of a UTOPIA level 2 pin interface that interfaces internally to the slave-transmit queue. Figure 2 shows the UTOPIA slave-transmit block diagram. The slave-transmit queue is accessed through the UXQ data port. The PDMA controller services the slave-transmit queue with 32-bit writes when a transmit event is generated by the UTOPIA transmit section.

When the UTOPIA slave interface detects its address on the transmit address bus, UXADDR[4:0], it drives the UXCLAV signal to indicate to the master that a cell is or is not available for transmit. In the following cycles, when the master chooses (after completion of any ongoing data transfers) to receive the data from this UTOPIA slave, the master asserts the slave address along with the enable signal, $\overline{\text{UXENB}}$. Next, the slave starts transmitting the data on its UXDATA[15:0] pins, by asserting the start-of-cell signal, UXSOC. Figure 5 shows the UTOPIA slave-transmit interface timing. The clock for the UTOPIA slave-transmit interface, UXCLK, is an input driven by the external master.

Figure 5. ATM Controller Slave Transmit Timing Diagram



3.3 Slave-Transmit Queue

The slave-transmit queue prepares the UTOPIA interface to be ready to transmit data whenever the master requests a transmit. The slave-transmit queue generates a transmit event (UXEVT) when it is not full. This transmit event triggers the PDMA controller to perform 32-bit writes to the slave-transmit queue. A total of 14 word writes are required to fill one standard ATM cell packet in the queue.

As soon as the first write to the queue occurs, the transmit event is cleared. The next transmit event is generated if the queue is not full. This allows the PDMA controller to begin the next cell-packet write without having to wait for the current cell to be fully written to the queue. This process repeats as described below.

The transmit event is generated and cleared as follows:

1. UXEVT is generated when the queue is not full. The queue is not full when there is space for at least one cell packet (56B).
2. UXEVT is cleared when the first write (by the PDMA controller) of that cell occurs.
3. UXEVT is regenerated immediately (without waiting for the previous cell to be fully written) if the queue is not full.
4. Go to step 2.

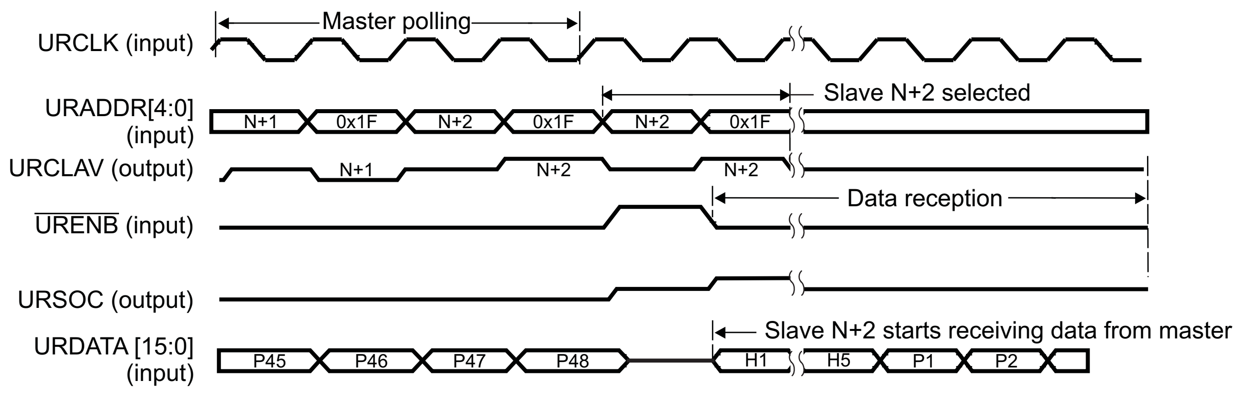
The UTOPIA slave indicates the availability of a cell to transmit to the master by asserting its UXCLAV signal when there is at least one cell available in the slave-transmit queue. If the slave cannot provide the next cell in a contiguous fashion, it de-asserts its UXCLAV signal in the cycle following the completion of the current cell transmission. The UXCLAV signal remains asserted if the slave has another cell available to transfer to the master. The master can disable $\overline{\text{RXENB}}$ on its side (connected to the $\overline{\text{UXENB}}$ pin for this ATMC slave), which causes the UTOPIA slave to hold off the next cell transfer until the master commands the transfer.

3.4 Slave-Receive Operation

The UTOPIA slave-receive block consists of a UTOPIA level 2 pin interface that interfaces internally to a slave-receive queue. The UTOPIA slave-receive block diagram is shown in Figure 2. The slave-receive queue is accessed through the URQ data port. The PDMA controller services the slave-receive queue with 32-bit reads when a receive event is generated by the UTOPIA receive section.

When the master polls for slaves in the system that can receive its cells, the UTOPIA slave responds with an active cell-available signal on its URCLAV pin if it has space in the slave-receive queue to receive a complete cell. The master transmits to this slave or continues to poll to find a suitable slave for its data. The UTOPIA slave responds to its assigned address by asserting its appropriate URCLAV state a cycle after its address is detected on the receive address bus, URADDR[4:0]. The master then outputs the slave address that has an active RCLAV signal and also provides the enable signal ($\overline{\text{URENB}}$ on slave) to enable slave-receive operation. The UTOPIA receive slave starts receiving data in the cycle when the master asserts its start-of-cell signal, URSOC. The bytes are assembled into words and written into the slave-receive queue. Figure 6 shows the UTOPIA slave-receive interface timing. The clock for the UTOPIA slave-receive interface, URCLK, is an input driven by the external master.

Figure 6. ATM Controller Slave Receive Timing Diagram



3.5 Slave-Receive Queue

When the master initiates the transfers to the slave, the slave-receive queue generates a receive event (UREVT) to the PDMA controller when at least one cell worth of data is available. As soon as the first read is performed by the PDMA controller, the receive event is cleared. The next receive event is generated when the next cell is fully available and the process repeats.

The receive event is generated and cleared as follows:

1. UREVT is generated when a complete cell is available.
2. UREVT is cleared when the first read (by the PDMA controller) of that cell occurs.
3. UREVT is regenerated when the next complete cell is available in the slave-receive queue.
4. Go to Step 1.

The UTOPIA slave agrees for reception from the master by asserting its URCLAV signal when there is at least one cell space available in the slave-receive queue. If the slave cannot receive the next cell immediately, it de-asserts its URCLAV signal at least four URCLK cycles before the end of this cell transfer. If it remains asserted, it indicates that the slave can receive another cell from the master. The master can disable TXENB on its side ($\overline{\text{URENB}}$ for this ATMC slave) at its discretion.

3.6 UTOPIA Events Generation

The UTOPIA transmit and receive queues generate not-full and not-empty events to the PDMA controller. The events are generated when the queues have space available for at least one cell and not when the queues are completely full or empty. This allows for more throughput and better performance because the transmit and receive data are continuously transferred without having to wait for a full or empty queue. For details on the generation of these events, see Section 3.3 and Section 3.5. The PDMA controller services the UTOPIA in response to these events, as discussed in Section 4. UTXCHN[5:0] and URXCHN[5:0] denotes the channel that triggers the PDMA for transfer from UTOPIA.

3.7 Multi-PHY (MPHY) Operation

The UTOPIA interface supports multi-PHY operation as per UTOPIA level 2 specification. The MPHY mode is enabled when the MPHY bit in the UTOPIA control register (UCR) is set to 1 (default state). In MPHY mode, the slave ID (SLID) bit in UCR indicates the PHY address of the UTOPIA. Either the DSP or the external master programs the SLID bits. MPHY operation is based on cell-level handshaking.

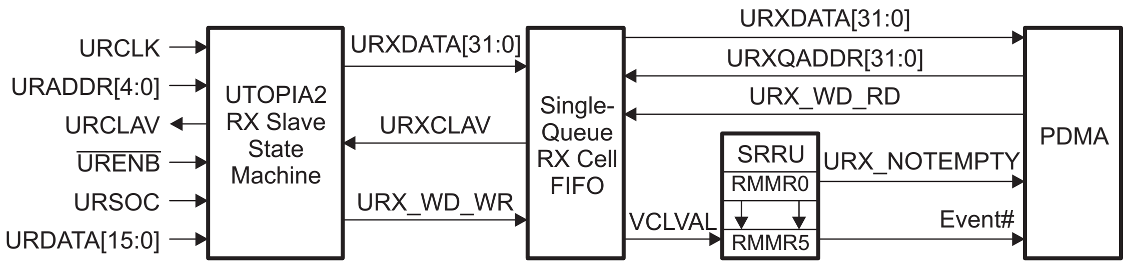
As shown in Figure 5 and Figure 6, the external ATM master polls for available slave devices before the beginning of the actual data transaction. The UTOPIA output signals URCLAV, UXCLAV, UXSOC, and UXDATA[15:0] are in high-impedance state when the UTOPIA slave is not selected by the master. When the UTOPIA slave detects its address at the UXADDR[4:0] or URADDR[4:0] pins, it asserts the UXCLAV or URCLAV signal, respectively.

When used in single-PHY mode (MPHY = 0 in UCR), there is no need to program the address.

3.8 Slave Receive Routing Unit (RRU)

The receive cells in the RX FIFO is transferred to six different buffers. ATM cells are required to be separated into six buffers for processing six channels of data. You program the segregation of cells into different buffers through a set of mask-match registers for each buffer. These registers are part of the receive routing unit in the slave state machine. The RRU module interfaces to the RX FIFO on one side and the PDMA on the other. The RRU does not have buffer space, but generates up to six appropriate PDMA events when a cell is fully available for a particular mask and match value you defined.

Figure 7. Slave Receive with Receive Routing Unit



3.8.1 User-Defined Routing

ATM cells are routed based on byte and bit offsets you define within the 53-byte (standard) to 64-byte (user-defined) ATM cell. The routing is based on a 16-bit field (maximum) within the ATM cell header (standard or user-defined cell).

The byte (RBYT) and bit offsets (RBIT) are used to select the 16-bit routing value that is programmable in the receive routing select register (RRSR) at location 0x01B40040 (register offset 0x40). The RRSR register is set up when the device is in reset or the module (in this case the receive module) is in reset.

The default value of 0x0 in RRSR indicates no routing is performed. The ATM cell is treated as starting with byte number 1 corresponding to HeaderByte1. Therefore specifying RRSR [RBYT] = 0 signifies no routing requirements. The RRSR [RBYT] value indicates the least significant byte of the 16-bit field of interest and RRSR [RBIT] number indicates the least significant bit within that 16-bit field of interest that is used for routing decisions. For example, if you prefer VCI-based routing, the RRSR value would be 0x00040004. The LSB of the VCI value in the ATM cell is located in HeaderByte4 in bit position 4. The 16-bit value is copied from this position to be used for match-mask operation for that cell.

The 6-bit RBYT field allows routing values to be placed anywhere in the ATM cell. However, if the routing value is placed starting at a byte location greater than 32, it can incur a delay of up to five peripheral bus clocks for the receive event to be generated for that cell. The delay is avoided by always placing routing values in the header bytes (5 in a normal ATM cell and up to 16 in a user-defined cell), which is the norm in most applications. If an invalid RBYT value is programmed, the routing operation is unpredictable. Invalid values are any values beyond the cell boundary except for value 0.

For details on the RRSR register, see Section 15.1.6.

3.8.2 Six-Way Buffering

There are six receive mask-and-match registers (RMMR) to perform 6-way buffering by the PDMA. Although the field of interest is always extracted as a 16-bit value, you have the option of choosing the required number of bits by programming the RMMR0-5 registers. ATM cells with 16-bit routing field values that match with one of the six mask-and-match values are serviced by one of the six PDMA events. The mask-and-match registers are 32-bit wide with 16-bit mask value on the lower half and 16-bit match value in the upper half of the register, as shown in Figure 13. The default value for these registers is 0. This implies that all cells are passed through to a single buffer unless the SSRU is programmed for user-defined routing through RRSR and RMMRs.

The mask and match process is on a cell-by-cell basis and starts as soon as the 16-bit routing value is extracted based on RRSR value. The result of the mask-and-match operation gates the complete-cell-available read event to the PDMA. The mask-match process is as follows:

1. The following occur simultaneously:
   - The 16-bit VCI field of all incoming cells is logically ANDed with the 16-bit mask value in each of the mask-and-match registers, starting with RMMR0 up to RMMR5.
   - The 16-bit match value is logically ANDed with the 16-bit mask value for each of the six sets of registers.
2. The results of the two independent AND operations are compared and, if they are equal, the appropriate PDMA event is generated after a complete cell is received.

This round-robin (with RMMR0 as high priority and RMMR5 as low priority) mask-and-match operation prevents multiple matches and data being replicated on multiple buffers. If the result of the mask-and-match operation is true (1), it implies that the cell, indeed, belongs to this device and, in addition, generates a PDMA event for that channel (one of six).

For details on the RMMR0-5 registers, see Section 15.1.5.

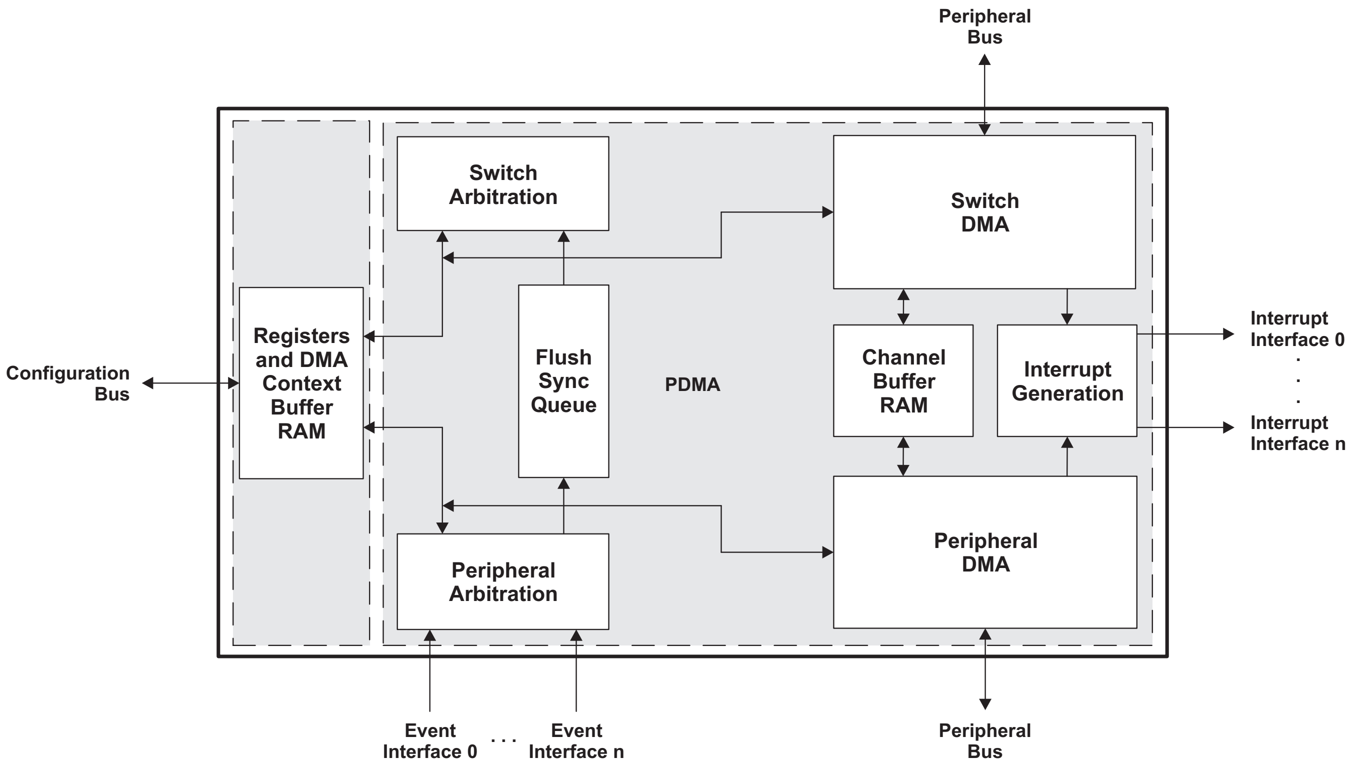
4 PDMA Controller Servicing UTOPIA

The PDMA module is a DMA controller element of a distributed DMA topology. A PDMA module is used to provide data transfers between a DSP subsystem and UTOPIA. The interface of the PDMA module and the DSP subsystem is through a switch.

4.1 PDMA Functional Overview

The PDMA is used to create a distributed DMA topology. In a distributed DMA topology, each peripheral in the system has its own DMA controller, the PDMA module. With a PDMA module a large number of channels are established and maintained. In a typical central DMA scheme each peripheral is configured to a single physical DMA channel. However, in many cases the I/O stream needs of the peripheral is best characterized and supported with multiple channels. The PDMA module enables this capability, scaled to the requirements of the particular peripheral. Figure 8 illustrates the block diagram of the PDMA module, along with its interfaces.

Figure 8. PDMA Block Diagram



The PDMA peripheral bus interfaces include one peripheral bus used for configuration, one peripheral data bus used for data transfers with slave peripherals (modules), and one peripheral data bus used for data transfers through the switch. The module also includes two peripheral event interfaces. Each peripheral event interface corresponds to an event group. Each event interface consists of up to ten ID lines and three synchronization inputs. The ID lines are used for PDMA channel selection in connection with the transfer synchronization. An interrupt interface, consisting of six interrupt signals, is added for each event interface. An interrupt identification port carries the PDMA channel number responsible for the interrupt in conjunction with the asserted interrupt signal.

4.2 PDMA Channel Context

Each data transfer specified by the PDMA context is defined by switch and peripheral addresses, transfer counts, and transfer control. The transfer control definition includes the switch and peripheral address modification and burst length to support burst transfers. Burst transfers allow the PDMA operation to make more effective use of the available transfer bandwidth.

PDMA channel context should only be written to a disabled channel to configure it and enable transfers, or to disable a channel that is enabled. Before the context of a channel is modified it must first be placed in a disabled state. If a PDMA channel is enabled and an attempt is made to enable the channel with a new context, the new context is ignored.

When the channel context is written to enable a channel that is in the disabled state, the STRT bit of the context is also registered. The channel is now enabled, although it is not yet active. For channels that correspond to the receive path for peripheral to memory transfers, the channel is registered as active on the next clock cycle. Channels that correspond to the transmit path for peripheral to memory transfers do not become active until data has been transferred from memory and stored in the PDMA buffer. Once data is in the PDMA buffer, the channel is registered as active on the next clock cycle. The added condition of data in the PDMA buffer for the transmit path avoids loss of memory buffer synchronization or other premature error conditions that could occur if the channel was activated prematurely.

When the channel context is written to disable a channel that is in the enabled state, the channel is disabled. That is, the enabled and active registered bits for the channel are cleared. The context for the channel is updated to reflect the disabled state. These updates take place when the channel context is written through the proxy registers and when the channel becomes disabled at the completion of a block transfer.

4.3 Data Transfer Modes

The PDMA offers two modes of data transfer operation. The first, ABU, is an auto-buffering mode. The second is a block (or non-ABU) mode. The ABU field in the Transfer Control register selects the data transfer operation mode and controls the operation of the DMA transfer counter for each channel.

4.3.1 ABU Mode

The ABU mode is used to implement auto-buffering functions. Auto-buffering enables a single buffer to be used in a continuous or circular manner. When the last address location in the buffer is accessed, the next access to the buffer is made to the first address location in the buffer.

4.3.1.1 ABU Buffer Size

The buffer size register contains the size of the buffer at the switch address in the lower 16 bits. The buffer size register contains the size of the buffer at the peripheral address in the upper 16 bits. Each field defines the buffer size in bytes and can be up to 64K. Each field in the buffer size register is interpreted as a 16-bit unsigned integer and valid buffer sizes range from 0008h to 0FFFCh. The buffer can be any size in this range that is an integer multiple of the element size in bytes and is not limited to a power of 2.

4.3.1.2 ABU Buffer Address

The buffer has a minimum address, called the base address, and a maximum address. The difference between the base address and the maximum address is the buffer size. The base address must be based on a power of 2. The base address must be located on an address boundary in which the N least significant bits of the address are zero. That is, where N is the smallest integer that satisfies the 2N being greater than the buffer size. The address boundaries for all available buffer sizes are shown in Table 3. Circular buffers cannot cross 64K address boundaries (16K 32-bit elements).

Table 3. ABU Buffer Examples for 32-bit Transfer Elements

ABU Buffer Size (hexadecimal)	Bytes (decimal)	Buffer Base Address (binary)
0x0008-0x000C	8:12	XXXX XXXX XXXX XXXX 0000b
0x0010-0x001C	16:28	XXXX XXXX XXXX XXX0 0000b
0x0020-0x003C	32:60	XXXX XXXX XXXX XX00 0000b
0x0040-0x007C	64:124	XXXX XXXX XXXX X000 0000b
0x0080-0x00FC	128:252	XXXX XXXX XXXX 0000 0000b
0x0100-0x01FC	256:508	XXXX XXXX XXX0 0000 0000b
0x0200-0x03FC	512:1020	XXXX XXXX XX00 0000 0000b
0x0400-0x07FC	1024:2044	XXXX XXXX X000 0000 0000b
0x0800-0x0FFC	2048:4092	XXXX XXXX 0000 0000 0000b
0x1000-0x1FFC	4096:8188	XXXX XXX0 0000 0000 0000b
0x2000-0x3FFC	8192:16380	XXXX XX00 0000 0000 0000b
0x4000-0x7FFC	16834:32764	XXXX X000 0000 0000 0000b
0x8000-0xFFFC	32768:65532	XXXX 0000 0000 0000 0000b

The following example helps to illustrate the application of buffer size with respect to base address boundaries:

For a buffer size of 8 (decimal), the next higher power of 2 is 16, so the buffer base address must be aligned with 16 long-word (32-bit element) boundaries (for example: 0x0000, 0x0040, 0x0080).The DMA transfer always starts at the addresses specified in the channel's switch or peripheral address registers. The address register can point to any location inside the defined range of the buffer. After each access, the appropriate address register is modified according to the specified address mode.

4.3.1.3 ABU Address Modification

The address modification bits (SMOD and PMOD), determine whether a circular buffer or single element buffer is being addressed on either side. The possible combinations of circular and single element buffers are as shown in Table 4.

Table 4. ABU Addressing Modes

SMOD	Address Modification	PMOD	Address Modification	Description
0	Post-increment	0	Post-increment	Dual circular buffers
0	Post-increment	1	No modification	Switch circular buffer, peripheral single buffer
1	No modification	0	Post-increment	Switch single buffer, peripheral circular buffer
1	No modification	1	No modification	Dual single buffers

When increment addressing is used, the buffer size is determined by the corresponding buffer size field and, as transfers proceed, the address in the circular buffer is incremented by 1. When the address reaches the end of the buffer, it wraps back to the beginning automatically. The number of transfers is not specified in this mode, so the address wraps indefinitely until the DMA channel is disabled.

4.3.2 Block or Non-ABU Mode

In block mode, when auto-buffering is not enabled, the PDMA transfers a single block of elements. When the transfer of the block is complete, the PDMA channel must be reinitialized before it is used for another transfer. A block consists of a number of bytes as defined in the count field of the transfer count register.

4.3.2.1 Block Buffer Size

The count defines the size of a block in bytes. Up to 128 Mbytes are transferred as a single block. As each element is transferred the count is decremented by the number of bytes corresponding to the element size. The buffer size, in bytes, must be an integer multiple of the element size in bytes. When the count reaches 0 the transfer is complete and the channel is stopped.

4.3.2.2 Block Buffer Address

The base address of the block must be 32-bit aligned. There is no additional power of 2 restrictions on buffer addresses based on buffer size in this mode. A buffer with a 32-bit aligned address can be located anywhere in memory. Addresses are defined by the switch address and peripheral address in the PDMA channel context.

4.3.2.3 Block Address Modification

As each element is transferred the addresses are modified. Address modification can either be a post increment or no modification.

4.4 PDMA Setup for UTOPIA Transmitter

The UTOPIA transmitter generates a UXEVT synchronization event to the PDMA controller when at least one cell-packet space is available in the slave-transmit queue. Per the UXEVT synchronization event, one frame of cell-packet data is transferred to the slave-transmit queue. A standard cell-packet consists of 14 words (56 bytes), while a nonstandard cell-packet consists of 14, 15, or 16 words (56, 60, or 64 bytes). The PDMA access to the UTOPIA is always 32 bits. The PDMA source address should point to the UTOPIA source buffer in the DSP memory (internal or external). The PDMA destination address should point to the slave-transmit queue data port UXQ.

Example 1 provides a code example of an PDMA setup to service a UTOPIA transmitter for a single cell-packet.

Example 1. PDMA3 Setup for UTOPIA Transmitter Code Example

```
/* Configures PDMA Transfer Control for Transfer Parameters */
PdmaHwSetup (cell_num_words);

/*Pdma TX Channel Set up to configure Channel context */
PdmaChSetup ((Uint32) XmtAddr, (Uint32) RcvAddr, Uint32 cell_num_bytes_pad,
    Uint32 Cell_Num, Uint32 Sint);

/* Mapping TX Event ID to PDMA Transfer Control Registers */
CSL_pdmaHwSetup (hpdma,&PdmaSetup_TX,TX_EVENT);

/* Populating PDMA Channel context to Channel context registers */
CSL_chHwSetup (hpdma,&PdmaSetup_TX,TX_CHANNEL_NUM);

/* Pdma Channel Enable*/
CSL_pdmaHwControl (hpdma,PDMA_GLOBAL_ENABLE,(Uint32)NULL);

//*****************************************************************************\
* PDMA Hardware Set Up for TX Channel
\*****************************************************************************/
/* Configures Global and Peripheral control registers to transfer num_words 32B *\
/* words in a cell */
void PdmaHwSetup(Uint32 num_words)
{
       /* Pdma halted. */
        PdmaGblSetup.strt=0x0;

        /* Emulation Mode Disabled */
        PdmaGblSetup.soft=0x0;
        PdmaGblSetup.free=0x0;

        /* Little Endian Mode selected */
        PdmaPeriCtlSetup_TX.bend=0x0;

        /* Clear the statistics Register */
        PdmaPeriCtlSetup_TX.clrs=0x0;

        /* Select Peripheral Event Control and Synchronization */
        PdmaPeriCtlSetup_TX.pe=0x1;
        PdmaPeriCtlSetup_TX.sync=0x1;

        /* Priority Setup */
        PdmaPeriCtlSetup_TX.pri=0x3;

        /* Disables auto-buffering mode */
        PdmaPeriCtlSetup_TX.abu=0x0;

        /* Switch to Peripheral Transfer Enabled */
        PdmaPeriCtlSetup_TX.dir=0x0;

        /* Configures burst length as size of cell in words */
        PdmaPeriCtlSetup_TX.pblen=(num_words-1);
        PdmaPeriCtlSetup_TX.sblen=(num_words-1);

        /* Select Constant addressing mode at Peripheral
           side and increment addressing mode at Switch Side */
        PdmaPeriCtlSetup_TX.pmod=0x1;
```

Example 1. (continued)

```
        PdmaPeriCtlSetup_TX.smod=0x0;
}

//***************************************************\
* PDMA TX Channel Configuration
\***************************************************/
void PdmaChSetup(Uint32 XmtAddr, Uint32 RcvAddr, Uint32 num_bytes, Uint32 num_cell,
    Uint32 sint )
{
        /* Enable the Channel Context */
        ChTferCtl_TX.strt = 0x1;

        /* Disables the auto buffering mode*/
        ChTferCtl_TX.imod=0x0;

        /* Enable the interrupt*/
        ChTferCtl_TX.ie=0x1;

        /* Asserts one of the 6 interrupt lines from the event Grp*/
        ChTferCtl_TX.sint=sint;

        /*Initialize to 0 in the block mode*/
        ChTferCtl_TX.bcz=0;

        /* 32-bit Peripheral and Switch Interface */
        ChTferCtl_TX.psiz=0x2; ChTferCtl_TX.ssiz=0x2;

        /*Initialize FIFO Flag, Write and Read pointer offset to 0 */
        ChTferCtl_TX.ff= 0x0;
        ChTferCtl_TX.wrptr=0x0;
        ChTferCtl_TX.rdptr=0x0;.

        /*Source Address of the Buffer in Switch side*/
        ChanContext_TX.SAR=XmtAddr;

        /* UTOPIA TX Queue Address*/
        ChanContext_TX.PAR=TXQue_Addr;

        /* Size of circular buffer */
        ChanContext_TX.CNT.BUFF= (num_cell*num_bytes);
}
```

4.5 PDMA Setup for UTOPIA Receiver

The UTOPIA receiver generates a UREVT synchronization event to the PDMA controller when the slave-receive queue has space for at least one cell-packet. Per UREVT synchronization event, one frame of cell-packet data is read from the slave-receive queue by way of the data port URQ. A standard cell-packet consists of 14 words (56 bytes), while a nonstandard cell-packet consists of 14, 15, or 16 words (56, 60, or 64 bytes). The PDMA destination address should point to the destination buffer in the DSP memory (internal or external).

Example 2 shows a code example of an PDMA3 setup to service a UTOPIA receiver for a single cell-packet.

Example 2. PDMA3 Setup for UTOPIA Receiver Code Example

```
/* Configures PDMA Transfer Control for Transfer Parameters */
PdmaHwSetup (cell_num_words);

/*Pdma RX Channel Set up to configure Channel context */
PdmaChSetup((Uint32) XmtAddr, (Uint32) RcvAddr, Uint32 num_bytes,
    Uint32 num_cell, Uint32 sint )

/* Mapping RX Event ID to PDMA Transfer Control Registers */
CSL_pdmaHwSetup (hpdma,&PdmaSetup_RX,RX_EVENT);

/* Populating PDMA Channel context to Channel context registers */
CSL_chHwSetup (hpdma,&PdmaSetup_RX,RX_CHANNEL_NUM);

/*Pdma Gbl Enable*/
CSL_pdmaHwControl (hpdma, PDMA_GLOBAL_ENABLE,(Uint32)NULL);

//****************************************************\
* PDMA Hardware Set Up for RX Channel
\****************************************************/
/* Configures Global and Peripheral control registers to transfer num_words
     32B words in a cell */
void PdmaHwSetup(Unit32 num_words)
{
        /* Pdma halted. */
        PdmaGblSetup.strt=0x0;

        /* Emulation Mode Disabled */
        PdmaGblSetup.soft=0x0;
        PdmaGblSetup.free=0x0;

        /* Little Endian Mode selected */
        PdmaPeriCtlSetup_RX.bend=0x0;

        /* Clear the statistics Register */
        PdmaPeriCtlSetup_RX.clrs=0x0;

        /* Select Peripheral Event Control and Synchronization */
        PdmaPeriCtlSetup_RX.pe=0x2;
        PdmaPeriCtlSetup_RX.sync=0x1;

        /* Priority Setup */
        PdmaPeriCtlSetup_RX.pri=0x3;

        /* Enable auto-buffering mode */
        PdmaPeriCtlSetup_RX.abu=0x1;

        /* Peripheral to Switch Transfer Enabled */
        PdmaPeriCtlSetup_RX.dir=0x1;

        /* Configures burst length as size of cell in words */
        PdmaPeriCtlSetup_RX.pblen=(num_words-1);
        PdmaPeriCtlSetup_RX.sblen=(num_words-1);

        /* Select Constant addressing mode at Peripheral
           side and increment addressing mode at Switch Side */
        PdmaPeriCtlSetup_RX.pmod=0x1;
        PdmaPeriCtlSetup_RX.smod=0x0;
```

Example 2. (continued)

```
}

//***************************************************\
* PDMA RX Channel Configuration
\***************************************************/
void PdmaChSetup(Uint32 XmtAddr, Uint32 RcvAddr, Uint32 num_bytes,
    Uint32 num_cell, Uint32 sint )
{
        /* Enable the Channel Context */
        ChTferCtl_RX.strt = 0x1;

        /* Disables the auto buffering mode*/
        ChTferCtl_RX.imod=0x0;

        /* Enable the interrupt*/
        ChTferCtl_RX.ie=0x1;

        /* Asserts one of the 6 interrupt lines from the event Grp*/
        ChTferCtl_RX.sint=sint;

        /*Intialize to 0 in the block mode*/
        ChTferCtl_RX.bcz=0;

        /* 32 bits at the Periperal side and Switch Interface*/
        ChTferCtl_RX.psiz=0x2;
        ChTferCtl_RX.ssiz=0x2;

        /*Initialize FIFO Flag, Write and Read pointer offset to 0 */
        ChTferCtl_RX.ff= 0x0;
        ChTferCtl_RX.wrptr=0x0;
        ChTferCtl_RX.rdptr=0x0;.

        /*Dst Address of the Buffer in Switch side*/
        ChanContext_RX.SAR=RcvAddr;

        /* UTOPIA RX Queue Address*/
        ChanContext_RX.PAR=RXQue_Addr;

        /*Buffer size of the peripheral buffer and switch buffer*/
        ChanContext_RX.CNT.BUFF=
        CSL_FMK(PIM_CNT_PBUF_SIZE,Buff_Size_Circular|
        CSL_FMK(PIM_CNT_SBUF_SIZE,(num_cell*num_bytes));
}
```

5 PDMA Interface Manager Functional Overview

The PDMA interface manager (PIM) is an interface bridge to the PDMA. The purpose of the PIM is to assure the integrity of PDMA programming in a multi-core environment. The PDMA module is also capable of acting as an event server for multi-core devices, where the peripheral module that the PDMA serves can generate an event that the PDMA holds until one of the cores programs a PDMA channel that can satisfy the event. The context for the PDMA channels is maintained in a 128-bit wide memory. These channels are programmed indirectly through a register interface that includes, among other registers, a 32-bit channel ID register and four 32-bit proxy registers that are used to program the actual context memory. The PIM defines a small amount of interface logic and registers that is based on CBA3.0 and assures the atomicity of these accesses from separate cores in a multi-core device.

The PIM is an interface bridge to a PDMA register file that allows multiple cores in a device to safely program individual PDMA channels. Through software allocation, each core is granted exclusive permission to one PDMA channel. The operation of a PDMA channel is defined by a 128-bit wide context RAM consisting of a switch address, peripheral address, transfer count/size, and transfer control. On a write to channel proxy registers in PIM, it configures the channel context registers for the corresponding channel in PDMA. Programming of this 128-bit channel context is completed by writing to five 32-bit registers in the PDMA register file. For example, after configuring the 128-bit channel context onto Channel Proxy register at offset 0x100, the PIM configures the PDMA Channel 0 context registers. On a write to the transfer control proxy, the contents of the proxy registers are used to write the selected channel context RAM location.

Similarly, a read to channel proxy registers in PIM effect a read to channel context registers in PDMA. The channel context of any channel can be read at any time, regardless of the state of the STRT field. On a read to the switch address proxy, the 128-bit entry in the context RAM for the selected channel is copied into the proxy registers. There is a small delay to allow the proxy registers to be loaded before the first read is able to complete. The remaining proxy registers are read on the next three consecutive cycles.

The MSB of the transfer control proxy is the STRT field (or enable/disable) for the PDMA channel, where STRT=1 defines an enabled channel and STRT=0 defines a disabled channel. If the current state of the STRT field is 1 then the context RAM is updated with the contents of the proxy registers only if the STRT field in the transfer control proxy register is 0. In other words, the context of an enabled PDMA channel cannot be changed by programming except to disable the channel. If the state of the STRT field is 0 then the write to the context RAM is made without specific consideration to the value of the STRT field in the transfer control proxy register.

6 Interrupt Generation and Servicing

6.1 UTOPIA Interrupt Generation

RX and TX interrupts of UTOPIA are not serviced by the CPU. UTOPIA error interrupt is serviced by the CPU.

6.2 PDMA Interrupt Generation

As PDMA transfers progress or complete the DSP can request an interrupt. When interrupts are asserted, the selected interrupt line is driven with an active high, single clock wide pulse. The channel identifier is asserted on the interrupt identifier port with the same timing as the interrupt signal. The interrupt signal is asserted by the PDMA two peripheral clock cycles after the PDMA receives a bus ready for the transfer corresponding to the interrupt condition. Control of interrupt generation depends on the PDMA application, mode of operation, and direction of transfer. The SINT field is used to select one of eight interrupt outputs for a group of PDMA channels.

Each of these interrupt outputs is connected to a different DSP subsystem. In this way, a PDMA that is used to perform data transfers between a UTOPIA and a DSP subsystem is able to assert an interrupt to the DSP subsystem that is the source or destination in the PDMA channel context. For a PDMA with two event interfaces, there are two interrupt output groups. This permits a unique interrupt to be connected to each DSP subsystem for interrupts generated by each group of PDMA channels. An interrupt identifier port is available to identify the specific channel responsible for the interrupt signal assertion.

Two fields (ABU and DIR) within the PDMA Peripheral Control Register or alternatively within the PDMA Memory-to-Memory Control Register and two fields (IE and IMOD) within the PDMA Transfer Control Register control how interrupts are handled during transfers. A third field, SINT, also in the PDMA Transfer Control Register is used to select which of several interrupt signals are actually asserted. IE is used to enable or disable interrupt generation on the completion of part or all of a transfer. If IE = 0, interrupts are disabled and no interrupt generation occurs. If IE = 1, interrupt generation is enabled and occurs based on the configuration of the IMOD bit. The available operation modes are shown below in table.

Table 5. DMA Block Transfer Interrupt Generation Modes

Mode	ABU	IE	IMOD	Interrupt Generation
ABU	1	1	0	At end of buffer only
ABU	1	1	1	At the half buffer and end of buffer
Block	0	1	0	At block transfer complete
Block	0	1	1	Reserved
Either	X	0	X	No interrupt generated

7 CPU Servicing UTOPIA

The RX and TX interrupts of the UTOPIA peripheral on TMS320TCI6486/TMS320C6472 devices are not serviced by the CPU. However, the PDMA and the error interrupts are required to be serviced by the CPU.

8 UTOPIA Clocking and Clock Detection

The transmit and receive clock for the UTOPIA interface is supplied by an external clock source such as the master ATM controller. This allows for accurate clocks as required by most applications. Internal to the DSP, the UTOPIA registers and queues are synchronized to the DSP peripheral clock at a CPU/3 rate.

Clock detection logic has been implemented to facilitate proper reset of the UTOPIA interface when the clock(s) is removed from the system. It uses a user-programmable Clock Detect register to trigger the clock detection based on the (R/X) CCNT bit-field parameters. Receive and transmit logic are independent of each other.

9 Special Transfer Conditions

This section explains how the UTOPIA slave interface handles some of the error conditions.

- **Runt cells**: Runt cells are those cells that are shorter than the standard ATM cell (53 bytes for 8-bit mode). This occurs when the device sending data asserts an SoC in the middle of a cell transfer. If the receive section of the UTOPIA detects an SoC before a complete cell is received, the byte count is reset and the runt cell is overwritten by the next new data. In the transmit direction, you cannot force an SoC; intentional runt cell generation is not supported.
- **Absence of UTOPIA clocks**: If, for any reason during a cell transfer in either direction, the receive clock (URCLK) or transmit clock (UXCLK) stop toggling, the corresponding section of the UTOPIA may be reset depending on the duration of absence. The queues are returned to their reset state, and all control registers are reset. In addition, the receive clock failed (RCFP) or transmit clock failed (XCFP) bit, respectively, is set in the error interrupt pending register (EIPR). An interrupt UINT is generated to the CPU if the corresponding bits in the error interrupt enable register (EIER) are set. This helps when recovering from conditions where the master card (that supplies the clocks) is pulled from the system.

  The receive clock present (RCPP) and transmit clock present (XCPP) bits in EIPR are set if the URCLK and UXCLK are detected, regardless of the state of the UTOPIA port. If the corresponding bits are enabled in EIER, an interrupt UINT is generated. This is useful in reenabling the UTOPIA ports once the UTOPIA clocks are detected.
- **Abrupt reset**: In slave mode, typically the master issues a reset command through the management interface to ensure a graceful stop of transfers. But if this is violated, data corruption occurs and the system software should comprehend this. In short, abrupt reset causes data loss/corruption. It is your responsibility to avoid such conditions.
- **FIFO read/write stall conditions**: There are two potential stall conditions when a read or a write to the transmit/receive queues is attempted when the transmit/receive queues are not ready (when UXEVT and UREVT are not active):
  – Writes to a FULL slave-transmit queue: Writing to the transmit queue that is full renders the queue not ready. In other words, writes are stalled until the queue is drained and space is available for further writes. Therefore, data is not overwritten. When such a condition occurs, the transmit queue stall interrupt pending (XQSP) status bit in EIPR is set to indicate the stall condition. The XQSP bit is a read-only bit and is cleared once the queue has space available, and writes can continue.
  – Reads from an EMPTY slave-receive queue: Attempting to read a queue that has no data results in stalling that operation until valid data is available. This also sets the receive queue stall interrupt pending (RQSP) bit in EIPR to indicate the read stall condition. This is a user error, because a read access is performed when there is no active UREVT. The RQSP bit is a read-only bit and is cleared as soon as valid data is available in the receive queue and the read is performed.

10 Endian Considerations

For 8-bit operation, shown in Table 6 through Table 11, bytes are assembled into words in the UTOPIA queues. Device endian configuration is selected during reset. Pin-level endian configuration ensures the desired endian mode for the DSP/CPU. For the UTOPIA interface to present the data transferred across its interface to the DSP in accordance with the device endian mode, the big-endian mode enable bit (BEND) in the UTOPIA control register (UCR) has to be programmed. By default/reset, the UTOPIA data is presented to the DSP in little-endian format (BEND = 0). If big-endian is preferred, the BEND bit should be programmed to 1. The data bytes are swapped in hardware based on the BEND value.

For 16-bit operation, shown in Table 12 through Table 15, the data transferred across the port (pins) follow big-endian format. In other words, the least significant (first) byte is driven on the upper half of the 16-bit bus. For little-endian the least significant (first) byte is stored first. For big-endian the most significant byte (the big end) is stored first. In 16-bit mode, the valid values for (R/X)UDC are 0 bytes to 10 bytes. This specification allows for RUDC and XUDC values of odd-number bytes. For example if XUDC=3 in 16-bit mode, the first byte (out of the first 16-bit data) should be considered invalid/dummy. This is followed by 16-bit data corresponding to two extra header bytes. In 16-bit mode 14-word accesses are performed for RUDC/XUDC values of 1-2, 15-word accesses for RUDC/XUDC values of 3 to 6, and 16-word accesses for RUDC/XUDC values of 7 to 10.

When the DSP is a UTOPIA slave in a system, it only communicates to the master. Therefore, the communication from the slave's perspective is always point-to-point. The cell-packet transfer formats are shown in Figure 3.

Depending on the user-defined cell (RUDC and XUDC bits) and endian format (BEND bit), bytes are placed in the transmit and receive queues as shown in Table 6 through Table 15.

The cell-packet formats in Table 6 through Table 15 indicate the data stored in the DSP memory (internal or external) and in the transmit/receive queues. Only the ATM data, including the header and payload information but not the Dummy bytes, is actually sent or received across the UTOPIA pins.

Table 6. 8-Bit UTOPIA Slave in Little-Endian Mode (BEND = 0) With RUDC/XUDC = 0

Queue Bits	31-24	23-16	15-8	7-0	Order
Address n	Header 1	Dummy	Dummy	Dummy	Word 0
Address n+4	UDF	Header 4	Header 3	Header 2	Word 1
Address n+8	Payload 4	Payload 3	Payload 2	Payload 1	:
::	::	::	::	::	:
Address n+48	Payload 44	Payload 43	Payload 42	Payload 41	:
Address n+52	Payload 48	Payload 47	Payload 46	Payload 45	Word 13

Table 7. 8-Bit UTOPIA Slave in Big-Endian Mode (BEND = 1) With RUDC/XUDC = 0

Queue Bits	31-24	23-16	15-8	7-0	Order
Address n	Dummy	Dummy	Dummy	Header 1	Word 0
Address n+4	Header 2	Header 3	Header 4	UDF	Word 1
Address n+8	Payload 1	Payload 2	Payload 3	Payload 4	:
::	::	::	::	::	:
Address n+48	Payload 41	Payload 42	Payload 43	Payload 44	:
Address n+52	Payload 45	Payload 46	Payload 47	Payload 48	Word 13

Table 8. 8-Bit UTOPIA Slave in Big-Endian Mode (BEND = 1) With RUDC/XUDC = 1

Queue Bits	31-24	23-16	15-8	7-0	Order
Address n	Dummy	Dummy	UDB 1	Header 1	Word 0
Address n+4	Header 2	Header 3	Header 4	UDF	Word 1
Address n+8	Payload 1	Payload 2	Payload 3	Payload 4	:
::	::	::	::	::	:
Address n+48	Payload 41	Payload 42	Payload 43	Payload 44	:
Address n+52	Payload 45	Payload 46	Payload 47	Payload 48	Word 13

Table 9. 8-Bit UTOPIA Slave in Little-Endian Mode (BEND = 0) With RUDC/XUDC = 7

Queue Bits	31-24	23-16	15-8	7-0	Order
Address n	UDB 4	UDB 3	UDB 2	UDB 1	Word 0
Address n+4	Header 1	UDB 7	UDB 6	UDB 5	Word 1
Address n+8	UDF	Header 4	Header 3	Header 2	:
Address n+12	Payload 4	Payload 3	Payload 2	Payload 1	:
::	::	::	::	::	:
Address n+56	Payload 48	Payload 47	Payload 46	Payload 45	Word 14

Table 10. 8-Bit UTOPIA Slave in Little-Endian Mode (BEND = 0) With RUDC/XUDC = 11

Queue Bits	31-24	23-16	15-8	7-0	Order
Address n	UDB 4	UDB 3	UDB 2	UDB 1	Word 0
Address n+4	UDB 8	UDB 7	UDB 6	UDB 5	Word 1
Address n+8	Header 1	UDB 11	UDB 10	UDB 9	Word 2
Address n+12	UDF	Header 4	Header 3	Header 2	:
Address n+16	Payload 4	Payload 3	Payload 2	Payload 1	:
::	::	::	::	::	:
Address n+60	Payload 48	Payload 47	Payload 46	Payload 45	Word 15

Table 11. 8-Bit UTOPIA Slave in Big-Endian Mode (BEND = 1) With RUDC/XUDC = 11

Queue Bits	31-24	23-16	15-8	7-0	Order
Address n	UDB 1	UDB 2	UDB 3	UDB 4	Word 0
Address n+4	UDB 5	UDB 6	UDB 7	UDB 8	Word 1
Address n+8	UDB 9	UDB 10	UDB 11	Header 1	Word 2
Address n+12	Header 2	Header 3	Header 4	UDF	:
Address n+16	Payload 1	Payload 2	Payload 3	Payload 4	
::	::	::	::	::	:
Address n+60	Payload 45	Payload 46	Payload 47	Payload 48	Word 15

Table 12. 16-Bit UTOPIA Slave in Little-Endian Mode (BEND = 0) With RUDC/XUDC = 2

Queue Bits	31-24	23-16	15-8	7-0	Order
Address n	Header 2	Header 1	UDB 2	UDB 1	Word 0
Address n+4	UDF 2	UDF 1	Header 4	Header 3	Word 1
Address n+8	Payload 4	Payload 3	Payload 2	Payload 1	Word 2
::	::	::	::	::	:
Address n+48	Payload 44	Payload 43	Payload 42	Payload 41	Word 12
Address n+52	Payload 48	Payload 47	Payload 46	Payload 45	Word 13

Table 13. 16-Bit UTOPIA Slave in Little-Endian Mode (BEND = 0) With RUDC/XUDC = 10

Queue Bits	31-24	23-16	15-8	7-0	Order
Address n	UDB 4	UDB 3	UDB 2	UDB 1	Word 0
Address n+4	UDB 8	UDB 7	UDB 6	UDB 5	Word 1
Address n+8	Header 2	Header 1	UDB 10	UDB 9	Word 2
Address n+12	UDF 2	UDF 1	Header 4	Header 3	Word 3
Address n+16	Payload 4	Payload 3	Payload 2	Payload 1	Word 4
::					:
Address n+48	Payload 44	Payload 43	Payload 42	Payload 41	Word 14
Address n+52	Payload 48	Payload 47	Payload 46	Payload 45	Word 15

Table 14. 16-Bit UTOPIA Slave in Big-Endian Mode (BEND = 1) With RUDC/XUDC = 2

Queue Bits	31-24	23-16	15-8	7-0	Order
Address n	UDB 1	UDB 2	Header 1	Header 2	Word 0
Address n+4	Header 3	Header 4	UDF 1	UDF 2	Word 1
Address n+8	Payload 1	Payload 2	Payload 3	Payload 4	Word 2
::	::	::			:
Address n+48	Payload 41	Payload 42	Payload 43	Payload 44	Word 12
Address n+52	Payload 45	Payload 46	Payload 47	Payload 48	Word 13

Table 15. 16-Bit UTOPIA Slave in Big-Endian Mode (BEND = 1) With RUDC/XUDC = 6

Queue Bits	31-24	23-16	15-8	7-0	Order
Address n	UDB 1	UDB 2	UDB 3	UDB 4	Word 0
Address n+4	UDB 5	UDB 6	Header 1	Header 2	Word 1
Address n+8	Header 3	Header 4	UDF 1	UDF 2	Word 2
Address n+12	Payload 1	Payload 2	Payload 3	Payload 4	:
::	::	::	::	::	:
Address n+52	Payload 41	Payload 42	Payload 43	Payload 44	Word 13
Address n+56	Payload 45	Payload 46	Payload 47	Payload 48	Word 14

11 Emulation Support

11.1 Emulation Control for UTOPIA

The UTOPIA port also supports emulation mode. Emulation support helps in system debug with clock cycle accuracy. Emulation modes are achieved with the programmable SOFT and FREE bits in the UTOPIA Power and Emulation Management register (UPWREMU); FREE in UPWREMU[0] and SOFT in UPWREMU[1]. When the CPU is halted during emulation, UTOPIA also halts based on the setting of the SOFT and FREE bits. For details on the UPWREMU registers, see Section 15.1.8.

11.2 Emulation Control for PDMA

Emulation control bits are included in the PDMA global control and status register. There are two bits, FREE and SOFT. FREE=1 defines a free-running mode in which the state of SOFT is ignored. When FREE=0, the state of the SOFT bit is used to determine the response of the PDMA when an emulation halt occurs for the devise. These bits are only applied for memory-to-memory control. The PDMA continues in the free-running mode (the bits are ignored) for all peripheral event groups.

12 Reset Strategy

12.1 UTOPIA Reset

The UTOPIA interface is in reset state during device reset. The UTOPIA interface can also be reset through software by programming the UREN and UXEN bits in the UTOPIA control register (UCR) when the device is out of reset. The peripheral reset through software ensures that the state machine is in a known state and there is no activity in the corresponding section. The software reset does not reset registers to their default values.

The UTOPIA pins have no internal pull-up or pull-down resistors; therefore, all input pins should be pulled externally to bring inputs to a known state. The Reset Value column of Table 16 shows the recommended state for the UTOPIA pins during reset. The address pins are pulled high to give the address of a null PHY/slave (address 11111b) during reset. At reset, all outputs are driven to a high-impedance state to facilitate MPHY operation.

Table 16. UTOPIA Pin Reset Values

UTOPIA Pin	ATM Controller SLAVE (Direction)	Reset Value
UXCLK	In	Low
UXADDR[4:0]	In	High
UXCLAV	Out	High impedance
$\overline{\text{UXENB}}$	In	High
UXSOC	Out	High impedance
UXDATA[15:0]	Out	High impedance
URCLK	In	Low
URADDR[4:0]	In	High
URCLAV	Out	High impedance
$\overline{\text{URENB}}$	In	High
URSOC	In	Low
URDATA[15:0]	In	Low

12.2 PDMA Reset

The PDMA is reset only by a peripheral bus reset. When a reset occurs, any active peripheral bus transactions are terminated and the state machines are reset to their initial or reset state. Register bits are initialized to zero. Memory contents are not affected by reset.

13 UTOPIA Slave Initialization Sequence

A device reset or a programmable reset through the UTOPIA control register (UCR) resets the UTOPIA interface. To initialize the UTOPIA interface for slave operation, the following steps are required:

- The UTOPIA master device in the system provides the clock input to URCLK and UXCLK. The UTOPIA port cannot be initialized without these clocks.
- Ensure that the DSP is out of reset.
- Program the PDMA channel(s) for data transmission and reception to/from the UTOPIA interface.
- Set up the UTOPIA configuration registers, as required.
- Either the DSP or the external ATM master can write the address of this slave/PHY in UCR. The external ATM master can write to UCR through the HPI or PCI interfaces.
- Enable PDMA TX and RX channels. Take the interface out of reset by setting the UREN bit to enable the receive interface and setting the UXEN bit to enable the transmit interface. The transmit and receive interfaces are independent of each other. In typical systems, both are used.

14 ATM Adaptation Layer (AAL) Functions

The UTOPIA interface provides a standard hardware interface between an ATM layer device (master) and a PHY device (slave). The ATM adaptation layer functions such as segmentation and re-assembly (SAR) for AAL2, AAL5 should be implemented in software.

15 Registers

15.1 UTOPIA2 Global Registers

The UTOPIA port is configured through the configuration registers listed in Table 17. The data for transmit and receive queues are accessible through the PDMA controller or CPU at the data port listed in Table 18. For the memory address of these registers, see the *TMS320TCI6486 Communications Infrastructure Digital Signal Processor* data manual (SPRS300) or the *TMS320C6472 Fixed-Point Digital Signal Processor* data manual (SPRS612).

Table 17. UTOPIA Configuration Registers

Offset	Acronym	Register Name	See
0000	UCR	UTOPIA Control Register	Section 15.1.1
0014	CDR	Clock Detect Register	Section 15.1.2
0018	EIER	Error Interrupt Enable Register	Section 15.1.3
001C	EIPR	Error Interrupt Pending Register	Section 15.1.4
0020	RRMR0	Receive Routing Unit MASK and MATCH Register 0	Section 15.1.5
0024	RRMR1	Receive Routing Unit MASK and MATCH Register 1	Section 15.1.5
0028	RRMR2	Receive Routing Unit MASK and MATCH Register 2	Section 15.1.5
002C	RRMR3	Receive Routing Unit MASK and MATCH Register 3	Section 15.1.5
0030	RRMR4	Receive Routing Unit MASK and MATCH Register 4	Section 15.1.5
0034	RRMR5	Receive Routing Unit MASK and MATCH Register 5	Section 15.1.5
0038	RRMR6	Receive Routing Unit MASK and MATCH Register 6	Section 15.1.5
003C	RRMR7	Receive Routing Unit MASK and MATCH Register 7	Section 15.1.5
0040	RRSR	Receive Routing Select Register	Section 15.1.6
0100	UPIDR	UTOPIA Peripheral ID Register	Section 15.1.7
0101	UPWREMU	UTOPIA Power Management and Emulation Register	Section 15.1.8

Table 18. UTOPIA Data Ports

Offset	Acronym	Register Name	See
0000	URQ	UTOPIA Receive Queue	Section 3.4
0004	UXQ	UTOPIA Transmit Queue	Section 3.2

15.1.1 UTOPIA Control Register (UCR)

The UTOPIA interface is configured through the UTOPIA control register (UCR) and contains UTOPIA status and control bits. The UCR is shown in Figure 9 and described in Table 19.

Figure 9. UTOPIA Control Register (UCR)

31	30	29	28	24
BEND	Reserved		SLID	
R/W-0	R-1		R/W-0	

23	22	21	18	17	16
Reserved		XUDC		Reserved	UXEN
R-0		R/W-0		R-0	R/W-0

15	14	13	12	8
Reserved	MPHY	U16M	Reserved	
R-0	R/W-1	R/W-1	R-0	

7	6	5	2	1	0
Reserved		RUDC		Reserved	UREN
R-0		R/W-0		R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19. UTOPIA Control Register (UCR) Field Descriptions

Bit	Field	Value	Description
31	BEND		Big-endian mode enable bit for data transferred by way of the UTOPIA interface.
		0	Data is assembled to conform to little-endian format.
		1	Data is assembled to conform to big-endian format.
30-29	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
28-24	SLID		Slave ID bits. Applicable in multi-PHY mode (MPHY = 1). This 5-bit value is used to identify the UTOPIA in a MPHY setup. Does not apply to single-PHY slave operation (MPHY = 0).
		0	
		1Fh	Null PHY/slave address.
23-22	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
21-18	XUDC		Transmit user-defined cell bits. Valid values are 0 to 11, the remaining values are reserved.
		0	XUDC feature is disabled. The UTOPIA interface transmits a normal ATM cell of 53 bytes.
		1h-Bh	UTOPIA interface transmits the programmed number (1 to 11) of bytes as an extra header. A UDC can have a minimum of 54 bytes (XUDC = 1h) up to a maximum of 64 bytes (XUDC = Bh).
		Ch-Fh	Reserved
17	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
16	UXEN		UTOPIA transmitter enable bit.
		0	UTOPIA port transmitter is disabled and in reset state.
		1	UTOPIA port transmitter is enabled.
15	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
14	MPHY		UTOPIA receive/transmit multi-PHY mode enable bit. MPHY mode is the default.
		0	Single PHY mode is selected for receive and transmit UTOPIA.
		1	Multi-PHY mode is selected for receive and transmit UTOPIA.
13	U16M		Sets default value operation on both the receive and transmit UTOPIA interface.
		0	Default value 8-bit operation is expected.
		1	Default value 16-bit operation is expected.

Table 19. UTOPIA Control Register (UCR) Field Descriptions (continued)

Bit	Field	Value	Description
12-6	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
5-2	RUDC		Receive user-defined cell bits. Valid values are 0 to 11, the remaining values are reserved.
		0	RUDC feature is disabled. The UTOPIA interface expects a normal ATM cell of 53 bytes.
		1h-Bh	UTOPIA interface expects to receive the programmed number (1 to 11) of bytes as an extra header. A UDC can have a minimum of 54 bytes (RUDC = 1h) to a maximum of 64 bytes (RUDC = Bh). Receives with header bytes ranging from 1 to 10 bytes.
		Ch-Fh	Reserved
1	Reserved		Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
0	UREN		UTOPIA receiver enable bit.
		0	UTOPIA port receiver is disabled and in reset state.
		1	UTOPIA port receiver is enabled.

15.1.2 Clock Detect Register (CDR)

The clock detect register (CDR) and the UTOPIA clock detection feature allow the DSP to detect the presence of the URCLK and/or UXCLK. The CDR is shown in Figure 10 and described in Table 20.

If a URCLK or a UXCLK edge is not detected within the respective time period specified in CDR, an error bit, RCFP or XCFP, respectively, is set in the error interrupt pending register (EIPR). In addition, the RCPP and XCPP bits in EIPR indicate the presence of the URCLK and UXCLK, respectively. For usage of these interrupts to the CPU, see Section 9.

Figure 10. Clock Detect Register (CDR)

31 – 24	23 – 16
Reserved	XCCNT
R-0	R/W-FFh

15 – 8	7 – 0
Reserved	RCCNT
R-0	R/W-FFh

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 20. Clock Detect Register (CDR) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
23-16	XCCNT		Transmit clock count bits specify the number of peripheral clock cycles for which the external UTOPIA transmit clock (UXCLK) must have a low-to-high transition to avoid a reset of the transmit interface. If a UXCLK clock edge is undetected within XCCNT peripheral clock cycles, the transmit UTOPIA port is reset by hardware. The XCF error bit (XCFP) in the error interrupt pending register (EIPR) is set.
		0	Transmit clock detect feature is disabled.
		1h-FFh	Transmit clock detect feature is enabled. This 8-bit value is the number of peripheral clock cycles before the next UTOPIA clock edge (UXCLK) must be present. There must be 255 peripheral clock cycles before the next UTOPIA clock edge (UXCLK).
15-8	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
7-0	RCCNT		Receive clock count bits specify the number of peripheral clock cycles for which the external UTOPIA receive clock must have a low-to-high transition to avoid a reset of the receive interface. If a URCLK clock edge is undetected within RCCNT peripheral clock cycles, the receive UTOPIA port is reset by hardware. The RCF error bit (RCFP) in the error interrupt pending register (EIPR) is set.
		0	Receive clock detect feature is disabled.
		1h-FFh	Receive clock detect feature is enabled. This 8-bit value is the number of peripheral clock cycles before the next UTOPIA clock edge (URCLK) must be present. There must be 255 peripheral clock cycles before the next UTOPIA clock edge (URCLK).

15.1.3 Error Interrupt Enable Register (EIER)

If an error condition is set in the error interrupt enable register (EIER) and the corresponding error is set in the error interrupt pending register (EIPR), an interrupt is generated to the CPU. The EIER is shown in Figure 11 and described in Table 21.

Figure 11. Error Interrupt Enable Register (EIER)

31–19	18	17	16
Reserved	XCPE	XCFE	XQSE
R-0	R/W-0	R/W-0	R/W-0

15–3	2	1	0
Reserved	RCPE	RCFE	RQSE
R-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21. Error Interrupt Enable Register (EIER) Field Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
18	XCPE		Transmit clock present interrupt enable bit
		0	Transmit clock present interrupt is disabled.
		1	Transmit clock present interrupt is enabled.
17	XCFE		Transmit clock failed interrupt enable bit
		0	Transmit clock failed interrupt is disabled.
		1	Transmit clock failed interrupt is enabled.
16	XQSE		Transmit queue stall interrupt enable bit
		0	Transmit queue stall interrupt is disabled.
		1	Transmit queue stall interrupt is enabled.
15-3	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
2	RCPE		Receive clock present interrupt enable bit
		0	Receive clock present interrupt is disabled.
		1	Receive clock present interrupt is enabled.
1	RCFE		Receive clock failed interrupt enable bit
		0	Receive clock failed interrupt is disabled.
		1	Receive clock failed interrupt is enabled.
0	RQSE		Receive queue stall interrupt enable bit
		0	Receive queue stall interrupt is disabled.
		1	Receive queue stall interrupt is enabled.

15.1.4 Error Interrupt Pending Register (EIPR)

The UTOPIA error conditions are recorded in the error interrupt pending register (EIPR). The EIPR is shown in Figure 12 and described in Table 22. A write of 1 to the XCPP, XCFP, RCPP, or RCFP bit clears the corresponding bit. A write of 0 has no effect. The XQSP and RQSP bits are read-only bits and are cleared automatically by the UTOPIA interface once the error conditions cease. The error conditions in EIPR can generate an interrupt to the CPU, if the corresponding bits are set in the error interrupt enable register (EIER).

The EIPR is shown in Figure 12 and described in Table 22.

Figure 12. Error Interrupt Pending Register (EIPR)

31 — 19	18	17	16
Reserved	XCPP	XCFP	XQSP
R-0	R/W-0	R/W-0	R-0

15 — 3	2	1	0
Reserved	RCPP	RCFP	RQSP
R-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22. Error Interrupt Pending Register (EIPR)

Bit	Field	Value	Description
31-19	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
18	XCPP		Transmit clock present interrupt pending bit indicates if the UTOPIA transmit clock (UXCLK) is present. XCPP is valid regardless if the transmit interface is enabled or disabled.
		0	UXCLK is not present.
		1	UXCLK is present. If the corresponding bit in EIER is set, an interrupt UINT is sent to the CPU.
17	XCFP		Transmit clock failed interrupt pending bit is activated only when the UTOPIA transmit interface is enabled (UXEN in UCR = 1).
		0	UXCLK is present.
		1	UXCLK failed. No UXCLK is detected for a period longer than that specified in the XCCNT field of CDR. If the corresponding bit in EIER is set, an interrupt UINT is sent to the CPU.
16	XQSP		Transmit queue stall interrupt pending bit
		0	No transmit queue stall condition
		1	Transmit queue stalled, a write is performed to a full transmit queue. The write is stalled until the queue is drained and space is available. Data is not overwritten. XQSP is cleared once the queue has space available and writes can continue.
15-3	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
2	RCPP		Receive clock present interrupt pending bit indicates if the UTOPIA receive clock (URCLK) is present. RCPP is valid regardless if the receive interface is enabled or disabled.
		0	URCLK is not present.
		1	URCLK is present. If the corresponding bit in EIER is set, an interrupt UINT is sent to the CPU.
1	RCFP		Receive clock failed interrupt pending bit is activated only when the UTOPIA receive interface is enabled (UREN in UCR = 1).
		0	URCLK is present.
		1	URCLK failed. No URCLK is detected for a period longer than that specified in the RCCNT field of CDR. If the corresponding bit in EIER is set, an interrupt UINT is sent to the CPU.
0	RQSP		Receive queue stall interrupt pending bit
		0	No receive queue stall condition
		1	Receive queue stalled, a read is performed from an empty receive queue. The read is stalled until valid data is available in the queue. RQSP is cleared as soon as valid data is available and the read is performed.

15.1.5 Receive Routing Unit Mask and Match Registers (RMMR0-5)

The field of interest is always extracted as a 16-bit value, you have the option of choosing the required number of bits and mask value by programming the RMMR0-5 registers. ATM cells with 16-bit routing field values that match with one of the six mask-and-match values are serviced by one of the six PDMA events.

The RMMR(0-5) is shown in Figure 13 and described in Table 23.

Figure 13. Receive Routing Unit MASK and MATCH Registers (RMMR0-5)

31	16
MATCH	
R/W-0	

15	0
MASK	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23. Receive Routing Unit MASK and MATCH Registers (RMMR0-5) Field Descriptions

Bit	Field	Value	Description
31-16	MATCH	0-FFFFh	16-bit match value
15-0	MASK	0-FFFFh	16-bit mask value

15.1.6 Receive Routing Select Register (RRSR)

ATM cells are routed based on byte and bit offsets that you define within the 53-byte (standard) to 64-byte (user-defined) ATM cell. The byte (RBYT) and bit offsets (RBIT) are used to select the 16-bit routing value within the ATM cell header.

The RRSR is shown in Figure 14 and described in Table 24.

Figure 14. Receive Routing Select Register (RRSR)

31	22	16
Reserved	RBYT	
R-0	R/W-0	

15	3	2	0
Reserved		RBIT	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24. Receive Routing Select Register (RRSR) Field Descriptions

Bit	Field	Value	Description
31-22	Reserved		Reserved
21-16	RYBT	0-3Fh	Indicates the least significant byte of the 16-bit field of interest.
15-3	Reserved		Reserved
2-0	RBIT	0-7h	Indicates the least significant bits within the 16-bit field of interest that is used for routing decisions.

15.1.7 UTOPIA Peripheral ID Register (UPIDR)

The UTOPIA peripheral ID (UPIDR) register identifies the peripheral for revision number, class, and type. The UPIDR register is shown in Figure 15 and described in Table 25.

Figure 15. UTOPIA Peripheral ID Register (UPIDR)

31 – 24	23 – 16
Reserved	TYPE
R-0	R/W-3

15 – 8	7 – 0
CLASS	REV
R/W-3	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25. UTOPIA Peripheral ID Register (UPIDR) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved		Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatability.
23-16	TYPE	03	Peripheral type. Specifies the actual peripheral type within a peripheral class. For UTOPIA, the CLASS field defines the UTOPIA Level 2 interface.
5-8	CLASS	03	Peripheral class. Indicates if the peripheral is a Host interface, DMA controller, or a Network Interface, etc. UTOPIA belongs to the Class_ Network Interface that always carries a value of 3.
7-0	REV	01	Peripheral revision. Identifies the revision level of the specific instance of the UTOPIA peripheral. Begins with a value 0x1 and increments each time there is a re-design.

15.1.8 UTOPIA Power Management and Emulation Register (UPWREMU)

The UTOPIA power management and emulation (UPWREMU) register is shown in Figure 16 and described in Table 26.

Figure 16. UTOPIA Power Management and Emulation Register (UPWREMU)

31 — 16
Reserved
R-0

15 — 2	1	0
Reserved	SOFT	FREE
R-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 26. UTOPIA Power Management and Emulation Register (UPWREMU) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved		Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
1	SOFT		Controls the behavior of UTOPIA during CPU emulation halt when FREE = 0. No effect when FREE=1.
		0	Undefined
		1	In slave mode: all on-going transfers on the pins should complete. If an active cell is output before this emulation halt, the request is serviced before UTOPIA enters the suspend state. Any on-going CPU or DMA read/write services to UTOPIA complete. No new read/write events to CPU or DMA are generated. In master mode: Current data transfer on the pins complete and pending read/write requests to or from CPU/DMA complete before emulation halt. The master mode uses the TX and RX clocks as inputs. Therefore, clocks continue to run. The master does not initiate new transfers with the PHYs in the system. That is, Enable lines are not driven active even if active CLAVs return. Since the clocks do not stop, polling continues but transfers are not enabled.
0	FREE		The FREE bit in conjunction with the SOFT bit determines emulation operation.
		0	SOFT bit controls emulation during emulation halt.
		1	Emulation has no effect on the UTOPIA port operation. Peripheral operates normally.

15.2 PDMA Registers

The PDMA global registers are listed in Table 17.

The PDMA register file includes four proxy registers (listed in Table 28). The CPU programs the proxy registers, along with the channel number in the channel context pointer register. The PDMA transfers the contents of the proxy registers to the appropriate location in PDMA context RAM. The fields of the proxy registers are described in Section 15.2.6.

Table 27. PDMA Global Registers

Offset	Acronym	Register Name	See
0000	PIR	Peripheral Identification Register	Section 15.2.1
0004	GCSR	Global Control Status Register	Section 15.2.2
0020	SR0	Statistics Register 0	Section 15.2.3
0024	SR1	Statistics Register 1	Section 15.2.3
0060	PCR0	Peripheral Control Register 0	Section 15.2.4
0064	PCR1	Peripheral Control Register 1	Section 15.2.4
0100-010F	TXC0	TX Channel Proxy Register 0	
0110-011F	TXC1	TX Channel Proxy Register 1	
0120-012F	TXC2	TX Channel Proxy Register 2	
0130-013F	TXC3	TX Channel Proxy Register 3	
0140-014F	TXC4	TX Channel Proxy Register 4	
0150-015F	TXC5	TX Channel Proxy Register 5	
0180-018F	RXC0	RX Channel Proxy Register 0	
0190-019F	RXC1	RX Channel Proxy Register 1	
01A0-01AF	RXC2	RX Channel Proxy Register 2	
01B0-01BF	RXC3	RX Channel Proxy Register 3	
01C0-01CF	RXC4	RX Channel Proxy Register 4	
01D0-01DF	RXC5	RX Channel Proxy Register 5	
01E0-01EF	RXC6	RX Channel Proxy Register 6	
01F0-01FF	RXC7	RX Channel Proxy Register 7	

Table 28. PDMA Channel Proxy Registers

Offset	Acronym	Register Name	See
0000	SAR	PDMA Switch Address Register	Section 15.2.6.1
0004	PAR	PDMA Peripheral Address Register	Section 15.2.6.2
0008	BSR	PDMA Buffer Size Register	Section 15.2.6.3
000C	TCR	PDMA Transfer Control Register	Section 15.2.6.4

15.2.1 PDMA Peripheral Identification Register (PID)

The peripheral identification register (PID) is a constant register that contains the ID and ID revision number for that module. The PID stores version information used to identify the module. All bits within this register are read-only (writes have no effect) meaning that the values within this register should be hard-coded with the appropriate values and must not change from their hard-coded values.

The PID register is shown in Figure 17 and described in Table 29.

Figure 17. Peripheral Identification Register (PID)

31 24	23 16
Reserved	TID
R	R-0000 0001

15 8	7 0
CID	PREV
R-0000 0100	R

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 29. Peripheral Identification Register (PID) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved		Reserved
23-16	TID		Peripheral Type ID: 0x01 PDMA Module
15-8	CID		Peripheral Class ID: 0x04 DMA Controller
7-0	PREV		Peripheral Revision Number: starts with 0x01

15.2.2 PDMA Global Control and Status Register (GCSR)

The global control and status register is used enable the operation of a PDMA and for emulation control for PDMA (see Section 11.2)

The GCSR register is shown in Figure 18 and described in Table 30.

Figure 18. PDMA Global Control and Status Register (GCSR)

31	30 16
STRT	Reserved
R/W-0	R-0

15 2	1	0
Reserved	SOFT	FREE
R-0	R/W	R/W

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 30. PDMA Global Control and Status Register (GCSR) Field Descriptions

Bit	Field	Value	Description
31	STRT		Start Control. The PDMA is halted (or disabled) by device reset.
		0	PDMA is halted.
		1	PDMA is enabled.
30-2	Reserved		Reserved
1	SOFT[1]		Soft emulation halt control
		0	Reserved.
		1	Halt PDMA transfers at the end of the burst in progress or immediately if no burst is in progress.
0	FREE[1]		Free running mode control
		0	Free running mode is disabled. PDMA operation during emulation halt is defined by the SOFT bit.
		1	Free running mode is enabled. The PDMA continues all transfers during emulation halt.

(1) Applies only for memory-to-memory transfers. SOFT and FREE are ignored for all PE groups used to support peripherals.

15.2.3 PDMA Statistics Registers (SR0-1)

The PDMA maintains statistics registers that may be useful in some applications. There is one pair of 16-bit statistics registers per event group. Figure 19 shows a pair of 16-bit statistics registers. The lower 16 bits provide an error count. The error count is used to count the number of times a peripheral sent a synchronization event for a channel that was not enabled. The upper 16 bits provide a stall count. The stall count is used to count the number of times a peripheral sent a synchronization event for a channel in which the channel buffer data or space status (depending on direction) resulted in the current event request not being serviced immediately. There is one pair of 16-bit statistics registers per peripheral requester.

The SR0-5 registers are shown in Figure 19 and described in Table 31.

Figure 19. PDMA Statistics Registers (SR0-1)

31 ... 16
WAIT
R

15 ... 0
ERROR
R

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 31. PDMA Statistics Registers (SR0-1) Field Descriptions

Bit	Field	Value	Description
31-16	WAIT		Channel stall (wait) count. The counter is incremented each time a PDMA stall occurs. Asserting the CLRS bit in the peripheral control register clears the counter. Except when PE=11, a stall occurs when the PDMA is unable to fill a transmit channel buffer or empty a receive channel buffer in time to service the next peripheral event and the PDMA operation requires the PDMA to wait for available data or space. There are no conditions for PE=11 that are logged as stalls.
15-0	ERROR		Channel not enabled (error) count. The counter is incremented each time a PDMA error occurs. Asserting the CLRS bit in the peripheral control register clears the counter. Except for PE=11, an error occurs when the PDMA is given an event ID for a disabled channel. For PE=11, an error occurs when the transmit channel buffer is empty or the receive channel buffer is full for the corresponding peripheral event.

15.2.4 PDMA Peripheral Control Registers (PCR0-1)

The PDMA provides peripheral control registers to supplement the information that is maintained as part of the channel context and to manage the operation of the PDMA. There is one peripheral control register for each event group.

The PCR0-5 registers are shown in Figure 20 and described in Table 32.

Figure 20. PDMA Peripheral Control Registers (PCR0-1)

31-27	26	25	24	23-22	21	20	19-18	17	16
Reserved	BEND	CLRS	Rsvd	PE	RSV	SYNC	PRI	ABU	DIR
R-0	R/W	R/W	R-0	R/W	R/W	R/W	R/W	R/W	R/W

15-12	11	10-8	7-4	3	2-0
PBLEN	PMOD	Reserved	SBLEN	SMOD	Reserved
R/W	R/W	R-0	R/W	R/W	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 32. PDMA Peripheral Control Registers (PCR0-1) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved		Reserved. Write as 0; read as 0.
26	BEND		Endian mode selection.
		0	Little-endian mode; no endian conversion
		1	Big-endian mode. Endian conversion is based on the switch and peripheral element sizes as defined in the channel context. Data is always right justified on the bus.

PSIZ	SSIZ	Conversion
8	8	None
8	16	Swap bytes
8	32	Swap bytes
16	8	Swap bytes
16	16	None
16	32	Swap 16-bit words
32	8	Sway bytes
32	16	Swap 16-bit words
32	32	None

Bit	Field	Value	Description
25	CLRS		Clear statistics register
		0	No effect. Always read as 0.
		1	Clears the statistics register
24	Reserved		Reserved. Write as 0; read as 0.
23-22	PE		Peripheral event control. Determines how the event group of PDMA channels responds to events.
		00	Always use channel 0 of the event group.
		01	Arbitrate among the PDMA channels of the event group on when the transfer count of the current channel becomes zero. Use the current PDMA channel on transfer events.
		10	The event ID is used to select the PDMA channel. The PDMA only responds if the PDMA channel is active.
		11	The event ID is used to select the PDMA channel. The PDMA always responds, even if the PDMA channel is inactive.
21	RSV		Reserved. Write as 0; read as 0.
20	SYNC		Peripheral synchronization start
		0	No synchronization
		1	Synchronized transfers

Table 32. PDMA Peripheral Control Registers (PCR0-1) Field Descriptions (continued)

Bit	Field	Value	Description
19-18	PRI		Priority
		00b	Lowest priority
		11b	Highest priority
17	ABU		Auto-buffering mode is used with the PDMA with most peripherals. If auto-buffering is enabled an interrupt is generated at the half-buffer and end-of-buffer points of the switch buffer, provided IE is set. At the end of buffer point, the switch and peripheral addresses and the transfer count are restored.
		0	Auto-buffering disabled
		1	Auto-buffering enabled
16	DIR		Direction of transfer
		0	Switch to peripheral
		1	Peripheral to switch
15-12	PBLEN	1-16	Peripheral burst length. Specifies the burst length value to the peripheral bus interface for the number of elements to be transferred in one burst. The 4-bit value specifies the burst length from 1-16 in a -1 format.
11	PMOD		Peripheral address modification
		0	Increment
		1	Constant
10-8	Reserved		Reserved. Write as 0; read as 0.
7-4	SBLEN	1-16	Switch burst length. Specifies the burst length value to the peripheral bus interface for the number of elements to be transferred in one burst. The 4-bit value specifies the burst length from 1-16 in a -1 format.
3	SMOD		Switch address modification
		0	Increment
		1	Constant
2-0	Reserved		Reserved. Write as 0; read as 0.

15.2.5 Proxy Registers

The PDMA register file includes four proxy registers. The CPU programs the proxy registers, along with the channel number in the channel context pointer register. The PDMA transfers the contents of the proxy registers to the appropriate location in PDMA context RAM. The fields of the proxy registers are described in Section 15.2.6.

Note that PDMA proxy registers are programmed through the PDMA Interface Manager (PIM). For details of the PIM, see Section 5.

15.2.6 PDMA Context Registers

Each channel within the PDMA module has a set of context registers as shown in Table 33. There is one set of context registers for each channel of the PDMA module. Each set of context registers is mapped to a 16-byte address boundary.

Table 33. PDMA Context Registers

Channel Offset Address	Description
0x0	Switch address
0x4	Peripheral address
0x8	Transfer count/buffer size
0xC	Transfer control

An 16-channel PDMA example context register memory map for a PDMA is shown in Table 34.

Table 34. PDMA Context Memory Map

PDMA Relative Address	Description
0x0000	PDMA channel 0
0x0010	PDMA channel 1
0x0020	PDMA channel 2
0x0030	PDMA channel 3
0x0040	PDMA channel 4
0x0050	PDMA channel 5
0x0060	PDMA channel 6
0x0070	PDMA channel 7
0x0080	PDMA channel 8
0x0090	PDMA channel 9
0x00A0	PDMA channel 10
0x00B0	PDMA channel 11
0x00C0	PDMA channel 12
0x00D0	PDMA channel 13
0x00E0	PDMA channel 14
0x00F0	PDMA channel 15

Each data transfer specified by the PDMA context is defined by switch and peripheral addresses, transfer counts, and transfer control. Section 15.2.6.1 through Section 15.2.6.4 show each of these registers.

15.2.6.1 PDMA Switch Address Register (SAR)

The switch address register specifies the switch address to/from the transfers that are to be started.

The SAR register is shown in Figure 21 and described in Table 35.

Figure 21. PDMA Switch Address Register (SAR)

31	16
SADDR	
R/W	

15	0
SADDR	
R/W	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 35. PDMA Switch Address Register (SAR) Field Descriptions

Bit	Field	Value	Description
31-0	SADDR		Switch address bits

15.2.6.2 PDMA Peripheral Address Register (PAR)

The peripheral address register specifies the peripheral address to/from the transfers to be started.

The PAR register is shown in Figure 22 and described in Table 36.

Figure 22. PDMA Peripheral Address Register (PAR)

31	16
PADDR	
R/W	

15	0
PADDR	
R/W	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 36. PDMA Peripheral Address Register (PAR) Field Descriptions

Bit	Field	Value	Description
31-0	PADDR		Peripheral address bits

15.2.6.3 PDMA Buffer Size Register (BSR)

The PDMA buffer size register specifies switch and peripheral burst length in ABU mode or total byte count to be transferred to/from the switch interface in block mode.

The BSR register in ABU mode is shown in Figure 23 and described in Table 37.

Figure 23. PDMA Buffer Size Register (BSR in ABU Mode)

31 – 16
PBUF
R/W

15 – 0
SBUF
R/W

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 37. PDMA Buffer Size Register (BSR in ABU Mode) Field Descriptions

Bit	Field	Value	Description
31-16			Peripheral buffer size. Represents the size of the circular buffer in peripheral memory. Buffer size values are limited to multiples of four bytes.
15-0			Switch buffer size. Represents the size of the circular memory buffer in DSP local memory. See Section 4.3.1 for an explanation of buffer requirements. Buffer size values are limited to multiples of four bytes.

The BSR register in block mode is shown in Figure 24 and described in Table 38.

Figure 24. PDMA Transfer Count Register (BSR in Block Mode)

31 – 27	26 – 16
Reserved	COUNT
R	R/W

15 – 0
COUNT
R/W

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 38. PDMA Transfer Count Register (BSR in Block Mode) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved		Reserved. Write as 0; read as 0.
26-0	COUNT		Byte count. Represents the total number of bytes that are to be transferred to/form the switch interface. Byte count values are limited to multiples of the switch element size, as shown, where N is the number of elements. **Element Size** / **Byte Count**: 8 / N; 16 / 2*N; 32 / 4*N

15.2.6.4 PDMA Transfer Control Register (TCR)

The PDMA transfer control register is used for the selection of interrupt signals that are used to interrupt selected CPUs in a multi-CPU device.

The TCR register is shown in Figure 25 and described in Table 39.

Figure 25. PDMA Transfer Control Register (TCR)

31	30 29	28	27	26 24	23 22	21 20	19 18	17	16
STRT	Reserved	IMOD	IE	SINT	PSIZ	SSIZ	Reserved	FF	BCZ
R/W	R	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W

15	14 8	7	6 0
Reserved	WRPTR	Reserved	RDPTR
R	R/W	R	R/W

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 39. PDMA Transfer Control Register (TCR) Field Descriptions

Bit	Field	Value	Description
31	STRT		Start to enable and disable transfers with the channel context.
		0	Disable
		1	Enable
30-29	Reserved		Reserved. Write as 0; read as 0.
28	IMOD		Interrupt mode defines when an enable interrupt is generated.
			ABU 0 mode:
		0	At the end of the block
		1	Reserved
			ABU 1 Mode:
		0	At the end of the buffer only
		1	At the half-buffer and the end of the buffer
27	IE		Interrupt enable signifies the DMA should send an interrupt to the requesting module when the transfer is complete.
26-24	SINT		Subsystem interrupt elects one of eight interrupt lines from the event group set that is used to interrupt a DSP CPU when an interrupt is generated. This selection is used when interrupts are generated as a result of switch port transfers. The interrupt selection must correspond to a DSP CPU that is the source/destination of the transfer.
23-22	PSIZ		Peripheral element size
		00	8 bits
		01	16 bits
		10	32 bits
		11	Reserved
21-20	SSIZ		Switch element size
		00	8 bits
		01	16 bits
		10	32 bits
		11	Reserved
19-18	Reserved		Reserved. Write as 0; read as 0.
17	FF[1]	0	
		1	
16	BCZ[1]		Block count equal zero. Block mode only. Initialize to 0.
15	Reserved		Reserved. Write as 0; read as 0.
14-8	WRPTR[1]		Write pointer. Current channel buffer write offset. Initialize to 0.
7	Reserved		Reserved. Write as 0; read as 0.
6-0	RDPTR[1]		Read pointer. Current channel buffer read offset. Initialize to 0.

[1] This field is modified by the PDMA state machine. It should be initialized to 0 when the channel is enabled.

Appendix A Revision History

This revision history highlights the technical changes made to the document in this revision.

Table 40. TCI6486/C6472 UTOPIA2 Revision History

See	Additions/Modifications/Deletions
Global	Added C6472 device