NETWORK ACCESS PRODUCTS

Bt8230

ATM Segmentation and Reassembly Controller–SRC

Rockwell Semiconductor Systems

Advance Information

This document contains information on a product under development. The parametric information contains target parameters that are subject to change.

Bt8230

ATM Segmentation and Reassembly Controller—SRC

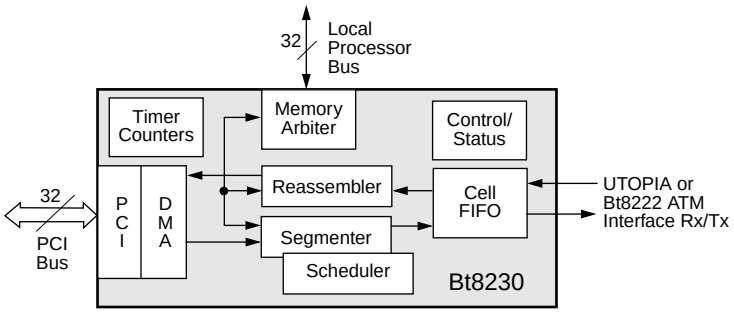
The Bt8230 Segmentation and Reassembly Controller (SRC) incorporates a host interface, Asynchronous Transfer Mode (ATM) Adaptation Layer processing, and line interfaces in a single device. The feature set includes a Peripheral Component Interconnect (PCI) bus interface, segmentation and reassembly controllers, a local memory controller, a Direct Memory Access (DMA) coprocessor, and an automatic scheduling algorithm. Each segmentation channel manages an independent bit rate of up to 200 Mbps per channel for simplex connections. This feature set makes the Bt8230 ideal for file server ATM adapters, routers/hubs, and other Wide Area Network (WAN) applications. The PCI bus interface makes the SRC suitable for workstation Network Interface Cards (NICs) as well.

The Bt8230 supports 16,000 open connections simultaneously with robust Operation and Maintenance (OAM), signaling, and Interim Local Management Interface (ILMI) features. Other key features include support for random Virtual Path Indicator/ Virtual Channel Indicator (VPI/VCI) assignment, interleaved AAL5 and AAL3/4 Segmentation and Reassembly (SAR), and termination of signaling and ILMI into local memory to maintain management connections independently. The Bt8230 fulfills all the requirements of the ATM Forum User-Network Interface (UNI) 3.1 standards and the related American National Standards Institute (ANSI) and ITU standards. The device is pin-compatible with the Bt8233 SRC, which supports new traffic management algorithms for Available Bit Rate (ABR) service as defined in TM 4.0.

The Bt8230 architecture provides several implementation options, allowing users to balance the cost and functions of their systems. In standalone mode, the Bt8230 is capable of full line-rate performance, presenting a low-cost option ideal for workstation ATM NICs. Combined with the Bt8222 ATM Receiver/Transmitter, the Bt8230 delivers an ATM solution capable of full-duplex operation at 155.52 Mbps. This is possible without the cost constraints of using a large amount of local memory to buffer incoming ATM cells. When using the local processor, host-driver software needs decrease, and host performance is improved since more control functions are handled locally. In slave Universal Test and Operation Physical Interface (UTOPIA) mode, the Bt8230 can be connected directly to an ATM backplane for applications exceeding 155 Mbps.

The Bt8230EVS evaluation system provides a working reference design and an example software driver. It also has facilities for generating and terminating ATM traffic.

Functional Block Diagram



Distinguishing Features

- Simultaneous ATM and Switched Multimegabit Digital Service (SMDS) SAR
- ATM Forum UNI 3.1–compliant
- Pin-compatible with TM 4.0 options
- AAL0, AAL3/4, and AAL5 SAR
- Formats AAL3/4 and AAL5 CPCS fields and performs all checks
- Performance monitoring per GR-1248 and ITU I.610, dated 3/93
- Supports 16,384 active Virtual Circuit Channels (VCCs)
- Robust per-VCC statistics supports Simple Network Management Protocol (SNMP) Management Information Base (MIB) requirements
- Internal MIB counters
- 155 Mbps full-duplex throughput
- UTOPIA master, UTOPIA slave, or Bt8222-compatible ATM Physical Layer (PHY) interface
- PCI host interface (master and slave mode), Revision 2.0
- Automatic scheduling algorithm operates individually on each channel
- Host or local segmentation and reassembly
- Variable Bit Rate (VBR) and Unspecified Bit Rate (UBR) traffic types
- Scatter/gather DMA to host memory
- Optional local processor for signaling, OAM, and ILMI management functions
- Zero- or one-wait-state local memory
- Boundary scan to facilitate board-level testing
- Low-power Complementary Metal-Oxide Semiconductor (CMOS) process in a 208-pin Plastic Quad Flat Rate (PQFP)
- Complete working reference design, software, and documentation package available

Applications

- ATM interface for routers and hubs
- ATM/PCI interface cards
- Test and WAN equipment
- Service access multiplexers

Print date: October 1997

Table of Contents

List of Figures

List of Tables

1.0 Product Description

The Bt8230 Segmentation and Reassembly Controller (SRC) incorporates a host interface, Asynchronous Transfer Mode (ATM) Adaptation Layer processing, and line interfaces in a single device. This highly integrated ATM device combines several features such as a PCI bus interface, segmentation and reassembly controllers, a local memory controller, a Direct Memory Access (DMA) coprocessor, and an automatic scheduling algorithm. The Bt8230 supports all requirements contained in ATM Forum User Network Interface Specification (UNI) 3.1 and the related American National Standards Institute (ANSI) and International Telecommunications Union (ITU) standards. The Bt8230's architecture enables it to efficiently handle high bandwidth throughput across the spectrum of two different ATM Adaptation Layers (AALs) and two ATM service categories. Once initialized and given a segmentation or reassembly task, the Bt8230 operates autonomously. This SRC combines a Peripheral Component Interconnect (PCI) bus interface, segmentation and reassembly controllers, a local memory controller, a DMA coprocessor, and a proprietary scheduling algorithm that enables each segmentation channel to achieve an independent bit rate of up to 200 Mbps per channel for simplex connections.

Today's terminal adapters, routers/hubs, and other Wide Area Network (WAN) applications require the features offered by the Bt8230. The device supports thousands of connections simultaneously with robust Operation and Maintenance (OAM), signaling, and Interim Local Management Interface (ILMI) features. Other key features include support for random Virtual Path Identifier/ Virtual Channel Identifier (VPI/VCI) assignment, interleaved AAL5 and AAL3/4 support, and termination of signaling and ILMI into local memory to maintain management connections independent of the host.

The Bt8230 incorporates an easy-to-use traffic priority management scheme that allows each connection to be individually monitored. Both segmentation and reassembly processes deal with host buffers as pipelines of information. Partial packets may be added to the transmit stream or released as they are received for additional processing by the host. The host buffer structures resemble those found in common operating systems This means that the data can be processed by the host without having to physically copy the information to another memory location. A send–immediate mode supports low-latency connections. The Bt8230 transmit-scheduling algorithm supports the currently defined "leaky buckets."

The optional local processor architecture provides the flexibility for system designers and programmers to track the rapidly changing ATM standards. OAM, ILMI, and signaling can all be ported to this processor to off-load the host system. At the same time, a processor-less system is a cost-effective option with the Bt8230 architecture. Therefore, this architecture is especially suited to ATM network interfaces for file servers, routers/hubs, Digital Service Units (DSUs), and

WAN Data Terminal Equipment (DTE), as well as cost-sensitive PCI ATM cards. Combined with the Bt8222 ATM Receiver/Transmitter, the Bt8230 provides an ATM solution capable of full-duplex operations at bit rates up to 155.52 Mbps ATM speed. This is accomplished without the cost constraints of extensive local memory required to buffer incoming ATM cells.

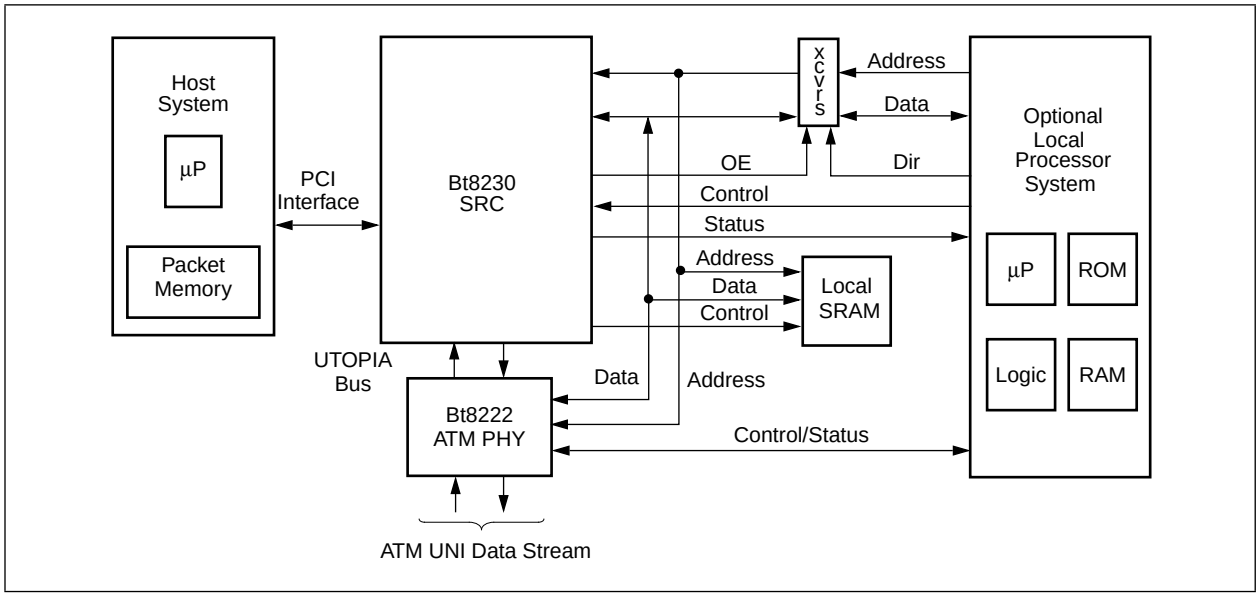
The Bt8230 is implemented as a 33 MHz CMOS device in a 208-pin Plastic Quad Flat Pack (PQFP) with direct Transistor-Transistor Logic (TTL)-level interfaces to the PCI bus. The device is designed to operate with TTL level inputs and Complementary Metal-Oxide Semiconductor (CMOS) level outputs. The Bt8230 allows incoming ATM cells to be transferred directly into the main system Central Processing Unit (CPU) without local buffering. The Bt8230 architecture uniquely balances the strengths of the PCI bus with the need to achieve sustained 155.52 Mbps ATM speeds.

With the Bt8230 and Bt8222, a highly integrated and economical PCI ATM adapter can be built. The interface card could, therefore, consist of the following:

- A Bt8230
- An ATM link interface or framer chip (Bt8222)
- An optional external control processor
- A suitable physical transceiver device capable of driving optical fibers or twisted pair cable
- Interface connectors
- Static Random Access Memory (SRAM) for control memory

The SRAM size is driven primarily by the number of connections to be supported. Approximately 250 kbytes are needed per 1000 connections. Figure 1-1 shows the Bt8230 connected to an ATM physical device and an optional local processor.

Figure 1-1. Bt8230 System Block Diagram with Optional Local Processor

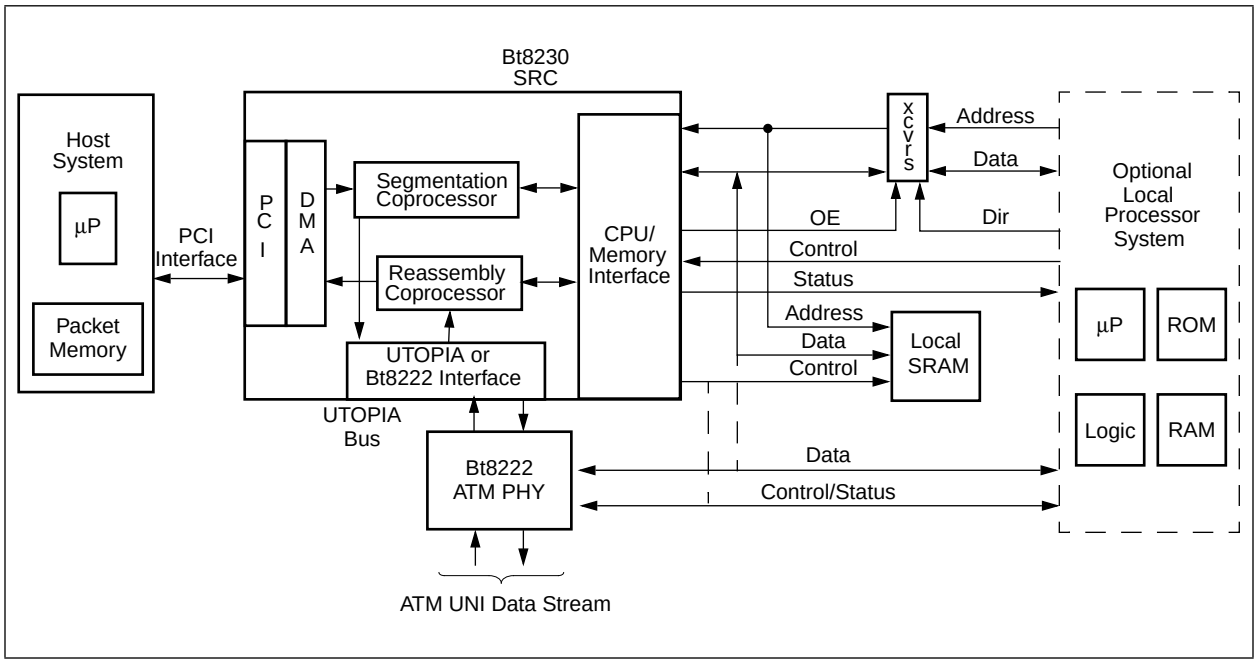


1.1 Components of the Bt8230

As shown in Figure 1-2, several components combine to make the Bt8230 a highly efficient ATM processor:

- DMA coprocessor the segmentation coprocessor, the reassembly coprocessor, the PCI bus interface, the local memory interface, the local processor interface and the ATM physical interface.
- Segmentation coprocessor
- Reassembly coprocessor
- Local processor interface
- ATM physical interface

Figure 1-2. Bt8230 Functional Blocks



1.1.1 DMA Coprocessor

The Bt8230 incorporates an on-chip DMA coprocessor that works in close conjunction with the segmentation and reassembly coprocessors. Once a transfer has been initiated by either the reassembly or segmentation coprocessor, the DMA coprocessor is responsible for gaining access to the PCI bus, transferring the requested data, and notifying the segmentation or reassembly coprocessor that the transfer is complete. The DMA coprocessor transfers all data using the read and write burst buffers in the PCI bus interface, while the PCI bus master logic executes the PCI bus burst protocol.

1.1.2 Segmentation Coprocessor

The segmentation coprocessor transmits 52-octet data to the ATM physical interface block. This coprocessor segments large (up to 64 kbyte) ATM AAL Service Data Units (SDUs) located in host or local Random Access Memory (RAM) into a controlled stream of 44- or 48-byte ATM cell payloads, which are then formatted into 52-byte ATM cells, not including the Header Error Control (HEC) byte. (A blank HEC byte is added by the ATM physical interface block.) This task is performed with the help of state tables in local RAM memory. Rate entries for each segmentation connection allow the segmentation coprocessor to automatically multiplex cells from a number of outgoing channels onto a single outgoing link while preserving the bandwidth relations between the channels. The segmentation coprocessor can also supply the Common Part Conversion Sublayer (CPCS)-Protocol Data Unit (PDU) header, trailer, and pad fields as part of the segmentation process.

1.1.3 Reassembly Coprocessor

The ATM physical block strips the HEC byte before passing the cell to the reassembly coprocessor. The reassembly coprocessor then processes the 52-octet cells. It controls the writing of the CPCS payload to host memory and performs all necessary Segmentation and Reassembly (SAR) and CPCS checks.

The reassembly coprocessor uses a scatter method to write the payload portion of the ATM cell to host memory. It maintains a free buffer queue and status queue in local memory to control the scatter operation. The free buffer queue is updated by the host processor to point to available cell buffers in host memory. The reassembly coprocessor updates the status queue with information on how the cell buffers are used. A hash table and a reassembly Virtual Circuit Channel (VCC) table located in local memory operate the various CPCS connections and are maintained by the reassembly coprocessor. The hash table allows the reassembly coprocessor to quickly locate the appropriate reassembly VCC table based upon the received VPI/VCI value in the ATM cell header. In addition, it permits the assignment of any VPI/VCI value to a particular connection.

1.1.4 PCI Bus Interface

The PCI bus interface responds to read and write requests by the host CPU as a PCI slave, allowing access to the chip resources by software on the host. The Bt8230 is capable of acting as a DMA bus master on the PCI bus. As a result, the PCI bus interface implements the full set of address, data, and control signals required to drive the bus as a master. It also contains the logic required to support arbitration for the PCI bus. This interface is PCI Version 2.0-compliant.

1.1.5 Local Memory Interface

The Bt8230 contains a memory controller that allows the device to access up to 8 Mbytes of SRAM memory. The memory controller also coordinates access to the internal control and status registers by the host and local processors. Both zero and single wait state access to SRAM are supported, which enables various price and performance trade-offs to be made. In addition, SRAM devices organized by_4, by_8, and by_16 can be used. Access to internal control and status registers follows the timing requirements of SRAM access, which simplifies system implementations.

1.1.6 Local Processor Interface

The local processor interface in the Bt8230 allows an external Reduced Instruction Set Computer (RISC) CPU to be directly connected to the device. The RISC serves as a local controlling intelligence that can handle initialization, connection management, overall data management, error recovery, and OAM functions. The interface is designed to connect directly to Intel i960CA/CF/Jx processors, but is generic enough to connect to other popular processors with little additional logic. The processor interface is "loosely coupled," meaning that the processor connects to the Bt8230 through bidirectional transceivers and buffers for the address and data buses. This allows the processor fast access to Bt8230 memory and registers, but insulates the Bt8230 from processor instruction and data cache fills. It also allows the processor to control multiple Bt8230 or physical devices, if desired.

The local processor interface shares the data and address bases with the local memory interface. In addition, two chip select signals are provided to select physical layer components such as the Bt8222. In this way, the Bt8222 control registers are completely accessible from the host processor via the PCI bus.

1.1.7 ATM Physical Interface

The ATM physical interface communicates with and controls the ATM link interface chip, which carries out all the transmission convergence and physical media dependent functions defined by the ATM protocol. This block strips the HEC byte from the received cell and adds a blank HEC byte to the transmit cell. No HEC processing is performed in the Bt8230. Three modes of operation are provided: standard Universal Test and Operations Physical Interface (UTOPIA) for ATM (UTOPIA), slave UTOPIA, and Bt8222. Standard UTOPIA mode conforms to the UTOPIA standard. Slave UTOPIA mode reverses the control direction to allow the ATM physical chip to drive the interface. Bt8222 mode allows the Bt8222 physical framer part to be connected directly to the Bt8230.

1.2 Logic Diagram and Pin Descriptions

The Bt8230 is packaged in a 208–pin PQFP. A functionally-partitioned logic diagram of the Bt8230 is shown in Figure 1-3. Pin descriptions, names, and input/output assignments are detailed in Table 1-1.

Figure 1-3. Bt8230 Logic Diagram

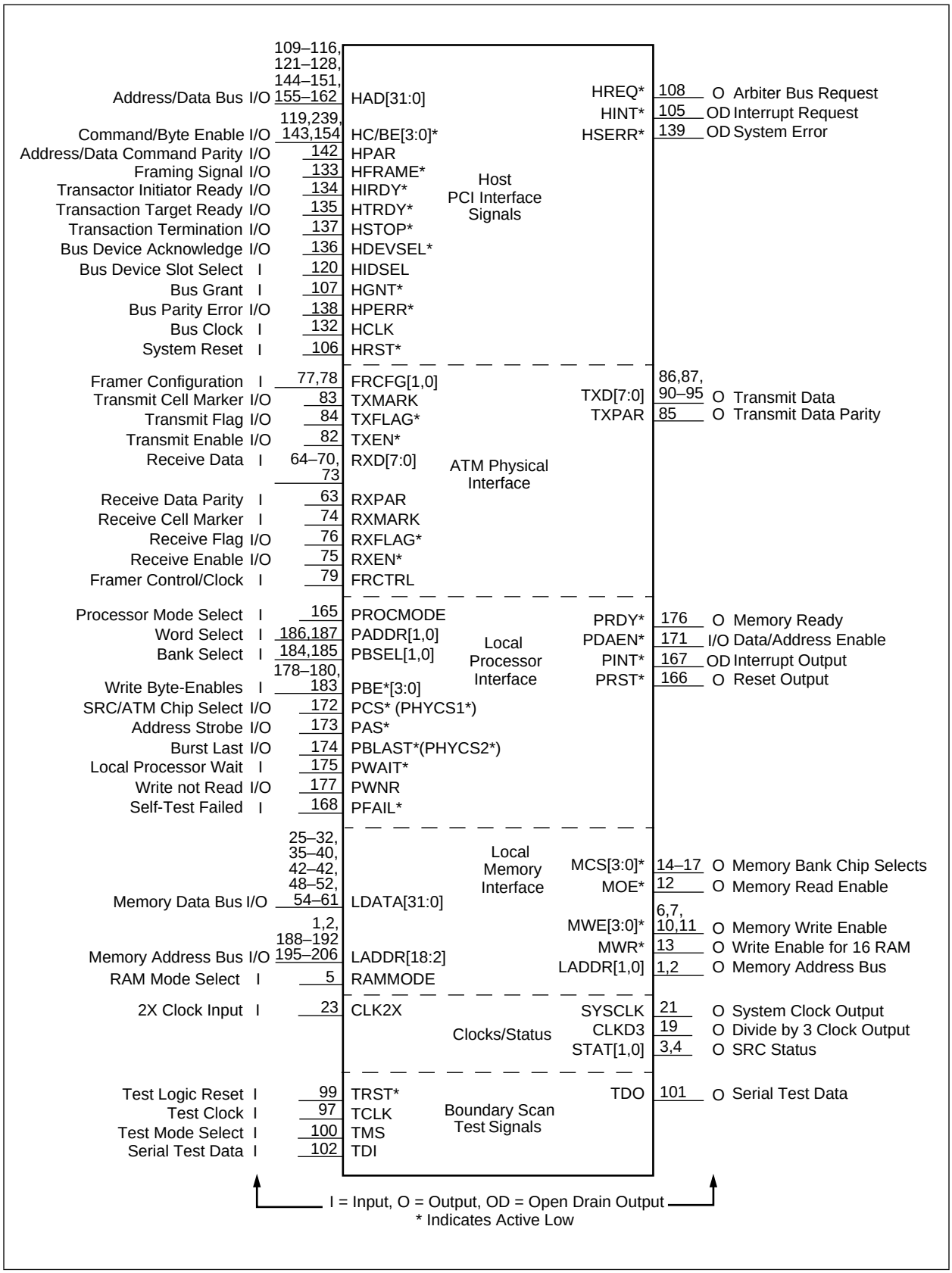


Table 1-1. Hardware Signal Definitions *(1 of 6)*

	Pin Label	Signal Name	I/O	Definition
Host PCI Interface Signals	HAD[31:0]	Multiplexed Address/Data Bus	I/O	Used by the PCI host or the Bt8230 to transfer addresses or data over the PCI bus.
	HC/BE[3:0]*	Command/Byte Enable	I/O	Outputs a command (during PCI address phases) or byte enables (during data phases) for each bus transaction.
	HPAR	Address/Data Command Parity	I/O	Supplies the even parity computed over the HAD[31:0] and HC/BE[3:0]* lines during valid data phases. It is sampled (when the Bt8230 is acting as a target) or driven (when the Bt8230 acts as an initiator) one clock edge after the respective data phase.
	HFRAME*	Framing Signal	I/O	A high-to-low HFRAME* transition indicates that a new transaction is beginning (with an address phase). A low-to-high transition indicates that the next valid data phase will end the current transaction.
	HIRDY*	Transaction Initiator Ready	I/O	Used by the transaction initiator or bus master (either the Bt8230 or the PCI host) to indicate ready for data transfer. A valid data phase ends when both HIRDY* and HTRDY* are sampled/asserted on the same clock edge.
	HTRDY*	Transaction Target Ready	I/O	Used by the transaction target or bus slave (either the Bt8230 or the PCI bus memory) to indicate that it is ready for a data transfer. A valid data phase ends when both HIRDY* and HTRDY* are sampled/asserted on the same clock edge.
	HSTOP*	Transaction Termination	I/O	Driven by the current target or slave (either the Bt8230 or the PCI bus memory) to abort, disconnect, or retry the current transfer. The HSTOP* line is used by the PCI master in conjunction with the HTRDY* and HDEVSEL* lines to determine the type of transaction termination.
	HDEVSEL*	Bus Device Acknowledge	I/O	Driven by a target to indicate to the initiator that the address placed on the HAD[31:0] lines (together with the command on the HC/BE[3:0]* lines) has been decoded and accepted as a valid reference to the target's address space. Once asserted, it is held by the Bt8230 (when acting as a slave) until HFRAME* is deasserted; otherwise, it indicates (in conjunction with HSTOP* and HTRDY*) a target abort.
	HIDSEL	Bus Device Slot Select	I	Signals the Bt8230 that it is being selected for a configuration space access.
	HGNT*	Bus Grant	I	Asserted to indicate to the Bt8230 that it has been granted control of the PCI bus, and may begin driving the address/data and control lines after the current transaction has ended (indicated by HFRAME*, HIRDY* and HTRDY* all deasserted simultaneously).
	HPERR*	Bus Parity Error	I/O	Asserted by the Bt8230 as a bus slave or asserted by a target addressed by the Bt8230 when it acts as a bus master to indicate a parity error on the HAD[31:0] and HC/BE[3:0]* lines. It is asserted when the Bt8230 is a bus slave and sampled when the Bt8230 is a bus master on the second clock edge after a valid data phase.

Table 1-1. Hardware Signal Definitions *(2 of 6)*

	Pin Label	Signal Name	I/O	Definition
Host PCI Interface Signals (continued)	HCLK	Bus Clock	I	Supplies the PCI bus clock signal.
	HRST*	System Reset	I	Performs a hardware reset of the Bt8230 and associated peripherals when asserted. Must be asserted for 16 cycles of HCLK. The CLKD3 output is active while the HRST* signal is asserted, the SYSCLK output is active during this period.
	HREQ*	Arbiter Bus Request	O	Asserted by the Bt8230 to request control of the PCI bus.
	HINT*	Interrupt Request	OD	Signals an interrupt request to the PCI host, and is tied to the INTA* line on the PCI bus.
	HSERR*	System Error	OD	Indicates a system error or a parity error on the HAD[31:0] and HC/BE[3:0]* lines during an address phase. This pin is handled in the same way as HPERR*, and is only driven by the Bt8230 when it acts as a bus slave.
ATM Physical Interface	FRCFG[1,0]	Framer Configuration	I	Configuration pins FRCFG[1,0] determine what framer interface Bt8230 supports. 00 = Bt8222 interface 01 = UTOPIA interface 10 = Slave UTOPIA interface 11 = Reserved, do not use These pins when floating to 11 can cause the SYSCLK (i.e., /2 clock) to not be divided and output a clock at the same rate as the 2x input clock.
	TXMARK	Transmit Cell Marker	I/O	In both UTOPIA and slave UTOPIA modes, the TXMARK line is asserted by the Bt8230 when the starting byte of a 53-byte cell is being output. In Bt8222 mode, this pin is asserted by the framer to indicate that it is expecting the starting byte of a 53-byte ATM cell to be transferred on the TXD[7:0] lines as the next valid data byte.
	TXFLAG*	Transmit Flag	I/O	In UTOPIA mode, TXFLAG* indicates that the transmit buffer in the downstream link interface chip is full, and no more data can be accepted. In slave UTOPIA mode, this pin indicates to the link interface chip that the Bt8230 transmit buffer is empty. In Bt8222 mode, TXFLAG* indicates that no more data is available in the transmit buffer of the Bt8230.
	TXEN*	Transmit Enable	I/O	Indicates that valid data has been placed on the TXD[7:0], TXPAR, and TXMARK lines in the current clock cycle when the Bt8230 is in UTOPIA or slave UTOPIA mode. This pin is an output in UTOPIA mode and an input in slave UTOPIA mode. In Bt8222 mode, the TXEN* input indicates that the downstream framer device is requesting the Bt8230 to transfer a byte of data on the TXD[7:0] lines.
	RXD[7:0]	Receive Data	I	Transfers incoming data bytes from the link interface or framer chip to the Bt8230 in all framer modes.
	RXPAR	Receive Data Parity	I	Should be driven with the 8-bit odd parity computed over the RXD[7:0] lines by the link interface or framer chip in all framer modes.
	RXMARK	Receive Cell Marker	I	Indicates that the current byte being transferred on the RXD[7:0] lines is the starting byte of a 53-byte cell.

Table 1-1. Hardware Signal Definitions *(3 of 6)*

	Pin Label	Signal Name	I/O	Definition
ATM Physical Interface (continued)	RXFLAG*	Receive Flag	I/O	In UTOPIA mode, RXFLAG* indicates that the receive buffer in the downstream link interface chip is empty, no more data can be transferred, and the RXD[7:0], RXPAR, and RXMARK lines are invalid. In slave UTOPIA mode, this pin indicates to the framer chip that the receive FIFO in the Bt8230 is full. In Bt8222 mode, the RXFLAG* output indicates that the Bt8230 internal FIFO buffer is full and any subsequent attempts to transfer data will be ignored.
	RXEN*	Receive Enable	I/O	In UTOPIA mode, RXEN* indicates that the Bt8230 is ready to receive data on the RXD[7:0], RXPAR, and RXMARK lines in the next clock cycle. This pin is an output in UTOPIA mode, and an input in slave UTOPIA mode. In Bt8222 mode, the RXEN* input is driven active by the framer to signify that valid data has been presented on the RXD[7:0], RXPAR, FRCTRL, and RXMARK lines.
	FRCTRL	Framer Control/Clock	I	In UTOPIA and slave UTOPIA modes, the FRCTRL line should be driven with a clock that is synchronous to that used by the framer device for interfacing to the Bt8230. The TXD[7:0], TXPAR, TXMARK, TXFLAG* TXEN*, RXD[7:0], RXPAR, RXMARK, RXFLAG*, and RXEN* lines must be synchronous to this clock in UTOPIA mode, and maintain the specified setup and hold times with reference to its rising edge. In Bt8222 mode, this pin marks a cell as invalid and the Bt8230 discards the cell.
	TXD[7:0]	Transmit Data	O	Carries outgoing data bytes to the framer chip in all framer modes.
	TXPAR	Transmit Data Parity	O	Outputs the 8-bit odd parity computed over the TXD[7:0] lines in all framer modes.

Table 1-1. Hardware Signal Definitions *(4 of 6)*

	Pin Label	Signal Name	I/O	Definition
Local Processor Interface	PROCMODE	Processor Mode Select	I	When grounded, this input selects the local processor mode. When pulled to a logic high, the standalone mode is selected.
	PADDR[1,0]	Word Select Inputs	I	The PADDR[1,0] inputs are connected to the word select field of the CPU address bus (address bits [3, 2] for the Intel i80960CA processor, which can perform 4-word burst transactions). These inputs are used by the Bt8230 to allow single-cycle bursts to be performed without requiring very short memory access times.
	PBSEL[1,0]	Bank Select Inputs	I	Selects one of four banks of memory to be accessed. They are decoded by the memory controller to generate the appropriate chip/bank selects to the external memory.
	PBE[3:0]*	Write Byte-Enables	I	Supply byte enables for each local processor memory access. These pins are only relevant during writes by the local processor to local memory. Each byte enable line corresponds to a specific byte lane in the LDATA[31:0] data bus: PBE[0]* corresponds to LDATA[7:0], PBE[1]* to LDATA[15:8], PBE[2]* to LDATA[23:16], and PBE[3]* to LDATA[31:24].
	PCS* (PHYCS1*)	SRC/ATM Chip Select ATM PHY Chip Select (in standalone mode)	I/O	In local processor mode with PROCMODE tied low, PCS* is the SRC chip select input. In standalone mode, this pin is PHYCS1*, which may be connected to the chip select input of the Bt8222 Physical Device (PHY) device.

Table 1-1. Hardware Signal Definitions *(5 of 6)*

	Pin Label	Signal Name	I/O	Definition
Local Processor Interface (continued)	PAS*	Address Strobe	I/O	Indicates a local processor address cycle. In standalone mode, PAS* is used to drive the AS* pin of the Bt8222 device.
	PBLAST* (PHYCS2*)	Burst Last ATM PHY Chip Select (in standalone mode)	I/O	In local processor mode, this input is used by the processor to indicate the end of a transaction. If burst transfers and wait cycles generated by PWAIT* are not required, tie this input low. During standalone mode, this output is a second chip select, PHYCS2*.
	PWAIT*	Processor Wait	I	Used by the local processor or external logic to insert wait states for read or write transactions.
	PWNR	Write/not Read	I/O	The PWNR input indicates the direction of a local processor transfer. A logic one indicates a write while a logic zero indicates a read. During standalone mode, this output provides the same function for the Bt8222.
	PFAIL*	Self-Test Failed	I	The local processor can indicate a failure of its internal self-test or initialization processes by asserting the PFAIL* input to the Bt8230.
	PRDY*	Memory Ready	O	Signals that the memory or control register has accepted the data on a write, or that data is available to latch by the local processor on a read cycle.
	PDAEN*	Data/Address Enable	I/O	Connected to the output enable input of the bidirectional transceivers and buffers that isolate the Bt8230 data and address bus from the local processor. In standalone mode, this input is connected to the physical device's interrupt output(s).
	PINT*	Interrupt Output	OD	Asserted by the Bt8230 to the local processor to signal an interrupt request in local processor mode.
	PRST*	Reset Output	O	Asserted by the Bt8230 to the local processor whenever the HRST* input is asserted, or when the LP_ENABLE bit in the CONFIG0 register [0x14] is a logic low.

Table 1-1. Hardware Signal Definitions *(6 of 6)*

	Pin Label	Signal Name	I/O	Definition
Local Memory Interface	LDATA[31:0]	Memory Data Bus	I/O	Data I/O bus. Used for memory reads and writes as well as Control and Status Register access by the local processor.
	LADDR[18:2]	Memory Address Bus	I/O	Address I/O bus. Used for memory reads and writes as well as Control and Status Register access by the local processor.
	LADDR[1,0]	Memory Address Bus	O	The two LSBs of LADDR. Used for memory reads and writes as well as CSR access by the local processor.
	MCS[3:0]*	Memory Bank Chip Selects	O	Selects one of four addressable banks of SRAM memory.
	MOE*	Memory Read Enable	O	Indicates that a read cycle is proceeding and that the memory device output buffers should be enabled, driving data onto the LDATA[31:0] lines.
	MWE[3:0]*	Memory Write Enables	O	Memory byte write enables for by_4 or by_8 SRAMs. For by_16 devices, these outputs are byte enables that are active on writes and reads.
	MWR*	Write Enable for 16 RAM	O	Memory write enable for by_16 SRAMs.
	RAMMODE	RAM Mode Select	I	Selects RAM chips supported. 1 = by_16 memory devices. 0 = by_4 or by_8 memory devices.
Clocks and Status	CLK2X	2x Clock Input	I	Double frequency (from SYSCLK) TTL clock input (66 MHz max.).
	SYSCLK	System Clock Output	O	This divide-by-2 of CLK2X is the internal system clock as well as the external system clock (33 MHz max.).
	CLKD3	Divide by 3 Clock Output	O	This output clock is a 50% duty cycle one-third divide of CLK2X; it may be used for the UTOPIA interface clock (22 MHz max.).
	STAT[1,0]	SRC Status	O	Bt8230 internal status outputs, including BOM indication. Internal status controlled by the STAT_MODE[3:0] field in the CONFIG0 Register (see Section 3.5.19).
Boundary Scan Test Signals	TRST*	Test Logic Reset	I	When a logic low, this signal asynchronously resets the boundary scan test circuitry and puts the test controller into the reset state. This state allows normal system operation. Tie to ground when boundary scan is not implemented.
	TCLK	Test Clock	I	Generated externally by the system board or by the tester. Tie to ground when boundary scan is not implemented.
	TMS	Test Mode Select	I	Decoded to control test operations.
	TDO	Serial Test Data	O	Outputs serial test pattern data.
	TDI	Serial Test Data	I	Input for serial test pattern data.
Supply Voltage	PWR	Power	–	Sixteen pins are provided for supply voltage.
	GND	Ground	–	Twenty-two pins are provided for ground.

2.0 Bt8230 Architecture Overview

This chapter provides an overview of the Bt8230's architecture, which is the basis for all of the SRC's functions. The first section describes the queue and data buffer system. The next section discusses the segmentation and reassembly functions in the context of the queue structures. The remaining sections cover traffic scheduling, OAM/PM, and device inputs and outputs. The Host/SRC architecture and the algorithms for task submission and status reporting have been optimized to meet the needs of high performance ATM interfaces. The Bt8230 is fully pin-compatible with the Bt8233 SRC. This allows an easy upgrade path from a UNI 3.1-compliant system implemented with the Bt8230 to a more advanced system meeting the additional requirements of the Traffic Management Specification (TM) 4.0.

2.1 Queue and Buffer Architecture

This section describes the queue and buffer structures used by the Bt8230 and the host system. The Host Transmit Queue resides in Lost Memory, and the Local Transmit Queue resides in the local memory. However, there are other queues used by the Bt8230 labeled “Host” and “Local.” In these cases, “Host” refers to data coming from the host buffers, and “Local” refers to data coming from local buffers. This is regardless of the type of traffic being transferred, or the location of the queue.

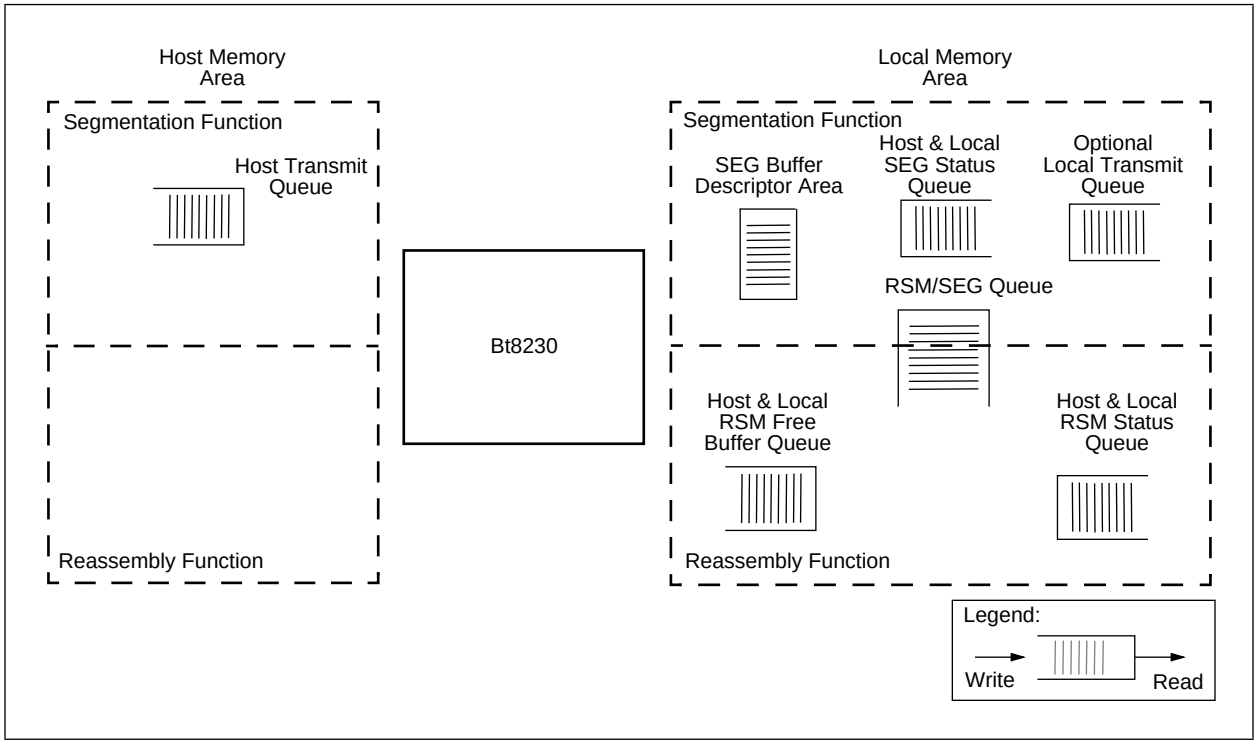
2.1.1 Queue Structure

The flow of the scheduling, segmentation, and reassembly processes in the Bt8230 is monitored, coordinated and controlled through the use of several circular queues, serviced by the Bt8230, the host or the local processor. Of these queues, only the Host Transmit Queue exists in host memory. The following queues are in local memory:

- Host and Local SEG Status Queue
- Host and Local RSM Status Queue
- Reassembly/Segmentation Queue
- Host and Local RSM Free Buffer Queues
- SEG Buffer Descriptor Area
- Optional Local Transmit Queue

Figure 2-1 shows the location of each queue in host or local memory.

Figure 2-1. Bt8230 Queue Architecture

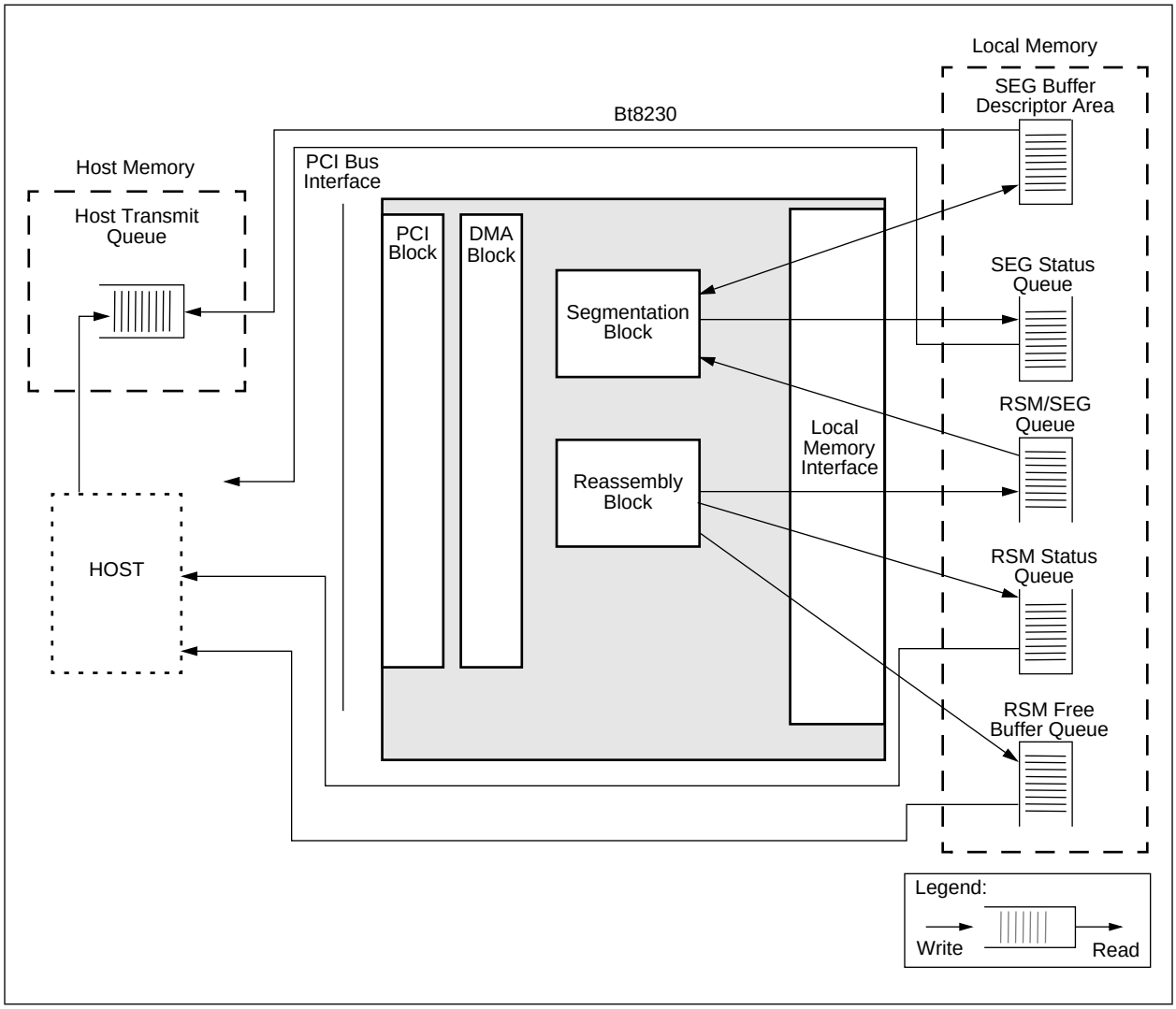


The functions of each queue are described below:

- The *Host Transmit Queue* is used by the host to submit chains of segmentation buffer descriptors to the Bt8230 for segmentation. The segmentation coprocessor then processes these Transmit Queue entries as part of the segmentation function.
- The *SEG Buffer Descriptor Area* is where the Bt8230 stores a copy of the contents of the Host Transmit Queue.
- Two queues provide control information for the segmentation and reassembly processes. The Bt8230 reports segmentation status to the *Segmentation Status Queue* and reassembly status to the *Reassembly Status Queue*. The host further processes these entries.
- Typically, the *Local Transmit Queue* is used when the local processor is employed to send control information.
- The *Reassembly/Segmentation Queue* is written to by the reassembly coprocessor and read by the segmentation coprocessor. The queue includes data on OAM-PM cells to be transmitted. The RSM/SEG Queue is a private queue for the SRC which the host cannot read or write.
- The host furnishes data buffers to the reassembly coprocessor by posting their location and availability to the *RSM Free Buffer Queue*. The reassembly coprocessor uses the RSM Free Buffer Queue entries to allocate data buffers for received ATM cells during reassembly.

These queues provide an asynchronous communication path between the host and the Bt8230. They also pass information among the major functional blocks of the Bt8230. Figure 2-2 illustrates these interactions. The arrows indicate which system entity writes to the queue and which entity reads from the queue.

Figure 2-2. Interaction of Queues with Bt8230 Functional Blocks



2.1.2 Status Queue Relation to Buffers

The status queues employed by the Bt8230 are written by the SRC and read by the host to further process the segmentation and reassembly data flow in progress or just completed. Each status queue entry includes data (such as error flags, status bits, etc.) which the host uses in its succeeding process steps. The status queue entry also holds a pointer to the first buffer descriptor of the segmented or reassembled data buffer(s) which comprises an entire PDU. This information conveys from the SRC to the host the successful or unsuccessful segmentation/reassembly of a PDU.

Figure 2-3 illustrates the association between the Segmentation Status Queue and the segmentation data buffers in Message mode. This figure shows one 3-buffer PDU on one virtual channel, represented by three entries written to the Host Transmit Queue and a single entry in the SEG Status Queue. The SRC performs segmentation processing on the PDU and writes a SEG Status Queue entry informing the Host of the status of the segmentation process. In Streaming mode, three status entries are written, one after the segmentation of each buffer is completed.

Figure 2-3. Segmentation Status Queues Related to Data Buffers

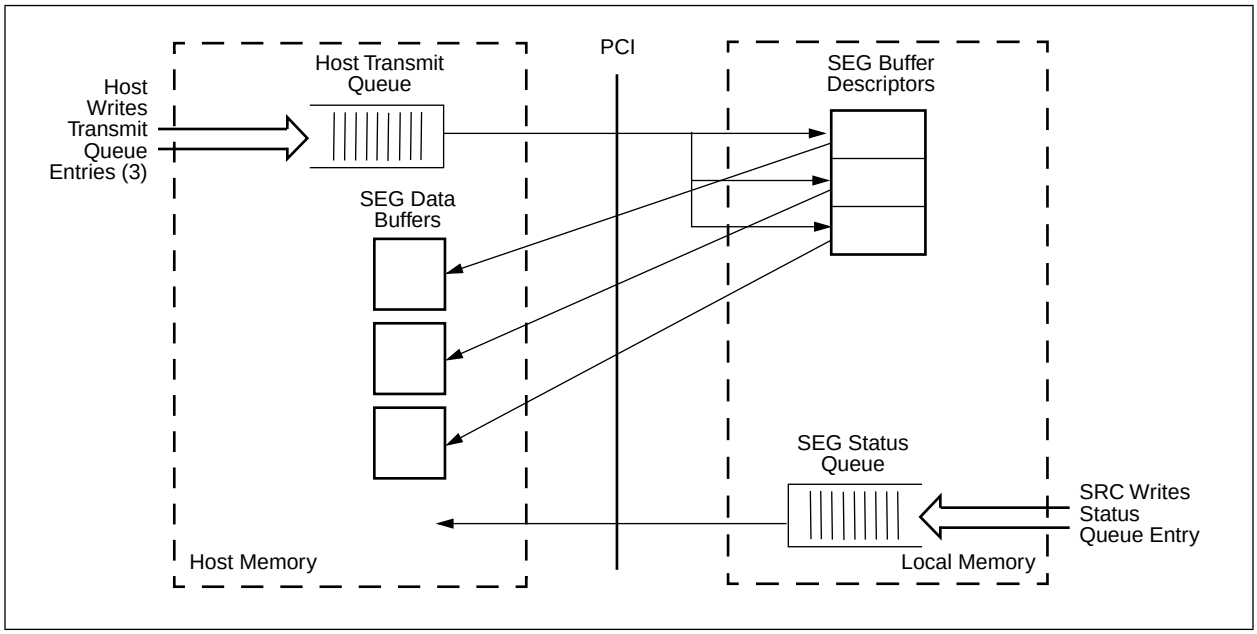
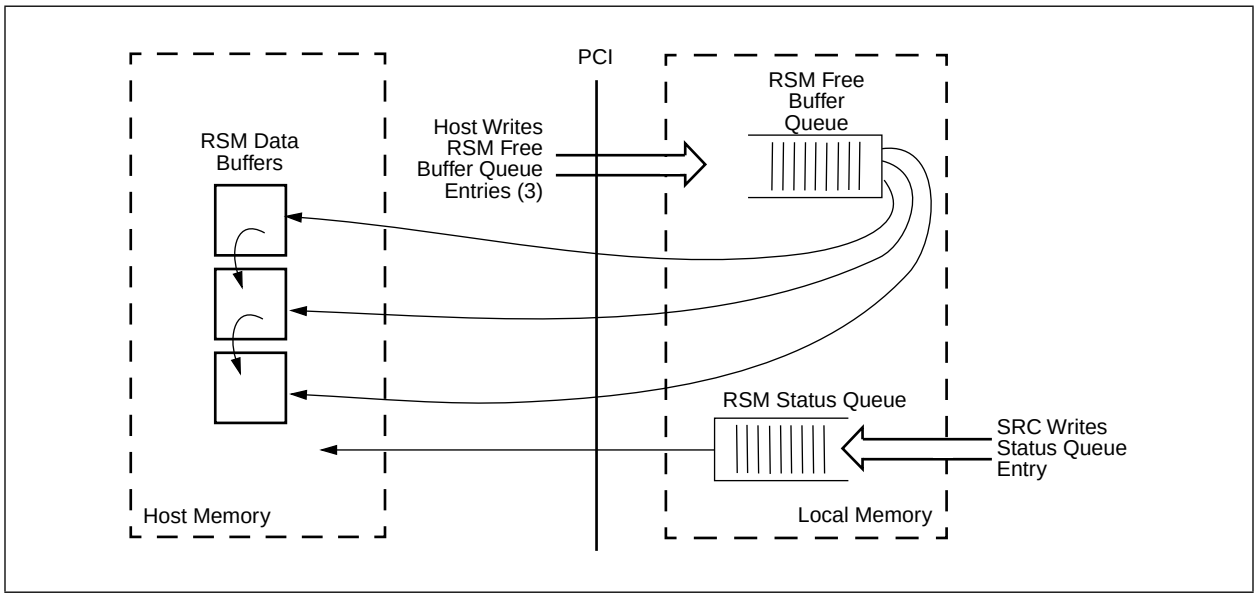


Figure 2-4 illustrates the association between the Reassembly Status Queue and the reassembly data buffers in Message mode. In this figure, the host submits free buffers to the Bt8230 by writing pointers to them in the Free Buffer Queue entries. The SRC links the buffer descriptors pointing to the three data buffers containing the reassembled PDU. It then writes the RSM Status Queue entry containing the pointer to the first buffer descriptor for that PDU. The host further processes the PDU using that data. In Streaming mode, three status entries are written, one after the reassembly of each buffer is completed.

Figure 2-4. Reassembly Status Queues Related to Data Buffers



2.2 Automated Segmentation Engine

The Bt8230 can segment up to 64 k VCCs simultaneously. It provides full support of the AAL5 and AAL3/4 protocols as well as a transparent or NULL adaptation layer, AAL0. The segmentation coprocessor block independently segments each channel and multiplexes the VCCs onto the line with cell level interleaving. For each cell transmission opportunity, the Traffic Management Table is used to determine which VCC's data the segmentation coprocessor will send.

Each segmentation channel is specified as a single entry in the Segmentation VCC Table located in local memory. These VCC Table entries define the negotiated or contracted characteristics of the traffic for that channel. They are initialized by the host, either during system initialization, or on-the-fly during operation. An initialized Segmentation VCC Table entry effectively establishes a connection on which data can be segmented.

The host initializes the buffer descriptor chain. It then writes to the host Transmit Queue. The segmentation coprocessor reads the Transmit Queue and copies its contents into a buffer descriptor in local memory. Next, the coprocessor reads the buffer descriptor when it is time to process data. It retrieves data from the buffer indicated by the pointer in the buffer descriptor.

The segmentation coprocessor then operates autonomously, formatting the cells on each channel according to the host-defined Segmentation VCC Table entries for each channel. The segmentation coprocessor formats the ATM cell header for each cell based on the settings in the Segmentation VCC Table entry for that VCC. For AAL5 and AAL3/4 traffic, it also generates the PDU-specific fields in the trailer (and header for AAL 3/4) of the CPCS-PDU and places these header and/or trailer fields in the appropriate cell for the segmented PDU. The formatted cells are passed through the Transmit FIFO to the PHY interface for transmission. For AAL0 traffic, which is intended for client-proprietary use, the segmentation coprocessor segments the SDU to ATM cell payload boundaries and generates ATM cell headers, but generates no other overhead fields. The user has per-channel, per-PDU control of 52-Octet (Raw Cell) Mode segmentation, wherein the segmentation coprocessor reads the entire 52-octet ATM cell from the segmentation buffer and does not generate the ATM headers for the cells.

The Bt8230 reports segmentation status to the host in the Segmentation Status Queue. The Bt8230 writes a segmentation status queue entry on either PDU boundaries or buffer boundaries, selectable on a per-VCC basis. PDU boundary status reporting is called Message Mode, while buffer boundary status reporting is called Streaming Mode.

2.3 Automated Reassembly Engine

The reassembly coprocessor processes cells received from the ATM Physical Interface Block. The coprocessor extracts the AAL SDU payload from the received cell stream and reassembles this information into buffers supplied by the host system and allocated per-VCC. The Bt8230 supports AAL3/4, AAL5, and AAL0 reassembly, as well as 52-Octet Mode. For AAL5 and AAL 3/4, the reassembly coprocessor extracts and checks all PDU protocol overhead. Using a hashing method, the Bt8230 reassembles up to 16 k VCCs simultaneously at a rate up to 200 Mbps on simplex connections and 155 Mbps on full-duplex connections.

Each active reassembly channel is specified as a single entry in the Reassembly VCC Table in local memory. Each entry defines the negotiated or contracted characteristics of the reassembly traffic for a particular channel. Each entry is initialized by the host during system initialization, or on-the-fly. The Bt8230 uses the VCC Table to store temporary information to assist in the reassembly process. An initialized Reassembly VCC Table entry effectively establishes a connection on which the Bt8230 can reassemble data.

The user can, on a per-channel basis, establish 52-Octet Mode reassembly. In this mode, the HEC octet is deleted to align the 53-octet cell to 32 bit boundaries. The RSM coprocessor reassembles the entire 52-octet ATM cell into the reassembly buffer.

The Bt8230 provides per-channel control of the reassembly process, including the following:

- Cell filtering on inactive channels
- Mechanisms to establish per-VCC firewalling by allocating buffer credits on a per-channel basis (limits the possibility of one VCC consuming all of the memory resources)
- Per-VCC monitoring of the length of the reassembled PDU with status reporting if the length exceeds a set maximum length for that channel
- Per-VCC statistics

The system designer can set the reassembly status reporting for any channel to either Message Mode or Streaming Mode. In Message Mode, a status entry is written only when the last buffer in a message completes reassembly. In Streaming Mode, a status entry is written for each buffer as it completes reassembly.

2.4 Traffic Scheduling

The Bt8230 multiplexes data from several sources into a single transmit cell stream. Each source has traffic parameters specified in terms of the Generic Cell Rate Algorithm (GCRA). The user stores the GCRA parameters in the Seg VCC Table. The SRC refers to them to ensure transmitted data does not exceed the limits they specify.

The GCRA defined in ATM Forum UNI 3.0 uses two parameters, *I* and *L*. *I* is the interval between cells, and *L* is the limit parameter that specifies the allowed variation in the cell arrival times. The state of the algorithm is stored in a variable called *TAT (k)* (theoretical arrival time of the k^{th} cell). Cells with the arrival times $t_a(k)$ are accepted as conforming if $t_a(k) \leq TAT(k) - L$. The new value for *TAT(k)* after a conforming cell is as follows:

$$TAT(k+1) = t_a(k) + I \quad \text{if } t_a(k) > TAT(k)$$
$$TAT(k+1) = TAT(k) + I \quad \text{if } t_a(k) \leq TAT(k)$$
$$TAT(0) = t_a(0)$$

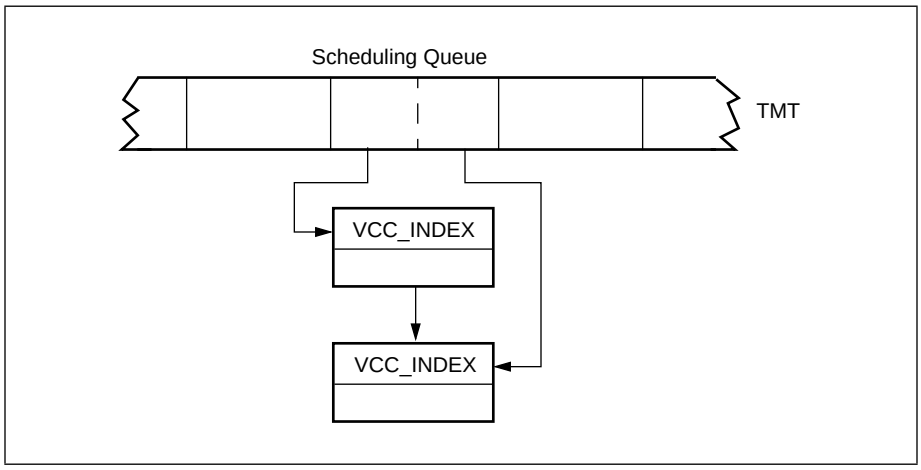
2.4.1 Traffic Management Table

The scheduling mechanism used is a circular Traffic Management Table (TMT). Each position in the management table represents a single cell time and holds the source (VCC_INDEX) that should be transmitted during the cell time. After a cell has been sent from a source, the SRC uses source parameters *I* and *L* to make an entry in the management table for the next cell from the source. The position in the management table is then updated to the next entry. The SRC performs this operation automatically.

If a TMT position has no scheduled entries, an idle cell or a UBR cell may be sent. Sources that have data but have not yet been scheduled are placed on a start queue. During an idle cell, the next cell for a new source can be scheduled. Scheduling the next cell instead of using the first idle cell position ensures that the peak cell rate will not be violated if a source halts and later restarts. A future transmission is not scheduled after the last cell of data for a source. This effectively removes the source from the TMT.

A scheduling conflict occurs when a cell is scheduled for a cell position that already has a different source scheduled. This is resolved by using a linked list of sources at each TMT entry, as shown in Figure 2-5.

Figure 2-5. Scheduling with Linked Source List



Each entry in the TMT contains a pointer to the first and last source in the linked list attached to the node. This allows a newly scheduled source to be added at either the beginning (for high priority) or end (for low priority) of the chain. For more information on high priority, refer to the FAST-START bit (VCC_CTRL[15]) description in the Transmit Ring Entry (see Table 3-6).

The scheduling algorithm assumes that each TMT entry will send a single cell. Sending more than one cell during a TMT entry causes all subsequent cells to be sent later than scheduled. These late cells are scheduled to go out earlier during the next transmission in order to regain their original positions. The system tracks the total number of extra cells that have been sent.

When a TMT entry is reached that has no scheduled cells, and extra cells have been sent, the TMT entry is skipped (no idle cell is sent). This provides a mechanism for getting the schedule back to the planned times. Later cells can then be sent earlier within the limit set by the connection's burst tolerance. In this way, Rockwell's proprietary algorithm keeps track of when cells are actually sent and makes the necessary adjustments to maintain compliance with the GCRA.

2.4.2 VBR Traffic

The Bt8230 takes advantage of the asynchronous nature of ATM by reserving bandwidth for VBR channels at average cell transmission rates without preassigning "hardcoded" schedule slots. This dynamic scheduling allows VBR traffic to be statistically multiplexed onto the ATM line, resulting in better use of the shared bandwidth resources. Through the combination of VBR parameters and priorities, it is possible to support real-time VBR services.

The outgoing cell stream for each VBR VCC is scheduled according to the GCRA algorithm. The GCRA *I* and *L* parameters control the per-VCC Peak Cell Rate (PCR) and Cell Delay Variation Tolerance (CDVT) of the outgoing cell stream on any channel. This guarantees compliance to policing algorithms applied at the network ingress point. The user can control the granularity of rate by dictating the number of schedule slots in the schedule table.

2.4.3 UBR Traffic

The UBR service category is intended for non-real-time applications which do not require tightly constrained delay and delay variation, such as traditional computer communications applications like file transfer and e-mail. Those VCCs which have not been assigned to the VBR service category as covered previously are scheduled as UBR traffic.

2.5 Implementation of OAM-PM Protocols

The Bt8230 provides internal support for the detection and generation of OAM traffic, including Performance Monitoring (PM) OAM. The Bt8230 supports the F4 and F5 OAM flows according to ITU I.610, 3/93. It monitors up to 128 channels and generates in-rate PM-OAM cells.

The Bt8230 includes a Local Processor Interface, providing the capability of local memory segmentation and reassembly. The user can thus route OAM traffic, including PM traffic, to and from this optional local processor, thereby off-loading ATM network management from the host. To facilitate this, the Bt8230 provides the option of user-defined global status queues for both segmentation and reassembly and a global buffer queue for reassembly, to which the user can assign local memory addresses. The Bt8230 will then process OAM traffic via the local processor, thereby isolating the host from these management functions and focusing host processing power on ATM user data traffic.

2.6 Standards-Based I/O

This section describes the various standards-based I/O mechanisms in the Bt8230.

2.6.1 PCI Bus I/O

The PCI bus interface implements the full set of address, data, and control signals required to drive the PCI bus as a master. It also contains the logic required to support arbitration for the PCI bus. This interface is PCI Version 2.0-compliant.

2.6.2 ATM PHY I/O

The Bt8230's ATM physical interface communicates with and controls the ATM link interface chip which carries out all the transmission convergence and physical media dependent functions defined by the ATM protocol. Three modes of operation are provided: standard UTOPIA, slave UTOPIA, and Bt8222 Cell Mux. Standard UTOPIA mode conforms to the UTOPIA Level 1 standard for an ATM Layer device. Slave UTOPIA mode reverses the control direction for use in place of a PHY on switch fabrics. Bt8222 Cell Mux mode is a FIFO interface which allows the Bt8230 to interface in a glueless connection to the Bt8222 PHY chip. Using this mode, the Bt8230 can share the Bt8222 PHY in parallel with up to three other entities. This architecture enables the concentration of Bt8230-generated traffic with other cell-based traffic such as AAL1 Circuit Emulation.

2.6.3 Local Memory I/O

To simplify system implementations, the Bt8230 integrates a complete memory controller designed for direct interface to common SRAM. The Bt8230's memory controller operates at 33 MHz and can access up to 8 Mbytes of SRAM memory. The memory controller also arbitrates access to the internal control and status registers by the host and local processors. The memory banks can be configured to a variable number of sizes. These features provide a great deal of flexibility in local memory architecture.

2.6.4 Local Processor I/O

The Local Processor Interface in the Bt8230 allows an optional external CPU to be directly connected to the device. In this way, the CPU can serve as a local controlling intelligence that can handle initialization, connection management, overall data management, error recovery, and OAM functions. The use of a local processor for these functions allows ATM message data to flow to and from host system memory in a substantially larger bandwidth, because the local processor is handling the "out of band" functions described above.

The processor interface is "loosely coupled," meaning that the processor connects to the Bt8230 through bidirectional transceivers and buffers for the address and data buses. This allows the processor fast access to Bt8230 memory and registers, but insulates the Bt8230 from processor instruction and data cache fills. It also allows the processor to control multiple Bt8230s or physical devices if desired.

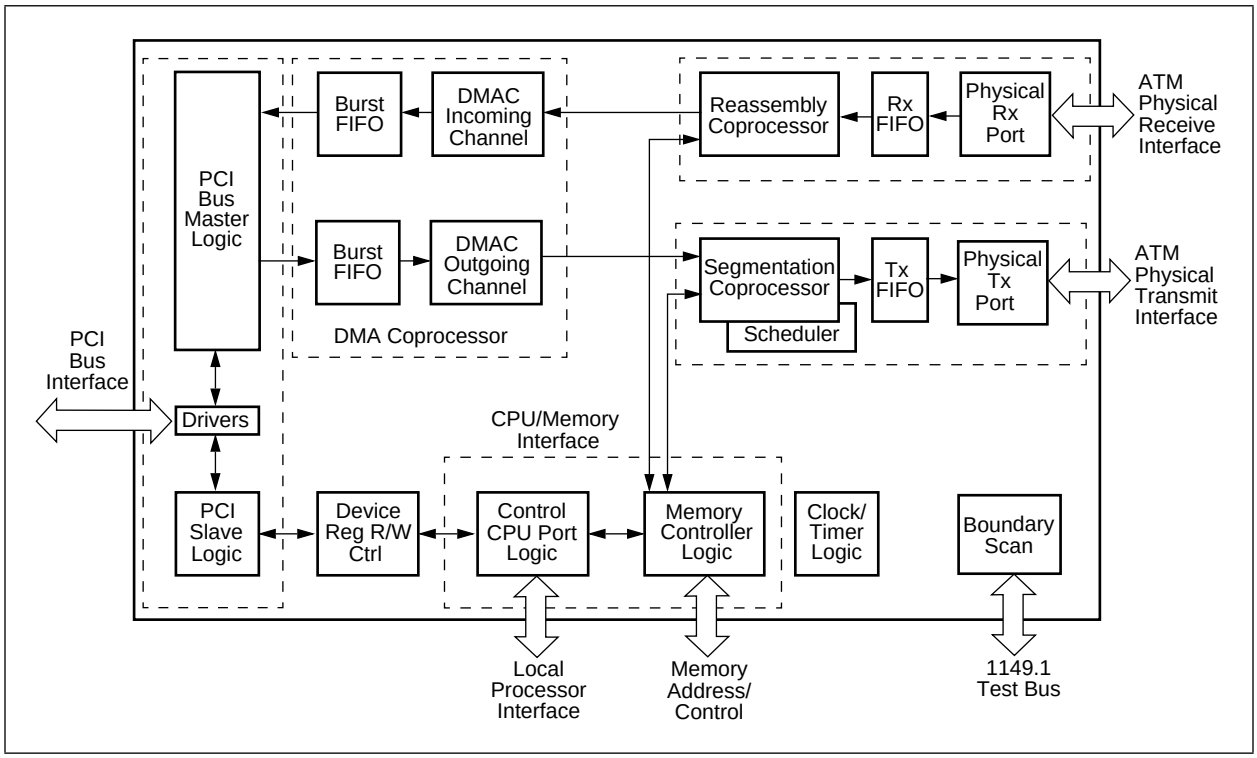
2.6.5 Boundary Scan I/O and Loopbacks

The Bt8230 includes five pins for Joint Test Action Group (JTAG) Boundary Scan, used for board-level testing. To facilitate system diagnostics, the Bt8230 incorporates an internal loopback from the segmentation coprocessor to the reassembly coprocessor.

3.0 Functional Description

This section, which describes the functions of the Bt8230, assumes a familiarity with and access to the relative ANSI and ITU communications standards (see Appendix A for a list of relative standards). The Bt8230 consists of three loosely-coupled coprocessors: a DMA coprocessor that performs data transfers to and from the memory of the host system, a segmentation coprocessor, and a reassembly coprocessor. Both the segmentation and reassembly coprocessors handle high-speed I/O between the Bt8230 and an external ATM PHY chip. The Bt8230 contains local processor support logic that allows a local processor to be connected to the Bt8230. Its PCI bus interface is responsible for accepting read and write commands from the DMA coprocessor, passed through the burst First In, First Out (FIFO) buffers. This interface also performs actual data transfers on the PCI bus. The Bt8230 also incorporates a boundary scan testing facility. Figure 3-1 depicts the overall Bt8230 functional blocks and organization.

Figure 3-1. Bt8230 Functional Block Diagram



3.1 DMA Coprocessor

The DMA coprocessor performs high-speed, sustained data transfers to and from the host memory space. It is controlled by the segmentation and reassembly coprocessors. The major functions of the DMA coprocessor are to transfer data, via the PCI bus, from the host memory to the segmentation coprocessor and from the reassembly coprocessor to the host memory space. In all modes of operation, the DMA coprocessor maintains a high level of performance. It uses burst transfers, when possible, To accomplish the following:

- Maximize use of the host bus bandwidth
- Perform byte switching to accommodate misaligned transfers
- Carry out concurrent input and output transfers (alternating burst reads and burst writes) to support simultaneous input and output data streams

3.1.1 DMA Read

For outgoing messages, DMA read cycles move data from host memory to the segmentation coprocessor using a gather-DMA method. The maximum burst size is thirteen 32-bit words, which equals one cell. The burst size can be reduced by setting the MAX_BUR_LEN field in the PCI Configuration Register to a value less than 13 words.

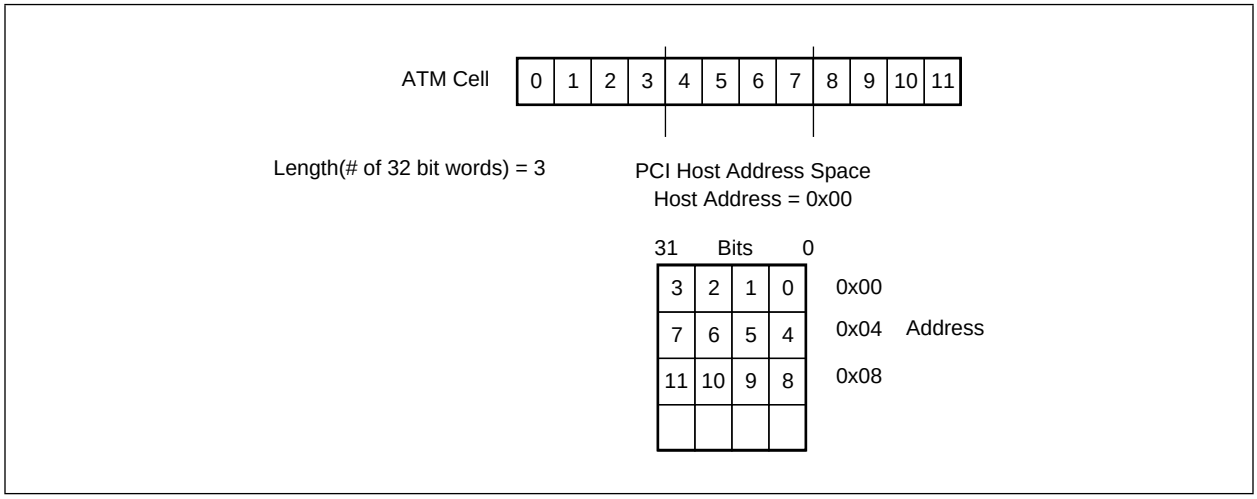
3.1.2 DMA Write

For incoming messages, DMA write cycles move data from the reassembly coprocessor to host memory using a scatter DMA method. The maximum burst size is fourteen 32-bit words, which corresponds to one ATM cell plus a status word appended if it is a PM cell. The burst size can be reduced by setting the MAX_BUR_LEN field in the PCI Configuration Register to a value less than 14 words.

3.1.3 Misaligned Transfers

The reassembly and segmentation coprocessors handle data internally on word boundaries (addresses). However, the DMA coprocessor is capable of handling PCI bus transfers of data that are not aligned on word boundaries. In addition, the length of the transfer is specified in bytes, not 32-bit words, even though the data bus widths are all 32 bits. This is achieved by the byte-switching logic used in the Bt8230. When the Bt8230 specifies a host address with the Least Significant Bits (LSBs) = 00, it is implied that the data is word-aligned. Figure 3-2 shows how a byte-aligned address would map into the PCI host address space for a little endian system. Setting the ENDIAN field [bit 12] in the Configuration Register 0 [CONFIG0; 0x14] allows you to choose between big and little endian systems.

Figure 3-2. Little Endian Aligned Transfer



When the Bt8230 specifies a host address with the LSBs ≠ 00, it is implied that the data is misaligned. Figure 3-3 illustrates how a misaligned address would map into the PCI host address space for a little endian system.

Figure 3-3. Little Endian Misaligned Transfer

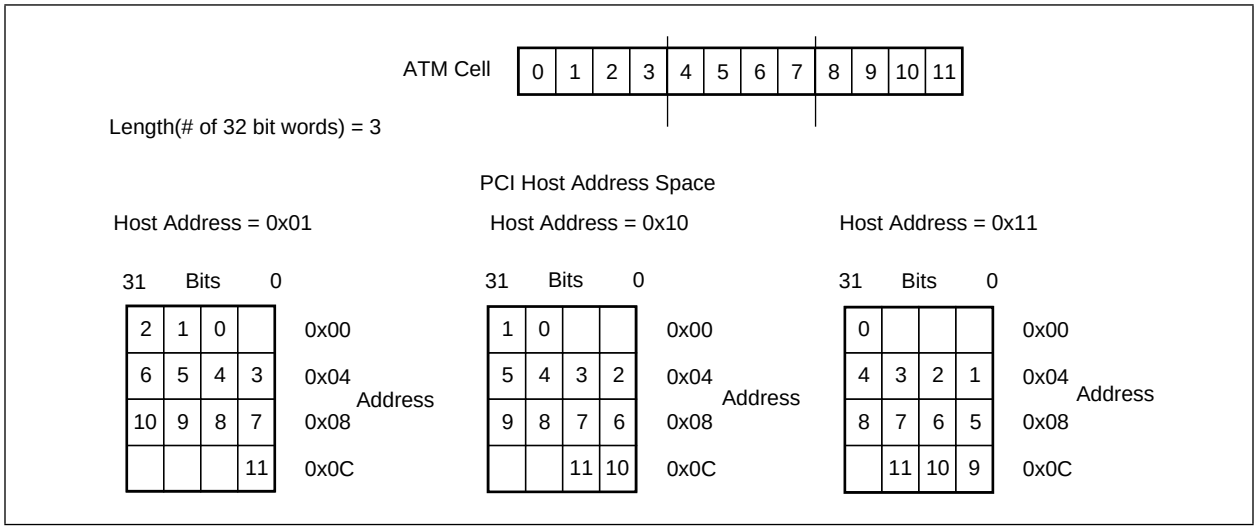
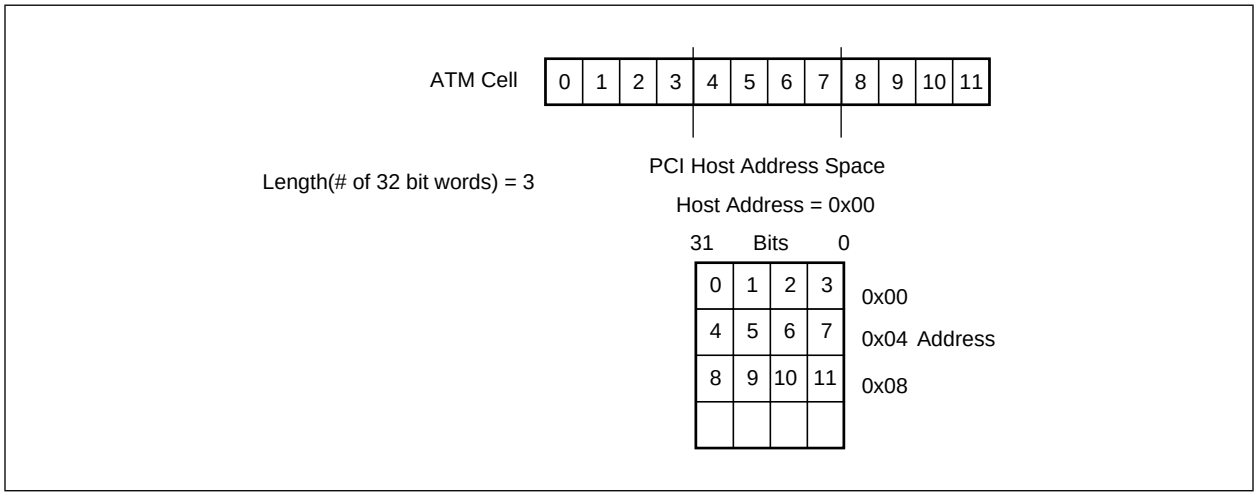


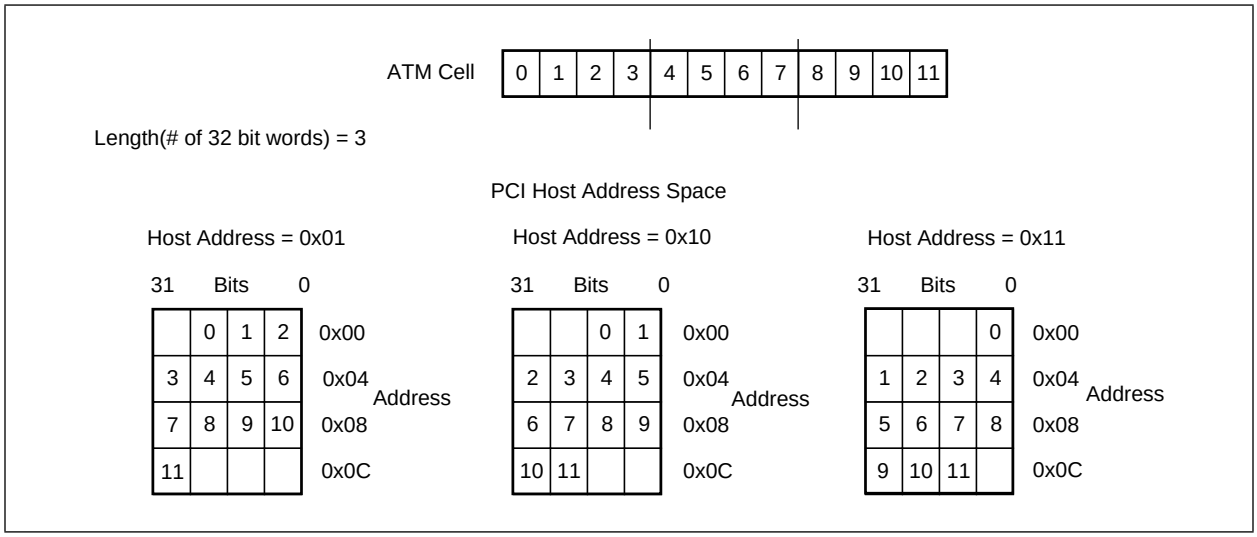
Figure 3-4 shows how a byte-aligned address would map into the PCI host address space for a big endian system.

Figure 3-4. Big Endian Aligned Transfer



When the Bt8230 specifies a host address with the LSBs ≠ 00, it is implied that the data is not aligned. Figure 3-5 shows how an unaligned address would map into the PCI host address space for a big endian system.

Figure 3-5. Big Endian Misaligned Transfer



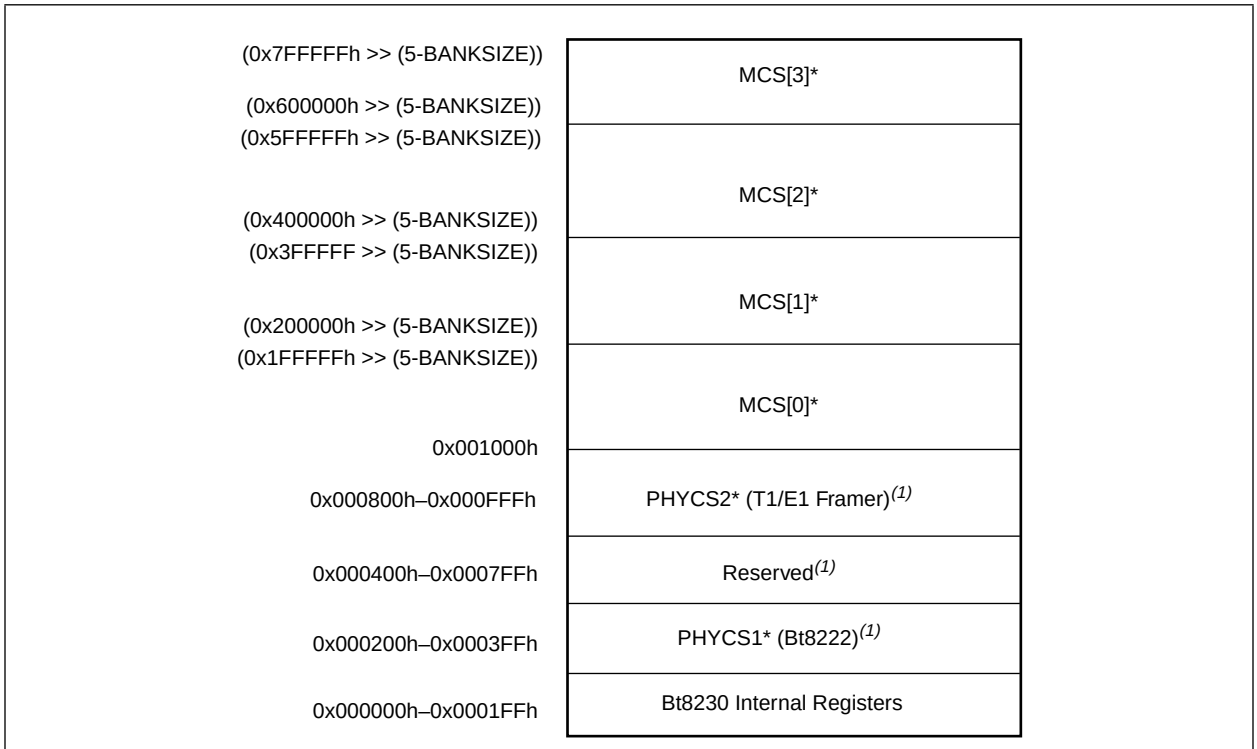
3.2 Memory Interface

To simplify system implementations, the Bt8230 integrates a complete memory controller designed for direct interface to common SRAMs. The control and status registers, as well as the physical interface devices in the standalone mode of operation, are mapped into the bottom of the memory map. Consequently, accesses to these resources are also controlled by the memory controller. Figure 3-6 shows the Bt8230 address map.

The Bt8230 can access up to 8 Mbytes (minus 1000h bytes to allow for SRC registers and reserved area overlap) of external memory using SRAM devices. The amount of memory required is heavily dependent on the number of VCCs implemented, as well as the number of VCCs that are currently active. Memory requirements are provided in detail in subsection 3.4.1. Further discussion on how to size queues and allocate memory can be found in Appendix B.

All memory accesses to the Bt8230 are 32-bit word accesses. If designers are using less than the full 8 Mbytes of address space and want to avoid losing the memory space shared with the register and reserved areas due to the overlap, they can choose to implement a memory configuration that avoids the use of MCS[0]*.

Figure 3-6. Bt8230 Memory Map



Notes: (1). These device selects are available in standalone mode only; otherwise, this memory is mapped to MCS[0]*.
2. All addresses in this illustration are byte addresses.

3.2.1 Memory Bank Characteristics

The external memory is organized in 1 to 4 banks of up to 2 Mbytes each. The system may use any number of banks to fulfill the memory requirements. The banks must be of the same size and organization. The local processor selects between the banks via the PBSEL[1,0] inputs. The BANKSIZE[2:0] field [bits 9–7] in the CONFIG0 register denotes the size of the memory banks and allows the Bt8230 to incorporate the various bank sizes into contiguous memory. Table 3-1 gives the coding of the BANKSIZE[2:0] control bits.

Table 3-1. Memory Bank Size

BANKSIZE	Bank Memory Organization	Total Bank Size (Bytes)	PBSEL[1,0] Connection	Typical Implementation
111	Reserved			
110	Reserved			
101	512 k x 32	2 M	A[22:21]	Four 512 k x 8
100	256 k x 32	1 M	A[21:20]	Two 256 k x 16, Eight 256 k x 4
011	128 k x 32	512 k	A[20:19]	Four 128 k x 8
010	64 k x 32	256 k	A[19:18]	Two 64 k x 16, Eight 64 k x 4
001	32 k x 32	128 k	A[18:17]	Four 32 k x 8
000	16 k x 32	64 k	A[17:16]	Two 16 k x 16, Eight 16 k x 4

The memory controller works with standard by_8 and by_4 SRAM devices, as well as with by_16 devices. Grounding the RAMMODE input selects the by_4 or by_8 mode of operation. Pulling RAMMODE to a logic 1 selects by_16 operation. When by_16 operation is selected, the MWE[3:0]* outputs become byte enables for both reads and writes. Figure 3-7 shows a typical half-Mbyte bank implementation using by_8 SRAM devices. Figure 3-8 shows a typical 1–Mbyte bank using by_16 RAM. To connect different sized RAM banks, simply use more or less address bits. All other control remains the same.

NOTE: The number and type of SRAM chips used affect the address and data bus capacitance and, consequently, the ac timing specifications and the required SRAM speed. Also note that the use of by_4 devices causes more address bus loading than the use of by_8 or by_16 devices. See Chapter 5.0 for detailed timing information.

Figure 3-7. 0.5 Mbyte SRAM Bank Utilizing by_8 Devices

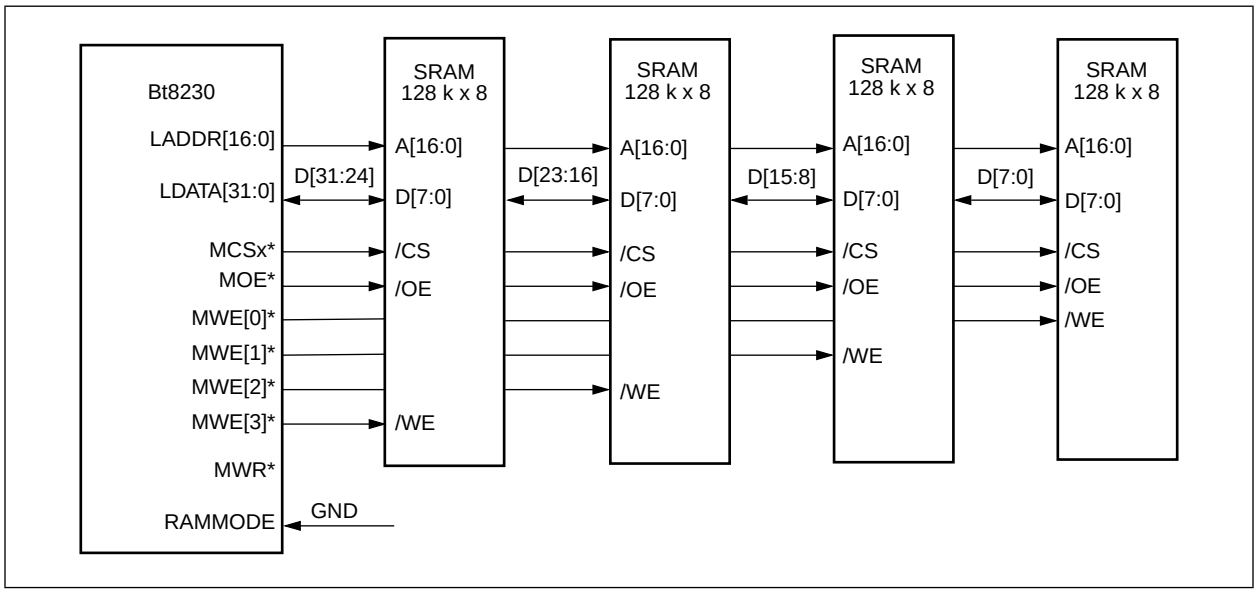
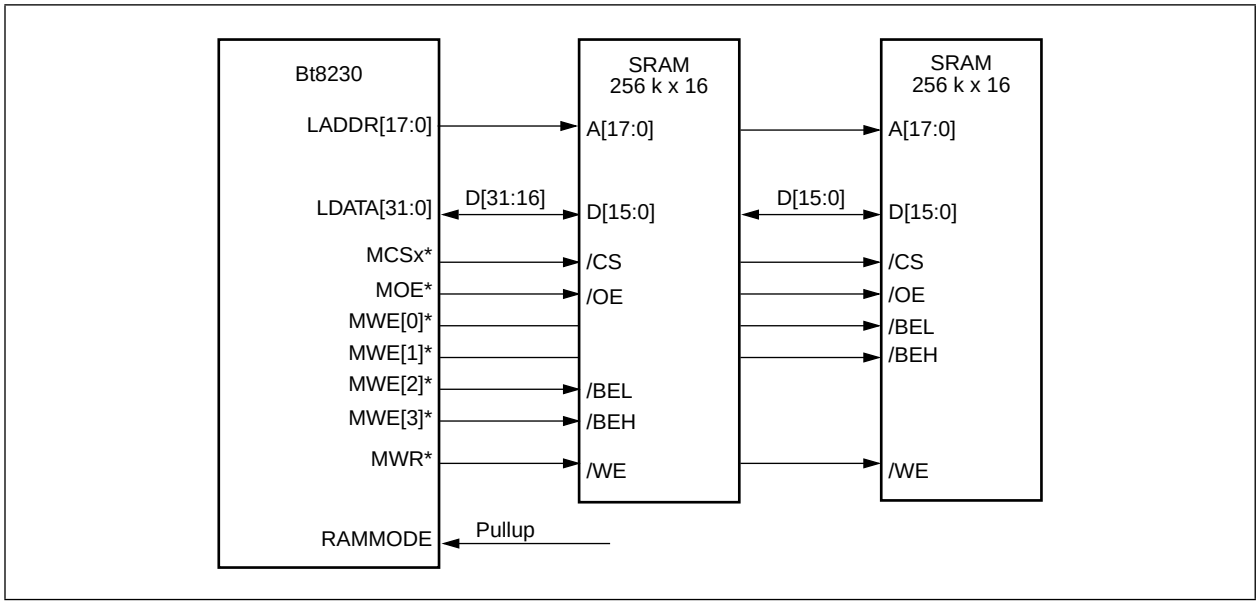


Figure 3-8. 1 Mbyte SRAM Bank Utilizing by_16 Devices



The memory map contains space allocated for the Bt8222 physical interface IC and for the Bt8370 T1/E1 framer. This mapping is only valid when the PROCMODE input pin is pulled high, indicating standalone operation with no local processor present. Standalone operation is detailed in Section 3.3.5. When PROCMODE is set to logic low and the local processor is present, addresses 0x200h–0xFFFh are available for general use and are mapped to MCS[0]*****.

The MEMCTRL field [bit 10] in the CONFIG0 register selects the number of wait states that the memory controller uses to access the SRAM. A logic 0 indicates 0 wait state or single-cycle memory, while a logic 1 indicates 1 wait state or two-cycle memory. The power-on default is MEMCTRL = 1, selecting 1 wait state or two-cycle memory accesses.

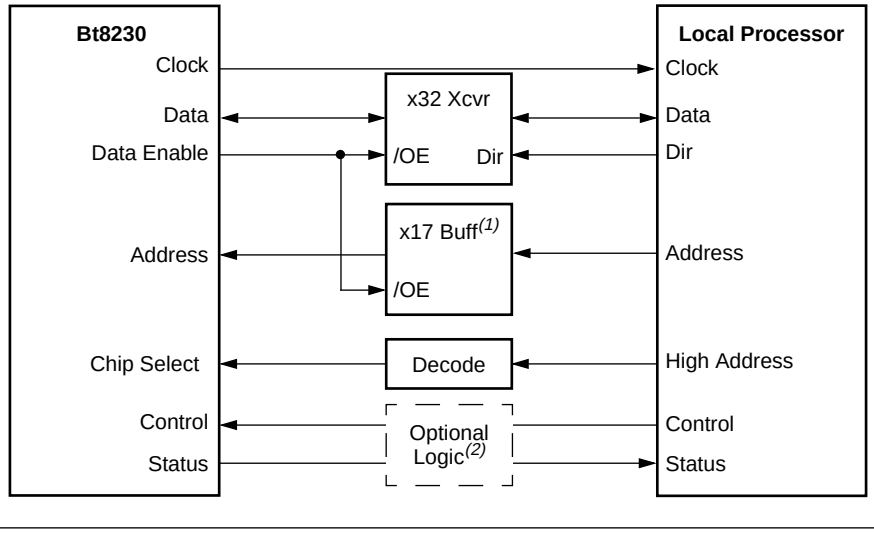
Accesses made to the control registers by the local processor follow the convention for SRAM accesses; that is, either 0 or 1 wait state depending on MEMCTRL programming. Subsequently, the local processor sees no functional timing differences between accesses to registers or SRAM. The internal register accesses from the PCI slave interface are always 0 wait state.

SRAM access time requirements are directly proportional to the system clock speed, as well as the amount and organization of the memory. The required system clock speed for a given application is dependent on the physical line rate, number of VCCs, and the percentage of idle cells versus assigned cells. Refer to Section 3.3.6 for more information. Memory access times and other requirements are specified at three typical implementations of 1, 2, and 4 banks of by_8 SRAM. In terms of address bus loading, 1 bank of by_8 SRAM equals 1-half bank of by_16 or two banks of by_4. In this way, the system designer can choose the appropriate SRAM characteristics to suit the amount of memory and organization required for the application. See Chapter 5.0 for timing information.

3.3 Local Processor Interface

The Bt8230 integrated circuit can be used in conjunction with an external processor that performs initialization, link management, monitoring, and control functions. The local processor interface consists of a “loosely coupled” architecture where it interfaces to the Bt8230 through bidirectional transceivers and buffers. These transceivers and buffers are controlled by the local processor and the Bt8230, as shown in Figure 3-9. This architecture allows the processor access to all of the Bt8230 local SRAM memory and control registers, while insulating the Bt8230 from processor instruction and data cache fills. This also allows the local processor the option to control multiple Bt8230 and/or physical interface devices.

Figure 3-9. Bt8230—Local Processor Interface



Notes: (1). Required to address full 8 MByte range, typically fewer address lines are used.
(2). Required for non-i960 processors.

The processor interface is a generic synchronous interface based on the Intel i960CA 32-bit architecture, and is completely compatible with the i960CA/CF and the new i960Jx processors. Other synchronous and asynchronous processors (e.g., from Motorola, AMD, IDT) can be interfaced using external circuitry. The only requirements are that the processor have a 32-bit bus and that the control signals be synchronized to the System Clock (SYSCLK).

To access the Bt8230 local memory or control registers, the processor must arbitrate with the Bt8230 for access to the memory controller. Due to the requirements of reassembly and segmentation access to SRAM, and the implications of PCI bus utilization, the local processor has the lowest priority in the memory arbitration scheme. Since the local processor is typically used for low bandwidth supervision and maintenance functions, this should be acceptable.

When the local processor accesses the Bt8230's control registers or local memory, a local processor memory request is generated within the Bt8230. The memory arbiter then coordinates this request with requests from other memory consumers and grants the memory bus to the local processor at the appropriate time. The local processor is held off during this process by the insertion of a variable number of wait states; accomplished by the i960 withholding READY* or RDYRCV*. Once the local processor is granted the memory system, the transceivers are enabled to allow the local processor's address and data to access the SRAM or control registers. The conclusion of the data transaction is signaled by the assertion of PRDY*. Wait states may be inserted by the processor at any time by asserting PWAIT**. The last data cycle in a burst is indicated by the PBLAST* signal. In this manner, non-i960 processor half-speed buses or slow transceivers can be accounted for.

LP_BWAIT [bit 11] in the CONFIG0 register automatically adds a single wait state between the first access in a burst and subsequent accesses. This can be used to simplify the design of memory controllers for processors that do not produce a wait output and which require more time between data cycles in a burst.

3.3.1 Interface Pin Description

The local processor bus interface consists of the control, address, and status signals described in Table 3-2. As a reference, see Table 1-1, "Hardware Signal Definitions," on page 8.

Table 3-2. Processor Interface Pins

Signal	Dir(1)	Description
PROCMODE	I	Processor interface mode select input—A logic low on this input enables the local processor mode of operation.
PCS*	I	Processor interface chip select—A logic low on this signal in conjunction with a logic low on PAS* at the rising edge of SYSCLK initiates a memory request to the memory controller.
PAS*	I	Processor address strobe— A logic low on this signal in conjunction with a logic low on PCS* latches the value of PWNR, PBSEL[1,0], PADDR[1,0], and PBE[3:0]* at the rising edge of SYSCLK.
PWNR	I	Processor write/read select—A logic 1 on this input indicates a write cycle, a logic 0 indicates a read cycle. Latched at rising edge of SYSCLK when PAS* and PCS* are active.
PADDR[1,0]	I	Word select address inputs—Indicates the word address for a single cycle access, or the first word for a multi-cycle burst access. Latched at rising edge of SYSCLK when PAS* and PCS* are active.
PBSEL[1,0]	I	Bank select inputs—Decode to select MCS[3:0]*; see Figure 3-6 and Table 3-1 for details. Latched at rising edge of SYSCLK when PAS* and PCS* are active.
PBE[3:0]*	I	Byte select inputs—Active low. Allows individual bytes of selected word to be written. Not active on reads. Latched at rising edge of SYSCLK when PAS* and PCS* active. PBE[3]* controls writes to LDATA[31:24], PBE[2]* controls writes to LDATA[23:16], etc.
PWAIT*	I	Processor wait input—Allows processor to insert variable number of wait states to extend memory transaction. Must be active on rising edge of SYSCLK with PRDY* active to insert wait cycle. May be used to interface to half speed or slow processor bus or to allow the use of slow transceivers. If the insertion of wait states is not required, set this input to a logic high. This signal may only be active (logic low) when PBLAST* is a logic high.
PBLAST*	I	Processor burst last input—Indicates the last word of a cycle. Must be active on rising edge of SYSCLK with PRDY* active to indicate last cycle. If burst accesses and wait cycles generated by PWAIT* are not required, this signal should be set to a logic low.
PRDY*	O	Processor interface ready signal—A logic low on this signal at rising edge of SYSCLK indicates that the present cycle has been completed. If a read cycle, the data is valid to latch by the processor; if a write cycle, the data has been written and may be removed from the bus. When PRDY* is active, wait states may be inserted with PWAIT*, or a single or burst cycle may be terminated by PBLAST*(2).
PDAEN*	O	Processor data and address bus enable—Connects to output enable input of bidirectional transceiver and buffer to enable data and address for processor cycles. May also be used to indicate that the processor has successfully arbitrated for access to the memory controller(3).

Notes: (1). Direction given with respect to the Bt8230.
(2). This output corresponds to the READY* or RDYRCV* input in the i960 architecture.
(3). The processor system is responsible for controlling the direction of the bidirectional data bus transceiver. In the i960 architecture, this may be controlled by the DT/R* signal.

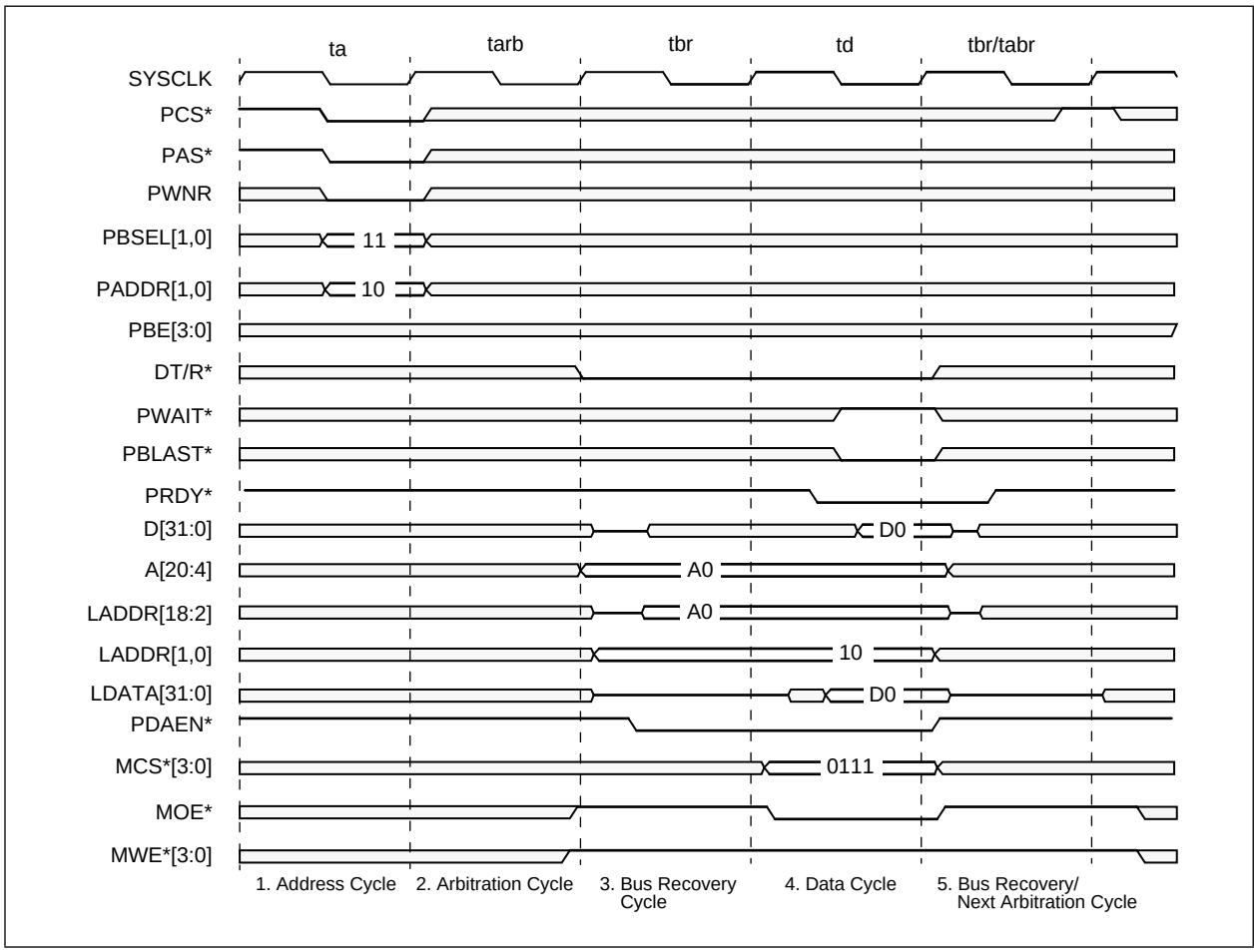
3.3.2 Bus Cycle Descriptions

Throughout the bus cycle descriptions, cycle refers to a single SYSCLK cycle ending with a rising edge. An arbitration cycle is one in which the memory requests from the local processor and internal memory consumers are compared and the one with the highest priority is granted the memory access on the next cycle. The priority for the users of local memory is: 1) Reassembly (RSM), 2) Segmentation (SEG), 3) PCI, and 4) Local. A memory access that was previously arbitrated can occur on an arbitration cycle. Once the local processor has successfully acquired the memory controller, it holds the bus until it is relinquished by the assertion of PBLAST* on the last data cycle. Therefore, local processor burst transfers will always be completed and can theoretically be of arbitrary length. However, in practice, burst transfers should be limited to four or less words. The maximum arbitration delay for a local processor access is on the order of 20 cycles, but it will typically be from 1 to 4 cycles. This parameter is heavily influenced by the SYSCLK frequency, line rate, number of VCCs, idle cell ratio, and SRAM access speed. Therefore, a system design in which local processor accesses must occur within a fixed time period is not recommended.

3.3.2.1 Single Read Cycle, 0 Wait State

Figure 3-10 illustrates a single read cycle with 0 wait states. During the address cycle (cycle 1) at the rising edge of SYSCLK with PCS* and PAS* active, a memory request is generated by the processor interface circuitry. Also at this time, the read/write select, bank select, and word select inputs (PWNR, PBSEL[1,0], and PADDR[1,0]) are internally latched. The byte enables (PBE[3:0]*) are “don’t cares” during reads. During cycle 2, this local processor memory request is processed by the memory arbitration circuitry. If no other memory consumers request an access on the same cycle, the local processor is granted access on cycle 3. To take into account bus transceiver turnaround, however, cycle 3 is always a wait or bus recovery state. This gives sufficient time for the address from the processor to access the SRAM. For 0 wait state SRAM, unless a wait state is inserted by the processor, the data is available to be latched into the processor on cycle 4, which is indicated by the assertion of PRDY*. Cycle 5 is an arbitration cycle for the internal memory consumers which may have requested access during the processor access. It also serves as a bus recovery cycle for the processor. Once the PCS*, PAS*, PWNR, PBSEL[1,0], and PADDR[1,0] are sampled at cycle 1, they are “don’t cares” for the remainder of the access. In addition, DT/R* is an output supplied by the local processor to indicate the direction of the data transceivers, and the Bt8230 PDAEN* signal is active to enable data and address.

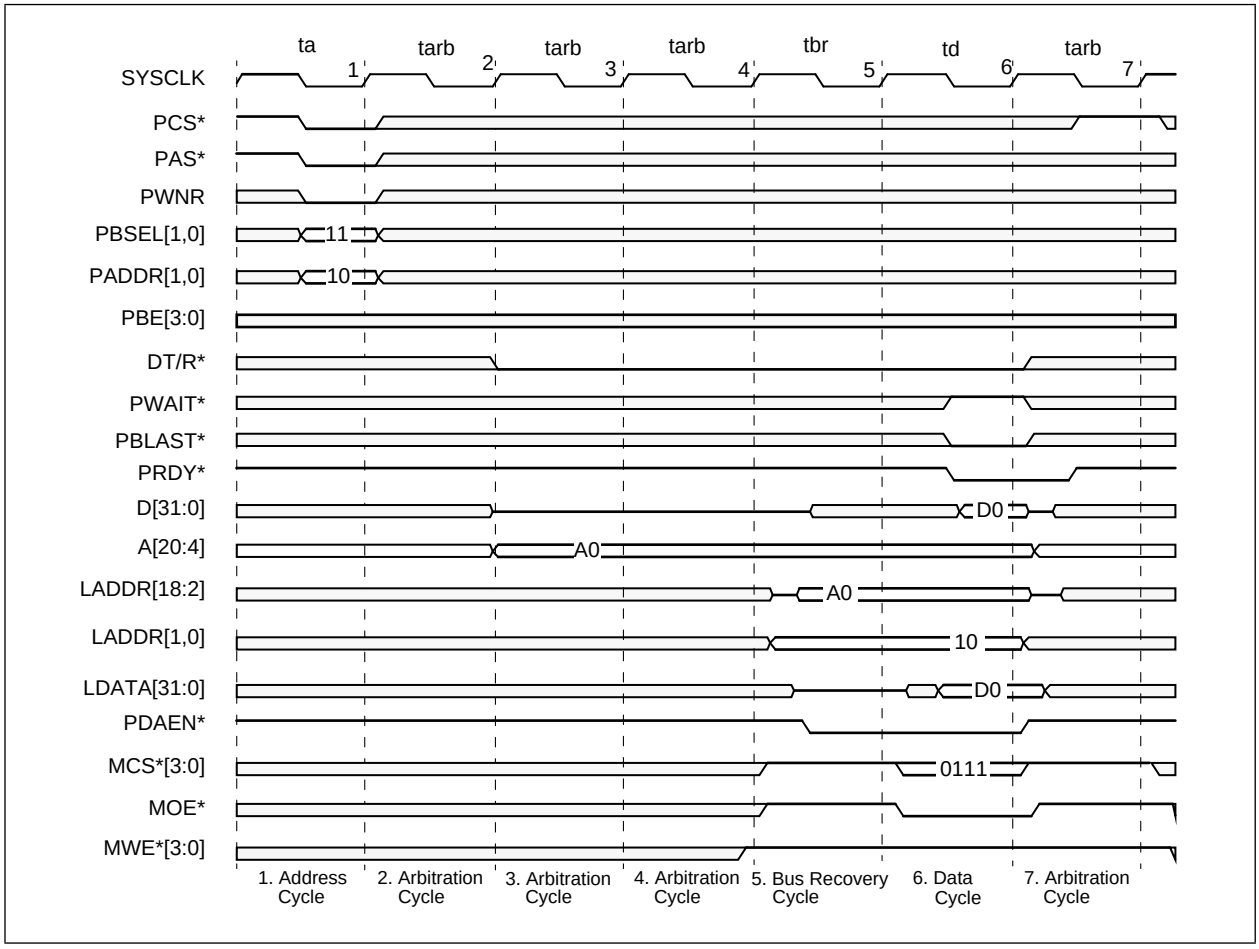
Figure 3-10. Local Processor Single Read Cycle



3.3.2.2 Single Read Cycle, Wait States Inserted By Memory Arbitration

Figure 3-11 illustrates a local processor single read cycle with arbitration wait states. This example is similar to the preceding one, except that here the local processor is not able to access the RAM immediately because of higher priority memory requests on cycles 2 and 3. On cycle 4, the memory controller allows the local processor access to the address and data bus, and the transaction takes place at the end of cycle 6.

Figure 3-11. Local Processor Single Read Cycle with Arbitration Wait States

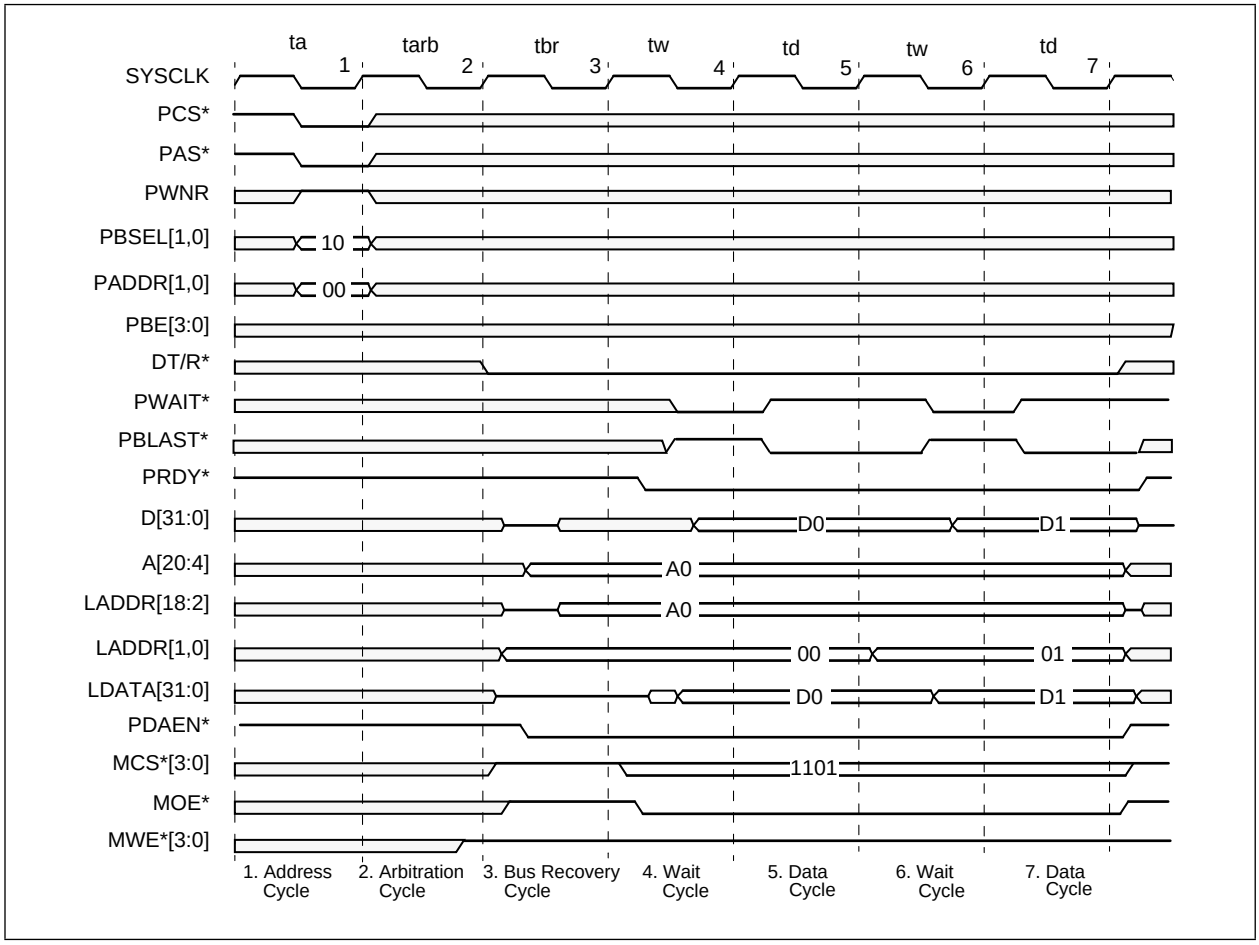


3.3.2.3 Double Read Burst With Processor Wait States

In Figure 3-12, the processor inserts wait states on cycle 4 and cycle 6 to allow additional time for the reads to occur. At the rising edge of SYSCLK on cycle 4 and cycle 6, the combination of PWAIT* low and PRDY* low extends the read by one more cycle.

NOTE: That the local processor word-select inputs (PADDR[1,0]) are latched at cycle 1, and the Bt8230 word-select address lines, LADDR[1,0], are incremented automatically at the beginning of cycle 6.

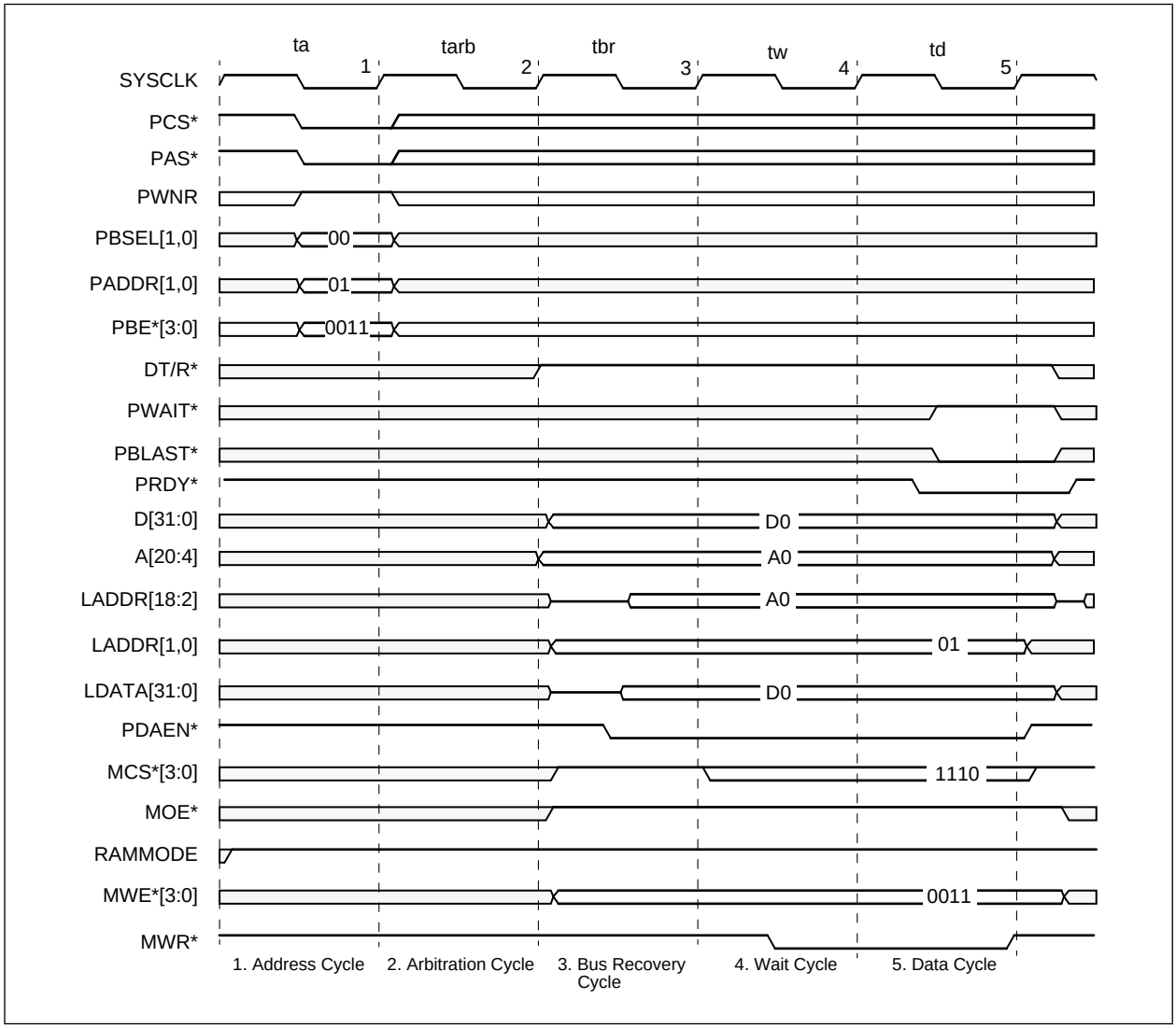
Figure 3-12. Local Processor Double Read with Wait States Inserted



3.3.2.4 Single Write With One-Wait-State Memory

In Figure 3-13, the local processor performs a single write into one-wait-state memory. There is no arbitration delay. RAMMODE is a logic high, indicating that by_16 RAM is used. Here the PBE[3:0]* inputs are latched at cycle 1 and are used to select the byte enables that are active during the cycle when MWR* is active. In this case, the two most significant bits are active, indicating a 16-bit write to the two most significant bytes.

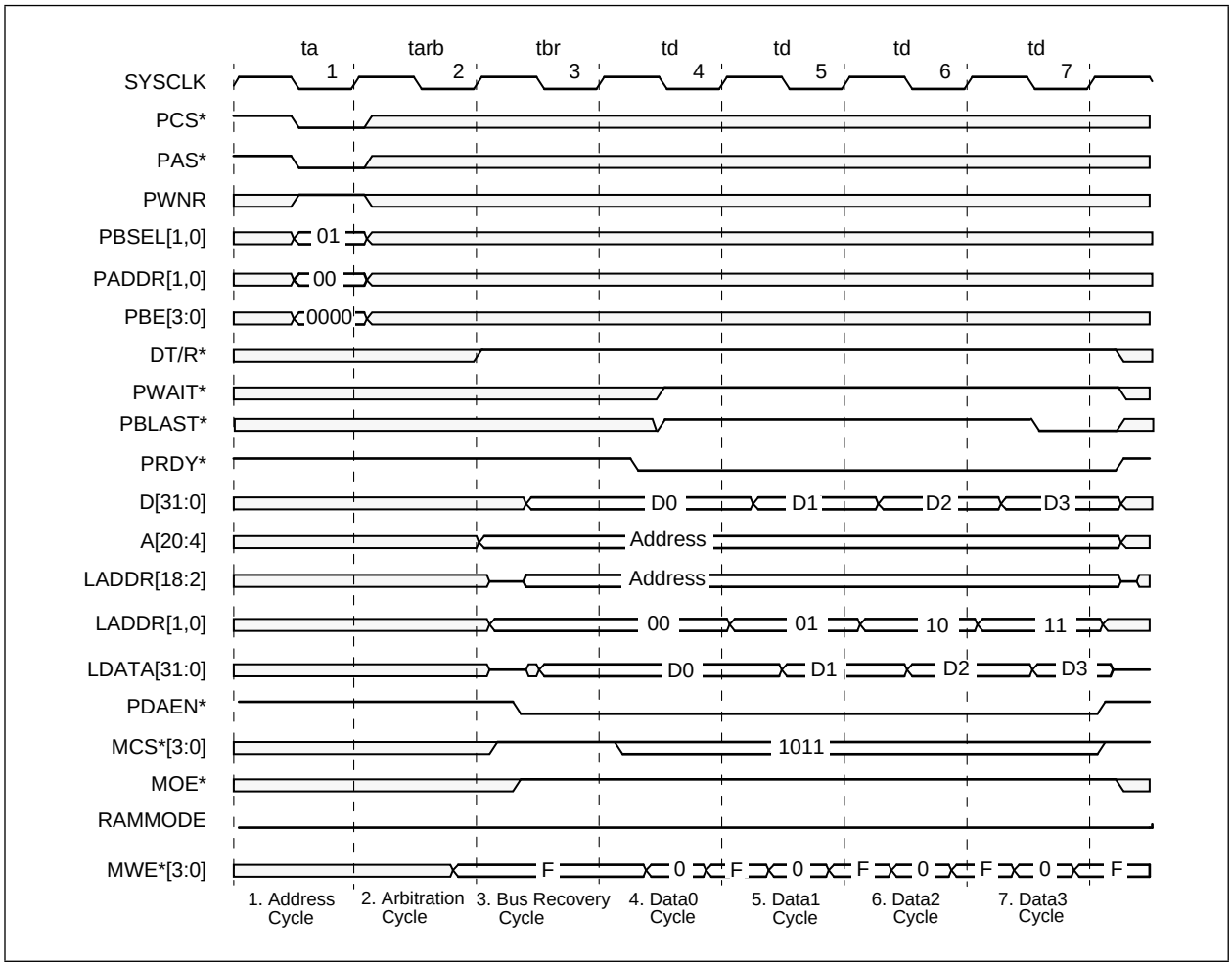
Figure 3-13. Local Processor Single Write with One Wait State by_16 SRAM



3.3.2.5 Quad Burst Write, No Wait States

In Figure 3-14, a quad burst write access to 0-wait-state memory is shown. RAMMODE is logic low, selected by_8 or by_4 RAM mode. Here PBE[3:0]* is latched on cycle 1, indicating that the write is active on all bytes and the MWE*[3:0] outputs are active as write strobes while MWR* is not used. Local memory word select addresses, LADDR[1,0], are incremented automatically by the Bt8230 on each successive write cycle. Although the i960 architecture has the limitation that a quad word transfer must start on a quad word boundary, the Bt8230 does not have that limitation. The PADDR[1,0] bits may be any value and are incremented as long as the burst transfer proceeds.

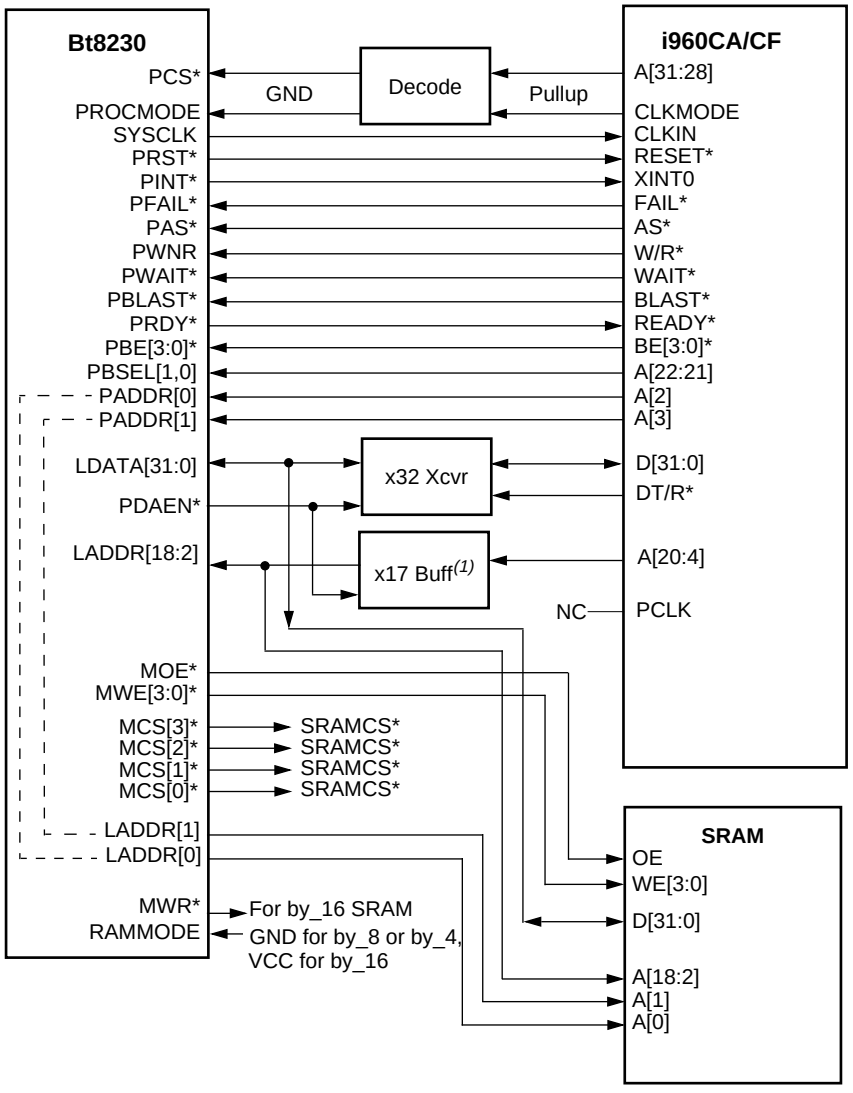
Figure 3-14. Local Processor Quad Write, No Wait States



3.3.3 i80960CA/CF Processor Interface

Figure 3-15 illustrates the signal interface between the Bt8230 device and the i960CA/CF processor. The memory region decoded for PCS* should be set for N_{RAD} and N_{WAD} = 2, N_{RDD} and N_{WDD} = 0 or 1 depending on the use of 0 or 1 wait state SRAM, and N_{XDA} = 1. In addition, external ready control must be enabled, and burst can be enabled or disabled at the system designers option. Pulling up the i960 CLKMODE input to a logic 1 selects the divide-by-one clock mode, making i960 PCLK synchronous to SYSCLK.

Figure 3-15. i960CA/CF to the Bt8230 Interface



Note: (1). Required for full 8 MByte address range

This configuration is for addressing the entire 8 MB of SRAM. In the majority of systems, the SRAM requirements are considerably less. The implications are that the PBSEL[1,0] inputs can be driven by lower order address lines, and there are less than 17 address lines to buffer. Therefore, in most applications, the data transceivers can utilize two by_16 parts, such as a 74ABT16245, and the address buffer can utilize a single by_16 74ABT16244.

NOTE: The i960CA/CF signals a failure of its internal self-test upon reset or power-up by asserting its FAIL* output. This line is connected to the PFAIL* pin of the Bt8230, and the status of this pin is reflected in the Host Interrupt Status Register [HOST_ISTAT0; 0xC0].

3.3.4 i80960Jx Operating Mode

The major difference between the i80960Jx and the i80960CA processors is that the Jx uses a multiplexed address/data bus structure while the CA/CF is non-multiplexed. However, in the Bt8230 system, the demultiplexing of addresses/data takes place on the processor side of the address buffers and, therefore, does not affect the Bt8230. Otherwise, the Jx has the same bus control signals as the CA/CF with the exception of the WAIT* signal, which the Jx does not possess. The insertion of wait states, if required, must be accomplished by an external memory controller which, in any case, is required for a Jx implementation.

3.3.5 standalone Operation

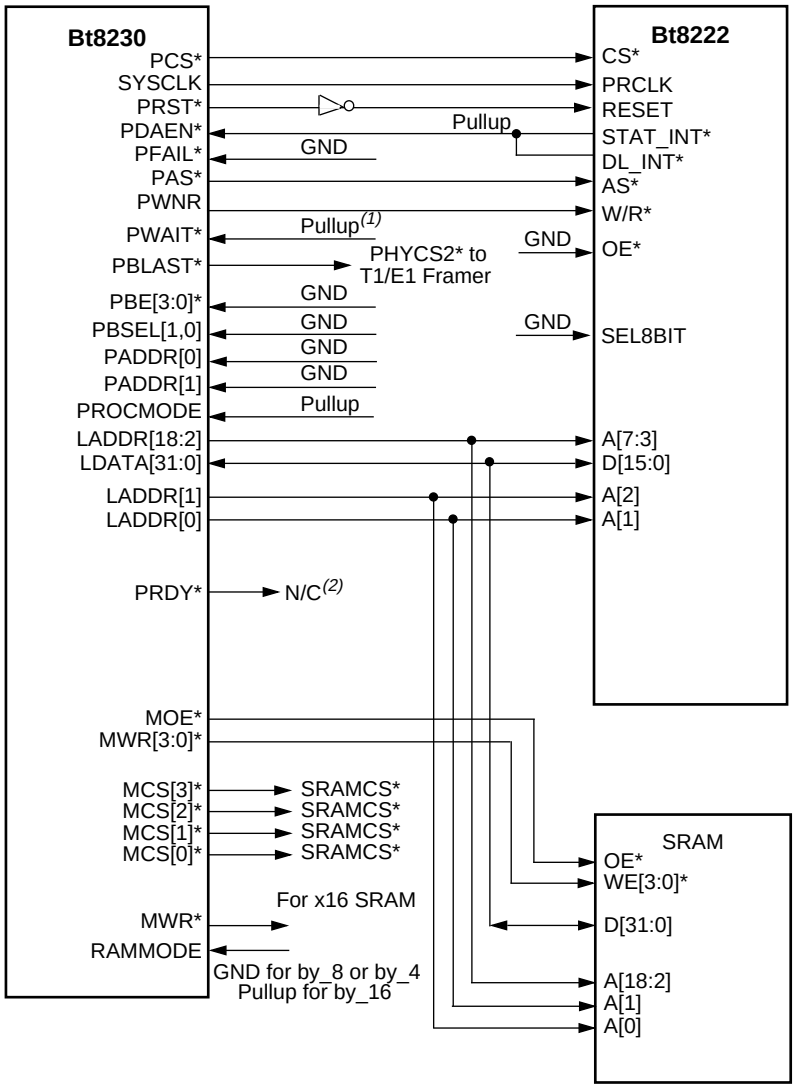
When the local processor is not used, the Bt8230 is said to be in standalone mode. Standalone interface pins and descriptions are given in Table 3-3. Figure 3-16 shows the signal interface between the Bt8230 and the Bt8222 ATM receiver/transmitter device with no local processor. The PCS*, PAS*, and PWNR pins are now outputs providing chip select, address strobe, and write/read control to the Bt8222. PDAEN* is now an input connected to the interrupt sources of the Bt8222. PBLAST* is a second chip select which can be used to connect the Bt8370 T1/E1 framer Line Interface Unit (LIU) since the Bt8222 does not contain the LIU function. The PRDY* output is active and indicates the cycles in which the data transaction occurs. The PWAIT* input is active and can be used to prolong the cycle as shown in Figure 3-17. Physical interface devices other than the Bt8222 can be connected by using PWAIT* to extend the read or write cycle and by using external logic to translate the Bt8230 control signals.

Table 3-3. standalone Interface Pins

Signal	Dir(1)	Description
PROCMODE	I	Processor interface mode select input. A logic 1 enables standalone operation without a local processor.
PCS*	O	Chip select output for PHY device number 1. Synchronous to SYSCLK. See Figure 3-16.
PBLAST*	O	Chip select output for PHY device number 2. Synchronous to SYSCLK. See Figure 3-16.
PAS*	O	PHY address strobe. Synchronous to SYSCLK.
PWNR	O	PHY write/read select. A logic 1 on this output indicates a write cycle, a logic 0 indicates a read cycle. Synchronous to SYSCLK.
PRDY*	O	PHY interface ready signal. A logic low on this signal at rising edge of SYSCLK indicates that the data cycle has been completed
PWAIT*	I	PHY wait input. Allows external logic to insert wait states to extend data cycles. Only active when PRDY* is active.
PDAEN*	I	PHY interrupt input, active low, level sensitive(2).
PADDR[1,0]	I	Not used, pull to logic 0.
PBSEL[1,0]	I	Not used, pull to logic 0.
PBE[3:0]*	I	Not used, pull to logic 0.

Notes: (1). Direction given with respect to the Bt8230.
(2). See the HOST_ISTAT0 register for details.

Figure 3-16. Bt8230 to the 8222 Microprocessor Interface (standalone Operation)



Notes: (1). Can be driven by external circuitry to extend cycles.
(2). Can be used by external circuitry.

Figure 3-17. Bt8230/PHY Functional Timing with Inserted Wait States

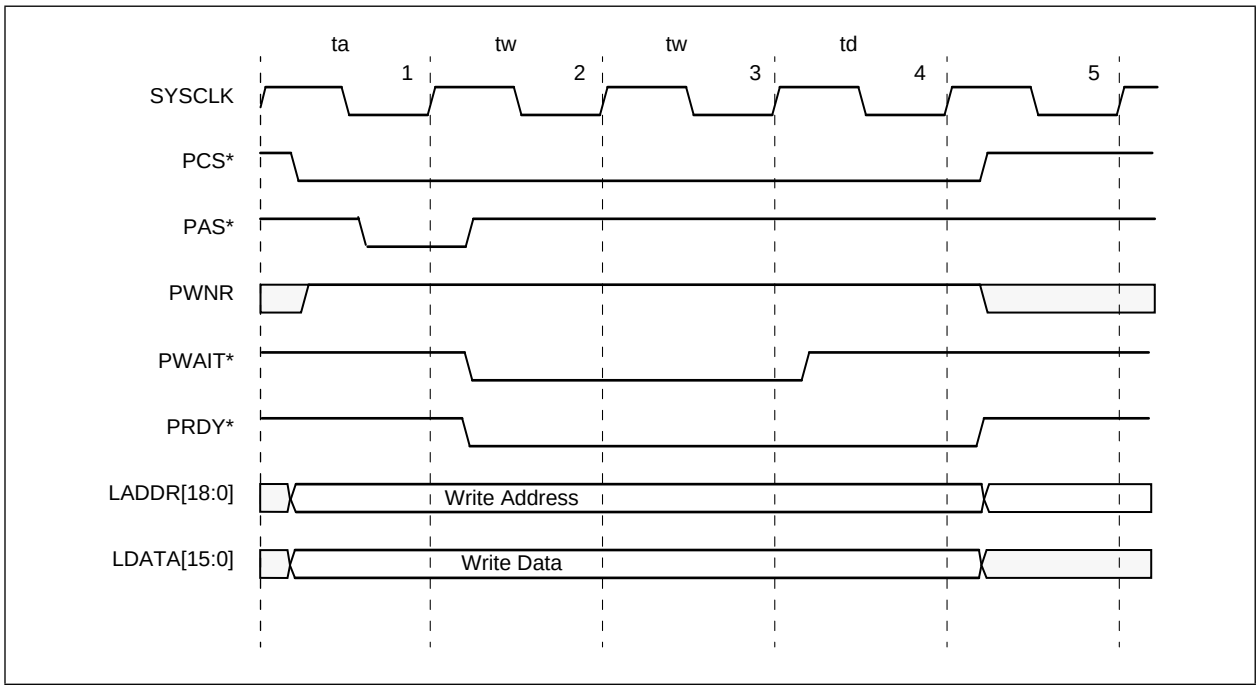
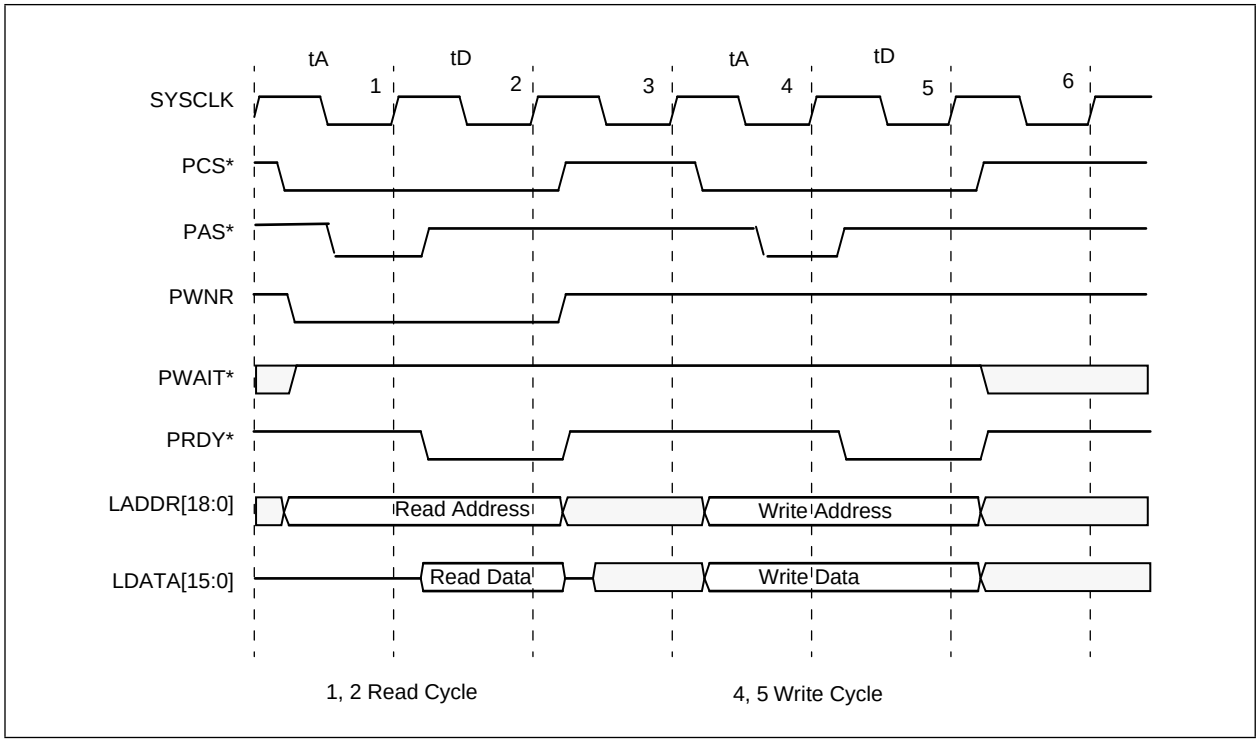


Figure 3-18 shows a read and write Bt8222 access. At cycle 1 (the rising edge of SYSCLK) the Bt8222 samples PCS*, PAS*, and PWNR low, indicating a read cycle. By the next rising edge of SYSCLK at cycle 2, the data is output by the Bt8222 to be latched by the Bt8230. The same procedure occurs for a write except that at cycle 4, PWNR is sampled high. The Bt8222 then latches the data to the appropriate internal register on the next SYSCLK rising edge at cycle 5.

Figure 3-18. Bt8230/Bt8222 Read/Write Functional Timing

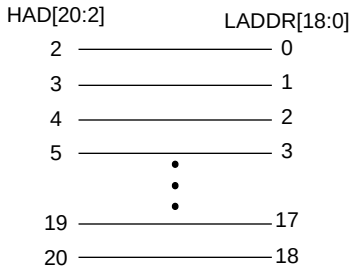


3.3.5.1 Address Mapping in standalone Mode

Because all memory accesses to the Bt8230 are 32-bit word addresses, the Bt8230 must provide address bit mapping to accommodate by_8 and by_16 devices that are connected via the PHYCS1* and PHYCS2* device interfaces.

When using the address space provided by either of these interfaces, mapping takes place between a PCI address and a LADDR (device) address, as shown in Figure 3-19.

Figure 3-19. PCI and LADDR Mapping



For example, using the PHYCS1* address space, address 0x000200h would appear on LADDR[7:0] as 0x80. Care must be taken in this mode to ensure proper byte lane mapping between the attached device and the processor.

3.3.6 System Clocking

The Bt8230 derives all of its timing from a 2x clock input, CLK2X. This clock is internally divided by 2 to create the system clock, SYSCLK. This system clock is used internal to the device and is output to the system to provide the clock to an external processor or PHY device. All processor interface signals are synchronous to SYSCLK. In addition, a CLKD3 (CLK2X asymmetrically divided by 3) output is provided and can be used as the clock for the UTOPIA ATM physical interface. Alternatively, SYSCLK can be used as the clock source for the UTOPIA ATM physical interface if the frequency is 25 MHz or less. In either case, the clock signal would be looped externally to the FRCTRL input. For example, if CLK2X is 66 MHz, then CLKD3 is 22 MHz, which is suitable for the UTOPIA interface. If CLK2X is 50 MHz, then SYSCLK is 25 MHz, which is suitable for the UTOPIA interface. The CLK2X frequency required for a given application is a function of the physical line rate, number of VCCs, active concurrent VCCs, and the SRAM cycle time.

3.3.7 Real-Time Clock Alarm

A real-time clock counter and alarm registers are built into the Bt8230. This real-time clock consists simply of a 7-bit prescaler (configured via the DIVIDER field in the CONFIG0 register). This register accepts the SYSCLK input and outputs a constant (nominally 1 MHz) pulse train and a 32-bit read/write counter (the Real Time Clock Register [CLOCK; 0x00]). The counter counts the number of pulses output by the prescaler since the system was initialized. When the prescaler is set to generate a 1 MHz pulse train, the CLOCK counter counts in 1 µs intervals. An interrupt is generated when the CLOCK counter overflows; i.e., more than 2^{32} pulses have occurred since it was cleared to 0. If this happens, the CLOCK counter simply wraps around to 0 and starts counting over. The control processor or host software is responsible for noting the overflow.

One simple real-time alarm is implemented in the Bt8230. This consists of the Alarm Register 1 [ALARM1; 0x04] which is continuously compared to the Clock Register [CLOCK; 0x00]. When a match is detected, the corresponding interrupt is generated to the local processor. The local processor can then respond to this interrupt and reload a new value into the ALARM1 register.

3.3.8 Bt8230 Reset

The Bt8230 must be reset by the host processor prior to system initialization for proper operation. This can be done in one of two ways: by asserting the external HRST* pin (which is normally connected to the system power-up reset circuitry), or by setting the GLOBAL_RESET [bit 30] in the Configuration Register 0 [CONFIG0; 0x14]. The HRST* pin must be deasserted, and GLOBAL_RESET must be cleared before beginning the Bt8230 initialization process.

Asserting the HRST* input pin automatically causes the local processor reset pin to be driven active, resetting the local processor. The reset to the local processor will stay active until LP_ENABLE [bit 31] in the CONFIG0 register is set to a logic high. When using the GLOBAL_RESET bit, the processor must manually set or clear the appropriate bits in the CONFIG0 register.

NOTE: The segmentation and reassembly coprocessors should be disabled when reset occurs. Do not enable these coprocessors until control structures are initialized.

3.4 Segmentation Coprocessor

The segmentation coprocessor is responsible for segmenting host or local data buffers into ATM cells for transmission. For each ATM cell time slot, an internal scheduler determines if there is a connection that should send a cell or, optionally, insert an idle cell. The payload for a cell can be read from host memory across the PCI bus or from local Bt8230 memory. The segmentation coprocessor adds the ATM header and any programmed adaptation layer formatting. The cell is then added to the transmit FIFO for eventual transmission. The segmentation coprocessor is capable of generating all CPCS-PDU overhead for both AAL3/4 and AAL5.

Most segmentation traffic should originate in host memory buffers. The option to segment from a Bt8230 local memory buffer is intended to allow a local processor to generate maintenance and signaling messages in local memory.

The major components of the segmentation coprocessor are:

- A segmentation controller that performs the segmentation and cell multiplexing processing required by the AAL3/4 and AAL5 and ATM protocol layers, creating interleaved streams of segmentation and reassembly (SAR) PDUs by segmenting multiple CPCS PDUs.
- A 128-word transmit FIFO that accepts and buffers segmented cells prior to transmission to the framer.
- A CRC-10 generator responsible for computing a 10-bit Cycle Redundancy Check (CRC) over the 48-byte payload field of each transmitted cell. This CRC is used for the AAL3/4 and Operation and Maintenance (OAM) SAR-PDU CRC.
- A CRC-32 generator responsible for computing a 32-bit CRC over the 48-byte payload field of each transmitted cell. This CRC is used for the AAL5 CPCS-PDU CRC.

A flowchart of the internal hardware process is shown in Figure 3-20. A flow chart of the host interaction with the segmentation coprocessor is shown in Figure 3-21.

Each VCC can send cells in VBR mode or in a UBR mode. The parameters for each VBR VCC are individually specified in terms of the ATM Forum UNI 3.1 Specification, Generic Cell Rate Algorithm (GCRA) (See “Traffic Scheduling” on page 23.). This algorithm specifies a rate in terms of two parameters: *I* and *L*. *I* is the average intercell interval for the VCC. *L* specifies how early a VCC may send a cell relative to the intercell interval *I*. Higher values of *L* allow greater intercell variation and increased burstiness of the traffic. The traffic management method used in the segmentation coprocessor will use the allowed intercell variation in *L* only as required to multiplex multiple VCCs. Setting a high value for *L* will not increase the burstiness of a VCC unless the VCC is multiplexed with other VCCs. It required, the values of *I* and *L* for each VCC can be changed during segmentation.

All VCCs that are specified as UBR connections share the bandwidth that is not allocated by the VBR VCCs. The UBR VCCs are maintained on a circular queue. The VCC at the top of the queue will transmit an ATM cell during each cell interval that is not used by any VBR VCC. The VCC at the top of the queue is then placed at the bottom of the queue.

The segmentation coprocessor reads the payload data for each VCC using a gather DMA. This allows the data for each VCC to be located in multiple noncontiguous buffers, either in host or local memory. Each buffer can be any size up to 65535 bytes, and a cell payload can cross a buffer boundary.

Figure 3-20. Bt8230 Transmit Hardware Flow Chart

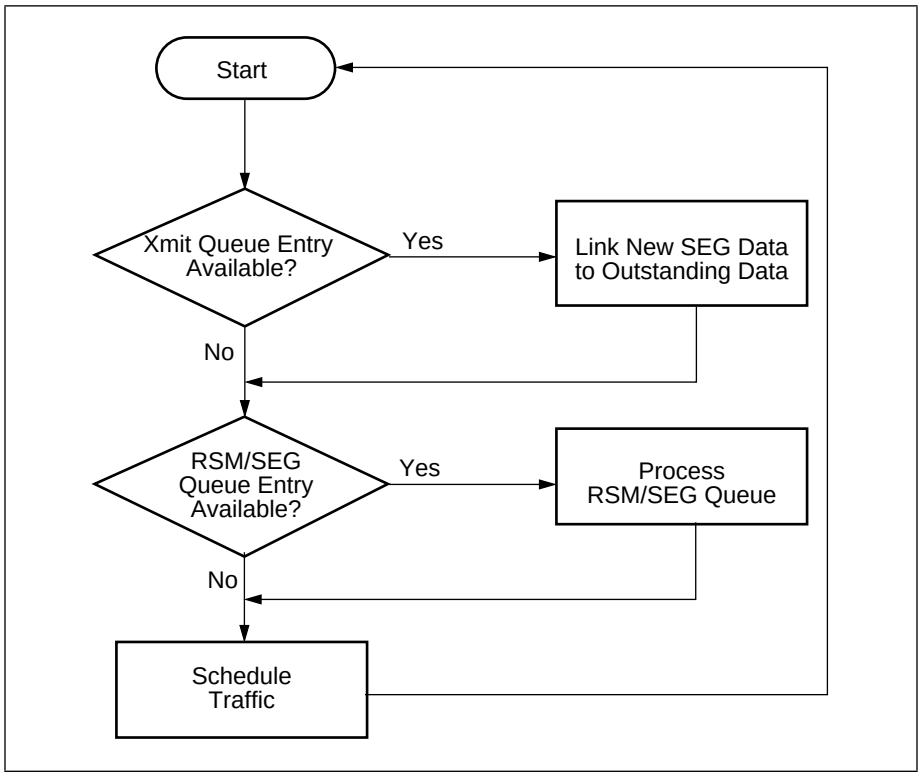
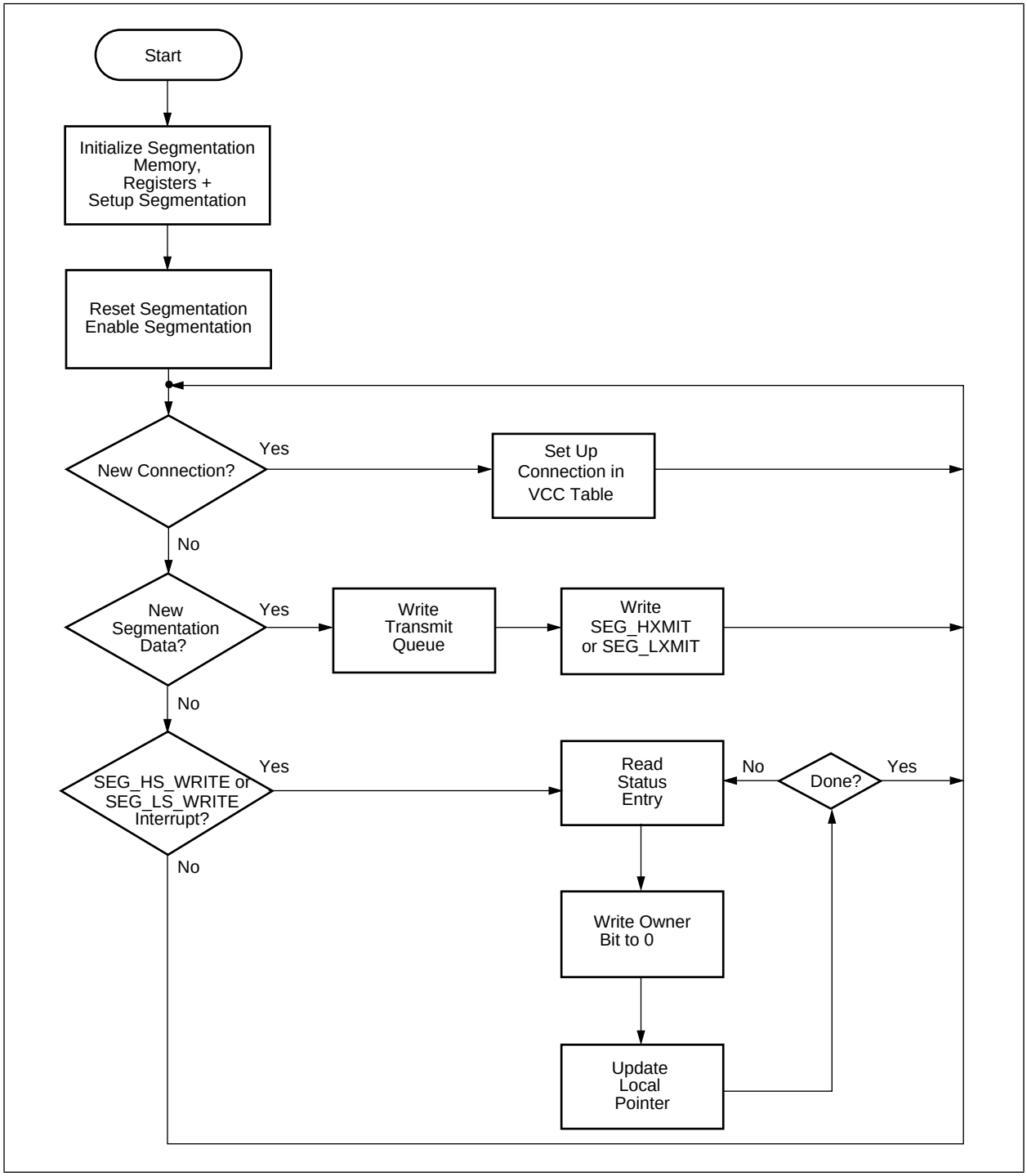


Figure 3-21. Bt8230 Transmit Software Flow Chart



3.4.1 Memory Setup

The segmentation coprocessor uses seven areas of local memory to store segmentation state information:

1. Host Status Queue area
2. PM state area
3. Local Status Queue area
4. Reserved area
5. VCC Table area
6. Buffer Descriptor area
7. Local Transmit Queue

Each area can be located anywhere in the memory space. The base address for each area except the buffer descriptor area is set by the user in a register. Each buffer descriptor is located individually with a separate pointer in another buffer descriptor or in the VCC table. The size of each memory area is determined by the maximum limits set in the Segmentation Control Register [SEG_CTRL; 0x20]. The base pointer, size, initialization required, and purpose of each memory area are shown in Table 3-4.

The user must set the segmentation size limits in the SEG_CTRL register. This fixes the size of the host status queue, the local status queue, the VCC table area, and reserved area. Bt8230 local memory for each of these areas must be allocated, and the Segmentation Status Queue Base [SEG_SBASE; 0x24] and Segmentation Virtual Connection Base Register [SEG_VBASE; 0x28] registers must be set with the corresponding base addresses. Each segmentation descriptor can be allocated anywhere in unused Bt8230 local memory. Buffer descriptors are discussed further in subsection 3.4.5.

Table 3-4. Bt8230 Segmentation Local Memory Configuration

Base Pointer	Description
Host Status Queue	
SEG_HSBASE[15:0] field in SEG_SBASE register	Size: (MAXHS * 8 bytes) + <Max number of PM channels> * 16 bytes Initialization: Set to all 0. Purpose: Indicates to the host that a host buffer is completely segmented. PM words are maintained at the top of the queue.
Local Status Queue	
SEG_LSBASE[15:0] field in SEG_SBASE register	Size: MAXLS * 8 bytes Initialization: Set to all 0. Purpose: Indicates to the local processor that a local buffer is completely segmented.
Reserved Segmentation Area	
SEG_RSVDB[15:0] field in SEG_VBASE register	Size: (MAXI + MAXPND) * 4 bytes Initialization: Set to 0xFFFFFFFF. Purpose: Internal state information for segmentation.
VCC Table	
SEG_VCCB[15:0] field in SEG_VBASE register	Size: <Number of VCCs> * 32 bytes Maximum size is 65535 * 32 bytes Size can change as VCCs are added and removed. Initialization: Each VCC as described in subsection 3.4.3. Purpose: Stores state and rate information for each VCC.
Buffer Descriptors	
NXT_FREE[20:0] field in SEG_FR_BD register	Size: <Number of active descriptors> * 16 bytes Size can change as buffers are added and removed. Initialization: Each descriptor as described in subsection 3.4.5. Purpose: Stores size, location, and segmentation options for each active segmentation buffer.
Local Transmit Queue	
SEG_LRBASE[15:0] field in SEG_LRBASE register	Size: MAXLR * 16 bytes Size can change as buffers are added and removed. Initialization: Each descriptor as described in subsection 3.4.5. Purpose: Stores size, location, and segmentation options for each active segmentation buffer.

3.4.2 Initialization of Buffer Descriptors

As part of the Bt8230 initialization, local memory for buffer descriptors must be designated. Each active transmit buffer requires a buffer descriptor stored in local memory. Therefore, the segmentation descriptors space must be large enough to hold the maximum expected number of active buffers. At initialization, all buffer descriptors must be linked together into a single chain by using the NEXT_PTR field in the buffer descriptors. The NEXT_PTR field of the last buffer descriptor must be set to 0. The NXT_FREE[20:0] field in the Segmentation Free Buffer Descriptor Register [SEG_FR_BD; 0x34] is set to the address of the first buffer descriptor in the chain. Table 3-5 provides the format of each buffer descriptor. Table 3-6 lists the individual field descriptions.

Table 3-5. Buffer Descriptor Format

<table>
<tr><td colspan="5">BUFF_ADDR (32)</td></tr>
<tr><td colspan="4">BASIZE(16)</td><td rowspan="2">VCC_INDEX (16)</td></tr>
<tr><td>Rsvd (5)</td><td>VCI_DATA (4)</td><td>PTI_DATA (3)</td><td>GFC_DATA (4)</td></tr>
<tr><td colspan="4">VCC_CTRL (16)</td><td>LENGTH (16)</td></tr>
<tr><td>Rsvd (9)</td><td colspan="3">NEXT_PTR (21)</td><td>Rsvd (2)</td></tr>
<tr><td colspan="5">Note: The gray line is a word selector. The word above is used when the buffer contains an AAL3/4 BOM with GEN_PDU bit active; otherwise, the word below the line is used. Both words are never used at the same time.</td></tr>
</table>

Table 3-6. Buffer Descriptor Format Field Descriptions *(1 of 2)*

Name	Description
BUFF_ADDR	Address in the host or local memory space of beginning of segmentation buffer. Host or local source is determined by the LOCAL option (VCC_CTRL[14]) in the VCC_CTRL field.
BASIZE	Used for the BAsize field in the AAL3/4 header when the GEN_PDU option is selected (VCC_CTRL[5]).
VCC_INDEX	Identifies the VCC in the VCC table.
VCI_DATA	Data for WR_VCI option (VCC_CTRL[10]). Not used for AAL3/4 BOM buffer when GEN_PDU bit is active. Normally used to generate OAM cells.
PTI_DATA	Data for WR_PTI option (VCC_CTRL[11]). Not used for AAL3/4 BOM buffer when GEN_PDU bit is active. Normally used to generate OAM cells.
GFC_DATA	Data for WR_GFC option (VCC_CTRL[12]). Not used for AAL3/4 BOM buffer when GEN_PDU bit is active.

Table 3-6. Buffer Descriptor Format Field Descriptions *(2 of 2)*

Name	Description
VCC_CTRL	Segmentation options for the buffer. The VCC_CTRL field controls segmentation options on a per-buffer basis. These options should only be changed at a cell or CPCS-PDU boundary. The options are: VCC_CTRL[15] FAST_START—When data is available for VCC, send cell as soon as possible. This is useful for transmission of latency-sensitive traffic such as AAL0. GCRA mode VCCs will send cell immediately instead of waiting for rate parameter/time to elapse before sending first cell. UBR VCCs will be placed to the top instead of the bottom of the UBR queue only if VCC is not currently in queue. VCC_CTRL[14] LOCAL—Buffer address is a buffer in Bt8230 local memory instead of host memory. Local buffers must begin on word-aligned addresses. VCC_CTRL[13] SET_CI—Sets the middle bit of the ATM header PTI field for all cells. VCC_CTRL[12] WR_GFC—Overwrites the ATM header GFC field for all cells with GFC_DATA. Global GFC changes (active for all buffers of VCC) can be set in VCC structure ATM header. VCC_CTRL[11] WR_PTI—Overwrites the ATM header PTI field for all cells with PTI_DATA. Used to generate F5 and RM OAM cells. OAM cells are not included in PM TUC or BIP16 calculations. VCC_CTRL[10] WR_VCI—Sets the ATM header VCI value for all cells to VCI_DATA (MSBs of VCI are set to 0). Used to generate F4 OAM cells. OAM cells are not included in PM TUC or BIP16 calculations. VCC_CTRL[9] SET_CLP—Sets the ATM header CLP bit for all cells to 1. VCC_CTRL[8] CELL—Read entire 52-octet ATM cell from segmentation buffer. The ATM_HEADER stored in the VCC structure (Table 3-7) is not used in this mode. VCC_CTRL[7] ABORT—Generates an abort cell and end processing of descriptor. VCC_CTRL[6] STM_MODE—When set high, a SEG_xS_WRITE interrupt occurs on each buffer. When low, the interrupt occurs on PDU boundaries. VCC_CTRL[5] GEN_PDU—Generates AAL3/4 header and trailer and AAL5 trailer. Setting this bit on an AAL5 cell will also enable the CRC32 calculation. VCC_CTRL[4] CRC_10—Overwrites last 10 bits of a cell with CRC10 calculation. VCC_CTRL[3] AAL3/4—Selects AAL3/4 operation. AAL5 operation is used if not set. This option also allows OAM cells by selecting AAL5 operation. VCC_CTRL[2] Reserved—Set to 0. VCC_CTRL[1] BOM—Buffer contains beginning of message (AAL3/4 only). VCC_CTRL[0] EOM—Buffer contains end of message.
LENGTH	Number of bytes of data contained in the buffer. For local buffers, if the EOM bit (VCC_CTRL[0]) is not set, this length must be set to an integral multiple of 4 bytes. Host buffers can be on non-modulo 4 length. This location is not changed by the Bt8230.
NEXT_PTR	Pointer to next descriptor for the VCC. The two LSBs of the pointer are assumed to be 0 (word-aligned).

3.4.3 Initializing Segmentation VCCs

State information for each segmentation VCC is kept in a VCC structure entry in the segmentation VCC table. Each connection is assigned a 16-bit VCC index by the user when the connection is set up. This index, which begins at 0, allocates the VCC structure in the VCC table for a particular connection. The VCC structure for a connection is at address (SEG_VCCB * 128) + (VCC_INDEX * 32).

To set up a segmentation VCC, choose an unused VCC index and write the appropriate fields in the VCC structure. Table 3-7 gives the 32-byte VCC structure. Table 3-8 lists the VCC structure field descriptions.

Table 3-7. Segmentation VCC Structure

RUN (1)	Rsvd(1)	RATE_MODE (1)	Rsvd(1)	PM (7)	CURR_DESCR (21)
UU (8)			Rsvd(3)		LAST_DESCR (21)
ATM_HEADER (32)					
CRC (32)					
CPI (8)		BETAG (8)	ST (2)	SN (4)	MID(10)
PDU_LEN (16)				BUFFER_LEN (16)	
L_HIGH (8)			I (24)		
Rsvd (16)				L_LOW (16)	
Rsvd (32)					

Note: The gray line is a word selector. In AAL5, the word above the gray line is used. In AAL3/4, the word below the gray line is used. Both words are never used at the same time.

Table 3-8. Segmentation VCC Structure Field Descriptions

Field Name	Description
RUN	A flag indicating that segmentation is proceeding. This flag is set to 1 by the Bt8230 when the host supplies data for the VCC; it is set to 0 by the Bt8230 when there is no data available for the VCC.
RATE_MODE	Controls transmit rate-shaping mode 0 GCRA—Transmitted traffic conforms to GCRA(I,L) 1 UBR—VCC sends cells during available slots.
PM	Performance Monitoring index. 127 VCCs can be selected for automatic OAM PM generation. Each monitored VCC has a unique performance monitoring index. An index of 0x7F indicates that the performance monitoring is not enabled for the VCC.
CURR_DESCR	Pointer to the current buffer descriptor for the VCC. This field is automatically updated by the Bt8230. The two LSBs of the pointer are assumed to be 0 (word-aligned).
UU	AAL5 user-to-user indication.
LAST_DESCR	Pointer to the last buffer descriptor for the VCC. This field is automatically updated by the Bt8230. The two least significant bits of the pointer are assumed to be 0 (word-aligned).
ATM_HEADER	Used for each ATM cell for the VCC. The transmitted header may be modified by option bits in the current buffer descriptor.
CRC	Cyclic Redundancy Check. For AAL5 VCCs, this field holds the accumulated value of the 32-bit CRC. For AAL3/4 this word contains the CPI, BETAG, ST, SN, and MID fields as described below.
CPI	AAL3/4 Common Part Indicator (CPI) field.
BETAG	AAL3/4 BTag and ETag field.
ST	AAL3/4 Segment Type field.
SN	AAL3/4 Sequence Number field.
MID	AAL3/4 Message ID field.
PDU_LEN	CPCS-PDU Trailer Length field. This field is generated by the Bt8230.
BUFFER_LEN	Buffer Length. Number of bytes of data read from the host or local segmentation buffer.
L_HIGH	*L* Field High. Eight MSBs of the rate control *L* field.
I	Rate Control *I* Field. This field is the corresponding parameter in the ATM UNI 3.1 GCRA specification (Section 3.6.2.4.1). This parameter is a fixed point number with 10 fractional bits and is in units of cell slots.
L_LOW	*L* Field Low. The 16 LSBs of the rate control *L* field. This field is the corresponding parameter in the ATM UNI 3.1 GCRA specification (Section 3.6.2.4.1). This parameter is a fixed-point number with 10 fractional bits and is in units of cell slots.

To set up a connection, write the following fields:
For all VCCs:
- ATM_HEADER
- Write all Rsvd fields to 0
- Set RATE_MODE for desired transmit rate shaping
- Set RUN to 0
- Set PM field to pm index or 0x7F to disable
- Set CURR_DESCR to 0
- Set Word 7 Rsvd field (next to L_LOW) to 0xFFFF
- Set BUFFER_LEN to 0
- Set PDU_LEN to 0

For rate-controlled VCCs:
- I
- L_HIGH, L_LOW
- (For more information on I and L, see Section 2.4.2 and Section 3.4.7.)

For AAL3/4 VCCs:
- CPI
- MID
- SN (if specific beginning value desired)

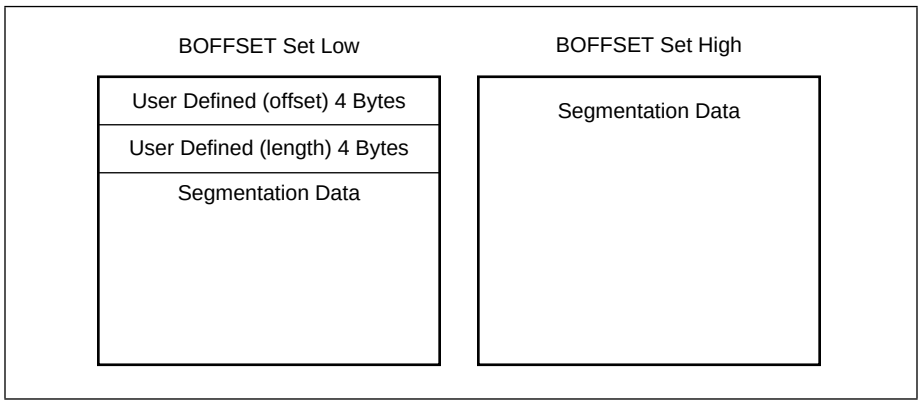
For AAL5 VCCs:
- UU
- CRC to 0xFFFF_FFFF

All other fields in the VCC structure are generated by the Bt8230.

3.4.4 Segmentation Buffer Format

The segmentation buffers consist of two user-defined words followed by the segmentation data as shown in Figure 3-22 if bit BOFFSET is 0 in the SEG_CTRL register. The two user-defined words are to be used for buffer management and to contain the offset to the first word of data and the total length of the buffer. The actual data contained in the buffer might be less than the total size of the buffer, and the number of bytes of data is given by the length field in the buffer descriptor. When the BOFFSET bit in the Segmentation Control Register [SEG_CTRL; 0x20] is a logic 1, no user-defined space is available in the buffers.

Figure 3-22. Segmentation Buffer Format



3.4.5 Adding Segmentation Data

Once a VCC structure has been initialized, segmentation data can be supplied to the VCC. The Bt8230 is able to perform an intelligent DMA operation to gather the data for a single connection from noncontiguous areas in host or local memory. This is accomplished by the use of segmentation transmit rings. There are two transmit rings: the Host Transmit Queue, located in host memory and the Local Transmit Queue, located in local memory. Each entry on a transmit queue is 16 bytes and has the same format as a segmentation buffer descriptor. The format of each transmit queue descriptor is given in Table 3-9.

Table 3-9. Transmit Queue Descriptor Format

<table>
<tr><td colspan="5">BUFF_ADDR (32)</td></tr>
<tr><td colspan="4">BASIZE(16)</td><td rowspan="2">VCC_INDEX (16)</td></tr>
<tr><td>Rsvd (5)</td><td>VCI_DATA (4)</td><td>PTI_DATA (3)</td><td>GFC_DATA (4)</td></tr>
<tr><td colspan="4">VCC_CTRL (16)</td><td>LENGTH (16)</td></tr>
<tr><td colspan="5">User Defined (32)</td></tr>
<tr><td colspan="5">Note: The gray line is a half-word selector. The half-word above is used when the buffer contains an AAL3/4 BOM with GEN_PDU bit active; otherwise, the half-word below the line is used. Both half-words are never used at the same time.</td></tr>
</table>

Bit mapping for host transmit descriptors is the same in big and little endian modes. In addition, the entries must be word (32 bits) aligned. To queue one or more buffers, the host writes the first three words of a transmit queue entry with the buffer address and control information for a segmentation buffer. The last word of each transmit queue entry is not used by the Bt8230. The host writes the queue entries in ascending order and restarts at the beginning of the queue after the end of the queue is reached. To initiate processing of one or more queued entries, the host writes the SEG_HNEW[7:0] field in the Host Transmit Queue Register [SEG_HXMIT; 0x40] with the number of queued entries. When the Bt8230 processes a transmit entry, it copies the information from the transmit queue entry into a buffer descriptor stored in Bt8230 local memory. Once processed, the transmit queue entry can be reused by the host, even before the corresponding segmentation buffer has been processed by the Bt8230. The number of transmit queue entries that are pending processing is given by the SEG_HPND[13:0] field in the SEG_HXMIT register.

The operation of the Local Transmit Queue is analogous to the Host Transmit Queue except that the queue is located in Bt8230 local memory, and the Local Transmit Queue Register [SEG_LXMIT; 0x44] is used instead of the SEG_HXMIT register. The Local Transmit Queue is provided to allow easy message insertion by the local processor. Typically, these messages are local (segmented from local memory) signaling and OAM messages. The WR_PTI (VCC_CTRL[11]) and WR_VCI (VCC_CTRL[10]) control bits in the transmit queue VCC_CTRL field can be used to multiplex OAM cells with data traffic on a single segmentation VCC structure.

If these options are used, the OAM buffers must be inserted at CPCS-PDU boundaries in the data traffic. This can be accomplished by either having each CPCS-PDU contained within a single buffer, or by placing OAM buffers only after CPCS-PDU buffers which contain the end of a PDU. When the CPCS-PDU

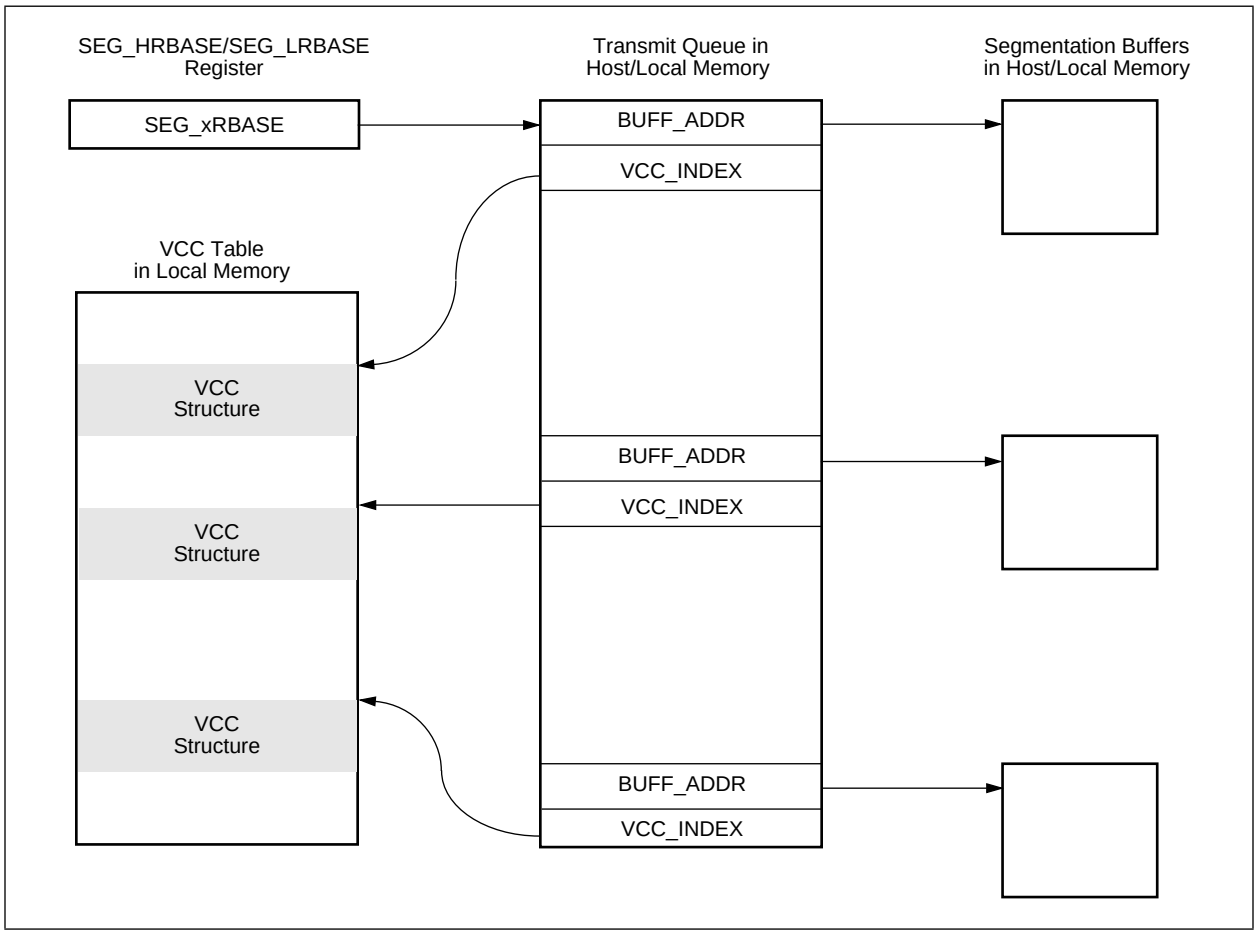
buffers reside on the Host and use the Host Transmit Queue, and the OAM buffers reside in the local memory and use the Local Transmit Queue, SEG_XMT_ALT in the SEG_CTRL register can be used to ensure that the OAM buffers are inserted only at the end of a CPCS-PDU. The Host should write (and notify the Bt8230 via the SEG_HNEW[7:0] field) the Host Transmit Queue only with complete CPCS-PDUs. Each PDU can comprise more than one buffer, but all buffers for the PDU should be added at once. With the SEG_XMT_ALT bit set to 0, the Bt8230 will process the local transmit queue with the OAM buffers only when the Host Transmit Queue is empty (at the end of a PDU). Processing a transmit queue entry does not mean that the buffer has completed segmentation, only that the buffer information has been added to the local control information.

OAM buffers can also be added on a separate VCC structure from the data traffic. This alleviates any need to place the OAM buffer on CPCS-PDU boundaries at the cost of an additional VCC structure in memory. A single VCC structure can be used for OAM cells on all VCCs by using the CELL segmentation mode where the ATM header is included in the buffer data. If OAM cells use a separate VCC structure from data traffic, the SEG_XMT_ALT bit can be set to one. This causes the Bt8230 to service the Host and Local Transmit Rings alternately and can provide lower latency service for the Local Transmit Queue.

The Bt8230 cannot prevent the host or local processor from overwriting transmit queue entries that have not been serviced by the Bt8230. If the SEG_HNEW[7:0] or SEG_LNEW[7:0] fields are written with a value that would cause the corresponding transmit queue to overflow, the new data is ignored. In addition, HRING_OVFL (for Host Transmit Queue) or LRING_OVFL (for Local Transmit Queue) is set in the Host Interrupt Status Register [HOST_ISTAT0; 0xC0] or Local Processor Interrupt Status Register [LP_STAT0; 0xE0], respectively. These status bits can optionally cause interrupts.

The Bt8230 will not service either transmit queue if there are no free segmentation buffer descriptors. This condition will clear automatically when a buffer completes segmentation and releases the corresponding buffer descriptor. This condition sets the BD_FULL bit in the HOST_ISTAT0 and LP_STAT0 registers and optionally causes an interrupt. Figure 3-23 illustrates how the data structures are linked.

Figure 3-23. Bt8230 Segmentation Linked Buffer Structure



3.4.6 Descriptor Status

When STM_MODE = 1 and the segmentation coprocessor completes segmentation of a buffer, it writes an entry onto a status queue. If the buffer descriptor LOCAL option (VCC_CTRL[14]) is not set, the status entry is written on the host status queue; otherwise, the status entry is written on the local status queue. A buffer is complete when the data is all segmented, an abort cell is sent (due to an ABORT option), or a length error occurs. A length error indicates that the maximum CPCS-PDU length was exceeded, and any additional data in the buffer was discarded. If STM_MODE in the buffer descriptor is a logic 0, a status entry is written only at the completion of a PDU. The status entry points to the last buffer of the PDU. The user must keep a linked list of buffers in order to recover all the buffers of a PDU. The format of both the host and local status queue entries is given in Table 3-10. Field descriptions of these entries are given in Table 3-11.

Table 3-10. Status Queue Entry Format

OWNER (1)	LEN_ERR (1)	STOP (1)	DONE (1)	PDU_DONE (1)	Rsvd (11)	VCC_INDEX (16)
BUFF_ADDR (32)						

Table 3-11. Status Queue Entry Format Field Descriptions

Field Name	Description
OWNER	Indicates that the host or local processor owns the descriptor.
LEN_ERR	Maximum PDU length is exceeded and descriptor aborted.
STOP	VCC has stopped because no more segmentation data is available.
DONE	Set when finished with buffer.
PDU_DONE	Set when finished with CPCS-PDU.
VCC_INDEX	VCC index value of connection.
BUFF_ADDR	Beginning address of completed buffer.

The Bt8230 will read the OWNER bit before writing a status entry. If the OWNER bit is 0, the entry is written with the OWNER bit set to 1. In addition, and the SEG_HS_WRITE (for a host status write), or SEG_LS_WRITE (for a local status write) status bit in the HOST_ISTAT0, or LP_ISTAT0 register is set to optionally cause an interrupt. If the OWNER bit is a one, a status overflow has occurred and Bt8230 segmentation halts until the condition is cleared. This sets the SEG_HS_FULL (for a host status write) or SEG_LS_FULL (for a local status write) status bit in the HOST_ISTAT0 or LP_ISTAT0 register. This overflow condition clears SEG_ENABLE in the SEG_CTRL register and SEG_RUN in the HOST_ISTAT0 and LP_ISTAT0 registers. After the overflow condition has been cleared, the segmentation coprocessor is restarted by setting the SEG_ENABLE bit to one. The host or local processor should write the OWNER bit to 0 after it has read each status entry. The Bt8230 can be set to always write status regardless of the setting of the OWNER bit by the SEG_HS_DIS (for the host status queue) or the SEG_LS_DIS (for the local status queue) in the SEG_CTRL register to 1.

3.4.7 Rate-Controlled Segmentation (VBR)

Rate-controlled VBR is selected by setting the RATE_MODE to GCRA in the VCC structure. All VCCs that have the RATE_MODE set to GCRA are transmitted according to the *I* and *L* parameters in the VCC structure, where *I* is the increment and *L* is the limit. The segmentation coprocessor multiplexes multiple rate-controlled VCCs by scheduling future transmissions for each VCC. The scheduling algorithm accommodates multiple VCCs contending for a single cell transmit opportunity and rates that are not integral numbers of cell slots. For each VCC, the segmentation coprocessor will attempt to send a cell every *I* cell slots. If several VCCs attempt to transmit during the same cell slot, they are sent in the same order as the last transmission for each VCC. The segmentation coprocessor will send cells for a VCC at intervals longer than the *I* parameter for the VCC for one of two reasons: 1) multiple VCCs attempted to send a cell during the same cell slot or 2) the *I* parameter is set to a fractional number of cells, and the segmentation coprocessor transmits the VCC on the next cell slot boundary. In either case, the segmentation coprocessor will decrease the cell interval for subsequent cells to maintain an average cell interval of *I*. The amount that a cell interval can be decreased is determined by the *L* parameter.

The segmentation coprocessor will never send traffic that would violate GCRA (*I, L*). The slowest rate that can be configured is 1/(MAXI –ACT_VBR) where ACT_VBR is the number of actively segmenting VBR channels. The MAXI[1,0] field is in the Segmentation Control Register [SEG_CTRL; 0x20]. For GCRA-only traffic, the MAXPND[1,0] field in the SEG_CTRL register should be set to 00. See section 3.6.2.4.1 of the ATM Forum UNI Specification, Version 3.1 for further details on GCRA.

NOTE: According to UNI 3.1, the GCRA is used to define the relationship between Peak Cell Rate (PCR) and the Cell Delay Variation Tolerance (CDVT). The PCR specifies an upper bound on traffic submitted on an ATM connection. The signaling message, according to ITU Recommendation I.371, defines PCR as cells-per-second in a floating point number with a 9-bit mantissa and a 5-bit exponent.

Since the *I* value is given in cell slot intervals, the PCR must be converted from cells-per-second to cell-slot-intervals. For example:

x = PCR (cells-per-second)
y = maximum payload throughput (cells-per-second)
I (cell-slot-interval) = y / x

The I.371 recommendation also specifies that there are 16,384 possible values for the floating point number (derived from the 14 bits that make up the floating point number). There must be less than a 0.19% difference between adjacent values (granularity).

An example of the VBR setup follows:

Desired Configuration
line rate = STS-3c transport (155.52 Mbps)
desired channel throughput = 10 Mbps
desired maximum burst size = 3 cells
MAXI_schedule_size = 16384 since SEG_CTRL(MAXI) = 11
ACT_VBR = 1000 (This is the number of actively segmenting VBR channels, not necessarily the total number of VBR channels configured.)

I Parameter Calculation

STS-3c payload throughput = 260/270 payload octets x 155.52 Mbps = 149.76 Mbps
I = 149.76 Mbps / 10 Mbps = 14.976

NOTE: This means that on average, one out of every 14.976 cells will contain payload information for this channel.

The *I* value is represented in the Bt8230 as a 24-bit, fixed-point value with a 14-bit integer and a 10-bit fraction. The *I* value is converted to this format by multiplying it by 1024 and discarding the fractional part. The number is then converted to hexidecimal.

SEG_VCC_TABLE(I) = 14.976 x 1024 = 15335.424 => 15335 = 0x3BE7

NOTE: The actual average channel throughput rate will be 149.76 Mbps x (1024 / 15335) = 10.0003 Mbps.

L Parameter Calculation

L = (MBS–1) x (I–1) where MBS is the maximum burst size in cells. (See ATM Forum Specification Version 3.1, Section 3.6.2.4.3.3.)

L = (3–1) (14.976) = 27.952
SEG_VCC_TABLE(L) = 27.952 x 1024 = 28622.848 => 28622 = 0x6FCE
SEG_VCC_TABLE (L_HIGH) = 0x00
SEG_VCC_TABLE (L_LOW) = 0x6FCE

Period-Setting Resolution

The period-setting resolution of the Bt8230 is determined by the fractional part of the internal representation of the *I* value (10 bits). The period. setting resolution determines the granularity of the data rates available. The difference between any adjacent values is the granularity. The following is an example with a payload throughput of 149.76 Mbps and a rate of 74,88 Mbps:
resolution = [1/(payload throughput)] x (1 / 1024)
= 1/149.76 Mbps x (1 / 1024)
= 6.677 nsec x (1 / 1024)
= 6.52 psec
desired data rate = 74.88 Mbps
desired rate / line rate = 149.76 Mbps / 74.88 Mbps = 2
2 x 1024 = 2048
149.76 Mbps x (1024 / 2048) = 74.88 Mbps
149.76 Mbps x (1024 / 2047) = 74.916571 Mbps
149.76 Mbps x (1024 / 2049) = 74.843443 Mbps

The granularity (difference) of these adjacent values is .049%.

Minimum Configurable Throughput

Min_Rate = payload rate / (MAXI_schedule_size – ACT_VBR) = 149.76 Mbps / (16384 – 1000) = 9.73 kbps
= 149.76 Mbps / (16,384 - 1000)
= 9.73 kbps

3.4.8 Unspecified Bit-Rate Segmentation

Setting the RATE_MODE to UBR in the VCC structure causes a VCC to be segmented from the unspecified bit-rate queue maintained internally by the segmentation coprocessor. This option allows “best effort” service. During each cell slot that does not have any rate-controlled (VBR) VCCs that attempt to transmit, a UBR cell is sent. If there is no UBR data to send, an idle cell is transmitted. The UBR queue contains all UBR VCCs that have data ready to segment. The VCC at the top of the queue transmits during the first available cell slot. The VCC is then removed from the top of the queue and placed at the bottom of the queue. The MAXPND field in the SEG_CTRL register needs to be set such that the size is greater than the number of simultaneous, active UBR connections.

The aggregate UBR transmission rate can be controlled with the Segmentation UBR Parameter Register [SEG_UBR; 0x30]. If the UBR_GCRA_EN bit of this register is set, the state of the UBR GCRA is checked before each potential UBR transmission. If the state of the UBR GCRA does not allow a cell to be sent, an idle cell is sent instead. The *I* and *L* parameters for the UBR GCRA are given by the SEG_UBR register. The *I* and *L* parameters are limited to: I + L is ≤ 0x3FFFF.

NOTE: The UBR rate was erroneously referred to as ABR in previous versions of this datasheet. All instances of this error have been corrected. However, Bt8230 register and bit names in software and software documentation still reflect the erroneous ABR title.

3.4.9 Segmentation Restart

A segmentation VCC may halt because the segmentation data has been exhausted. The VCC will automatically resume segmentation when more data is available from the Host or Local Transmit Rings.

3.4.10 OAM, ILMI, and Signaling

OAM, ILMI, and signaling messages are management messages generated by the host. These messages may be generated from Bt8230 memory instead of host memory by setting the LOCAL option (VCC_CTRL[14]) in the buffer descriptor. This can be done on a buffer-by-buffer basis. OAM F5 (Virtual Circuit Level flows) have the same VCI/VPI as normal traffic but a different Payload Type Indicator (PTI) field. This is accomplished by using the WR_PTI field (VCC_CTRL[11]) in the buffer descriptor. OAM F4 (Virtual Path Level flow) cells occur on a specific VCI by using the WR_VCI option (VCC_CTRL[10]) in the buffer descriptor. The CRC_10 option (VCC_CTRL[4]) in the buffer descriptor can be used for both F4 (Virtual Path Level flow) and F5 (Virtual Circuit Level flow) cells to generate the required 10-bit CRC. Signaling and ILMI messages occur on a specific VCI. A separate, VCC data structure can be set up to handle these messages.

OAM PM can be enabled for up to 127 VCCs by setting the PM field in the VCC structure. For each monitored VCC, a block count and Bit Interleaved Parity (BIP)-16 is maintained. A forward monitoring OAM PM cell is automatically inserted at the end of each PM block. A backward-reporting cell is generated whenever the reassembly coprocessor writes a PM entry in the Reassembly/Segmentation (RSM/SEG) Queue. This cell will not include any forward monitoring information. For backward reporting without forward monitoring, set BLOCK_SIZE = 0. A PM table is maintained at the top of the Segmentation Host Status Queue. The format of each entry is shown in Table 3-12.

Table 3-12. PM Table Format

ATM_HEADER (32)	
TUC (16)	BIP (16)
TARGET (16)	BLOCK_SIZE (16)
Rsvd (24)	MSN (8)

The ATM_HEADER is used for all generated PM cells. The Total User Cells (TUC) and BIP fields are used for forward-monitoring and are updated by the Bt8230. TARGET is the TUC at the end of the block. A forward-monitoring cell is generated when the TUC reaches the target. BLOCK_SIZE is the size of each block and is used to update the TARGET field after a forward cell is sent. MSN is the forward-monitoring sequence number.

To set up performance monitoring on a VCC, the host selects an unused PM value (0–126) and initializes the corresponding PM table entry as follows:

- Set ATM_HEADER to the appropriate value.
- Set BIP to all 0s.
- Set TUC to desired value (can use 0).
- Set BLOCK_SIZE to appropriate value.
- Set MSN to desired value (can use 0).
- Set TARGET = TUC + BLOCK_SIZE.
- The reassembly VCC_INDEX and segmentation VCC_INDEX are equal on the PM channel. For AAL3/4 channels, the SEG_VCC_INDEX must equal the RSM_VCC_INDEX value in the RSM_VCC_TABLE pointed to by VCC_PNTR #1 in the respective AAL3/4 reassembly Hash Bucket (see Table 3-17).

For F4 PM flows, the RSM_VCC_INDEX must equal one of the SEG_VCC_INDEX values. The RSM_VCC_INDEX is determined by the special F4 OAM reassembly hash bucket that needs to be configured. For all VCs within the VP group, the PM values in the respective VCC table entries need to be assigned the same value.

Once the table entry has been initialized, performance monitoring processing is started by writing the PM index into the VCC structure. PM processing operates automatically. It is stopped by writing the PM field in the VCC structure to all ones.

3.4.11 Idle Cell Generation

If there are no rate-controlled VCCs that attempt to transmit during a given cell slot, and the UBR queue is empty, the segmentation coprocessor generates an idle cell. If there are no rate-controlled VCCs that attempt to transmit during a given cell slot, and the UBR queue is empty The segmentation coprocessor generates an idle cell. Every byte in an idle cell payload is generated as 0. If the SEG_432 bit in the SEG_CTRL register is set to a logic 1, the idle cell header equals 0x0000_0001 and the payload is 0x6A. This feature may be disabled by setting the INH_IDLE bit of the SEG_CTRL register [0x20] to 1. If idle cell generation is disabled, the Bt8230 cannot guarantee correct cell spacing on VBR channels.

3.4.12 ATM Header Generation

The ATM header for each non-idle ATM cell is constructed by the segmentation coprocessor according to the VCC_CTRL options in the current buffer descriptor and the ATM_HEADER field in the VCC structure. Table 3-13 gives the source of each field in the header. For a given field, later entries in this table have priority over earlier entries.

Table 3-13. Bt8230 Header Source Fields

Field	VCC_CTRL Options	Source
GFC	WR_GFC	ATM_HEADER GFC_DATA
VPI		ATM_HEADER
VCI[15:4]	WR_VCI	ATM_HEADER All 0
VCI[3:0]	WR_VCI	ATM_HEADER VCI_DATA
PT[2]	WR_PTI	ATM_HEADER PTI_DATA
PT[1]	WR_PTI SET_CI	ATM_HEADER PTI_DATA 1
PT[0]	WR_PTI EOM set, AAL3/4 not set	ATM HEADER PTI_DATA 1 on last cell of buffer
CLP	SET_CLP	ATM_HEADER 1
HEC		Generated as all 0s in ATM physical interface.

3.4.13 AAL3/4 SAR–PDU Generation

Setting the AAL3/4 option in a buffer descriptor causes the Bt8230 to generate the AAL3/4 ST, SN, Message Identification (MID), LI, and CRC fields for each ATM cell. Each field is generated as shown in Table 3-14.

Table 3-14. AAL3/4 SAR-PDU Generation

Field	Function
ST	BOM for first cell generated from buffer with BOM option set. EOM for last cell generated from buffer with EOM option set. SSM for cell generated from buffer with BOM and EOM set and a descriptor length field ≤ 44. COM for all other generated cells.
SN	Read from the SN field in the VCC structure. The SN field is incremented module 16 after each use.
MID	Read from the MID field in the VCC structure.
LI	Generated by the segmentation coprocessor.
CRC	Generated as all 0. The CRC10 field may be overwritten with the CRC-10 generator before transmission by setting the CRC_10 option in the buffer descriptor.

On the first cell for a buffer with BOM set in the buffer descriptor, the PDU_LEN field in the VCC structure is reset to 0. The PDU_LEN field is updated for every byte read from the segmentation buffer. On the last generated cell for a buffer with EOM set in the buffer descriptor, the segmentation coprocessor will pad the SAR-PDU payload with 0 to 44 bytes.

Setting the GEN_PDU and AAL3/4 options causes the Bt8230 to generate the AAL3/4 CPCS-PDU header and trailer. On the first cell of a buffer with BOM set, the CPCS-PDU header is generated as the first 4 bytes of the SAR-PDU payload. On the last cell of a buffer with EOM set, the segmentation processor writes an all–0's PAD field after the end of the segmentation buffer data to complement the CPCS-PDU payload to an integral number of 4-byte words. The segmentation coprocessor then adds the 4-byte CPCS-PDU trailer, and 0 fill octets for the remainder of the SAR-PDU payload. The CPCS-PDU trailer is generated in a separate cell if the CPCS-PDU payload plus PAD is an integral multiple of 44 bytes. The Common Part Indicator (CPI) is read from the CPI field in the VCC structure. The AL field is set to all 0s. The BTag and ETag fields are read from the BETAG field in the VCC structure. The BETAG field is incremented after it is used as an Etag field. The BAsize field is read from the BASIZE field in the buffer descriptor. The length field is read from the PDU_LEN field in the VCC structure.

If the PDU length in the VCC structure exceeds the maximum PDU length, the buffer is aborted and a length error is placed on the status queue. The maximum PDU length is:

```
EOM set and GEN_PDU not set 65544
all other cases 65535
```

3.4.14 AAL5 SAR-PDU Generation

If the AAL3/4 option is not set for a buffer descriptor, the SAR-PDU is formatted for AAL5. Following the segmentation of the last cell of a buffer with EOM set, the PDU_LEN and CRC in the VCC structure are reset. The CRC field is reset to all 1s. The PDU length and CRC fields in the VCC structure are updated for each transmitted cell. If the EOM option is set in the buffer descriptor, the SAR-PDU for the last cell is padded with 0s to fill 48 bytes. If the GEN_PDU and EOM option bits are set in the buffer descriptor, an all-zero PAD field and the AAL5 CPCS-PDU trailer are inserted at the end of the data from the buffer. The PAD field is sized so that CPCS-PDU payload plus PAD plus CPCS-PDU trailer is an integral multiple of 48 bytes. The CPCS-PDU trailer will be generated in a separate cell if the length of the CPCS-PDU payload modular 48 is 0 or in the range of 41–47 bytes.

The CPCS-UU field is read from the VCC structure, the CPI is encoded with all 0s, and the length and CRC fields are read from the VCC structure. The CRC field from the VCC structure is 1s complemented before it is used in the CPCS-PDU trailer.

If the PDU length in the VCC structure exceeds the maximum PDU length, the buffer is aborted, and a length error is placed on the status queue. The maximum PDU length is:

```
EOM set and GEN_PDU not set 65568
all other cases 65535
```

3.4.15 AAL0 SAR-PDU Generation

AAL0 cells may be sent as AAL5 cells by setting EOM and not setting the GEN_PDU in each buffer descriptor. This prevents length errors from occurring and allows the data from the host buffer to be segmented unchanged. If a VCC only occasionally has data available, the Fast Start option in the buffer descriptor can be used to transmit the data as soon as it is available. The CELL option in the buffer descriptor can be used to read entire ATM cells (except for the HEC field) from the segmentation buffer.

3.4.16 CRC-10 Generator

The segmentation coprocessor contains a CRC-10 generator that calculates a 10-bit CRC over the first 374 bits of the generated SAR-PDU. The output of the CRC-10 generator will overwrite the final 10 bits of the generated SAR-PDU if the CRC_10 option is set in the current buffer descriptor. This option would normally be set for AAL3/4 VCCs and for OAM cells.

3.4.17 Transmit Cell FIFO

There is a 128-word, transmit PHY, interface FIFO between the segmentation coprocessor and the PHY interface. This FIFO stores up to nine complete, 52-octet cells. It is filled by the segmentation coprocessor and emptied by the PHY. When data is either required by per-channel rate control, or is not available for segmentation, the segmentation coprocessor inserts idle cells into the transmit PHY interface FIFO. The transmit FIFO helps hide the bursty nature of the cell generation and DMA process from the ATM physical interface. It also decouples the ATM physical interface clock from the segmentation coprocessor. The HEC byte must be calculated in the physical device.

The total FIFO depth for segmentation equals the depth of the DMA master read burst FIFO plus the depth of the transmit PHY interface FIFO.

Total Depth = (16 words / 48 bytes per cell) + (128 words / 52 bytes per cell) ≈ 11.2 cells

3.5 Reassembly Coprocessor

The reassembly coprocessor is responsible for processing 52-octet cells from the receive ATM physical interface. It controls the writing of the CPCS payload to host memory and performs all necessary SAR and CPCS checks. The reassembly coprocessor maintains a hash table, hash bucket chains, and VCC tables in local memory to operate on the various CPCS connections. The hash function quickly locates the appropriate reassembly VCC table based on the received VPI/VCI value in the ATM cell header. The hashing process also allows complete freedom in assignment of VCI/VPI values.

The reassembly coprocessor uses an intelligent scatter method to write the payload portion of the ATM cell-to-host memory. It maintains a free-buffer queue and status queue in local memory to control the scatter operation. The free-buffer queue is updated by the host processor to point to available cell buffers in host memory. The status queue is updated by the reassembly coprocessor with information about how the cell buffers are used. A duplicate structure may be maintained to write cell payload data to local memory. This diverts OAM, ILMI, and signaling messages to the local memory. The local processor can then process these messages without affecting the host system or being affected by host system faults. The local processor updates the local free buffer queue to point to local cell buffers and reads the local status queue to access the local cell buffers.

The reassembly coprocessor can perform reassembly checks on AAL5, AAL3/4, AAL0, and OAM cells. For each type of cell, specific SAR and CPCS checks are performed, and the results are signaled to the appropriate processor. For an AAL5 connection, only CPCS checks, including the CRC32 check, are performed. For an OAM cell, only the CRC10 check is performed. AAL3/4 traffic is checked for various SAR and CPCS protocols, including CRC10. AAL0 traffic has no checks.

The reassembly coprocessor consists of the following subunits:

- VPI/VCI Hash Function—Quickly locates connection information stored in local memory. This method uses less memory than a straight index approach while still supporting the complete VPI/VCI address space.
- AAL Reassembly Processor—Performs required SAR-PDU and CPCS-PDU checks. SAR-PDU checks are performed on AAL3/4 and OAM cells and CPCS-PDU checks are performed for AAL3/4 and AAL5 connections.
- Transfer cell—Scatters cell payload data to host memory cell buffers through the DMA block or to local memory cell buffers. Separate free buffer queues are maintained to allocate cell buffers in host and local memory.
- Status Processing Unit—Assembles the appropriate status entry and writes it to local memory. Separate status queues are maintained to correspond with cell buffers in host and local memory.
- 64-word Receive Cell FIFO—Accepts and buffers incoming cells from the ATM physical interface prior to processing by the reassembly coprocessor.

A flow chart of the internal hardware process is shown in Figure 3-24. Various internal control registers and structures in local memory are maintained by the reassembly coprocessor to perform the external software process shown in Figure 3-25.

Figure 3-24. Bt8230 Receive Hardware Flow Chart

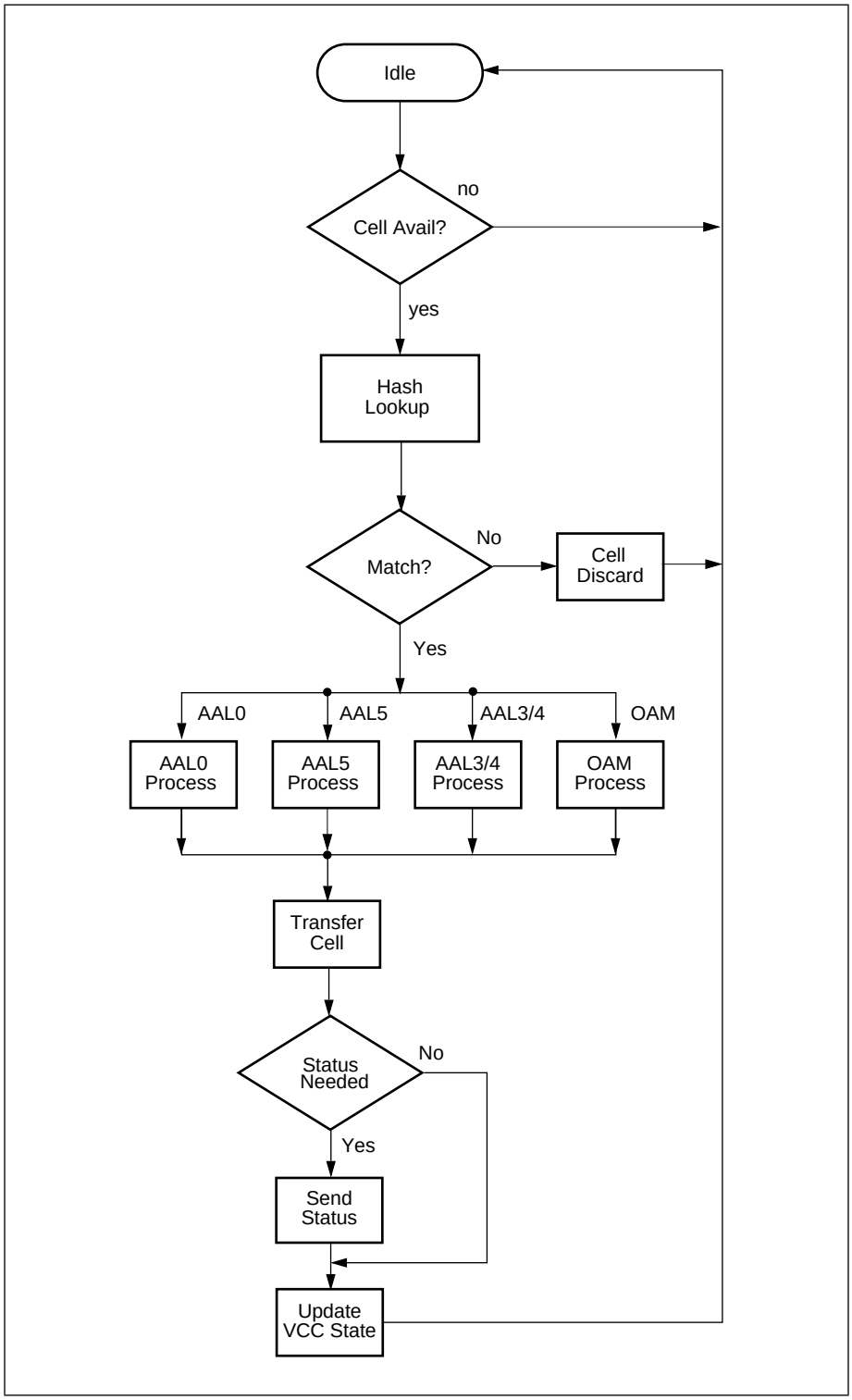


Figure 3-25. Bt8230 Receive Software Flow Chart

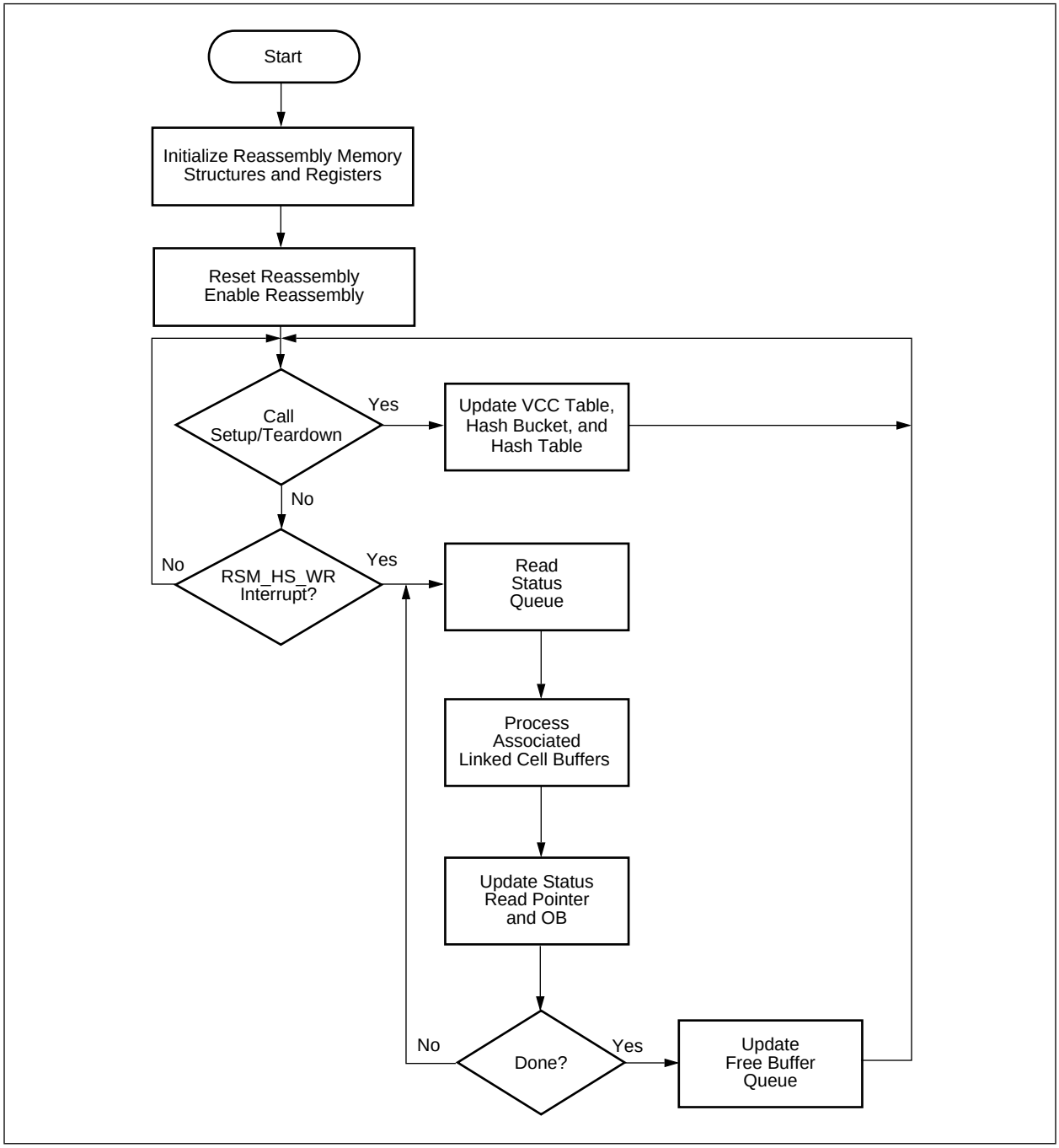
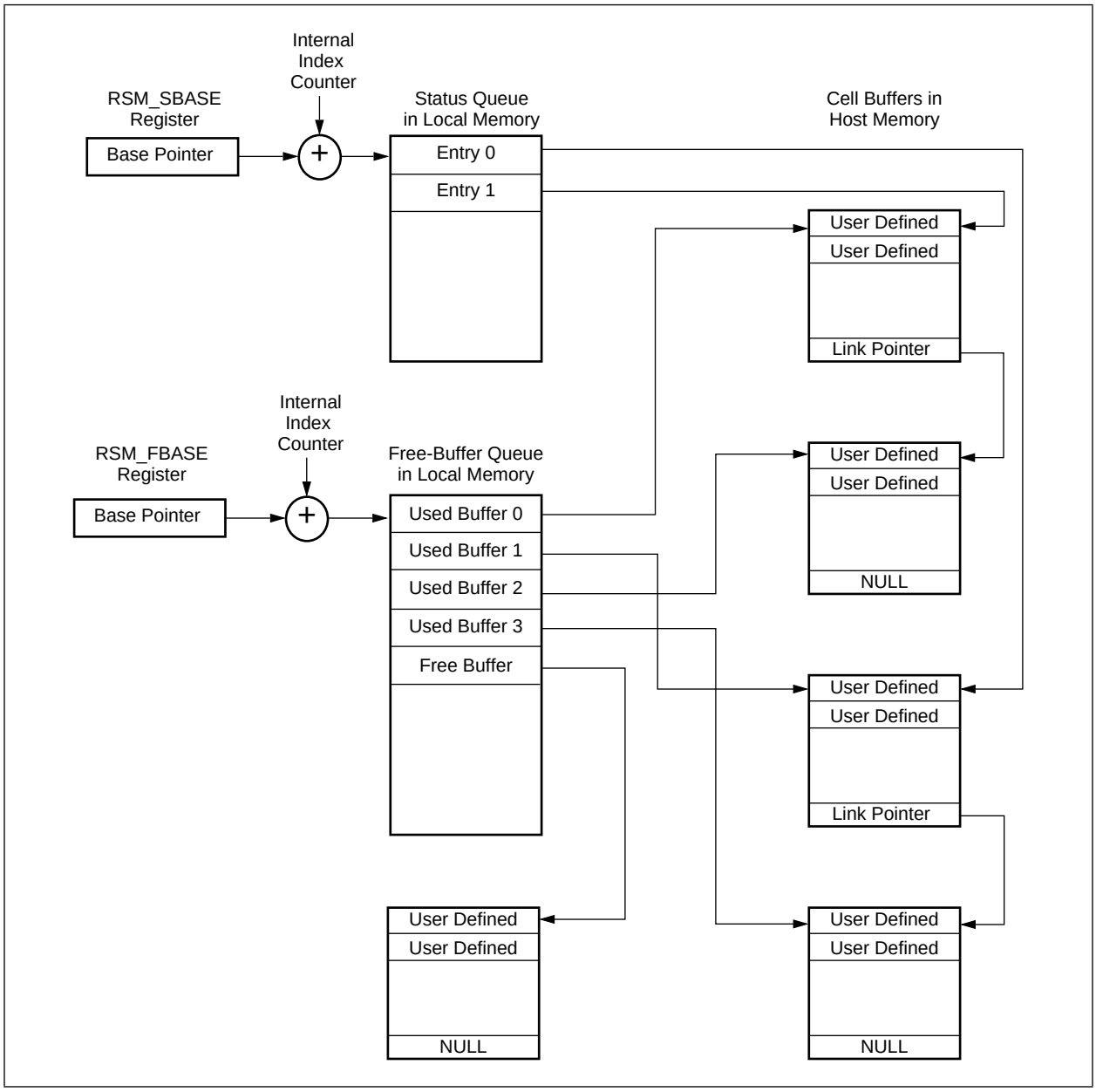


Figure 3-26 illustrates the registers and memory structures maintained to perform the intelligent scatter algorithm to host memory. The structure for local memory scatter is identical.

Figure 3-26. Register and Memory Structure for Intelligent Scatter Algorithm



3.5.1 Local Memory Set Up

The reassembly coprocessor uses six areas of contiguous local memory (Host Status Queue, Local Status Queue, Host Free Buffer Queue, Local Free Buffer Queue, RSM/SEG Queue, and Hash Table) and two dereferenced areas (VCC table and hashbuckets) to store reassembly state information. Each area can be located anywhere in the memory space. The base address for the queues and hash table is set by the user in a register. Each VCC table and hash bucket is located individually within the memory space. The size of each memory area is determined by the maximum limits set in the Reassembly Control Register [RSM_CTRL; 0x70]. The base pointer, size, initialization required, and purpose of each memory area are shown in Table 3-15.

Table 3-15. Bt8230 Reassembly Local Memory Configuration *(1 of 2)*

Base Pointer	Description
Host Status Queue	
HST_BASE[15:0] field in RSM_SBASE register [0x74]	Size: HS_SIZE * 16 bytes Initialization: Set to all 0s. Purpose: Indicates to the host processor that a CPCS has been received or an error condition has occurred.
Local Status Queue	
LST_BASE[15:0] field in RSM_SBASE register [0x74]	Size: LS_SIZE * 16 bytes Initialization: Set to all-0. Purpose: Indicates to the local processor that a CPCS has been received or an error condition has occurred.
Host Free Buffer Queue	
HFR_BASE[15:0] field in RSM_FBASE register [0x78]	Size: HF_SIZE * 8 bytes Initialization: Set per subsection 3.5.5. Purpose: Point to available reassembly cell buffers in host memory.
Local Free Buffer Queue	
LFR_BASE[15:0] field in RSM_FBASE register [0x78]	Size: LF_SIZE * 8 bytes Initialization: Set per subsection 3.5.5. Purpose: Point to available reassembly cell buffers in local memory.
Hash Table	
RSM_HBASE[15:0] field in RSM_HBASE register [0x7C]	Size: (TAB_SIZE * 4bytes) + 4 words + <Number of PM-OAM channels> * 4 bytes Initialization: Set per subsection 3.5.5 Purpose: Stores pointers to hash buckets.

Table 3-15. Bt8230 Reassembly Local Memory Configuration (2 of 2)

Base Pointer	Description
Hash Bucket	
Pointer in Hash Table	Size: <Number of VCCs> * 20 bytes. Add 4 bytes for each AAL3/4 non0 MID when MID multiplexing is enabled. Initialization: Set per subsection 3.5.5. Purpose: Stores Hash match information.
VCC Table	
VCC_PNTR in Hash Bucket	Size: <Number of VCCs> * 32 bytes Size can change as VCCs are added and removed. Initialization: Each VCC as described in subsection 3.5.5. Purpose: Stores state information for each VCC.
RSM/SEG Queue	
RS_QBASE[15:0] field in RS_QBASE register [0x88]	Size: 2048 bytes. Initialization: None. Purpose: Reserved queue for reassembly to segmentation communication.

3.5.2 Hashing

The Bt8230 implements a hash algorithm to locate the appropriate RSM VCC table entry for incoming ATM cells. During reassembly, the Bt8230 looks at the header of each incoming ATM cell to determine the connection (VCC) to which the cell belongs. This information is then used to establish where in the host memory the current reassembly buffer for this connection resides. The pointer to that buffer location (CBUFF_ADDR) is one of the fields in the reassembly VCC table entry for that connection. Because ATM cells will be received in an unknown order from the line, the Bt8230 must manage several incoming data streams and determine the VCC to which each cell belongs.

3.5.2.1 Hashing Algorithm

The Bt8230 hashing algorithm passes the 24-bit VPI/VCI pair received from the line through a truncating function. The hardware hashing function computes the hash table index as follows:

$$(\{VPI[7:0],VCI[15:12]\} \wedge VCI[11:0]) \,\&\, (2^{TAB\_SIZE}-1)$$

Bits 0 through 7 of the VPI are concatenated with bits 12 through 15 of the VCI to form a 12-bit field. This field is then exclusive-ORed with bits 0 through 11 of the VCI. The result is masked with a bitmask generated from the TAB_SIZE field of the Reassembly Control Register [RSM_CTRL; 0x70]. This function shortens the identifier to an N-bit field, where N is between 0 and 14. The Bt8230 then multiplies the N-bit offset by four and adds the result to the contents of the Reassembly Hash Table Base Register [RSM_HBASE; 0x7C] to produce the actual address of a pointer to a hash bucket chain. When TAB_SIZE is equal to 14, a special 14-bit hash index function is enabled. The hash table index is calculated as follows:

$$(\{VPI[7:0],VCI[15:14],'0000'\} \wedge VCI[13:0])$$

The hash bucket chain is processed sequentially for a hash bucket match. The hash bucket index comparator computes the logical-AND of the 32-bit MASK field of a hash bucket read from memory with the 32-bit ATM header. It then compares the result with the 32-bit HEADER field in the same hash bucket to generate a match signal. The match signal is used by the reassembly coprocessor to determine whether the required hash bucket has been found. If a match is obtained, the current hash bucket is used to process the cell; otherwise, the reassembly coprocessor reads a pointer to the next sequential bucket in the list from the NEXT field and repeats the matching process. If NEXT contains a null (0) pointer, the end of the list is assumed to have been reached. The matching is then declared unsuccessful, the cell is discarded, and the VPI_VCI_ERR counter at the top of the hash table is incremented.

NOTE: A bit bucket can be set up for cells with unknown VPI/VCIs. Instead of a NULL NEXT pointer in the last hash bucket, a common connection can be set up to handle these cells. By setting the AAL_TYPE to AAL0, a status entry will be written for every cell with unknown VPI/VCI, and the cell will be written to a cell buffer.

Once a hash bucket match occurs, the VCC_PNTR is used to access the RSM_VCC_TABLE. For AAL3/4, the pointer to the VCC table (VCC_PNTR) is found from the result of the following calculation:

$$\texttt{hash\_bucket\_base\_address} + 4 + (\texttt{MID} \;\&\; (2^{\texttt{MID\_BITS}} - 1))$$

MID is the AAL3/4 MID overhead field in the receive cell, and MID_BITs is the control field in the hash bucket.

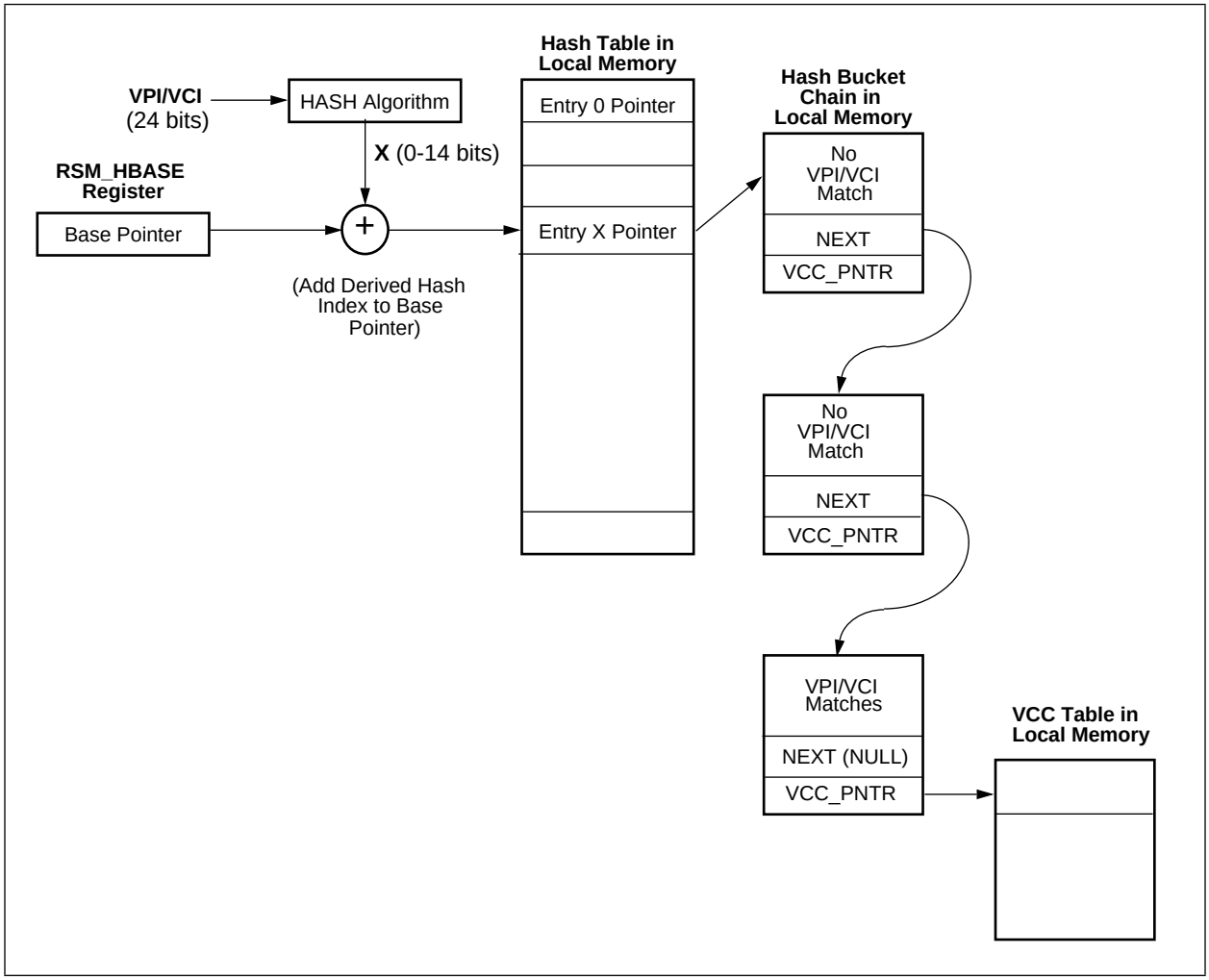
The VCC_PNTR fields are used to point to the VCC table entry of each connection. In AAL5, only one VCC_PNTR entry is used. AAL3/4 uses a VCC_PNTR entry for every CPCS-MID connection.

NOTE: The Bt8230 hash cache is not supported. The cache should be permanently disabled by setting the CACHE_ENABLE bit to a logic 0.

3.5.2.2 Hash Collisions

Obviously, the hash function's truncation of bits causes several VPI/VCI pairs to resolve to the same index value. This alias effect is often referred to as a "hash collision." The Bt8230 handles hash collisions by using a linked list of hash buckets for each hash table entry. The hash buckets contain VCC identifier information which is compared to the full header of the incoming cell. A mask, provided in each hash bucket, determines which bits will be compared. If a match is found, then that bucket identifies the proper VCC. Otherwise, the next bucket in the chain is searched by using the NEXT field (a pointer to the next hash bucket in the chain). The last bucket in every chain must contain a NULL NEXT field pointer. If a NULL pointer is encountered, and no match has been found in the hash bucket chain, the VCC is unassigned. Figure 3-27 illustrates the process flow of the hashing function.

Figure 3-27. Process Flow of the Hashing Function



3.5.2.3 Hash Table

The hash table contains pointers to discrete chains (or linked lists) of "hash buckets." The truncated identifier is used as an index into this table. The Bt8230 follows the pointer in the hash table entry and finds a hash bucket that identifies the VCC channel. (For more information, see Hash Collisions on page 84.)

The Reassembly Hash Table Base Address Register [RSM_HBASE; 0x7C] is expected to point to the starting address of this table of pointers. Global reassembly error counters are located at the top of the table. When these counters reach the maximum value, they wrap around to 0. A table of OAM performance monitoring words is also located at the top of the hash table. The format of the hash table is given in Table 3-16.

Table 3-16. Bt8230 Hash Table Data Structure

Address	Contents
RSM_HBASE + 0	Pointer to Hash Bucket Chain #0 (23 bits)
RSM_HBASE + 1	Pointer to Hash Bucket Chain #1 (23 bits)
RSM_HBASE + 2	Pointer to Hash Bucket Chain #2 (23 bits)
RSM_HBASE + (N–1)	Pointer to Hash Bucket Chain #(N–1) (23 bits)
RSM_HBASE + (N)	VPI_VCI_ERR (16 bits) Reserved(16 bits)
RSM_HBASE + (N+1)	CRC10_ERR (16 bits) MID_ERR(16 bits)
RSM_HBASE + (N+2)	LI_ERR (16 bits) SN_ERR (16 bits)
RSM_HBASE + (N+3)	BOM_EOM_ERR (16 bits) SSM_COM_ERR (16 bits)
RSM_HBASE + (N+4)	OAM channel 1 BIP16 (16 bits) BCNT (16 bits)
RSM_HBASE + (N+127)	OAM channel 124 BIP16 (16 bits) BCNT (16 bits)

The number of entries in the table (corresponding to the number of bins in the hashing function and denoted by N in the table above) is given by the TAB_SIZE[3:0] field [bits 20–17] in the Reassembly Control Register [RSM_CTRL;0x70] and is always a power of 2. This does not include the four global counter words and 124 OAM words at the top of the table. Each pointer in the table points to a linked list of hash buckets. Unused table locations should be set to NULL.

The number of hash buckets in each linked list is essentially unlimited. The end of the linked list is denoted by a null NEXT; i.e., one with a value of 0. The format of a hash bucket is given in Table 3-17. Field descriptions are given in Table 3-18.

Table 3-17. Hash Bucket Format

MASK (32)			
HEADER (32)			
Rsvd(3)	AAL_TYPE (2)	MID_BITS (4)	NEXT (23)
MSN(8)	Rsvd(10)	AAL_EN (7)	PMINDEX (7)
Rsvd(2)	CHECK_EN (7)	VCC_PNTR #1 (23)	
Rsvd(2)	CHECK_EN (7)	VCC_PNTR #2 (23)	
:			
:			
Rsvd(2)	CHECK_EN (7)	VCC_PNTR #M (23)	

Table 3-18. Hash Bucket Format Field Descriptions *(1 of 2)*

<table>
<tr><th>Field Name</th><th>Description/Function</th></tr>
<tr><td>MASK</td><td>Header comparison mask</td></tr>
<tr><td>HEADER</td><td>Header value for comparison</td></tr>
<tr><td>AAL_TYPE</td><td>Indicates type of AAL to be processed
00–AAL5
01–AAL0
10–AAL3/4</td></tr>
<tr><td>MID_BITS</td><td>Number of significant bits for the MID field. If the MID_BITS subfield is non-zero and the AAL_TYPE is AAL3/4, CPCS-MID field processing is enabled by the reassembly coprocessor. The interpretation of the MID_BITS subfield is as follows:
<table>
<tr><th>MID Bits</th><th>Active Bits of MID Field</th><th>Range of MID Field Values</th><th>MID Bits</th><th>Active Bits of MID Field</th><th>Range of MID Field Values</th></tr>
<tr><td>0000</td><td>0</td><td>MID Process-ing Disabled</td><td>1000</td><td>8</td><td>0–255</td></tr>
<tr><td>0001</td><td>1</td><td>0–1</td><td>1001</td><td>9</td><td>0–511</td></tr>
<tr><td>0010</td><td>2</td><td>0–3</td><td>1010</td><td>10</td><td>0–1023</td></tr>
<tr><td>0011</td><td>3</td><td>0–7</td><td>1011</td><td>10</td><td>1–1023</td></tr>
<tr><td>0100</td><td>4</td><td>0–15</td><td>1100</td><td></td><td>Reserved</td></tr>
<tr><td>0101</td><td>5</td><td>0–31</td><td>1101</td><td></td><td>Reserved</td></tr>
<tr><td>0110</td><td>6</td><td>0–63</td><td>1110</td><td></td><td>Reserved</td></tr>
<tr><td>0111</td><td>7</td><td>0–127</td><td>1111</td><td></td><td>Reserved</td></tr>
</table></td></tr>
</table>

Table 3-18. Hash Bucket Format Field Descriptions *(2 of 2)*

Field Name	Description/Function
NEXT	Pointer to next bucket, or null if end of list
MSN	Performance monitoring cell sequence number. Does not need to be initialized.
AAL_EN	Enable various cell processes. The AAL_EN field contains the following control bits: PM_EN | CRC10_EN | 52_EN | Rsvd | STM_MODE | LMEM_EN | FW_EN PM_EN — If this bit is set high, performance monitoring is enabled on this channel. CRC10_EN — If this bit is set high, enable AAL3/4 CRC10 field checking and error counting. 52_EN — If this bit is set high, the ATM and AAL overhead octets are written to the cell buffer. Rsvd — Reserved—Set to 0. STM_MODE — If this bit is set high, enable streaming mode for this channel. LMEM_EN — If this bit is set high, write CPCS-PDU to local memory. FW_EN — If this bit is set high, firewall is enabled on a per-VCC basis. Used in conjunction with FWALL_EN [bit 13] of the RSM_CTRL register.
PMINDEX	Pointer to a PM OAM word. Index with reference to top of hash table. Minimum value is 4; maximum value is 127.
CHECK_EN	Enable various AAL3/4 cell checks. The CHECK_EN field contains the following control bits: LI_EN | ST_EN | SN_EN | CPI_EN | BAT_EN | BAH_EN | LEN_EN LI_EN — If this bit is set high, enable AAL3/4 LI field checking and error counting. ST_EN — If this bit is set high, enable AAL3/4 ST field checking and error counting. SN_EN — If this bit is set high, enable AAL3/4 SN field checking and error counting.
VCC_PNTR	Address pointer to beginning of VCC table. Multiple pointers are needed in AAL3/4 when MID multiplexing is enabled.

3.5.2.4 Hash Performance

The Bt8230 hashing method significantly reduces the memory requirements of locating the appropriate VCC. The worst-case hash table size, occurring when a 14-bit index is used, is 16 k entries of pointers (at 4 bytes each), or 64 kbytes hash table size. With 16 k simultaneously open channels and a hash bucket size of 20 bytes, hash buckets for VCCs require approximately 327 kbytes. An additional 2.5 kbytes are required by the Bt8230 for performance monitoring channels. Therefore, the total memory required is approximately 393.5 kbytes, 1.6% of the memory required by the direct index approach. The hash algorithm also provides superior performance to a straightforward search method, since the hash function eliminates, on the average, all but ($[1/(2^N)]$ times the number of VCCs) possibilities immediately. This allows a much faster VCC lookup.

Using the Bt8230, system designers can make design trade-offs between performance and memory size. Memory size can be reduced by limiting the number of outstanding VCCs, and by choosing a small hash table. Performance is optimized by using larger hash tables, which reduces the frequency of hash collisions and average hash bucket chain depth.

Strategic setup and management of the hash buckets can also improve system performance. High throughput channels should be placed at the beginning of hash chains, since they receive a greater number of cells. The linked list structure of the hash buckets facilitates dynamic management as VCCs are set up and torn down.

The hash algorithm employed in the Bt8230 is optimized for User Network Interface (UNI). However, due to the comparison of the full 32-bit header when searching the hash bucket chain, the Bt8230 is capable of channel discrimination of the extended VPI field of the Network Node Interface (NNI).

If possible, VPI/VCI pairs should be chosen to minimize hash collisions, although this may not be feasible in a complex network. However, if a contiguous range of VPI/VCI pairs is chosen, the hashing algorithm implemented in the Bt8230 ensures that each pair resolves to a unique hash index even to the maximum size of the hash table (maximum 16,384 entries).

3.5.2.5 Hashing Structures

At initialization, all hash table entries that have not been written with a pointer should be set to NULL. At this time, a hash bucket can be created to "catch" unrecognized headers. This bucket should be at the end of ALL hash bucket chains. It should be set up with a mask that ensures all headers will satisfy the matching condition. System designers can direct the Bt8230 to write all 52 bytes of the cell (including the header) to memory by setting the 52_EN bit in the hash bucket entry to a logic 1. This allows the channel to be identified and recognized by software.

3.5.2.6 Adding Hash Buckets

The sequence used to add a hash bucket into a hash bucket chain is as follows:

1. Fill out the hash bucket.
2. If this is the last bucket in the chain, fill out the NEXT pointer in the hash bucket to NULL.
3. If this is not the last bucket in the chain, fill out the NEXT pointer in the hash bucket with the pointer to the bucket you want AFTER it in the chain.
4. In the bucket BEFORE it in the chain, write the NEXT pointer to the newly created bucket.
5. If the hash bucket being added is the first, write the pointer TO the added bucket into the hash table entry (RM_HBASE + x).

3.5.2.7 Deleting Hash Buckets

The sequence used to delete a hash bucket from a hash bucket chain is as follows:

1. If this is the last bucket in the chain, write the NEXT pointer in the hash bucket or hash table entry before the bucket being deleted to NULL.
2. If this is not the last bucket in the chain, write the NEXT pointer in the hash bucket or hash table entry before the bucket being deleted to point to the bucket following the deleted hash bucket.

Both the adding and deleting of hash buckets can be performed without interrupting the reassembly process.

3.5.3 Reassembly VCC Table

A VCC table is maintained for every active VPI/VCI and MID (see Table 3-19). The table maintains state information for an entire CPCS-PDU. Nine words are currently defined. The CRC_REM (AAL5) and BTAG (AAL3/4) words share the same physical location depending on the mode. The last word is only accessed by the local or host processor and is, therefore, a user-defined word. The processor may add on to the table since the reassembly coprocessor accesses the table by a pointer in the hash bucket. Field descriptions are given in Table 3-20.

Table 3-19. Reassembly VCC Table Format

VCC_INDEX (16)	Rsvd (6)	MAX_LEN_H (2)	Rsvd (5)	PDU_FLAG[2:0] (3)
BUFF_PNTR (32)				
MAX_LEN_L (15)			TOT_PDU_LEN (17)	
FW (1)	CBUFF_LEN (15)		CBUFF_CNT (16)	
CBUFF_ADDR (32)				
Rsvd (2)	BASIZE (16)		NEXT_ST (2)	NEXT_SN (4)	BTAG (8)
CRC_REM (32)				
CELL_CNT (16)			PDU_CNT (16)	
PDU_TS (16)			LAST_TS (16)	
LAST_COUNT (16)			CREDIT_TS (16)	

Note: The gray line is a word selector. In AAL3/4, the word above the gray line is used. In AAL5, the word below the gray line is used. Both words are never used at the same time.

Table 3-20. Reassembly VCC Table Format Field Descriptions

Field Name	Description
VCC_INDEX	VCC Index—Used to reference a VPI/VCI so that the host need not perform the hash function.
MAX_LEN_H	Reference MAX_LEN_L field description below.
PDU_FLAG[2:0]	PDU_FLAG[2] Cell Loss Priority(CLP)—Loss priority parameter ORed over PDU PDU_FLAG[1] Reserved PDU_FLAG[0] Beginning of Message(BOM)—Written by the reassembly coprocessor in streaming mode to tag the first buffer of a CPCS-PDU as containing a BOM.
BUFF_PNTR	Buffer Pointer—Points to the first linked cell buffer. In streaming mode, it points to the beginning of each cell buffer.
MAX_LEN_L	Maximum allowable length of a CPCS-PDU in bytes—Consists of two fields: low MAX_LEN_L (15 bits) and highMAX_LEN_H (2 bits).
TOT_PDU_LEN	Total PDU length in bytes—In AAL3/4 mode, it is the summation of the SAR-LI fields across the CPCS-PDU. In AAL5 mode, it is the length of the total CPCS-PDU.
FW	Firewall Condition—Indicates that a firewall condition has occurred.
CBUFF_LEN	Current Buffer Length—Unused space of the current buffer in words.
CBUFF_CNT	Current Buffer Count—Buffer firewall count.
CBUFF_ADDR	Current Buffer Address—Pointer to the current unused position of the cell buffer where cell payload data can be written.
BASIZE	BaSize—Used in AAL3/4 to record the BASIZE field in order to check against the LENGTH field.
NEXT_ST	Next Segment Type—Next AAL3/4 segment type expected.
NEXT_SN	Next Sequence Number—Next AAL3/4 sequence number expected.
BTAG	BTAG—Used in AAL3/4 to record the CPCS-BTAG field in order to check against the CPCS-ETAG field.
CRC_REM	Cycle Redundancy Check Remainder—CRC32 remainder used in AAL5 only.
CELL_CNT	Cell Count—Total number of cells received.
PDU_CNT	Protocol Data Unit Count—Number of unerrored CPCS-PDUs received.
PDU_TS	Protocol Data Unit Timestamp—Records the timestamp of when the current PDU started.
LAST_TS	Last Timestamp—Records the timestamp of the last cell received. Used for message time-out function.
LAST_COUNT	Last Count—Used by the local or host processor to record the value of the CELL_CNT field when the VCC table was last read. Not required for Bt8230 operation.
CREDIT_TS	Credit Timestamp—Used by the local or host processor to record the last time that credit was checked. Not required for Bt8230 operation.

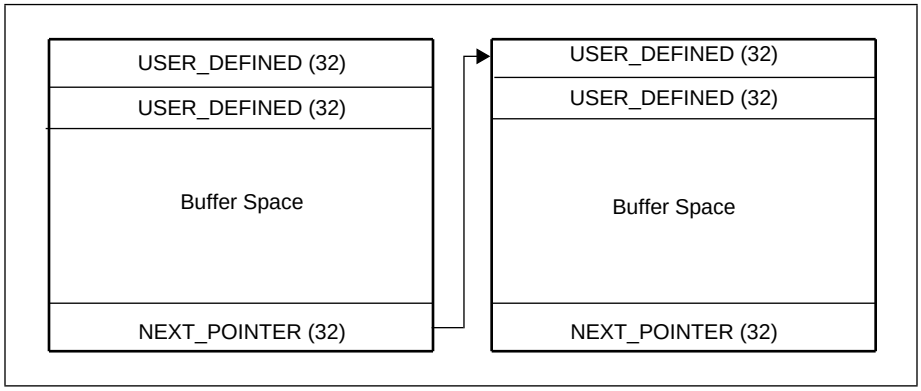
3.5.4 Scatter Method to Host Memory

The Bt8230 uses an intelligent scatter method to write cell payload data to host memory. The reassembly coprocessor maintains the scatter operation and controls the incoming DMA block during scatter DMA to host memory. Both message mode and streaming mode are supported. Message mode writes a status entry at the completion of a CPCS-PDU. The host can then traverse the linked cell buffers to collect the complete CPCS-PDU. In streaming mode, a status entry is written after each cell buffer is filled or the last cell of a CPCS-PDU has been received. Three data structures are maintained, one in the host memory and two in local memory. The linked cell buffers (HCELL_BUFF) reside in host memory, and the free buffer queue (HFR_BUFF_QUE) and status queue (HSTAT_QU) reside in local memory. The linked cell buffers contain the payload portion of the cell. The free buffer queue is a circular buffer that contains pointers to linked cell buffers and gives their length. The status queue contains information that allows the processor to operate on the linked cell buffers.

3.5.4.1 Linked Cell Buffers

The variable-length, linked cell buffers (HCELL_BUFF), pointed to by the free buffer queue entries, reside in host memory. The reassembly coprocessor writes the payload part of the cell to this buffer space, see Figure 3-28. The first two words of the buffer are user defined and can be used to store buffer length and offset. The last word of the buffer, NEXT_POINTER, is an optional pointer to the next linked buffer. This entry is written by the reassembly coprocessor after it has obtained a new buffer space for the corresponding connection or has been written to NULL for the last buffer in a message. The feature that writes the link pointer can be disabled by setting LINK_DIS in the RSM_CTRL register. The NEXT_POINTER should only be disabled in streaming mode.

Figure 3-28. Linked Cell Buffers



3.5.4.2 Host Free Buffer Queue, HFR_BUFF_QUE

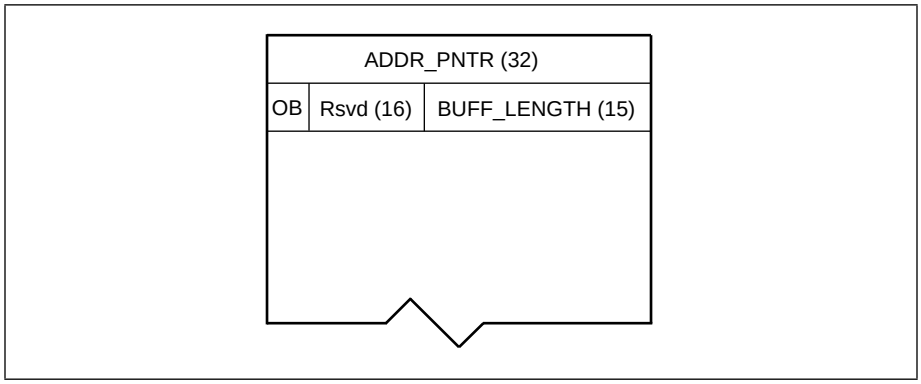
The Host Free Buffer Queue is used to process reassembled cell streams. The host initializes and maintains this queue. A free buffer queue entry consists of two words that contain the address pointer and buffer length of an available buffer space in host memory along with an Owner Bit (OB) (see Figure 3-29). The buffer length is in words (4 bytes). The length does not include the first two words and the last word of the buffer. This structure resides in local memory to allow low latency in setup of new buffer spaces. The owner bit is written by the reassembly coprocessor to a logical 1 when it has read an entry. The host processor needs to reset the owner bit to a logical 0 when it writes a new entry in the queue. The host will need to keep a free list of cell buffers to supply this queue as buffers are used. Only one cell buffer exists per Free Buffer Queue entry.

The reassembly coprocessor generates a free buffer queue read pointer from the HFR_BASE[15:0] field in the Reassembly Free Buffer Queue Base Register [RSM_FBASE; 0x78] and the HF_SIZE field in the RSM_CTRL register. The pointer takes on the value:

```
HFR_POINTER = HFR_BASE * 128 + HFR_COUNTER * 8
```

The HFR_COUNTER is an internal counter. The size of the counter is determined by the HF_SIZE field in the RSM_CTRL register. The counter can be 8, 10, 12, or 14 bits in length, and is reset to 0 by the RSM_RESET bit in the RSM_CTRL register.

Figure 3-29. Free Buffer Queue



3.5.4.3 Status Queue

The status queue is a circular buffer that resides in local memory. A 4-word entry is written to the queue before the reassembly coprocessor interrupts the host by setting the RSM_HS_WRITE bit in the HOST_ISTAT0 and LP_ISTAT0 registers. A status queue entry is written after any of the following occurs:

- A complete CPCS-PDU has been received in message mode.
- A cell buffer has been used in streaming mode.
- A PDU maximum length error condition has been detected.
- A message time-out has been detected.
- A n OAM cell has been received.

The format of a status queue entry is shown in Figure 3-30. Field descriptions are given in Table 3-21.

The reassembly coprocessor generates a status queue write pointer from the HST_BASE field in the RSM_SBASE register and the HS_SIZE field in the RSM_CTRL register. The pointer takes on the value:

```
HST_POINTER = HST_BASE * 128 + HST_COUNTER * 16
```

The HST_COUNTER is an internal counter. The size of the counter is determined by the HS_SIZE field in the RSM_CTRL register. The counter can be 8, 10, 12, or 14 bits in length. The counter is reset to 0 by the RSM_RESET bit [bit 30] in the RSM_CTRL register.

Figure 3-30. Status Queue Entry

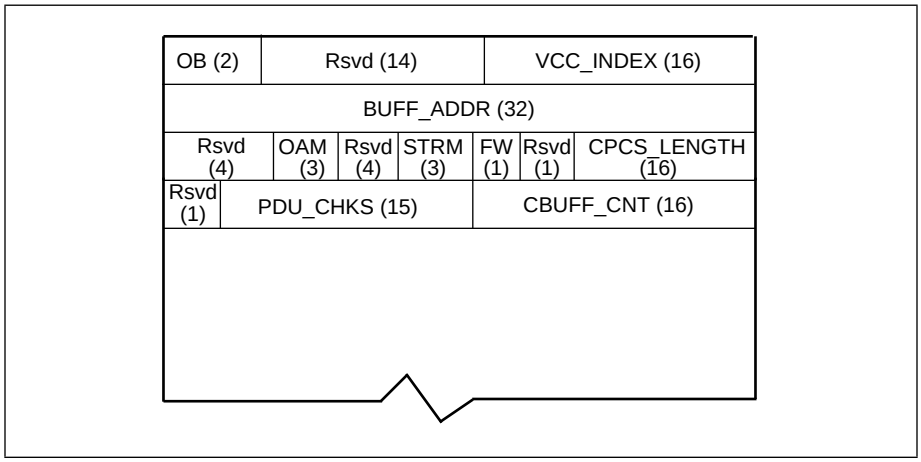


Table 3-21. Status Queue Entry Field Descriptions

Field Name	Description
OB	Owner Bits—Semaphore bits to prevent status queue overwrite and to mark used buffer locations. Written by the reassembly coprocessor to logical high. Reset by the host processor to a logical 0.
VCC_INDEX	VCC Index—Used to reference a VPI/VCI so that the host need not perform the hash function.
BUFF_ADDR	Buffer Address—In message mode, this field points to the first buffer of the linked cell buffers. In streaming mode, it points to each cell buffer used.
OAM	Operating and Maintenance—If this field is non-zero, it indicates that an OAM or management cell had been received as follows: 001–F4 OAM 010–F4 OAM End to End 100–F5 OAM 101–F5 OAM End to End 110–PTI = 6 111–PTI = 7
STRM	Streaming mode status field. Contains three bits as follows: Bit 2–EOM—In streaming mode, this bit identifies that the buffer contains a EOM Bit 1–BOM—In streaming mode, this bit identifies that the buffer contains a BOM. Bit 0–STRM_EN—Streaming mode is enabled on this channel
FW	Firewall—Firewall error occurred.
CPCS_LENGTH	Value of CPCS-length field.
PDU_CHKS	PDU_CHKS[14] MOD_ERROR—AAL3/4 CPCS-PDU is not 32 bit aligned. PDU_CHKS[13] BA_ERROR—AAL3/4 BASIZE ≠ LENGTH. In AAL5, indicates that total pdu length exceeds MAX_LENGTH field when AUU = 1. PDU_CHKS[12] AL_ERROR—AAL3/4 AL field not equal to 0. PDU_CHKS[11] LEN_ERROR—Total PDU length exceeds maximum length and not at end of message. PDU_CHKS[10] PAD_ERROR—Pad field length is not correct length. PDU_CHKS[9] CPI_ERROR—CPI field is not all 0. PDU_CHKS[8] TAG_ERROR—BTAG does not match ETAG in AAL3/4. PDU_CHKS[7] CRC_ERROR—AAL5 CPCS or OAM CRC error. PDU_CHKS[6] ST_ERROR—AAL3/4 segment type error. PDU_CHKS[5] SN_ERROR—AAL3/4 sequence number error. PDU_CHKS[4] LI_ERROR—AAL3/4 SAR-PDU length error PDU_CHKS[3] CLP—Cell Loss Priority parameter PDU_CHKS[2] CI—Congestion Indication PDU_CHKS[1] ABORT—Abort function detected PDU_CHKS[0] TO—Message time out
CBUFF_CNT	Value of CBUFF_CNT field in the VCC table.

3.5.5 Structure Initialization

The structures described below must be initialized before operation of the reassembly coprocessor is enabled. The suggested initialization is as follows:

1. Linked Cell Buffers
   - The first two words of each buffer may be written to user-defined values.
   - The last word of each buffer must be written to NULL.
2. Free Buffer Queue
   - Write BUFF_ADDR to point to a cell buffer configured in host memory.
   - Write BUFF_LENGTH to reflect the length of the cell buffer in words, not including the first two words or the last word.
   - Write OB to a logical 0.
   - Write the base address of the free buffer queue in the HFR_BASE[15:0] subfield of the Reassembly Free Buffer Queue Base Register [RSM_FBASE; 0x78]. Write the size of the free buffer queue in the HF_SIZE subfield of the RSM_CTRL register.
3. Status Queue
   - Write the OB field in each status entry to 00.
   - Write the base address of the status queue in the HST_BASE[15:0] subfield of the Reassembly Status Queue Base Register [RSM_SBASE; 0x74]. Write the size of the status queue in the HS_SIZE subfield of RSM_CTRL register.
4. Hash Table
   - Write the base address of the hash table in the Reassembly Hash Table Base Register [RSM_HBASE; 0x7C].
   - Entries not written with a pointer to a Hash Bucket chain should be written to NULL.
5. Hash Bucket
   - Initialize all words in each hash bucket to the appropriate values.
   - Write the global counters and PM words at the top of the hash table to NULL.
6. VCC table
   - Write the following fields in each VCC table.

     VCC_INDEX = User defined number
     CLP = 0
     BOM = 1
     In AAL5, MAX_LENGTH = (MAX_SDU_LENGTH + pad field length + trailer length)
     In AAL3/4, MAX_LENGTH = User defined number
     TOT_PDU_LEN = 0
     FW = 0
     CBUFF_CNT = Number of cell buffers allowed for this connection if FWALL_EN is set in the RSM_CTRL register.
     In AAL5, CRC_REM = FFFFFFFFh.
     In AAL3/4, NEXT_ST = 10
   CELL_CNT = 0
   PDU_CNT = 0

3.5.6 Cell Buffer and Status Processing

Upon receiving an interrupt, the host will read the HOST_ISTAT0 register to determine what event caused the interrupt. If the RSM_HS_WRITE bit is set, the status queue in local memory must be processed. The status queue supports both streaming and message modes. In message mode, a status entry is written upon the completion of the CPCS checks performed on a complete CPCS. In streaming mode, a status entry is written upon filling a cell buffer in order to allow the host to start processing a CPCS-PDU before the complete PDU is received. In this case, the STRM field in the status queue is written to indicate that the connection is in streaming mode and whether or not the buffer contains a BOM or EOM cell. The CPCS_LENGTH and PDU_CHECKS fields (not including the TO bit) are valid only for a cell buffer that contains the last cell of a CPCS-PDU. The BUFF_ADDR field points to the beginning of each buffer.

The host processes a status entry by maintaining a read pointer to the circular status queue. The host reads the status entry pointed to by the read pointer and then writes the Owner Bits (OB) of the status entry to a logical 0 and increments its read pointer. The host performs the necessary functions to process the cell buffer(s) referenced by the status entry. If the cell buffer firewall is enabled (FWALL_EN bit in RSM_CTRL), the host must check the CBUFF_CNT field in the status entry. If the value is nearing 0, the host needs to write more credit for the channel in the Reassembly Firewall Register [RSM_FW; 0x84]. It then reads the owner bits of the next status entry to determine if more status processing is needed. Both owner bits must be set to 1 for the host to process the status entry. The host continues this process until it has read an OB at a logical 0.

The host must also maintain a read pointer to the free buffer queue in local memory. After processing the Status Queue, the host reads the OB of the free buffer queue entry pointed to by its read pointer. If the OB is a logical 1, the host writes a new address pointer and length entry and writes the OB to a logical 0. The host continues this process until it has read an OB at a logical 0.

The host processes the linked cell buffers by reading the contents of each buffer until a NULL next pointer is encountered. This signals that the last cell buffer of the linked buffers has been reached.

3.5.7 Status Queue Full Condition

The OBs of the current status entry are read by the Bt8230 before the status is written. If either bit is a logical 1, the status entry is not written, and an error is reported on the RSM_HS_FULL bit in the HOST_ISTAT0 and LP_ISTAT0 registers. The reassembly coprocessor also halts operations. The host can be optionally interrupted by setting the EN_RSM_HS_FULL bit in the Host Interrupt Mask Register [HOST_IMASK0; 0xCC]. The host then needs to update the status queue and set the RSM_EN bit in order to restart the reassembly coprocessor. The reassembly coprocessor will then reread the OB bits of the current status entry and continue processing. The full buffer protection can be disabled by setting the RSM_HS_DIS bit in the RSM_CTRL register.

3.5.8 Cell Buffer Queue Empty Condition

The cell buffer queue empty condition occurs when there are no available cell buffers in the free buffer queue. The owner bit of the current free buffer queue is read before a cell buffer is obtained. If the bit is at a logical 1, the reassembly coprocessor terminates processing and sets either the RSM_HF_EMPT or RSM_LF_EMPT bit in the HOST_ISTAT0 or LP_ISTAT0 registers, respectively. The host may be optionally interrupted by setting the EN_RSM_HF_EMPT bit in the HOST_IMASK0 register. The host then needs to update the free buffer queue and set RSM_EN [bit 31] to restart the reassembly coprocessor. The reassembly coprocessor will then reread the OB bit of the current free buffer queue and continue processing. The empty queue protection can be disabled by setting the RSM_HF_DIS bit in RSM_CTRL.

3.5.9 Firewall Condition

A firewall is a barrier set up to contain designated traffic within a specified area. In the Bt8230 architecture, firewalling limits the number of cell buffers and thus the memory bus bandwidth that is available to an individual VCC. This provides a method of policing the channels to prevent any single channel from consuming disproportionate resources. Firewalling does not affect OAM cells.

The Bt8230's firewall feature is enabled at two levels. First, the feature is enabled globally by setting the FWALL_EN bit (RSM_CTRL[13]) to a logical 1. Next, the firewall feature is enabled on an individual channel by setting the FW_EN bit in that channel's VCC hash bucket to a logical 1.

Once both of the firewall-enabling controls are set, the Bt8230 checks the CBUFF_CNT field of the RSM VCC Table for the channel being used before taking a buffer from the Free Buffer Queue. This field is an optional parameter that is set by the user when initializing the VCC as a receive or full-duplex connection. If CBUFF_CNT is greater than 0, the Bt8230 decrements it and takes a buffer from the queue. If CBUFF_CNT is 0, a firewall condition has occurred on that channel.

When the firewall condition occurs, the Bt8230 sets the FW bit in the RSM VCC Table for that channel to a logical 1. It then issues a Status Queue entry with the FW bit in the RSM Status Queue set to a logical 1. If reassembly of a PDU was in progress on that VCC, the host will return a pointer to the first buffer in the reassembly buffer chain for that PDU. The incomplete PDU will be discarded and the buffers containing the PDU will be re-allocated to the free buffer queue.

The Bt8230 discards all cells on the firewalled channel until it receives a Beginning of Message (BOM) or Single Segment Message (SSM) cell indicating that a cell buffer is available. Upon receipt of a BOM or SSM, the Bt8230 will recheck the CBUFF_CNT field in the channel's VCC Table entry for a value greater than 0, recommencing the process described above. The host may restart reassembly on that channel by reinitializing CBUFF_CNT to a value greater than 0. If the BUFF_ADDR field in the status entry is non-zero, the field points to a linked list of cell buffers that should be discarded.

The BUFF_ADDR field of the reassembly Status Queue entry is always NULL when the firewall condition is triggered by a BOM or SSM indication in a received cell. If the cell has been stored into multiple receive buffers (i.e., buffers having a size less than 48 bytes), the buffers used to store the partial cell prior to the occurrence of the firewall condition will not be recoverable by the system. (The BUFF_ADDR field is NULL and does not point to the first buffer of the linked list of buffers containing the cell, so software cannot recover the buffer.)

NOTE: Do not use buffers less than 48 bytes in length when firewalling is enabled.

While a channel is actively reassembling, the host may need to add buffers to the VCC's firewall credits. This is done by using the Reassembly Firewall Register (RSM_FW, at address 0x84). The RSM_FW register includes in it a 21-bit address pointer, FWPNTR[20:0] (bits 22-2), and a 9-bit credit update field, FWCREDIT[8:0] (bits 31-23). FWPNTR points to the address in the VCC table of the VCC entry to which the user wishes to add firewall credits. The CBUFF_CNT field in the VCC Table is incremented by the value in the FWCREDIT field. The reassembly coprocessor will NULL the Reassembly Firewall Register after it has processed a time-out. The processor can only write to this register when the contents are NULL. Writing to it when the contents are non-zero will have no effect.

3.5.10 Scatter Method to Local Memory

The local processor can be used to perform ILMI, signaling, and flow control by setting the LMEM_EN bit in the hash bucket. When this bit is set, the reassembly coprocessor writes cells to local memory instead of host memory. The local processor can then gather the cells and reassemble the CPCS-PDU. The structure to perform local scatter is the same as host scatter. All structures reside in local memory. Table 3-22 lists the resources used in place of host memory resources.

In effect, the local processor will be interrupted when a status entry is written to the Local Status Queue. The local processor needs to perform the same functions as the host as described in the Table 3-22.

Table 3-22. Local Resources Used in Place of Host Memory Resources

Local	Host
LST_BASE in RSM_SBASE[0x74]	HST_BASE in RSM_SBASE[0x74]
LFR_BASE in RSM_FBASE[0x78]	HFR_BASE in RSM_FBASE[0x78]
RSM_LF_EMP in RSM_CTRL[0x70]	RSM_HF_EMP in RSM_CTRL[0x70]
RSM_LS_FULL in RSM_CTRL[0x70]	RSM_HS_FULL in RSM_CTRL[0x70]
LF_SIZE in RSM_CTRL[0x70]	HF_SIZE in RSM_CTRL[0x70]
LS_SIZE in RSM_CTRL[0x70]	HS_SIZE in RSM_CTRL[0x70]
LSTAT_QU	HSTAT_QU
LFR_BUFF_QUE	HFR_BUFF_QUE
LCELL_BUFF	HCELL_BUFF
LFR_POINTER	HFR_POINTER
LST_POINTER	HST_POINTER
RSM_LF_DIS in RSM_CTRL[0x70]	RSM_HF_DIS in RSM_CTRL[0x70]
RSM_LS_DIS in RSM_CTRL[0x70]	RSM_HS_DIS in RSM_CTRL[0x70]
RSM_LS_WRITE in LP_ISTAT0[0xE0]	RSM_HS_WRITE in HOST_ISTAT0[0xC0]

3.5.11 AAL5 CPCS Processing

AAL5 processing is used when the AAL_TYPE field in the hash bucket indicates AAL5. Both CPCS payload and trailer are written to memory.
The following processes occur for every cell:

- The CLP bit in the VCC table holds the accumulated value of the CLP parameter. Upon receipt of a cell, the reassembly coprocessor ORs the CLP bit in the VCC table with the CLP bit of the cell and writes the result back to the VCC table.
- The timestamp fields in the VCC table are updated each cell. The PDU_TS is written on the first cell of a PDU. The first cell is detected when the TOT_PDU_LEN field in the VCC table is 0. The LAST_TS field is written upon reception of a cell. Also, the CELL_CNT field in the VCC table is incremented upon receipt of a cell.
- The VCC table contains a TOT_PDU_LEN field. This field is constantly updated as cells of a CPCS-PDU are received. The field contains the total length of the PDU in octets. If this value ever exceeds MAX_LENGTH, the PDU is terminated and a status entry is written with the LEN_ERROR bit set to 1. If a cell with AUU = 1 exceeds the maximum length, the BA_ERROR bit is set to 1 instead of the LEN_ERROR bit. The MAX_LENGTH field is written by the host or local processor at setup to reflect the AAL5 MAX_SDU_DELIVER_LENGTH parameter padding and trailer lengths in octets. It should be an integer number of cells in length. The maximum value of MAX_LENGTH in this mode is 65568.
- The VCC table contains a CRC_REM field that holds the incremental 32-bit CPCS CRC calculation. Upon receiving a cell, the reassembly coprocessor reads the CRC_REM field into the reassembly 32-bit CRC generator. The cell payload is then used to calculate the next CRC remainder. After the current cell is processed, the new CRC remainder is written back to the CRC_REM field. The CRC polynomial is:

$$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1$$

Upon termination of a CPCS-PDU, AUU bit = 1, the following processes occur:

- The 32-bit CRC remainder is checked for a value of 0xC704DD7B. If in error, the CRC_ERROR bit in the status entry is set to 1.
- The SAR-CI parameter is copied into the CI field of the status entry.
- If the CPCS-LENGTH field is all 0s, the ABORT bit in the status entry is set to 1. The CPCS_LENGTH field in the status entry is written to the value of the CPCS-LENGTH field.
- The reassembly coprocessor performs a pad length check if LENGTH ≠ 0 as follows: TOT_PDU_LEN –CPCS_LENGTH –8 = [0,47] octets.
- If in error, the PAD_ERROR bit is set to 1 in the status entry. The CPI_ERROR bit in the status entry is set to 1 if the CPCS-CPI field is not all 0s.
- The PDU_CNT field in the VCC table is also incremented if no errors have occurred.

After the status entry is written, the VCC table is initialized to start processing another CPCS-PDU.

3.5.12 AAL3/4 CPCS Processing

AAL3/4 processing is used when the AAL_TYPE field in the hash bucket indicates AAL3/4. CPCS header, payload, and trailer are written to memory. The following checks occur before the cell is accepted for further processing:

- If the CRC10_EN bit is set to 1 in the hash bucket, the CRC10 field is checked. If in error, the cell is discarded and the CRC10_ERR counter at the top of the hash table is incremented. The CRC polynomial is:

  $x^{10} + x^9 + x^5 + x^4 + x^1 + 1.$

- The MID field value is checked against the number of active MID bits specified by the MID_BITS field in the hash bucket. If invalid bits are set, the cell is discarded, and the MID_ERR counter at the top the hash bucket is incremented.
- If the ST_EN bit is set to 1 in the hash bucket, the ST field is checked. The NEXT_ST field in the VCC table is used for this check. A value of 1 in NEXT_ST field indicates expecting BOM/SSM. A 00 value indicates expecting Continuation of Message (COM)/EOM. The following errors are detected:

  a. Receive BOM expecting COM/EOM—The current CPCS-PDU is terminated. A status entry is sent with the ST_ERROR bit set, CPCS_LENGTH = 0 and the BOM_EOM_ERR counter at the top of the hash table is incremented. A new CPCS-PDU is started with the current BOM.
  b. Receive SSM expecting COM/EOM—The current CPCS-PDU is terminated. A status entry is sent with the ST_ERROR bit set and CPCS_LENGTH = 0. The current SSM is processed as a valid CPCS-PDU. The SSM_COM_ERR counter at the top of the hash table is incremented.
  c. Receive COM expecting BOM/SSM—The cell is discarded, and the SSM_COM_ERR counter at the top of the hash table is incremented.
  d. Receive EOM expecting BOM/SSM—The cell is discarded, and the BOM_EOM_ERR counter at the top of the hash table is incremented.

- If EOM and LI = 63, a status entry is sent with ABORT set and CPCS_LENGTH = 0.
- If the LI_EN bit in the hash bucket is set to 1, the LI field is checked. The following errors are detected:

  a. Receive BOM and LI ≠ 44—The cell is discarded and the LI_ERR counter at the top of the hash table is incremented.
  b. Receive COM and LI ≠ 44—The cell is discarded and the current CPCS-PDU is terminated. A status entry is sent with the LI_ERROR bit set, CPCS_LENGTH = 0 and the LI_ERR counter at the top of the hash table is incremented.
  c. Receive EOM and (LI < 4) or (LI > 44—The cell is discarded and the current CPCS-PDU is terminated. A status entry is sent with the LI_ERROR bit set, CPCS_LENGTH = 0 and the LI_ERR counter at the top of the hash table is incremented.
  d. Receive SSM and (LI < 8) or (LI > 44)—The cell is discarded and the LI_ERR counter at the top of the hash bucket is incremented.

- If the SN_EN bit in the hash bucket is set to 1, the SN field is checked. If COM or EOM is received and the SN field does not equal the NEXT_SN field in the VCC table, the cell is discarded, the current PDU is terminated, and a status entry is written with the SN_ERROR bit set and CPCS_LENGTH = 0. The SN_ERR counter at the top of the hash table is also incremented.

After the cell checks are performed resulting in a valid cell, the following processes occur:

- Upon receiving a new CPCS-PDU:
  - **a** The CPI field is checked for a 0 value if CPI_EN is set to 1 in the hash bucket. If in error, the cell is discarded, the current PDU is terminated and a status entry is written with the CPI_ERROR bit set high and CPCS_LENGTH = 0.
  - **b** If a BOM and BAH_EN is set high, The BASIZE field is checked to be less than 37 octets. If less than 37 octets, the cell is discarded, the current PDU is terminated and a status entry is written with the BA_ERROR bit set high and CPCS_LENGTH = 0.
  - **c** If CPI and BASIZE fields are valid and LEN_EN is set to 1 in the hash bucket, then the reassembly coprocessor writes a value of BASIZE field + 7 to the MAX_LENGTH field in the VCC table. If LEN_EN is set to 0, the MAX_LENGTH field must be initialized upon connection setup. The value of the BASIZE field in the VCC table is also updated.
- The CLP field in the VCC table holds the value of the CLP parameter. Upon reception of a cell, the reassembly coprocessor ORs the CLP bit in VCC table with the CLP bit of the cell and writes the result back to the VCC table.
- The timestamp fields in the VCC table are updated each cell. The PDU_TS is written on the first cell of a PDU. The first cell is detected when the TOT_PDU_LEN field in the VCC table is 0. The LAST_TS field is written upon receipt of a cell. Also, the CELL_CNT field in the VCC table is incremented upon receipt of a cell.
- The VCC table contains a TOT_PDU_LEN field. This field is constantly updated as cells of a CPCS-PDU are received. The field contains the accumulation of the SAR-LI fields of the PDU in octets. During a BOM or COM, if this value ever exceeds MAX_LENGTH, the PDU is terminated and a status entry is written with the LEN_ERROR bit set to 1 and CPCS_LENGTH = 0.

Upon termination of a CPCS-PDU, ST = EOM or SSM, the following processes occur:

- The SAR-CI parameter is copied into the CI field of the status entry.
- The CPCS_LENGTH field in the status entry is written to the value of the CPCS_LENGTH field.
- The reassembly coprocessor performs a pad length check as follows:
  TOT_PDU_LEN –CPCS_LENGTH – 8 = [0,3] octets. If in error, the PAD_ERROR bit is set in the status entry.
- The reassembly coprocessor confirms that the total message length is on a 32-bit boundary. If in error, the MOD_ERROR bit in the status entry is set to 1.
- The AL_ERROR bit in the VCC table is set if the CPCS-AL field is not all 0s. The TAG_ERROR bit is set if the CPCS-BTAG field does not match the CPCS-ETAG field.
- If BAT_EN is set in the hash bucket, BASIZE is compared to the LENGTH field. If not equal, a status entry is sent with BA_ERROR bit set. If BAT_EN is a logical 0 and the LENGTH field exceeds the BASIZE field, the BA_ERROR bit is set.
- The PDU_CNT field in the VCC table is incremented if no errors occurred.

After the status entry is written, the VCC table is initialized to begin processing another CPCS-PDU.

3.5.13 AAL0 Processing

AAL0 processing is used when the AAL_TYPE field in the hash bucket indicates AAL0. Processing consists of writing the cell to host or local memory and writing a status entry for each cell. In the VCC table, the CELL_CNT field is incremented, and the current timestamp is written to LAST_TS. The fields used on the status entry are VCC_INDEX, BUFF_ADDR, OB, and CBUFF_CNT.

3.5.14 OAM Processing

By global control (OAM_MEM bit in the RSM_CTRL register), OAM cells can be written to either local or host memory. If OAM_EN is set in the RSM_CTRL register, the reassembly coprocessor will internally detect an OAM cell and route it accordingly. To save hash bucket locations, the following OAMs will be detected:

F4 OAM
F4 OAM End to End
F5 OAM
F5 OAM End to End
PTI = 6
PTI = 7

Connections with F5 or PTI 6 and 7 flows enabled must have the PTI field masked in the corresponding hash bucket for the connection since both user data cells and OAM cells are sharing the same VPI/VCI address. Note that PTI 6 and 7 fields are reserved in UNI 3.1 but are still detected by the RSM. F5 flow will already have an active entry in the VCC table so no extra hash bucket need be assigned. F4 flows will need to be explicitly terminated so the host/local will need to establish explicit hash buckets to handle these cells. Connections with F4 flows enabled must configure a hash bucket for VCI 3 and 4. For this connection, only the first word of the VCC table need be configured.

The complete 52-octet OAM cell is written to either host or local memory using the scatter method. The status entry indicates an OAM cell by a non-0 OAM field. The fields used in the status entry are VCC_INDEX, BUFF_ADDR, OAM, CRC_ERROR, and OB. The reassembly coprocessor treats OAM cells as 1-cell PDUs. The 10-bit CRC is checked and if in error, the CRC_ERROR bit is set in the status entry.

Once an OAM cell is detected, the cell is checked to determine whether it is a PM cell. Note that PM detection is only performed on F4 and F5 OAM cells. The PM word pointed to by the PMINDEX field in the hash bucket is read and the PM reporting fields Block Error Result and Lost/Misinserted Cell Count are appended to the cell payload. The reporting fields are also written to the RSM/SEG Queue for further processing by the segmentation coprocessor.

3.5.14.1 PM Operation

Upon receiving a PM activation cell, the local or host processor must enable PM on the applicable channel by setting the PM_EN bit high and writing the PMINDEX to point to the memory location that contains the BIP16 and BCNT values. The PMINDEX value has a minimum value of 4 and a maximum value of 127. Both BIP16 and BCNT should initially be written to 0. For F4 flows, each VCI channel in the VPI group must have the PM_EN bit set to 1 and the PMINDEX pointing to the same location. This is also true of the VCI 3/4 hash bucket set up to handle F4 flows. Once this has been performed, the processor must send an activation confirmed OAM cell to the originator. When a user data cell is received that has PM activated on the channel, the BIP16 and BCNT values are updated.

NOTE: The cell buffer size must be large enough to hold a complete cell when OAM detection is enabled.

3.5.15 Message Time Out

Message time out is supported by the LAST_TS field in the VCC table. The processor can peruse the VCC tables looking for active PDUs, with TOT_PDU_LEN not NULL. The processor then compares the LAST_TS field against the CLOCK register in the Bt8230. If the maximum reassembly time has elapsed, the processor alerts the reassembly coprocessor that a time-out has occurred through the RSM_TO register.

The processor reads the Reassembly Time Out Register [RSM_TO; 0x80] until it returns a NULL. The processor then writes the address of the first word of the VCC table of the timed-out channel in the TOPNTR field, the status destination to the LSTAT field and if the channel is an AAL3/4, set the TMO_AAL34 bit to a logical 1. The Bt8230 will then write a status entry with the TO bit set to a logical 1 and initialize the VCC table for a new CPCS-PDU. The TO bit in the status entry will be set to a logical 1 and the BUFF_ADDR field will point to the beginning of the first buffer in the linked cell list. In streaming mode, the BUFF_ADDR field will point to the beginning of the last cell buffer.

3.5.16 Credit Check

The credit check is mainly a microprocessor function. The local or host microprocessor would peruse the VCC tables. For each active channel, with TOT_PDU_LEN > 0, the LAST_TS and CREDIT_TS are read along with the CELL_COUNT and LAST_COUNT. The processor then calculates the number of cells per time interval as follows:

```
(CELL_COUNT – LAST_COUNT)  ÷ (LAST_TS – CREDIT_TS)
```

If the number of cells per time unit exceeds the channel specification, the host can be notified via the HOST_MBOX register interrupt. The processor then copies the LAST_TS value to the CREDIT_TS field and the CELL_CNT field to the LAST_COUNT field.

3.5.17 52-Octet Mode

If the 52_EN bit is set in the hash bucket, overhead octets will be written to the cell buffer along with the payload data. In AAL0 and AAL5, the 4-octet ATM header will be written before the payload data is written. In AAL3/4, the ATM header followed by the combined AAL header, and trailer will be written before the payload data. This also applies to OAM/MANAGEMENT cells for the connection if OAM_EN is set.

3.5.18 Receive Cell FIFO

There is a 64-word, receive PHY interface FIFO, between the reassembly coprocessor and the PHY interface. It receives 52-byte cells, formatted into thirteen 32-bit words, from the PHY. The receive cell FIFO is intended to buffer incoming cells from the ATM physical interface until the reassembly coprocessor can complete the processing of previously received cells. It also allows the ATM physical interface clock to be decoupled from the Bt8230 clock without creating metastability problems.

The total FIFO depth for reassembly equals the depth of the DMA master write burst FIFO plus the depth of the receive PHY interface FIFO.

Total Depth = (512 words / 48 bytes per cell) + (64 words / 52 bytes per cell) ≈ 47.58 cells

The receive data buffer depth is important because it impacts the Bt8230's tolerance of PCI bus latency. The PCI Local Bus specification calls for a "highly conservative" 30 microseconds of latency tolerance.

3.5.19 BOM Synchronization Signals

If the STATMODE[3:0] field in CONFIG0 is set to 0xA, the STAT[1,0] pins indicate the BOM. The encoding of the pins differentiate between AAL5, AAL0, and AAL3/4 messages according to Table 3-23. The STAT pins are valid during the address phase of SRC master writes.

Table 3-23. STAT Pin Indications

STAT[1,0]	Indication
00	No BOM/SSM
01	AAL5 BOM/SSM
10	AAL0 Cell
11	AAL3/4 BOM/SSM

3.6 ATM Physical Interface

The ATM physical interface is responsible for communicating with and controlling the ATM link interface chip, which carries out all the transmission convergence and physical, media-dependent functions defined by the ATM protocol. The block performs the following functions:

- Receives and transmits ATM physical interface logic. The ATM physical interface accommodates the Rockwell Bt8222 framer device, a UTOPIA-compatible framer or a Rockwell-conceived slave UTOPIA interface. It is responsible for converting between these devices and the internal data interfaces. The Slave UTOPIA interface connects the Bt8230 to a cell switched backplane.
- Receives cell synchronization logic, which validates cell boundaries in the incoming byte stream, strips off the HEC byte from the ATM header, and formats the remaining 52 bytes into thirteen 32-bit words before passing them to the incoming cell FIFO. The receive cell synchronization logic ensures that only complete cells are passed down to the remainder of the reassembly controller.
- Transmits cell synchronization logic, which converts the 32-bit data read from the transmit cell FIFO into an octet stream, generates appropriate cell delineation pulses for use by the transmit ATM physical interface, and inserts the blank HEC byte into the ATM header of each cell before transferring it to the framer interface.
- Generates and checks odd parity on the octet transmit and receive data buses.

The ATM physical interface contains receive and transmit framer interface logic and receive error generation logic. The ATM physical interface block also interfaces with the segmentation and reassembly coprocessors.

3.6.1 ATM Physical Interface Logic

The Bt8230 ATM physical interface logic consists of the I/O drivers required to communicate with the external framer device, together with adaptation logic required to convert between either the UTOPIA, Bt8222, or Slave UTOPIA interface protocol and the internal byte streams. Configuration pins (FRCFG[1,0]) determine whether the UTOPIA, Bt8222 interface, or slave UTOPIA protocol will be used.

3.6.2 ATM Physical I/O Pins

The operational mode desired is indicated to the Bt8230 by appropriately driving the FRCFG[1,0] input according to the Table 3-24.

Table 3-24. ATM Physical Interface Mode Select

FRCFG[1,0]	ATM Physical Interface Mode
0 0	Bt8222
0 1	UTOPIA
1 0	Slave UTOPIA
1 1	Reserved - Do Not Use

The interpretation of the ATM physical interface pins on the Bt8230 and the actual signals generated or received by the framer in Bt8222 mode are shown in the Table 3-25. The Bt8222 framer needs to be configured to supply a Start of Cell indication. This is done by setting the CELL_VAL[15] bit to a logical 1 in the Bt8222 device.

Table 3-25. Bt8230 Interface Signals

Bt8230 Signal	Bt8222 Signal	Active Polarity	Bt8230 Direction
TXD[7:0]	FDAT_IN[7:0]	—	Out
TXPAR	FDAT_IN[8]	—	Out
TXMARK	FCTRL_OUT[16]	High	In
TXFLAG*	FCTRL_IN[0]	Low	Out
TXEN*	FCTRL_OUT[12]	Low	In
RXD[7:0]	FDAT_OUT[7:0]	—	In
RXPAR	FDAT_OUT[8]	—	In
RXMARK	FCTRL_OUT[10]	High	In
RXFLAG*	FCTRL_IN[4]	Low	Out
RXEN*	FCTRL_OUT[0]	Low	In
FRCTRL	FCTRL_OUT[4]	High	In

The interpretation of the ATM physical interface pins on the Bt8230 and the actual signals generated or received by the framer in UTOPIA mode are shown in Table 3-26. Note that both the TxClk and RxClk signals of the UTOPIA interface may be derived from the CLKD3 output of the Bt8230 (which must also be connected to the FRCTRL input).

Table 3-26. UTOPIA Mode Signals

Bt8230 Signal	PHY Signal	Active Polarity	Bt8230 Direction
TXD[7:0]	TxData[7:0]	–	Out
TXPAR	TxPrty	–	Out
TXMARK	TxSOC	High	Out
TXEN*	TxEnb*	Low	Out
TXFLAG*	TxFull*	Low	In
RXD[7:0]	RxData[7:0]	–	In
RXPAR	RxPrty	–	In
RXMARK	RxSOC	High	In
RXEN*	RxEnb*	Low	Out
RXFLAG*	RxEmpty*	Low	In
FRCTRL	RxClk/TxClk	Rising Edge	In

The interpretation of the ATM physical interface pins on the Bt8230 and the actual signals generated or received by the framer in slave UTOPIA mode are given in Table 3-27.

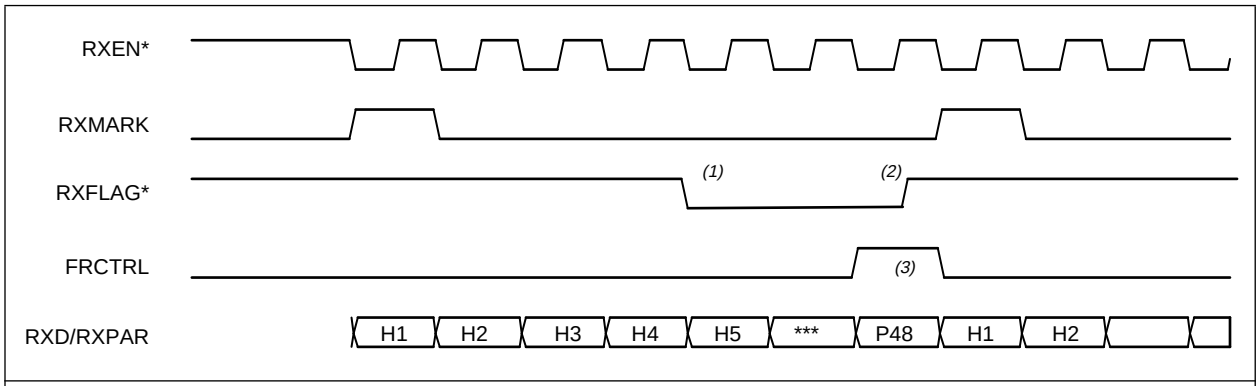
Table 3-27. Slave UTOPIA Mode Interface Signals

Bt8230 Signal	PHY Signal	Active Polarity	Bt8230 Direction
TXD[7:0]	TxData[7:0]	–	Out
TXPAR	TxPrty	–	Out
TXMARK	TxSOC	High	Out
TXEN*	TxEnb*	Low	In
TXFLAG*	TxEmp*	Low	Out
RXD[7:0]	RxData[7:0]	–	In
RXPAR	RxPrty	–	In
RXMARK	RxSOC	High	In
RXEN*	RxEnb*	Low	In
RXFLAG*	RxFull*	Low	Out
FRCTRL	RxClk/TxClk	Rising Edge	In

3.6.3 Bt8222 Mode Timing

As illustrated in Figure 3-31, received data is presented on the RXD[7:0], RXPAR, FRCTRL, and RXMARK by the framer and strobed into the Bt8230 using the RXEN* line. The adaptation logic computes the 8-bit odd parity over the RXD[7:0] lines and compares it to RXPAR. If in error, FR_PAR_ERR [bit 24] is set in the HOST_ISTAT0/LP_ISTAT0 registers. No data is discarded upon a parity error unless the RSM_PHALT bit in the RSM_CTRL register is set to a logic 1. If so, the reassembly coprocessor halts upon a parity error. The Bt8230 asserts the RXFLAG* line to indicate that its internal receive FIFO does not have room for another cell, and no more cells can be accepted. This normally results in cell loss. RXFLAG* is deasserted when there is room for a complete cell in the receive FIFO. The RXMARK input delimits the start of a cell. The FRCTRL signal is used by the adaptation logic to signal to the receive cell synchronizer that the current cell is errored or invalid and should be discarded. Even if the cell is invalid, a complete cell must be transferred or a synchronization error will occur. This signal is sampled on the last octet of each cell. The FR_RMODE bit in the CONFIG0 register must be set to a logical 1 in this mode.

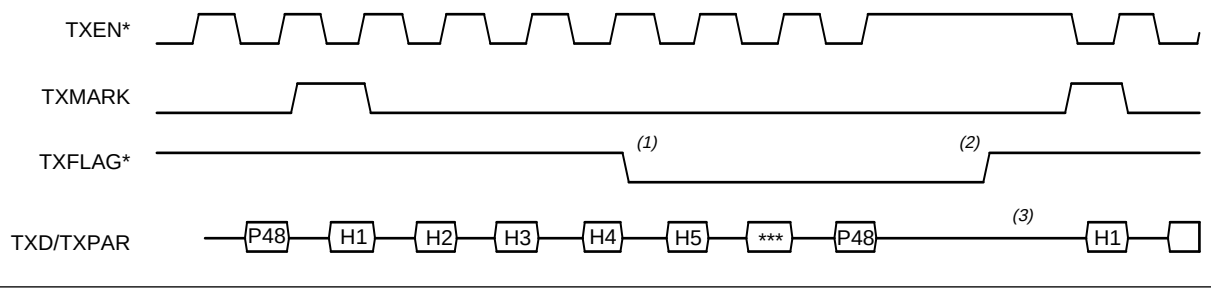
Figure 3-31. Receive Timing in Bt8222 Mode



Notes: (1). RXFLAG* goes active as soon as there is no room for another cell in the receive FIFO. However, there is still room for the current cell.
(2). RXFLAG* goes inactive when there is room for a complete cell in the receive FIFO. A free-running RXEN* is required when RXFLAG* is active.
(3). If FRCTRL is sampled high on the last octet of the cell, the cell is discarded.

Figure 3-32 illustrates how the TXEN* input from the framer is used as a data enable for the TXD[7:0] and TXPAR outputs and the TXMARK input. TXEN* also three-states the TXD and TXPAR outputs when a logical 1 so that multiple ATM SARs can be connected to the Bt8222 device. If another cell is not supplied by the transmit cell synchronization logic, the TXFLAG* output is asserted to indicate that a cell cannot be supplied in the next cell slot. The flag will be asserted after the first 4 bytes of the last cell are transmitted. As before, the TXPAR line carries the 8-bit odd parity computed over TXD[7:0]. The TXMARK line is asserted by the framer to indicate the start of each output cell.

Figure 3-32. Transmit Timing in Bt8222 Mode

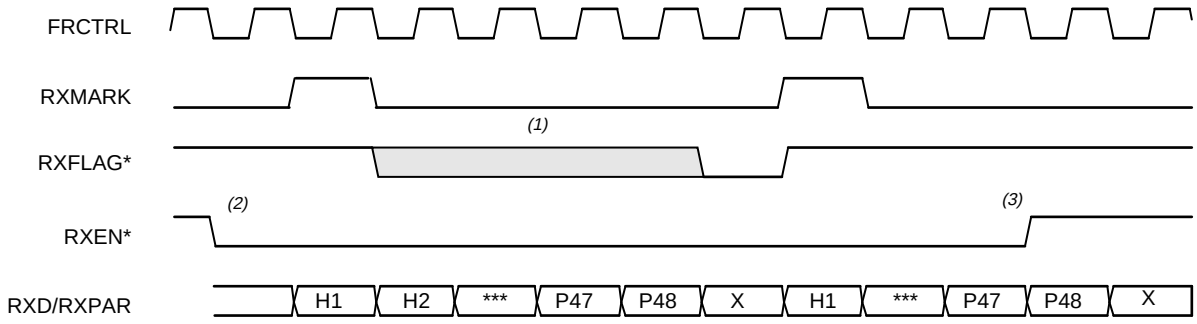


Notes: (1). TXFLAG* goes active when there is no longer a complete cell available in the transmit FIFO. This signal switches asynchronously to TXEN*.
(2). TXFLAG* goes inactive when a complete cell is in the transmit FIFO. This signal switches asynchronously to TXEN*.
(3). TXD and TXPAR are three-stated when TXEN* is a logic high.

3.6.4 UTOPIA Mode Cell Handshake Timing

If set to 1, the UTOPIA_MODE bit in the CONFIG0 register selects cell-level handshaking. Received data is latched from the RXD[7:0] and RXPAR lines on the rising edge of FRCTRL after RXEN* is sampled active (see Figure 3-33). The 8-bit odd parity computed over the RXD[7:0] lines is compared to the RXPAR input. If in error, the FR_PAR_ERR bit is set in the HOST_ISTAT0/LP_ISTAT0 registers. No data is discarded upon a parity error unless the RSM_PHALT bit in the RSM_CTRL register is set to a logic high. If so, the reassembly coprocessor halts upon a parity error. The RXMARK signals to the Bt8230 the start of cell. The RXFLAG* input is the physical layer FIFO empty signal. When it is active, a complete cell is not present in the physical receive FIFO. The physical layer device sets RXFLAG* to inactive when it has a complete cell to transfer. The Bt8230 sets RXEN* to a logic low if it can accept a complete cell. On the clock cycle after the last octet of a cell is transferred, the Bt8230 samples the RXFLAG* input. If a 0, the physical device does not have a cell to transfer. If RXFLAG* is a 1, the physical device has another cell to transfer and the Bt8230 immediately begins receiving the next cell if it can accept a complete cell. The FR_RMODE bit in CONFIG0 should be set to a logical 0 in this mode.

Figure 3-33. Receive Timing in UTOPIA Mode with Cell Handshake

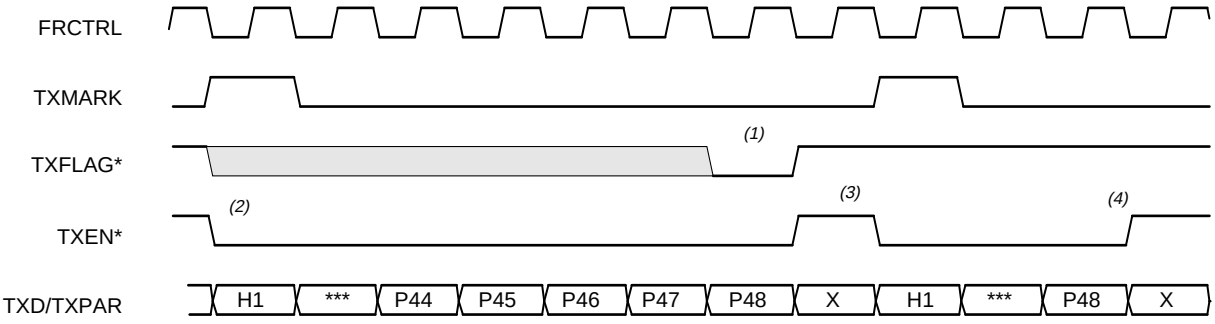


Notes: (1). Once a cell transfer is started, RXFLAG* is not sampled until the end of the cell.
(2). RXEN* goes active only if there is room in the receive FIFO for a complete cell.
(3). RXEN* goes inactive at cell boundaries if the receive FIFO cannot accept another cell.

Transmit data is driven on TXD[7:0] on the rising edge of FRCTRL when TXEN* is asserted, TXEN* is only asserted when there is data in the Bt8230 transmit FIFO. Simultaneously, the 8-bit odd parity computed over the TXD[7:0] lines is driven on to the TXPAR output. The TXMARK line is driven by the Bt8230 device to indicate start of cell. If the TXFLAG* input is asserted by the framer device, the framer device is full, and another cell is not transmitted to the physical framer. (See Figure 3-34.)

In UTOPIA mode, the FRCTRL input can be connected to the Bt8230 CLKD3 output; a 50% duty cycle clock derived by dividing CLK2X by 3.

Figure 3-34. Transmit Timing in UTOPIA Mode with Cell Handshake

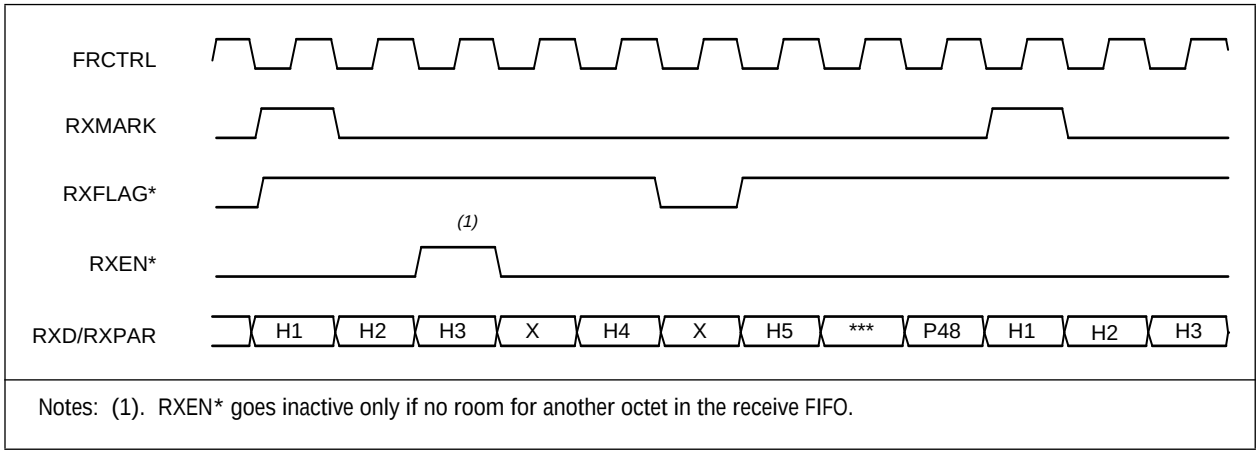


Notes: (1). Once transfer of a cell is started, TXFLAG* is sampled only on the last octet of a cell.
(2). TXEN* goes active if TXFLAG* is inactive at the previous rising clock edge and a complete cell is in the transmit FIFO.
(3). TXEN* goes inactive if TXFLAG* is active on the previous cycle.
(4). TXEN* goes inactive since a complete cell is not in the transmit FIFO.

3.6.5 UTOPIA Mode Octet Handshake Timing

If set to 0, the UTOPIA_MODE bit in the CONFIG0 register selects octet-level handshaking. Received data is latched from the RXD[7:0] and RXPAR lines on the rising edge of FRCTRL after RXEN* is sampled active (see Figure 3-35). The 8-bit odd parity computed over the RXD[7:0] lines is compared to the RXPAR input. If in error, the FR_PAR_ERR bit in the HOST_ISTAT0/LP_ISTAT0 registers is set. No data is discarded upon a parity error unless the RSM_PHALT bit in the RSM_CTRL register is set to a logical 1. If so, the reassembly coprocessor halts upon a parity error. The RXMARK signals the start of cell to the Bt8230. The RXFLAG* input is the physical layer FIFO empty signal. When it is active, no data is present in the physical receive FIFO. The physical layer device sets RXFLAG* to inactive when it has an octet to transfer. The Bt8230 sets RXEN* to a logical 0 if it can accept an octet in the next clock cycle. The FR_RMODE bit in the CONFIG0 register should be set to a logical 0 in this mode.

Figure 3-35. Receive Timing in UTOPIA Mode with Octet Handshake

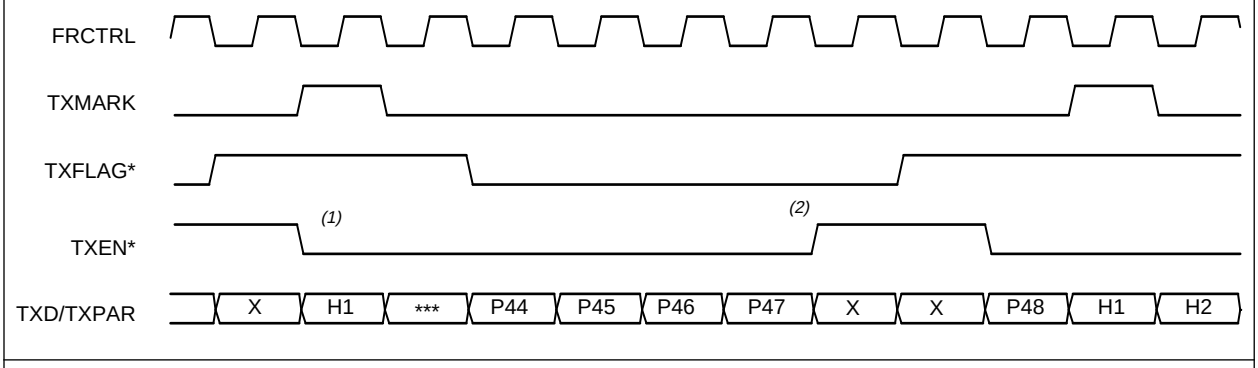


Notes: (1). RXEN* goes inactive only if no room for another octet in the receive FIFO.

Transmit data is driven on TXD[7:0] on the rising edge of FRCTRL when TXEN* is asserted, TXEN* is only asserted when there is data in the Bt8230 transmit FIFO. Simultaneously, the 8-bit odd parity computed over the TXD[7:0] lines is driven on to the TXPAR output. The TXMARK line is driven by the framer device to indicate start of cell. If the TXFLAG* input is asserted by the framer device, the framer device is full and can accept only one to four more octets. (See Figure 3-36).

In UTOPIA mode, the FRCTRL input may be connected to the Bt8230 CLKD3 output, which is a 50% duty cycle clock derived by dividing the CLK2X input by 3.

Figure 3-36. Transmit Timing in UTOPIA Mode with Octet Handshake



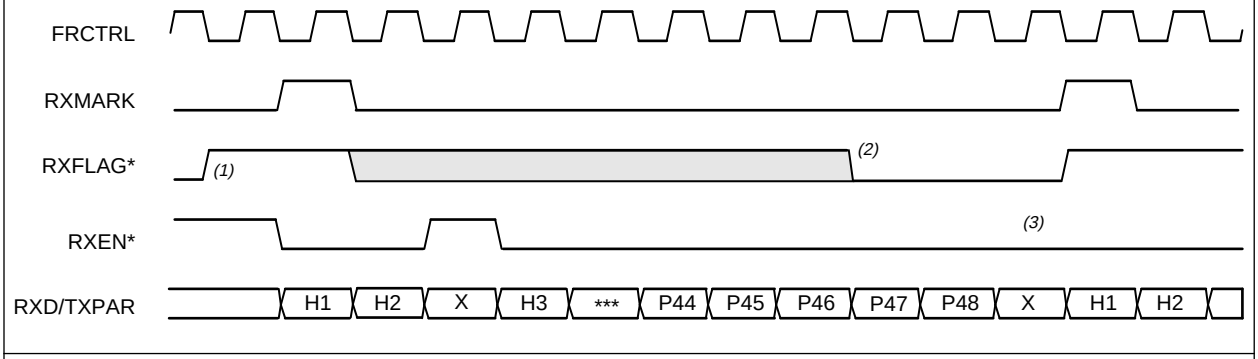
Notes: (1). TXEN* goes active if TXFLAG* is inactive at the previous rising clock edge and a complete cell is in the transmit FIFO.
(2). TXEN* goes inactive one to four clock cycles after TXFLAG* goes active.

3.6.6 Slave UTOPIA Mode

The slave UTOPIA mode is similar to the UTOPIA mode, except the direction of the enable signals and FIFO flags are reversed. This allows a switch fabric or backplane to directly control the physical port. The transmit and receive enable signals are generated by the physical layer instead of the Bt8230. The TXFULL* signal is changed to the TXEMPTY* signal and is an output of the Bt8230. The RXEMPTY* signal is changed to the RXFULL* signal and is also an output of the Bt8230. This mode only supports a cell-level handshake protocol.

Received data is latched from the RXD[7:0] and RXPAR lines on the rising edge of FRCTRL when RXEN* is active (see Figure 3-37). The 8-bit odd parity computed over the RXD[7:0] lines is compared to the RXPAR input. If in error, the FR_PAR_ERR bit is set in the HOST_ISTAT0/LP_ISTAT0 registers. No data is discarded upon a parity error unless the RSM_PHALT bit in the RSM_CTRL register is set to a logical 1. If so, the reassembly coprocessor halts upon a parity error. The RXMARK signals the start of cell to the Bt8230. The RXFLAG* output is the receive FIFO full signal. When it is active, the Bt8230 cannot accept another cell. The Bt8230 sets RXFLAG* to inactive when it has room in the receive FIFO for another cell. The physical device sets RXEN* to a logical 0 if it can transfer an octet. The FR_RMODE bit in the CONFIG0 register should be set to a logic low in this mode.

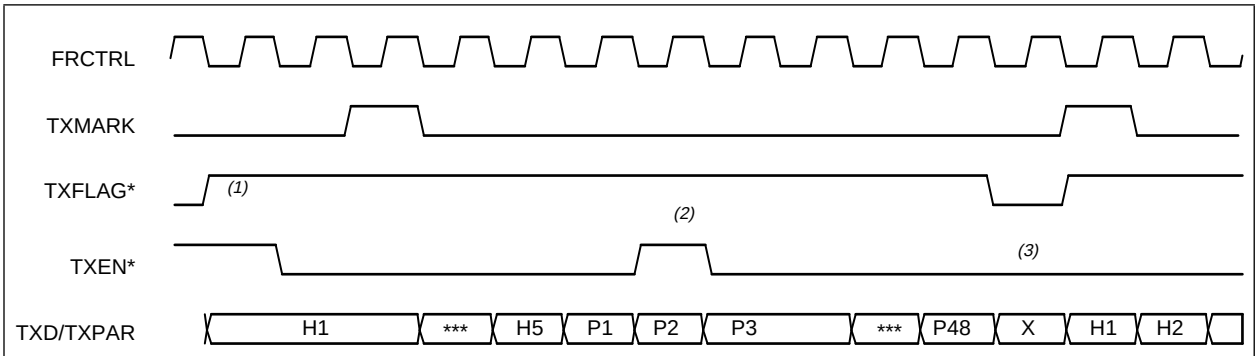
Figure 3-37. Receive Timing in Slave UTOPIA Mode



Notes: (1). RXFLAG* will go inactive when there is room in the receive FIFO for a complete cell.
(2). RXFLAG* will go active when there is no longer room in the receive FIFO for another complete cell.
(3). RXEN* need not be inactive when RXFLAG* is active.

Transmit data is driven on TXD[7:0] on the rising edge of FRCTR after TXEN* is sampled/asserted. Simultaneously, the 8-bit odd parity computed over the TXD[7:0] lines is driven on to the TXPAR output. The TXMARK line is driven by the framer device to indicate start of cell. If the TXFLAG* output is asserted by the Bt8230, the transmit FIFO does not contain a complete cell. (See Figure 3-38).

Figure 3-38. Transmit Timing in Slave UTOPIA Mode



Notes: (1). TXFLAG* goes inactive when a complete cell is in the transmit FIFO.
(2). Example of physical device deactivating TXEN*.
(3). Physical device may keep TXEN* active while TXFLAG* is active.

3.6.7 Loopback Mode

The physical interface can be internally looped by setting the FR_LOOP bit in the CONFIG0 register to a logical 1. This mode uses the internal system clock for operation. Therefore, a framer clock is not needed during loopback operation. All outputs to the Utopia/Bt8222 interface are tri-stated.

NOTE: Reset the Bt8230 after configuring loopback mode.

3.6.8 Receive Cell Synchronization Logic

The receive cell synchronization logic accepts a stream of octets (together with error and cell boundary indications) from the receive ATM physical interface and performs the following functions:

- Maintains a sequence counter that marks the various components of an ATM cell: the 5-byte ATM header, the 1-byte HEC field within the header, and the 48-byte payload. The sequence counter is also used by the ATM physical interface to check cell boundary synchronization. When the sequence counter wraps to 0 it should coincide with the RXMARK (SOC) signal. If not, then a framing error occurs.
- Extracts and discards the HEC byte from each 53-byte ATM cell, leaving 52 bytes of cell data.
- Formats consecutive 4-byte segments into 32-bit words. Thus, the header forms the first word, the first 4 bytes of the payload form the next word, and so on. A total of 13 32-bit words are created from each 52-byte cell after the HEC byte has been removed. The bytes within each word are left-justified (big-endian format); i.e., the first byte received is the most significant byte of the word.
- Ensures that a complete cell (exactly 52 bytes) is always written to the FIFO. If a synchronization error occurs, the FR_SYNC_ERR bit in the HOST_ISTAT0/LP_ISTAT0 registers is set. The ATM physical interface attempts to resynchronize with the data stream. If this error occurs in UTOPIA mode, the cell is placed in the FIFO. If it occurs in 8222 mode, the cell is not placed in the FIFO.
- Sets the RSM_OVFL bit in the HOST_ISTAT0/LP_ISTAT0 registers if an octet could not be transferred due to the receive FIFO being full. RSM_OVFL will not occur when using UTOPIA modes since RXENA is not asserted unless there is room in the RxFIFO.

3.6.9 Transmit Cell Synchronization Logic

The transmit cell synchronization logic copies cell data from the transmit cell FIFO to the transmit ATM physical interface while performing the following functions:

- Reads 32-bit words from the transmit cell FIFO and converts them to a stream of octets, with the most significant byte of each 32-bit word corresponding to the first byte derived from that word (big-endian format).
- Maintains a sequence counter that delineates the various components of each ATM cell (4-byte header, 48-byte payload) in the outgoing byte stream.
- Inserts a blank (all-0) HEC byte, used as a placeholder, into the outgoing byte stream representing each ATM cell. The HEC placeholder is inserted after the first 4 bytes (the ATM header) have been transferred. In Bt8222 mode, the Bt8222 device will calculate and insert the correct HEC value.
- Generates appropriate cell delineation pulses to the transmit ATM physical interface logic, for use in generating the TXMARK* output and also in verifying synchronization with the framer device.
- If the ATM physical transmit interface runs out of cells to transmit, the SEG_UNFL bit is set in the HOST_ISTAT0/LP_ISTAT0 registers.

The transmit cell synchronization logic supplies a continuous stream of octets to the transmit ATM physical interface unit, with cell delineation pulses at the starting byte of every cell. Only complete 53-byte cells are supplied to the ATM physical interface. If the transmit cell FIFO is empty, the transmit cell synchronization logic indicates that no more data can be transferred to the framer.

3.7 PCI Bus Interface

The PCI bus interface is compliant with PCI Local Bus Specification, Revision 2.0, except as described in this datasheet. With the exception of HRST* and HINT*, this interface is completely synchronous to the PCI bus clock (HCLK). All inputs are sampled at the rising edge of HCLK, and all outputs are driven by the Bt8230 to be valid before the next rising edge of HCLK.

The maximum PCI bus clock rate supported by the Bt8230 is 33 MHz. The PCI bus interface logic is clocked directly from the PCI bus clock, while the remainder of the Bt8230 logic runs off separate system clocks. Synchronizing registers and FIFOs are implemented in the PCI bus interface to transfer data between the PCI bus clock (HCLK) and the system clock (SYSCLK) domains.

PCI bus drivers are shared between the master and slave bus interface functional blocks. The PCI bus master logic (within the device) arbitrates via the PCI bus arbiter (external to the device) for access to the PCI bus. Access to the PCI bus automatically implies access to the bus drivers because no other master can be concurrently communicating with the slave logic. The bus master logic contends for the bus on a transaction-by-transaction basis.

The PCI bus interface responds to read and write requests by the host CPU, allowing access to chip resources by host software. The Bt8230 is also capable of acting as a DMA bus master on the PCI bus. As a result, the PCI bus interface implements the full set of address, data, and control signals required to drive the bus as a master, and contains the logic required to support arbitration for the PCI bus. Note that the DMA coprocessor and the PCI bus interface are closely linked and, hence, are shown as one unit.

The PCI bus interface functional blocks are as follows:

- I/O drivers and receivers that drive the pins connected to the PCI bus signals.
- PCI bus master logic that allows the bus interface to acquire mastership of the PCI bus and act as a transaction initiator. The bus master logic also contains a command decoder that interprets access commands generated by the DMA coprocessor, and a burst controller for controlling the duration of each read or write burst. In addition, the bus master logic contains address counters that allow it to restart and retry burst transfers if required by the transaction target.
- Burst FIFO buffers that store and transfer bursts of data words between the DMA coprocessor and the PCI bus master logic.
- PCI bus slave logic that responds to transactions initiated by other masters on the PCI bus with the Bt8230 as a target. The bus slave logic also synchronizes data passed back and forth across the clock boundary between the PCI bus interface and the internal chip logic.
- Configuration registers holding initialization parameters and PCI bus status information.
- Logic that allows the host CPU to read/write the internal Bt8230 registers via the PCI slave port.
- Logic to enable read/write access to the local memory space from the host CPU, again via the PCI slave port.

3.7.1 Nonimplemented PCI Bus Interface Functions

The PCI bus interface on the Bt8230 does not implement all the transaction types defined by the PCI bus specification. Only those sections of the protocol that are necessary for slave and DMA memory accesses are implemented. The following transaction types are not implemented:

- 64-bit transfers, as well as the Dual Address Cycle command.
- Snooping and cache support. The Memory Read Multiple, Memory Read Line and Memory Write and Invalidate commands are internally aliased to the Memory Read and Memory Write commands as per the PCI specification.
- Locked and exclusive accesses. The PCI LOCK* line is not driven by the Bt8230, and the PCI slave interface does not handle locked accesses by other bus masters in any special manner.
- I/O accesses (the I/O Read and I/O Write commands).
- Interrupt acknowledge cycles, including the Interrupt Acknowledge command.
- The Special Cycle command and special cycle transactions.
- Burst transfers that do not have simple, sequentially incrementing addresses for consecutive data phases. The PCI master logic always performs sequentially-incrementing burst transfers. The two LSBs of the PCI address lines (AD[1,0]) must be 0 during the address phase of any transfer made to the PCI slave logic (indicating sequentially incrementing burst addresses). If AD[1,0] is not equal to 0, the slave logic will signal a type A or B target disconnect after the first data phase, forcing the external master to perform a single word transfer as per the PCI specification.

3.7.2 PCI Bus Master Logic

The PCI bus master logic block accepts read and write commands from the DMA coprocessor (passed via the burst FIFO buffers) and in turn acquires mastership of the PCI bus and generates transactions to perform the actual data transfers. The bus master logic contains the following:

- A command decoder, which interprets commands issued from the DMA coprocessor
- A burst controller that counts off read and write cycles in each burst on the PCI bus (and also latches and drives the address and command during the address phase of each transfer)
- Arbitration logic that acquires control of the PCI bus
- A bus state machine that sequences and controls transfers

The bus master implements a software-configurable, maximum burst length counter. This counter is started at the beginning of each read or write burst transaction, and counts the actual number of words transferred during the burst. When it reaches the value set in the MAX_BUR_LEN field of the PCI configuration space, the burst is terminated, and a new address phase is begun.

It is possible for the addressed slave to request a disconnect or a retry during a read or a write transfer, using the defined PCI protocol sequence. In this case, the bus master logic will terminate the current burst, maintain its bus request, and restart the transfer at the point of termination. Disconnects and retries are not regarded as errors.

Five possible sources of error are present during any PCI bus master transaction. If any of the following five errors occur, the bus master logic will permanently terminate the transaction, flag an error, and cease to process any more commands until either the corresponding status flag in the PCI configuration space has been cleared or the PCI master has been reset.

1. Target Abort—The PCI transaction will terminate if the addressed target signals a target abort. In this case, the RTA and MERROR bits in the PCI Configuration Register space will be set, and the PCI_BUS_STATUS[4] bit in the SYS_STAT register will be set.
2. Master Abort—If the addressed target does not respond with an HDEVSEL* assertion, then a master abort is flagged. In this case, the RMA and MERROR bits in the PCI Configuration Register space will be set, and the PCI_BUS_STATUS[3] bit in the SYS_STAT register will be set.
3. Parity Error—If the data parity checked during read transfers is inconsistent with the state of the HPAR signal, then a parity error is signaled. In this case, the DPR and MERROR bits in the PCI Configuration Register space will be set, and the PCI_BUS_STATUS[2] bit in the SYS_STAT register will be set.
4. Interface Disabled—If the driver or application software on the PCI host CPU has disabled the Bt8230 PCI bus master logic (using the M_EN bit in the Command field of the PCI bus configuration registers), any attempt to perform a DMA transaction to the PCI bus will result in an error. In this case, the MERROR and INTF_DIS bits in the PCI configuration space will be set, and the PCI_BUS_STATUS[1] bit in the SYS_STAT register will be set.
5. Internal Failure—Upon a synchronization error between the DMA coprocessor and the PCI master logic, an internal failure will be flagged. In this case, the MERROR and INT_FAIL bits in the PCI configuration space will be set, and the PCI_BUS_STATUS[0] bit in the SYS_STAT register will be set.

As mentioned above, bus protocol errors can be cleared either by a software reset of the associated status flag or flags; i.e., RTA, RMA or DPR, or with a reset of the PCI bus master logic using the HRST* input pin. For example, a master abort error can be cleared by writing a logic 1 to the RMA status bit in the PCI Configuration Register space, causing the status bit to be cleared and the master to resume normal operation. The MERROR bit in the PCI Configuration Register drives the PCI_BUS_ERROR interrupt. To clear this interrupt, a logic high must be written to the MERROR bit location. The MERROR bit can also be cleared by a logic low on the HRST* input pin.

Several fields are provided in the PCI configuration space to aid in recovering from a PCI master error. The PCI host software can determine that an error occurred by checking the MERROR bit. It also can determine if the transaction was a read or write by inspecting the MRD bit, and the read or write address at which an error occurred by reading the CUR_MSTR_RD_ADDR or CUR_MSTR_WR_ADDR fields.

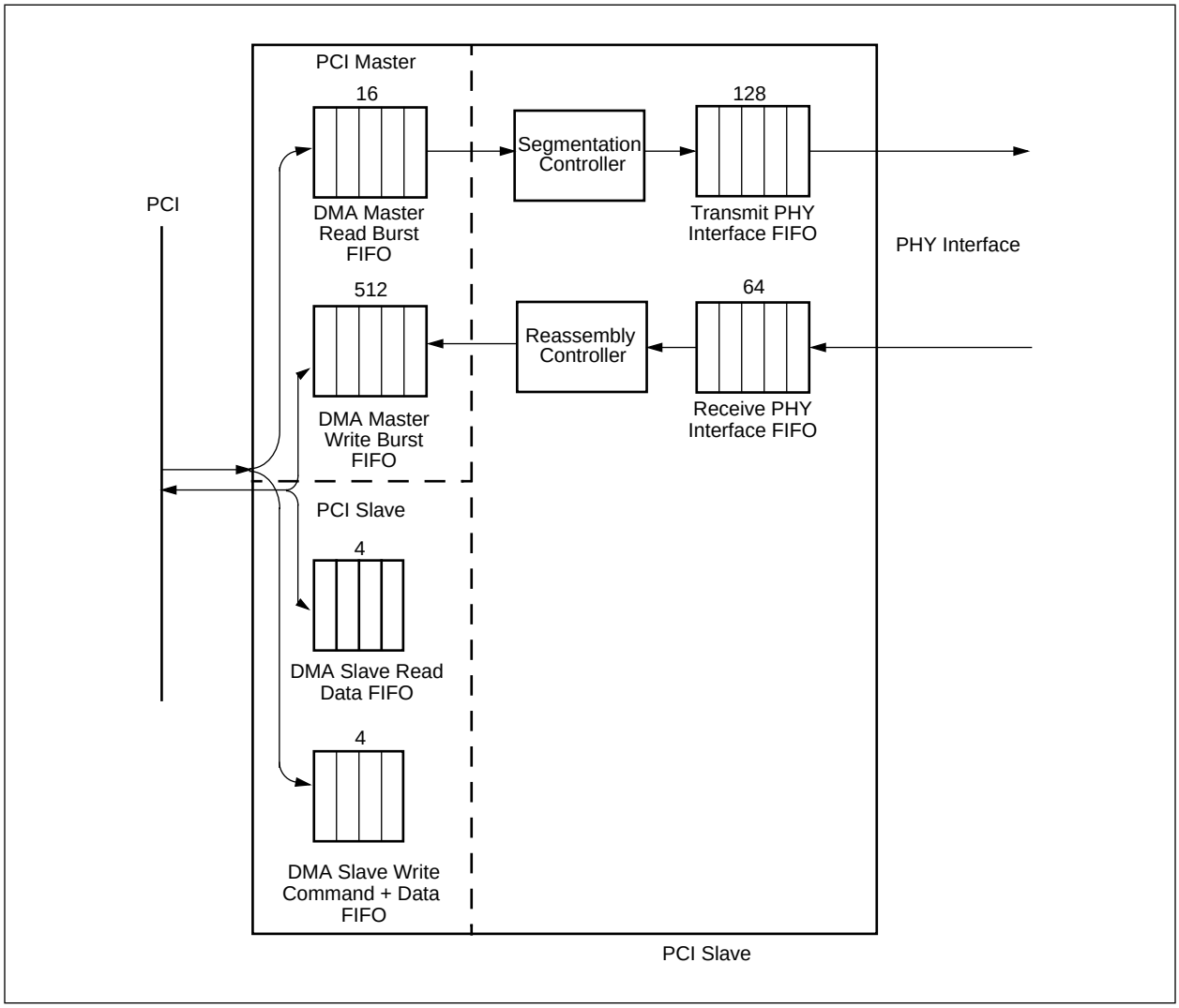
3.7.3 Burst FIFO Buffers

To conserve local memory bandwidth, the Bt8230 does not use its local SRAM as a buffer for incoming or outgoing data. Instead, six dedicated internal FIFOs buffer data:

- Two DMA master burst FIFOs (read, 16 words; write, 512 words)
- Two DMA slave burst FIFOs (4 words each)
- One FIFO between the PHY interface and the segmentation coprocessor (128 words) (see Section 3.4.17)
- One FIFO between the PHY interface and the reassembly coprocessor (64 words) (see Section 3.5.18)

Figure 3-39 illustrates the Bt8230 data FIFOs and their relationship to the PHY interface, PCI bus, and segmentation and reassembly coprocessors.

Figure 3-39. Data FIFOs



3.7.3.1 DMA Master Burst FIFOs

The DMA master read burst and the DMA master write burst FIFOs store actual cell data when the Bt8230 is acting as a DMA master. The DMA master read burst FIFO is 16 words deep. Cells requested from host memory by the segmentation coprocessor are stored in this FIFO. Once the cell is present, the segmentation coprocessor transfers it to the PHY interface FIFO, adding a header as appropriate. The DMA master write burst FIFO is 512 words (2048 bytes) long. This FIFO stores cells that have been processed by the reassembly coprocessor and are to be written to host memory. Typically, these cells will be 48 bytes long because the reassembly coprocessor strips the header off for CPCS-PDU reassembly.

3.7.3.2 DMA Slave Burst FIFOs

The two DMA slave burst FIFOs buffer data during DMA slave accesses to the Bt8230 and local memory resources. The DMA slave read data FIFO is a 4-word (16 bytes) FIFO that stores read data. The DMA slave write command + data FIFO stores up to 4 words of host PCI commands and write data. These two FIFOs provide the host processor with a data path without requiring that ATM cell data be flushed from the DMA master burst FIFOs.

3.7.3.3 Aggregate FIFO Depths

The total FIFO depth for segmentation equals the depth of the DMA master read burst FIFO plus the depth of the transmit PHY interface FIFO.

Total Depth = (16 words / 48 bytes per cell) + (128 words / 52 bytes per cell) ~ = 11.2 cells

The total FIFO depth for reassembly equals the depth of the DMA master write burst FIFO plus the depth of the receive PHY interface FIFO.

Total Depth = (512 words / 48 bytes per cell) + (64 words / 52 bytes per cell) ~ = 47.58 cells

The receive data buffer depth is important because it impacts the Bt8230's tolerance of PCI bus latency. The PCI Local Bus specification calls for a "highly conservative" 30 microseconds of latency tolerance.

3.7.4 PCI Bus Slave Logic

The PCI slave logic permits the host CPU on the PCI bus to access and modify the Bt8230 resources (the external local memory and internal registers). As the control processor also has access to these resources, the PCI slave logic must arbitrate for access prior to performing any read or write transaction. The slave logic also contains the PCI configuration registers. These registers control the PCI slave and master interfaces and can be read or written at any time by the PCI host. The slave logic implements the synchronizers required for rate-matching between the PCI bus clock and the internal Bt8230 system clock. Also, small FIFOs are used to speed up burst reads and writes performed by the host processor to local resources by buffering prefetched read data and absorbing latency during consecutive writes.

In general, the PCI slave interface functions as a normal memory-mapped PCI target, responding to Memory Read, Memory Write, Configuration Read, and Configuration Write commands from any initiator on the PCI bus. The slave interface will only respond to Memory Read and Memory Write commands if the MS_EN bit of the Command field in the PCI Configuration Register has been set.

The PCI slave logic does not implement special cycle commands, or respond to special cycles on the PCI bus. If a master performs a special cycle on the PCI bus the following occurs:

- The slave logic never asserts HDEVSEL*.
- Parity errors during the address phase of the special cycle command will be reported by asserting HSERR* in the normal fashion, if SE_EN and PE_EN in the command register are both set.
- Parity errors during the data phase are ignored.

3.7.5 PCI Host Address Map

The address map of the Bt8230 resources seen by the PCI bus is the same one seen by the local processor. Refer to Section 3.2 for further detail. The base address of the Bt8230 resource mapping is defined in the EXT_MEM_BASE field located in the PCI configuration space.

3.7.6 PCI Configuration Space

For details of the PCI configuration space, refer to Section 4.7.

In accordance with the PCI bus specification Revision 2.0, the Bt8230 PCI bus interface implements a 128-byte configuration register space. These configuration registers can be used by the host processor to initialize, control, and monitor the Bt8230 bus interface logic. The complete definitions of these registers and the relevant fields within them are given in the PCI bus specification. The descriptions of these register fields as implemented in the Bt8230 are provided in Chapter 4.0.

4.0 Registers

This chapter describes the Bt8230's control and status registers. These registers include the following:

- General purpose registers
- Segmentation registers
- Reassembly registers
- Host interface registers
- Local processor interface registers
- Internal Global Management Information Base (MIB) counters
- PCI bus interface configuration registers

Table 4-1provides a list of all the registers. Detailed register descriptions are provided in the sub-sections.

Table 4-1. Address Map

Address	Name	Type	Description
0x00	CLOCK	R/W B, L	Real Time Clock Register
0x04	ALARM1	R/W L, NL	Alarm Register 1
0x08	Reserved	–	Not Implemented
0x0C	SYS_STAT	R/O	System Status Register
0x10	Reserved	–	Not Implemented
0x14	CONFIG0	R/W B, L	Basic Configuration And Control Register
0x18	Reserved	–	Not Implemented
0x1C	Reserved	–	Not Implemented
0x20	SEG_CTRL	R/W B, L	Segmentation Control Register
0x24	SEG_SBASE	R/W B, L	Segmentation Status Queue Base Address Register
0x28	SEG_VBASE	R/W B, L	Segmentation Virtual Connection Area Base Address Register
0x2C	SEG_ST	R/O	Segmentation Status Position Register
0x30	SEG_UBR	R/W B, L	Segmentation Unspecified Bit Rate Parameter Register
0x34	SEG_FR_BD	R/W B, L	SEG Free Buffer Descriptor Register
0x38	SEG_HRBASE	R/W B, L	Segmentation Host Transmit Queue Base Address Register
0x3C	SEG_LRBASE	R/W B, L	Segmentation Local Transmit Queue Base Address Register
0x40	SEG_HXMIT	R/O-W/O B, L	Segmentation Host Transmit Queue Register
0x44	SEG_LXMIT	R/O-W/O B, L	Segmentation Local Transmit Queue Register

Table 4-1. Address Map

Address	Name	Type	Description
0x48–0x6C	Reserved	–	Not Implemented
0x70	RSM_CTRL	R/W B, L	Reassembly Control Register
0x74	RSM_SBASE	R/W B, L	Reassembly Status Queue Base Address Register
0x78	RSM_FBASE	R/W B, L	Reassembly Free Buffer Queue Base Address Register
0x7C	RSM_HBASE	R/W B, L	Reassembly Hash Table Base Address Register
0x80	RSM_TO	R/W R, L	Reassembly Time Out Register
0x84	RSM_FW	R/W R, L	Reassembly Firewall Register
0x88	RS_QBASE	R/W B, L	Reassembly/Segmentation Queue Base Address Register
0x8C–0xBC	Reserved	–	Not Implemented
0xA0	CELL_XMIT_CNT	R/O	MIB Counter - ATM Cells Transmitted
0xA4	CELL_RCVD_CNT	R/O	MIB Counter - ATM Cells Received
0xA8	CELL_DSC_CNT	R/O	MIB Counter - ATM Cells Discarded
0xAC	AAL5_DSC_CNT	R/O	MIB Counter - AAL5 PDUs Discarded
0xC0	HOST_ISTAT0	R/O	Host Interrupt Status Register
0xC4	Reserved	–	Not Implemented
0xC8	Reserved	–	Not Implemented
0xCC	HOST_IMASK0	R/W H, NL	Host Interrupt Mask Register
0xD0	Reserved	–	Not Implemented
0xD4	Reserved	–	Not Implemented
0xD8	HOST_MBOX	R/W L, NL	Host Mailbox Register
0xDC	Reserved	–	Not Implemented
0xE0	LP_ISTAT0	R/O	Local Processor Interrupt Status Register
0xE4	Reserved	–	Not Implemented
0xE8	Reserved	–	Not Implemented
0xEC	LP_IMASK0	R/W L, NL	Local Processor Interrupt Mask Register
0xF0	Reserved	–	Not Implemented
0xF4	Reserved	–	Not Implemented
oxF8	LP_MBOX	R/W H, NL	Local Processor Mailbox Register
0xFC	Reserved	–	Not Implemented

NOTE: Byte enables are ignored by the Bt8230 when writing to control and status registers. Each register description is prefaced with the appropriate abbreviated access types. The access type terminology given in Table 4-2 applies to all registers in this chapter.

Table 4-2. Access Type Terminology

Abbreviation	Description
R/W B	Read and write access for both host and local processors
R/W H	Read and write access for host, read only access for local processor
R/W L	Read and write access for local, read only access for host processor
R/W R	Read and write access for both processors with restrictions
R/O-W/O B	Part of this register is read only, part write only by both processors
R/O	Read only access for both processors
L	Lockable access by LP_LOCK and HOST_LOCK control bits
NL	Non-lockable access by any processor with write capability

4.1 General Purpose Registers

This section describes the Bt8230's general purpose registers which include the Real Time Clock Register, Alarm Register 1, System Status Register, and Configuration Register 0.

4.1.1 0x00—Real Time Clock Register (CLOCK)

Access Types: R/W B, L

The Real Time Clock Register is located at address 0x00. This register contains the 32-bit real time clock. It is incremented by the SYSCLK which has been divided by the DIVIDER[6:0] field in the CONFIG0 register. It can be written to any value by each processor, and can generate an interrupt on the RTC_OVFL status bit in the LP_ISTAT0 register when it overflows from 0xFFFFFFFF to 0x0.

Bit	Field Size	Name	Description
31–0	32	CLOCK[31:0]	Real-time clock value.

4.1.2 0x04—Alarm Register 1 (ALARM1)

Access Types: R/W L, NL

Alarm Register 1 is located at address 0x04. This register contains a 32-bit value which is compared against the CLOCK register. When the two registers are equal, the ALARM1 status bit in the LP_ISTAT0 register is latched and can be enabled to cause an interrupt to the local processor. To implement a periodic interrupt, add a constant value to this register after each interrupt.

Bit	Field Size	Name	Description
31–0	32	ALARM1[31:0]	ALARM1 comparison value.

4.1.3 0x0C—System Status Register (SYS_STAT)

Access Types: R/O

The System Status Register is located at address 0x0C. It provides read-only system status. This register reflects the device ID and version information for the part, as well as pin-programmable options that otherwise may not be visible to the processors. It also contains expanded information for the status located in the HOST_ISTAT0 and LP_ISTAT0 registers.

Bit	Field Size	Name	Description
31–17	15	Reserved	Not implemented at this time
16–12	5	PCI_BUS_STATUS [4:0]	The status bits are as follows: 4: Target Abort 3: Master Abort 2: Parity Error 1: Interface Disabled 0: Internal Failure See Section 3.7 for more detail. Error status is also condensed into the PCI_BUS_ERROR status bit in the HOST_ISTAT0 and LP_STAT0 registers.
11	1	RAMMODE	Reflects the state of the RAMMODE input pin.
10	1	PROCMODE	Reflects the state of the PROCMODE input pin.
9, 8	2	FRCFG[1,0]	Reflects the state of the FRCFG[1,0] input pins.
7–4	4	VERSION [3:0]	Version number -0x1 for Bt8230A, 0x2 for Bt8230B, 0x3 for Bt8230C.
3–0	4	DEVICE[3:0]	Device ID for the Bt8230, set to 0000.

4.1.4 0x14—Configuration Register 0 (CONFIG0)

Access Types: R/W B, L

Configuration Register 0 is located at address 0x14. This register provides all of the control and configuration bits that are not associated with the reassembly and segmentation coprocessors. The majority of these bits are configuration (which occurs at initialization time) and are not changed dynamically. The assertion of the HRST* system reset pin will clear all of the bits in the CONFIG0 register except for MEMCTRL, which will be set high.

Bit	Field Size	Name	Description
31	1	LP_ENABLE	When set, this bit causes the PRST* output pin to be high. This can be used to reset the local processor.
30	1	GLOBAL_RESET	When set, this bit causes reset of the segmentation and reassembly coprocessors as well as all latched status.
29	1	PCI_MSTR_RESET	When set, this bit resets the PCI master logic. Once active, this bit must stay active for 16 cycles of the HCLK input signal.
28–26	3	Reserved	Always set to 0.
25	1	LP_LOCK	When set, this bit disables write access by the host to Bt8230 registers and local memory, with the exception of the LP_MBOX and HOST_IMASK0 registers. Read accesses are not affected. This bit cannot be set by the host processor, nor can it be set if the HOST_LOCK bit is active.
24	1	HOST_LOCK	When set, this bit disables write access by the local processor to SRC registers and local memory, with the exception of the HOST_MBOX and LP_IMASK0 registers. Read accesses are not affected. This bit cannot be set by the local processor. This bit cannot be set if the LP_LOCK bit is active.
23	1	Reserved	Always set to 0.
22	1	PCI_READ_MULTIPLE	When this bit is set, PCI Master implements the PCI Read Multiple Command. Otherwise, the PCI Master implements the PCI Read Command.
21	1	PHY2_EN	Enables the generation of the PHYCS2* signal.
20	1	PCI_ARB	Selects PCI master arbitration scheme. When a logic high, it enables round-robin between read and write requests. When a logic low, reads have priority over writes.
19–16	4	STATMODE[3:0]	Selects which internal status to output on the STAT[1,0] output pins (see Section 3.5.19).
15	1	FR_RMODE	Controls reassembly start of cell processing. When set low, processing starts after the first two words of a cell are received. When set high, a complete cell must be in the reassembly FIFO before the cell is processed.
14	1	FR_LOOP	When set, this bit enables loopback of cells at the ATM physical interface. In Rev. A, the loopback mode only worked when the PHY interface was configured in UTOPIA mode, not Bt8222 mode. In Rev.B and higher, the loopback mode works in either configuration.
13	1	UTOPIA_MODE	Selects byte or cell UTOPIA handshake mode. 0 = Octet handshake 1 = Cell handshake

Bit	Field Size	Name	Description
12	1	ENDIAN	Selects between Little and Big Endian host data structures. 0 = Little Endian 1 = Big Endian
11	1	LP_BWAIT	Selects 0 or 1 wait states between consecutive data cycles during local processor burst accesses. Must be set to 0 in standalone mode.
10	1	MEMCTRL	Selects 0 or 1 wait state for local memory interface (1 or 2 cycle).
9–7	3	BANKSIZE[2:0]	Selects size of memory banks for contiguous memory support. See Section 3.2 for further explanation.
6–0	7	DIVIDER[6:0]	Prescaler for SYSCLK which advances the counter in the CLOCK Register. SYSCLK is divided by the divider value; if 0, it is divided by 128.
Notes: (1). In 8222 mode, this bit must be set high because if there is a sync error, the entire cell is discarded (FIFO write pointer is backed up). In UTOPIA mode, the cell is accumulated into the FIFO despite of the sync error.			

4.2 Segmentation Registers

This section describes the Bt8230's segmentation registers which include Segmentation Control Register, Segmentation Status Queue Base Register, Segmentation Virtual Connection Base Register, Segmentation Status Position Register, Segmentation Unspecified Bit Rate Register, Segmentation Free Buffer Descriptor Register, Segmentation Host Transmit Base Register, Segmentation Local Transmit Base Register, Segmentation Host Transmit Register, and Segmentation Local Transmit Register.

4.2.1 0x20—Segmentation Control Register (SEG_CTRL)

Access Types: R/W B, L

The Segmentation Control Register is located at address 0x20. It contains the general control bits for the segmentation coprocessor. The assertion of the HRST* system reset pin or GLOBAL_RESET bit in the CONFIG0 register will clear the SEG_ENABLE control bit. Table 4-3 shows the segmentation MAXx size encoding.

Bit	Field Size	Name	Description
31	1	SEG_ENABLE	Enables the segmentation coprocessor. If disabled, the segmentation coprocessor will halt on a cell boundary. This bit will also be set low internally when the local or host status queue overflows. In this case, the error condition should be corrected and the SEG_ENABLE bit set high to resume operation.
30	1	SEG_RESET	Resets the segmentation coprocessor and pointers.
29–18	12	Reserved	Program and read as 0.
17	1	INH_IDLE	Setting this bit to 1 inhibits idle cell generation. For Bt8230 Rev B or higher, this bit controls idle cell generation.
16	1	SEG_432	When a logic 0, idle cells contain all 0s. When a logic high, idle cells have an ATM header of 0x0000_0001 and payload octets of 0x6A in compliance with the I.432 standard.
15	1	BOFFSET	When a logic 0, segmentation buffers will include 8 bytes of user-defined data at the beginning of each buffer. When a logic high, no user-defined space is available in the buffers.
14	1	SEG_XMT_ALT	When a logic 0, the host transmit queue will be emptied before the local transmit queue is processed. When a logic high, processing of the rings will alternate when both rings have entries.
13	1	SEG_LS_DIS	Disables the segmentation coprocessor halt on local status queue full condition.
12	1	SEG_HS_DIS	Disables the segmentation coprocessor halt on host status queue full condition
11, 10	2	MAXHR[1,0]	Sets the Host Transmit Queue number of entries. See Table 4-3 for size encoding.
9, 8	2	MAXLR[1,0]	Sets the Local Transmit Queue number of entries. SeeTable 4-3 for size encoding.
7, 6	2	MAXI[1,0]	Maximum rate control I value (intercell interval) in VCC table for rate controlled connections. The slowest possible cell rate is 1/MAXI. See Table 4-3 for size encoding.
5, 4	2	MAXPND[1,0]	Maximum number of VCCs waiting to send a first cell after data (re)start. Also maximum number of UBR VCCs. See Table 4-3 for size encoding.

Bit	Field Size	Name	Description
3, 2	2	MAXHS[1,0]	Maximum number of status messages waiting to be read by the host. The size of the host status queue is MAXHS * 8 bytes. See Table 4-3 for size encoding.
1, 0	2	MAXLS[1,0]	Maximum number of status messages waiting to be read by the local processor. The size of the local status queue is MAXLS * 8 bytes. See Table 4-3 for size encoding.

Table 4-3. Segmentation MAXx Size Encoding

MAXx	Size
00	256
01	1024
10	4096
11	16384

4.2.2 0x24—Segmentation Status Queue Base Register (SEG_SBASE)

Access Types: R/W B, L

The Segmentation Status Queue Base Register is located at address 0x24. This register sets the base address in the Bt8230 local memory for the local and host segmentation status queues. Both base addresses are 128-byte aligned, and only the 16 most significant bits of the address are specified.

Bit	Field Size	Name	Description
31–16	16	SEG_LSBASE[15:0]	Base address for the segmentation local status queue. The size of the local status queue is MAXLS * 4 bytes.
15–0	16	SEG_HSBASE[15:0]	Base address for the segmentation host status queue. The size of the host status queue is MAXHS * 4 bytes.

4.2.3 0x28—Segmentation Virtual Connection Base Register (SEG_VBASE)

Access Types: R/W B, L

The Segmentation Virtual Connection Base Register is located at address 0x28. This register sets the base address in the Bt8230 local memory for the segmentation VCC table and the segmentation reserved area. Both base addresses are 128-byte aligned, and only the 16 most significant bits of the address are specified.

Bit	Field Size	Name	Description
31–16	16	SEG_RSVDB[15:0]	Base address segmentation reserved area. The size of the reserved area is (MAXI + MAXPND) * 4 bytes.
15–0	16	SEG_VCCB[15:0]	Base address for the segmentation VCC table.

4.2.4 0x2C—Segmentation Status Position Register (SEG_ST)

Access Types: R/O

This Segmentation Status Position Register is located at address 0x2C. This register shows the current position of the segmentation coprocessor in the local and host status queues. The values in the register are pointer offsets from the base address of the appropriate status queue. The current host status queue entry is at address SEG_HSB * 128 + (SEG_HPTR – 1) * 4. The current local status queue entry is at address SEG_LSB * 128 + (SEG_LPTR – 1) * 4.

Bit	Field Size	Name	Description
31, 30	2	Reserved	Always read as 0.
29–16	14	SEG_LPTR[13:0]	Current segmentation coprocessor position in local status queue.
15, 14	2	Reserved	Always read as 0.
13–0	14	SEG_HPTR [13:0]	Current segmentation coprocessor position in host status queue.

4.2.5 0x30—Segmentation Unspecified Bit Rate Register (SEG_UBR)

This register name and bit names in this register have been changed in this datasheet to promote a higher degree of clarity. The register was previously called SEG_ABR, and the bit names also contained ABR which has been changed to UBR. The original names still exist in software and software documentation.

Access Types: R/W B, L

The Segmentation Unspecified Bit Rate Register is located at address 0x30. It sets the maximum rate parameters for the overall segmentation UBR cell stream. The *I* and *L* parameters must be limited to *I* + *L* **less than or equal to** 0x3FFFF.

Bit	Field Size	Name	Description
31	1	UBR_GCRA_EN	Enables UBR control when set to 1. If this bit is set to 0, UBR cells will be sent during all non-scheduled cell slots.
30–13	18	UBRL[17:0]	Rate Control *L* Field. This field is the corresponding parameter in the ATM UNI 3.1 GCRA specification (Section 3.6.2.4.1). This parameter is a fixed point number with 8 fractional bits and is in units of cell slots.
12–0	13	UBRI[12:0]	Rate Control *I* Field. This field is the corresponding parameter in the ATM UNI 3.1 GCRA specification (Section 3.6.2.4.1). This parameter is a fixed point number with 8 fractional bits and is in units of cell slots.

4.2.6 0x34—Segmentation Free Buffer Descriptor Register (SEG_FR_BD)

Access Types: R/W R, L

The Segmentation Free Buffer Descriptor Register is located at address 0x34. This register can be written only if the SEG_RUN status bit in the HOST_ISTAT0 and LP_ISTAT0 registers is a logic low, indicating that the segmentation coprocessor is halted.

Bit	Field Size	Name	Description
31–23	9	Reserved	Always program and read as 0.
22–2	21	NXT_FREE [20:0]	The word-aligned address of the next free buffer descriptor.
1, 0	2	Reserved	Always program and read as 0.

4.2.7 0x38—Segmentation Host Transmit Base Register (SEG_HRBASE)

Access Types: R/W B, L

The Segmentation Host Transmit Base Register is located at address 0x38. This register sets the base address of the Host Transmit Queue.

Bit	Field Size	Name	Description
31–0	32	SEG_HRBASE[31:0]	Base address of the Host Transmit Queue. Must be word (32-bit) aligned.

4.2.8 0x3C—Segmentation Local Transmit Base Register (SEG_LRBASE)

Access Types: R/W B, L

The Segmentation Local Transmit Base Register is located at address 0x3C. This register sets the base address of the Local Transmit Queue.

Bit	Field Size	Name	Description
31–16	16	Reserved	Always program and read as 0.
15–0	16	SEG_LRBASE [15:0]	Base address of the Local Transmit Queue. Must be 128 byte aligned.

4.2.9 0x40—Segmentation Host Transmit Register (SEG_HXMIT)

Access Types: R/O-W/O B, L

The Segmentation Host Transmit Register is located at address 0x40. This register contains both the current number of entries on the Host Transmit Queue and the number of new entries to add to the transmit queue**.** The SEG_HPND field in this register is limited to the size set in the MAXHR field in the SEG_CTRL register; it will not roll over. Exceeding this value will cause the HRING_OVFL status in the HOST_ISTAT0 and LP_ISTAT0 registers to be set.

Bit	Field Size	Name	Description
31–30	2	Reserved	Always program and read as 0.
29–16	14	SEG_HPND[13:0]	SEG_HPND is the number of entries on the Host Transmit Queue that have not been processed. This field is read only.
15–8	8	Reserved	Always program and read as 0.
7–0	8	SEG_HNEW[7:0]	SEG_HNEW is the number of entries added by a processor to the Host Transmit Queue. These bits are automatically added to SEG_HPND and cleared. They are read as 0s.

4.2.10 0x44—Segmentation Local Transmit Register (SEG_LXMIT)

Access Types: R/O-W/O B, L

The Segmentation Local Transmit Register is located at address 0x44. This register contains both the current number of entries on the Local Transmit Queue and the number of new entries to add to the transmit queue. The SEG_LPND field in this register is limited to the size set in the MAXLR field in the SEG_CTRL register; it will not roll over. Exceeding this value will cause the LRING_OVFL status in the HOST_ISTAT0 and LP_ISTAT0 registers to be set.

Bit	Field Size	Name	Description
31–30	2	Reserved	Always program and read as 0.
29–16	14	SEG_LPND[13:0]	SEG_LPND is the number of entries on the Local Transmit Queue that have not been processed. This field is read only.
15–8	8	Reserved	Always program and read as 0.
7–0	8	SEG_LNEW[7:0]	SEG_LNEW is the number of entries added by a processor to the Local Transmit Queue. These bits are automatically added to SEG_LPND and cleared, they will read as 0s.

4.3 Reassembly Registers

This section describes the Bt8230's reassembly registers which include Reassembly Control Register, Reassembly Status Queue Base Register, Reassembly Free Buffer Queue Base Register, Reassembly Hash Table Base Register, Reassembly Time-out Register, Reassembly Firewall Register, and Reassembly/Segmentation Queue Register.

4.3.1 0x70—Reassembly Control Register (RSM_CTRL)

Access Types: R/W B, L

The Reassembly Control Register is located at address 0x70 and contains the general control bits for the reassembly coprocessor. The assertion of the HRST* system reset pin or the GLOBAL_RESET bit in the CONFIG0 register will clear the RSM_ENABLE control bit. Table 4-4 shows TAB_SIZE encoding and Table 4-5 gives reassembly buffer size coding.

Bit	Field Size	Name	Description
31	1	RSM_ENABLE	Reassembly enable. Initiates an incoming transfer if set, and halts it if reset. If this bit is reset while the reassembly coprocessor is running, it temporarily suspends the activities of the reassembly coprocessor logic. Suspension takes place on a cell boundary; i.e., between the completion of all processing and transfers required for the current cell and the start of processing for the next cell. The hold can be removed and the transfer resumed by setting the RSM_ENABLE bit. This bit will also be set low internally on certain reassembly error conditions. These include free buffer queue empty, status queue full, or parity error with PHALT_EN. In this case, the error condition should be corrected and the RSM_ENABLE bit set high to resume operation.
30	1	RSM_RESET	Reassembly reset. Forces a hardware reset of the reassembly coprocessor when asserted. It must be deasserted before the reassembly coprocessor will resume normal operation.
29–23	7	Reserved	Not implemented at this time.
22	1	TS_SEL	Timestamp Select. Used to select either 16 LSBs or 16 MSBs of the CLOCK register for reassembly timestamps PDU_TS and LAST_TS. A logic 0 selects the lower 16 bits. A logic 1 selects the upper 16 bits. (LSBs selection retains reverse software compatibility.)
21	1	LNK_DIS	When this bit is a logic 1, the SCR will not write the data buffer NEXT pointer. This bit should only be set in Streaming mode.
20–17	4	TAB_SIZE[3:0]	VPI/VCI hash table size. The TAB_SIZE field indicates the size, in powers of 2, of the hash table used by the reassembly coprocessor to manage incoming cell streams and perform the per-connection linked-list generation task. The values in the field are interpreted according to Table 4-4.
16	1	RSM_PHALT	Reassembly coprocessor halt on parity error detect. The reassembly coprocessor will halt the incoming channel logic if a parity error is detected and the RSM_PHALT bit is set.

Bit	Field Size	Name	Description
15	1	OAM_EN	Operating and maintenance enable. Enables detection and processing of OAM cells.
14	1	OAM_MEM	Operating and maintenance memory. OAM processing is sent to local memory instead of host memory when set.
13	1	FWALL_EN	Firewall enable. Enables free buffer queue firewalling of user cells. If set, this bit enables the per-connection, free-buffer queue firewall. Each connection that firewall is active in must have the FW_EN bit set to a logic high in the hash bucket.
12	1	CACHE_ENABLE	Cache not implemented. This bit must be set to 0.
11	1	RSM_LF_DIS	Local Free Buffer Queue Overwrite Disable. Disables the local Free Buffer Queue read checks if set.
10	1	RSM_HF_DIS	Host Free Buffer Queue Overwrite Disable. Disables the host free buffer queue read checks if set.
9	1	RSM_LS_DIS	Local Status Queue Overwrite Disable. Disables the local status queue write checks if set.
8	1	RSM_HS_DIS	Host Status Queue Overwrite Disable. Disables the host status queue write checks if set.
7, 6	2	HF_SIZE	Set Size Of Host Free Buffer Queue. Sets the number of entries of the host free buffer queue as shown in Table 4-5.
5, 4	2	HS_SIZE	Set Size of Host Status Queue. Sets the number of entries of the host status queue as shown in Table 4-5.
3, 2	2	LF_SIZE	Set Size of Local Free Buffer Queue. Sets the number of entries of the local free buffer queue as shown in Table 4-5.
1, 0	2	LS_SIZE	Set Size of Local Status Queue. Sets the number of entries of the local status queue as shown in Table 4-5.

Table 4-4. TAB_SIZE Encoding

TAB_SIZE	Hash Table Size, Entries
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024
11	2048
12	4096
13	Invalid
14	16384
15	Invalid

Table 4-5. Reassembly Buffer Size Coding

XX_SIZE	Buffer Size
00	256
01	1024
10	4096
11	16384

4.3.2 0x74—Reassembly Status Queue Base Register (RSM_SBASE)

Access Types: R/W B, L

The Reassembly Status Queue Base Register is located at address 0x74. This register determines the base address of both the host and local status queue structures. The base address is a 16-bit number. Since both local and host status queues reside in local memory (23 bits of byte addressing) the structures can start on 128-byte boundaries.

Bit	Field Size	Name	Description
31–16	16	LST_BASE[15:0]	Local status queue base address.
15–0	16	HST_BASE[15:0]	Host status queue base address.

4.3.3 0x78—Reassembly Free Buffer Queue Base Register (RSM_FBASE)

Access Types: R/W B, L

The Reassembly Free Buffer Queue Base Register is located at address 0x78. This register determines the base address of both the host and local free buffer queue structures. The base address is a 16-bit number. Since both local and host free buffer queues reside in local memory (23 bits of byte addressing) the structures can start on 128-byte boundaries.

Bit	Field Size	Name	Description
31–16	16	LFR_BASE[15:0]	Local free buffer queue base address.
15–0	16	HFR_BASE[15:0]	Host free buffer queue base address.

4.3.4 0x7C—Reassembly Hash Table Base Register (RSM_HBASE)

Access Types: R/W B, L

The Reassembly Hash Table Base Register is located at address 0x7C. This register consists of a 16-bit address pointer that points to the beginning of the hash table. Since the hash table resides in local memory, the table starts on 128-byte boundaries. It contains the base address of the hash table used by the reassembly coprocessor to process incoming cell streams associated with different virtual connections. The size of this hash table is given by the TAB_SIZE field in the RSM_CTRL register. It is always a power of 2.

Bit	Field Size	Name	Description
31–16	16	Reserved	Program and read as 0.
15–0	16	RSM_HBASE[15:0]	Hash table base address.

4.3.5 0x80—Reassembly Time-out Register (RSM_TO)

Access Types: R/W R, L

The Reassembly Time-out Register is located at address 0x80. This register is written by the local or host processor to point to the beginning of the VCC table of a connection that has timed out. The LSTAT bit is set high if a status entry is to be sent to the local status queue after the reassembly VCC table is reset. Otherwise, a status entry is written to the host status queue. The reassembly coprocessor will NULL this register after it has processed the time-out. The processor can only write this register when the contents are NULL. Writing when the contents are non-0 will have no effect.

Bit	Field Size	Name	Description
31–25	7	Reserved	Program and read as 0.
24	1	TMO_AAL34	If this bit is set high, the VCC table is initialized as an AAL3/4 connection; otherwise, it is initialized as an AAL5 connection.
23	1	LSTAT	If this bit is set high, status entry is sent to local status queue; otherwise, status entry is sent to host status queue.
22–2	21	TOPNTR[20:0]	Address pointer to timed out connection.
1, 0	2	Reserved	Program and read as 0.

4.3.6 0x84—Reassembly Firewall Register (RSM_FW)

Access Types: R/W R, L

The Reassembly Firewall Register is located at address 0x84. This register consists of a 21-bit address pointer, FWPNTR, and a credit update field, FWCREDIT. FWPNTR points to the base address of the VCC table that is to be incremented. The CBUFF_CNT field is incremented by the value in the FWCREDIT field. The reassembly coprocessor will NULL this register after it has processed the credit update. The processor should only write this register when the contents are NULL.

Bit	Field Size	Name	Description
31–23	9	FWCREDIT[8:0]	Credit update field.
22–2	21	FWPNTR[20:0]	Address pointer.
1, 0	2	Reserved	Program and read as 0.

4.3.7 0x88—Reassembly/Segmentation Queue Register (RS_QBASE)

Access Types: R/W B, L

The Reassembly/Segmentation Queue Register is located at address 0x88. This register contains the 128-byte aligned base address of the reassembly/segmentation queue structure.

Bit	Field Size	Name	Description
31–16	16	Reserved	Program and read as 0.
15–0	16	RS_QBASE[15:0]	Base address of the reassembly/segmentation queue.

4.4 Host Interface Registers

This section describes the Bt8230's host interface registers which include Host Interrupt Status Register, Host Interrupt Mask Register, and Host Mailbox Register.

4.4.1 0xC0—Host Interrupt Status Register (HOST_ISTAT0)

Access Types: R/O

The Host Interrupt Status Register is located at address 0xC0. This register contains all interruptible status for the host processor. The corresponding interrupt enables are located in the HOST_IMASK0 register. Status types are defined as:

L	Level-sensitive status—A logic 1 on the status bit will cause an interrupt when enabled by the corresponding IMASK bit. Reading the status does not clear the status or interrupt. The source of the condition causing the status must be cleared before the status or interrupt is cleared.
E	Event driven status—A 0→1 transition on the status bit causes an interrupt when enabled. Reading the status register clears the status bit and the interrupt.
DE	Dual Event status—A 0→1 and 1→0 transition on the status bit can be enabled to cause an interrupt. Reading the status register clears the status bit and the interrupt.
Note: Only host reads will reset the status bits in the HOST_ISTAT0 register.	

Bit	Field Size	Type	Name	Description
31	1	L	PFAIL	Reflects inverted state of processor PFAIL* input.
30	1	L	PHY_INTR	In standalone operation, this bit reflects the inverted state of the PDAEN input. PHY_INTR can be connected to a PHY interrupt source.
29	1	DE	LP_LOCK_EVENT	This bit is set on a 0→1 or 1→0 transition of the LP_LOCK control bit. Read the CONFIG0 register for the current state of LP_LOCK.
28	1	E	HOST_MBOX_ WRITTEN	This bit is set upon a write to the HOST_MBOX register by the local processor, and cleared by a read of the HOST_MBOX register.
27	1	E	LP_MBOX_READ	This bit is set upon the read of the LP_MBOX register by the local processor.
26	1	L	PCI_BUS_ERROR	This bit is set upon the occurrence of a PCI master bus error. Read the SYS_STAT register for more information of the type of error. Refer to Section 3.7 to reset this interrupt.
25	1	E	DMA_FULL	Set when the incoming DMA burst FIFO becomes full.
24	1	E	FR_PAR_ERR	Set on the occurrence of a parity error on the reassembly ATM physical interface.

Bit	Field Size	Type	Name	Description
23	1	E	FR_SYNC_ERR	Set on the occurrence of a synchronization error on the reassembly ATM physical interface.
22	1	–	Reserved	Read as 0.
21	1	E	CELL_XMIT_RLOVR	Set on the occurrence of a CELL_XMIT_CNT rollover.
20	1	E	RS_QUEUE_FULL	Reassembly/segmentation queue full condition.
19	1	L	RSM_RUN	Set when the reassembly machine is running. Will be high when RSM_ENABLE bit in RSM_CTRL is high or when processing the last cell after RSM_ENABLE is set low.
18	1	E	RSM_OVFL	Reassembly overflow. Indicates that a cell was lost due to a FIFO full condition.
17	1	L	RSM_HS_FULL	Set on the occurrence of a host status queue full condition.
16	1	L	RSM_LS_FULL	Set on the occurrence of a local status queue full condition.
15	1	L	RSM_HF_EMPT	Set on the occurrence of a host free buffer queue empty condition.
14	1	L	RSM_LF_EMPT	Set on the occurrence of a local free buffer queue empty condition.
13	1	E	RSM_HS_WRITE	Indicates reassembly host status has been written by the Bt8230.
12	1	E	RSM_LS_WRITE	Indicates reassembly local status has been written by the Bt8230.
11	1	E	CELL_RCVD_RLOVR	Set on the occurrence of a CELL_RCVD_CNT rollover.
10	1	E	CELL_DSC_RLOVR	Set on the occurrence of a CELL_DSC_CTN rollover.
9	1	E	AAL5_DSC_RLOVR	Set on the occurrence of a AAL5_DSC_CNT rollover.
8	1	L	SEG_RUN	Set when the segmentation machine is running. Will be high when SEG_ENABLE bit in SEG_CTRL is high or when processing the last cell after SEG_ENABLE is low.
7	1	E	SEG_UNFL	Segmentation underflow indicates that an idle cell was sent instead of a scheduled cell due to lack of PCI bandwidth.
6	1	E	HRING_OVFL	Host Transmit Queue overflow.
5	1	E	LRING_OVFL	Local Transmit Queue overflow.
4	1	L	BD_FULL	Buffer descriptor space full.
3	1	L	SEG_HS_FULL	Indicates that the segmentation host status queue is full.
2	1	L	SEG_LS_FULL	Indicates that the segmentation local status queue is full.
1	1	E	SEG_HS_WRITE	Indicates that the segmentation host status queue has been written by the Bt8230.
0	1	E	SEG_LS_WRITE	Indicates that the segmentation local status queue has been written by the Bt8230.

4.4.2 0xCC—Host Interrupt Mask Register (HOST_IMASK0)

Access Types: R/W H, NL

The Host Interrupt Mask Register is located at address 0xCC. This register contains the interrupt enables that correspond to the status in the HOST_ISTAT0 register. The assertion of the HRST* system reset pin will clear all of the HOST_IMASK0 interrupt enables.

Bit	Field Size	Name	Description
31	1	EN_PFAIL	Enables an interrupt when PFAIL status is a logic 1.
30	1	EN_PHY_INTR	Enables an interrupt when PHY_INTR status is a logic 1.
29	1	EN_LP_LOCK_EVENT	Enables an interrupt when LP_LOCK_EVENT status is a logic 1.
28	1	EN_HOST_MBOX _WRITTEN	Enables an interrupt when HOST_MBOX_WRITTEN status is a logic 1.
27	1	EN_LP_MBOX_READ	Enables an interrupt when LP_MBOX_READ status is a logic 1.
26	1	EN_PCI_BUS_ERROR	Enables an interrupt when PCI_BUS_ERROR status is a logic 1.
25	1	EN_DMA_FULL	Enables an interrupt when DMA_FULL status is a logic 1.
24	1	EN_FR_PAR_ERR	Enables an interrupt when FR_PAR_ERR status is a logic 1.
23	1	EN_FR_SYNC_ERR	Enables an interrupt when FR_SYNC_ERR status is a logic 1.
22	1	Reserved	Reserved, set to 0.
21	1	EN_CELL_XMIT_RLOVR	Enables an interrupt when CELL_XMIT_RLOVR status is a logic high.
20	1	EN_RSQUEUE_FULL	Enables an interrupt when RSQUEUE_FULL status is a logic 1.
19	1	EN_RSM_RUN	Enables an interrupt when RSM_RUN status is a logic 1.
18	1	EN_RSM_OVFL	Enables an interrupt when RSM_OVFL status is a logic 1.
17	1	EN_RSM_HS_FULL	Enables an interrupt when RSM_HS_FULL status is a logic 1.
16	1	EN_RSM_LS_FULL	Enables an interrupt when RSM_LS_FULL status is a logic high.
15	1	EN_RSM_HF_EMPT	Enables an interrupt when RSM_HF_EMPT status is a logic high.
14	1	EN_RSM_LF_EMPT	Enables an interrupt when RSM_LF_EMPT status is a logic high.
13	1	EN_RSM_HS_WRITE	Enables an interrupt when RSM_HS_WRITE status is a logic high.
12	1	EN_RSM_LS_WRITE	Enables an interrupt when RSM_LS_WRITE status is a logic high.
11	1	EN_CELL_RCVD_RLOVR	Enables an interrupt when CELL_RCVD_RLOVR status is a logic high.
10	1	EN_CELL_DSC_RLOVR	Enables an interrupt when CELL_DSC_RLOVR status is a logic high.
9	1	EN_AAL5_DSC_RLOVR	Enables an interrupt when AAL5_DSC_RLOVR status is a logic high.
8	1	EN_SEG_RUN	Enables an interrupt when SEG_RUN status is a logic high.
7	1	EN_SEG_UNFL	Enables an interrupt when SEG_UNFL status is a logic high.
6	1	EN_HRING_OVFL	Enables an interrupt when HRING_OVFL status is a logic high.
5	1	EN_LRING_OVFL	Enables an interrupt when LRING_OVFL status is a logic high.
4	1	EN_BD_FULL	Enables an interrupt when the BD_FULL status is a logic high.

Bit	Field Size	Name	Description
3	1	EN_SEG_HS_FULL	Enables an interrupt when SEG_HS_FULL status is a logic high.
2	1	EN_SEG_LS_FULL	Enables an interrupt when SEG_LS_FULL status is a logic high.
1	1	EN_SEG_HS_WRITE	Enables an interrupt when SEG_HS_WRITE status is a logic high.
0	1	EN_SEG_LS_WRITE	Enables an interrupt when SEG_LS_WRITE status is a logic high.

4.4.3 0xD8—Host Mailbox Register (HOST_MBOX)

Access Types: R/W L, NL

The Host Mailbox Register is located at address 0xD8. This register implements a mailbox for communication between the host and local processors. The register is written by the local processor and read by the host to pass messages in that direction. Writes to this register can interrupt the host while reads can interrupt the local processor.

Bit	Field Size	Name	Description
31–0	32	HOST_MBOX[31:0]	Messages flow from local processor to host.

4.5 Local Processor Interface Registers

This section describes the Bt8230's local processor interface registers which include Local Processor Interrupt Status Register, Local Processor Interrupt Mask Register, and Local Processor Mailbox Register.

4.5.1 0xE0—Local Processor Interrupt Status Register (LP_ISTAT0)

Access Types: R/O

The Local Processor Interrupt Status Register is located at address 0xE0 and contains all the interruptible status for the local processor. The corresponding interrupt enables are located in the LP_IMASK0 register. Status types are defined as:

L	Level sensitive status—A logic 1 on the status bit will cause an interrupt when enabled by the corresponding IMASK bit. Reading the status does not clear the status or interrupt. The source of the condition causing the status must be cleared before the status or interrupt is cleared.
E	Event driven status—A 0→1 transition on the status bit causes an interrupt when enabled. Reading the status register clears the status bit and the interrupt.
DE	Dual event status—A 0→1 and 1→0 transition on the status bit can be enabled to cause an interrupt. Reading the status register clears the status bit and the interrupt.
Note:	Only local processor reads will reset the status bits in the LP_ISTAT0 register.

Bit	Field Size	Type	Name	Description
31	1	E	RTC_OVFL	Clock register overflow.
30	1	E	ALARM1	Set when ALARM1 register matches CLOCK register.
29	1	L	HOST_LOCK_STAT	Set on a 0⟶1 or 1⟶0 transition of the HOST_LOCK control bit. Read the CONFIG0 register for the current state of HOST_LOCK.
28	1	E	LP_MBOX_WRITTEN	Set upon a write to the LP_MBOX register by the host processor.
27	1	E	HOST_MBOX_READ	Set upon the read of the HOST_MBOX register by the host processor.
26	1	L	PCI_BUS_ERROR	Set upon the occurrence of a PCI master bus error. Read the SYS_STAT register for more information of the type of error. Refer to Section 3.7 to reset this interrupt.
25	1	E	DMA_FULL	Set when the incoming DMA burst FIFO becomes full.
24	1	E	FR_PAR_ERR	Set on the occurrence of a parity error on the reassembly ATM physical interface.
23	1	E	FR_SYNC_ERR	Set on the occurrence of a synchronization error on the reassembly ATM physical interface.
22	1	-	Reserved	Reserved, read as 0.

Bit	Field Size	Type	Name	Description
21	1	E	CELL_XMIT_RLOVR	Set on the occurrence of a CELL_XMIT_CNT rollover.
20	1	E	RS_QUEUE_FULL	Reassembly/segmentation queue full condition.
19	1	L	RSM_RUN	Set when the reassembly machine is running. Will be high when RSM_ENABLE bit in RSM_CTRL is high or when processing the last cell after RSM_ENABLE is set low.
18	1	E	RSM_OVFL	Reassembly overflow. A cell was lost due to a FIFO full condition.
17	1	L	RSM_HS_FULL	Set on the occurrence of a host status queue full condition.
16	1	L	RSM_LS_FULL	Set on the occurrence of a local status queue full condition.
15	1	L	RSM_HF_EMPT	Set on the occurrence of a host free buffer queue empty condition.
14	1	L	RSM_LF_EMPT	Set on the occurrence of a local free buffer queue empty condition.
13	1	E	RSM_HS_WRITE	Indicates reassembly host status has been written by the Bt8230.
12	1	E	RSM_LS_WRITE	Indicates reassembly local status has been written by the Bt8230.
11	1	E	CELL_RCVD_RLOVR	Set on the occurrence of a CELL_RCVD_CNT rollover.
10	1	E	CELL_DSC_RLOVR	Set on the occurrence of a CELL_DSC_CNT rollover.
9	1	E	AAL5_DSC_RLOVR	Set on the occurrence of a AAL5_DSC_CNT rollover.
8	1	L	SEG_RUN	Set when the segmentation machine is running. Will be high when SEG_ENABLE bit in SEG_CTRL is high or when processing the last cell after SEG_ENABLE is set low.
7	1	E	SEG_UNFL	Segmentation underflow. Indicates that an idle cell was sent instead of a scheduled cell due to lack of PCI bandwidth.
6	1	E	HRING_OVFL	Host Transmit Queue overflow.
5	1	E	LRING_OVFL	Local Transmit Queue overflow.
4	1	L	BD_FULL	Buffer descriptor space full.
3	1	L	SEG_HS_FULL	Indicates that the segmentation host status queue is full.
2	1	L	SEG_LS_FULL	Indicates that the segmentation local status queue is full.
1	1	E	SEG_HS_WRITE	Indicates that the segmentation host status queue has been written by the Bt8230.
0	1	E	SEG_LS_WRITE	Indicates that the segmentation local status queue has been written by the Bt8230.

4.5.2 0xEC—Local Processor Interrupt Mask Register (LP_IMASK0)

Access Type: R/W L, NL

The Local Processor Interrupt Mask Register is located at address 0xEC. This register contains the interrupt enables that correspond to the status in the LP_ISTAT0 register. The assertion of the HRST* system reset pin clears all of the LP_IMASK0 interrupt enables.

Bit	Field Size	Name	Description
31	1	EN_RTC_OVFL	Enables an interrupt when RTC_OVFL status is a logic high.
30	1	EN_ALARM1	Enables an interrupt when ALARM1 status is a logic high.
29	1	EN_HOST_LOCK_STAT	Enables an interrupt when HOST_LOCK_STAT status is a logic high.
28	1	EN_LP_MBOX_ WRITTEN	Enables an interrupt when LP_MBOX_WRITTEN status is a logic high.
27	1	EN_HOST_MBOX _READ	Enables an interrupt when HOST_MBOX_READ status is a logic high.
26	1	EN_PCI_BUS_ERROR	Enables an interrupt when PCI_BUS_ERROR status is a logic high.
25	1	EN_DMA_FULL	Enables an interrupt when DMA_FULL status is a logic high.
24	1	EN_FR_PAR_ERR	Enables an interrupt when FR_PAR_ERR status is a logic high.
23	1	EN_FR_SYNC_ERR	Enables an interrupt when FR_SYNC_ERR status is a logic high.
22	1	Reserved	Set to 0.
21	1	EN_CELL_XMIT_RLOVR	Enables an interrupt when CELL_XMIT_RLOVR status is a logic high.
20	1	EN_RSQUEUE_FULL	Enables an interrupt when RSQUEUE_FULL status is a logic high.
19	1	EN_RSM_RUN	Enables an interrupt when RSM_RUN status is a logic high.
18	1	EN_RSM_OVFL	Enables an interrupt when RSM_OVFL status is a logic high.
17	1	EN_RSM_HS_FULL	Enables an interrupt when RSM_HS_FULL status is a logic high.
16	1	EN_RSM_LS_FULL	Enables an interrupt when RSM_LS_FULL status is a logic high.
15	1	EN_RSM_HF_EMPT	Enables an interrupt when RSM_HF_EMPT status is a logic high.
14	1	EN_RSM_LF_EMPT	Enables an interrupt when RSM_LF_EMPT status is a logic high.
13	1	EN_RSM_HS_WRITE	Enables an interrupt when RSM_HS_WRITE status is a logic high.
12	1	EN_RSM_LS_WRITE	Enables an interrupt when RSM_LS_WRITE status is a logic high.
11	1	EN_CELL_RCVD_RLOVR	Enables an interrupt when CELL_RCVD_RLOVR status is a logic high.
10	1	EN_CELL_DSC_RLOVR	Enables an interrupt when CELL_DSC_RLOVR status is a logic high.
9	1	EN_AAL5_DSC_RLOVR	Enables an interrupt when AAL5_DSC-RLOVR status is a logic high.
8	1	EN_SEG_RUN	Enables an interrupt when SEG_RUN status is a logic high.
7	1	EN_SEG_UNFL	Enables an interrupt when SEG_UNFL status is a logic high.
6	1	EN_HRING_OVFL	Enables an interrupt when HRING_OVFL status is a logic high.
5	1	EN_LRING_OVFL	Enables an interrupt when LRING_OVFL status is a logic high.

Bit	Field Size	Name	Description
4	1	EN_BD_FULL	Enables an interrupt when the BD_FULL status is a logic high.
3	1	EN_SEG_HS_FULL	Enables an interrupt when SEG_HS_FULL status is a logic high.
2	1	EN_SEG_LS_FULL	Enables an interrupt when SEG_LS_FULL status is a logic high.
1	1	EN_SEG_HS_WRITE	Enables an interrupt when SEG_HS_WRITE is a logic high.
0	1	EN_SEG_LS_WRITE	Enables an interrupt when SEG_LS_WRITE is a logic high.

4.5.3 0xF8—Local Processor Mailbox Register (LP_MBOX)

Access Types: R/W H, NL

The Local Processor Mailbox Register is located at address 0xF8. This register implements a mailbox for communication between the host and local processors. LP_MBOX is written by the host processor and read by the local processor to pass messages in that direction. Writes to this register can interrupt the local processor while reads can interrupt the host processor.

Bit	Field Size	Name	Description
31–0	32	LP_MBOX[31:0]	Local processor mailbox register. Messages flow from host processor to local processor.

4.6 Internal Global MIB Counters

The Bt8230 contains three internal MIB counters for global variables defined as part of UNI ILMI and ATOM MIBs, and an additional counter for discarded AAL5 PDUs. These read-only registers are cleared on the assertion of HRST* or by setting GLOBAL_RESET, bit 30 of CONFIG0. A status bit in each status register is set upon rollover of individual counters. The cell counters are 32-bit counters, the PDU counter is 16 bits wide.

4.6.1 0xA0—ATM Cells Transmitted (CELL_XMIT_CNT)

Access Types: R/O

Bit	Field Size	Name	Description
31–0	32	CELL_XMIT_CNT	Count of user data cells transmitted. (Rollover - Bit 21 of x_ISTAT0)

4.6.2 0xA4—ATM Cells Received (CELL_RCVD_CNT)

Access Types: R/O
This is a count of received cells that map to an assigned VCC Table entry.

Bit	Field Size	Name	Description
31–0	32	CELL_RCVD_CNT	Count of assigned received cells. (Rollover - Bit 11 of x_ISTAT0)

4.6.3 0xA8—ATM Cells Discarded (CELL_DSC_CNT)

Access Types: R/O
This is a count of cells which do not map to an assigned VCC Table entry.

Bit	Field Size	Name	Description
31–0	32	CELL_DSC_CNT	Count of received cells dropped. (Rollover - Bit 10 of x_ISTAT0)

4.6.4 0xAC—AAL5 PDUs Discarded (AAL5_DSC_CNT)

Access Types: R/O

PDUs may be discarded due to a Firewall condition or if the maximum length of an AAL5 PDU, as specified by the MAX_LEN fields in the Reassembly VCC Table, is exceeded.

Bit	Field Size	Name	Description
31–16	16	Reserved	N/A
15–0	16	AAL5_DSC_CNT	AAL5 PDUs discarded by reassembly coprocessor. (Rollover - Bit 9 of x_ISTAT0.)

4.7 PCI Bus Interface Configuration Registers

In accordance with the PCI bus specification, the SRC PCI bus interface implements a 128-byte configuration register space. These configuration registers can be used by the host processor to initialize, control, and monitor the SRC bus interface logic. The complete definitions of these registers and the relevant fields within them are given in the PCI bus specification. The implementation of these registers in the Bt8230 is given in Table 4-6. Table 4-7 provides the field descriptions.

Table 4-6. PCI Configuration Register Definition

Configuration Register Definition				Byte Address
DEVICE_ID (16 bits)		VENDOR_ID (16 bits)		0x00
STATUS (16 bits)		COMMAND (16 bits)		0x04
0x020300 (24 bits)			REV_ID (8 bits)	0x08
0x0000 (16 bits)		LAT_TIMER (8 bits)	0x00 (8 bits)	0x0c
EXT_MEM_BASE (8 bits)	0x000000 (24 bits)			0x10
0x00000000 (32 bits)				0x14–0x38
0x050201(24 bits)			ILINE (8 bits)	0x3C
0x00000 (20 bits)	SPECIAL_STATUS (4 bits)		MAX_BUR_LEN (8 bits)	0x40
CUR_MSTR_RD_ADDR (32 bits)				0x44
CUR_MSTR_WR_ADDR (32 bits)				0x48
0x00000000 (32 bits)				0x4C–0x7C

Table 4-7. PCI Configuration Registers Field Descriptions *(1 of 2)*

<table>
<tr><th>Field Name</th><th>Description/Function</th></tr>
<tr><td>DEVICE_ID</td><td>16-bit device identifier. Serves to uniquely identify the SRC to the host operating system. Set to 0x8230.</td></tr>
<tr><td>VENDOR_ID</td><td>16-bit vendor identifier code, allocated on a global basis by the PCI SIG. Set to 0x109E.</td></tr>
<tr><td>STATUS</td><td>PCI bus interface status register. The PCI host can monitor its operation using the STATUS field. This field is further divided into subfields as shown below.
<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27–25</td><td>24</td><td>23–16</td></tr>
<tr><td>DPE</td><td>SSE</td><td>RMA</td><td>RTA</td><td>0x0</td><td>DPR</td><td>0x80</td></tr>
</table>
DPE Detected Parity Error
SSE Signaled System Error
RMA Received Master Abort
RTA Received Target Abort
DPR Reported Data Parity Error</td></tr>
<tr><td>COMMAND</td><td>PCI bus interface control/command register. The PCI host can configure the SRC bus interface logic using the
COMMAND field. This field is further divided into subfields as shown below.
<table>
<tr><td>15–10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5–3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>0x00</td><td>FB_EN</td><td>SE_EN</td><td>0x0</td><td>PE_EN</td><td>0x0</td><td>M_EN</td><td>MS_EN</td><td>0x0</td></tr>
</table>
FB_EN Fast back-to-back enable across target
SE_EN SERR* pin output enable
PE_EN Parity report enable
M_EN Master enable- Need not be asserted for the Bt8230 interface to issue PCI bus interrupts.
MS_EN Memory space enable-Must be asserted before the Bt8230 registers or memory can be accessed.</td></tr>
<tr><td>REV_ID</td><td>Revision ID code for the Bt8230 chip.</td></tr>
<tr><td>LAT_TIMER</td><td>Power-up value is x0. All bits are writable. The suggested value is 0x10 in order to allow the transfer of a complete cell.</td></tr>
<tr><td>EXT_MEM_BASE</td><td>Base address of external memory being mapped into PCI address space.</td></tr>
<tr><td>ILINE</td><td>Interrupt line identifier.</td></tr>
</table>

Table 4-7. PCI Configuration Registers Field Descriptions *(2 of 2)*

<table>
<tr><th>Field Name</th><th>Description/Function</th></tr>
<tr><td>SPECIAL_STATUS</td><td>Device status not defined by the PCI specification. The field is further subdivided into subfields as shown below. Detailed descriptions of the subfields can be found in the PCI bus specification. The configuration registers are accessed starting from byte address 0 in the configuration space allotted to an adapter card containing the SRC chip. Access to the configuration registers is available only to the PCI host CPU and is independent of all other SRC logic.
<table>
<tr><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>INTF_DIS</td><td>INT_FAIL</td><td>MERROR</td><td>MRD</td></tr>
</table>
INTF_DIS — If the M_EN bit in the COMMAND field is a logic low, any attempt by the Bt8230 to perform a DMA transaction to the PCI bus will result in an error. INTF_DIS and MERROR bits will both be set at a logic high. This bit can be reset by writing a logic high to itself.
INT_FAIL — Set to a logic 1 when an internal PCI/DMA synchronization error has occurred. The MERROR bit will also be set to a logic high. This bit can be reset by writing a logic high to itself.
MERROR — Indicates that PCI bus master has encountered a fatal error and therefore has halted operation. Set when either RTA, RMA, DPE, INTF_DIS, or INT_FAIL errors occur. This bit can be reset by writing a logic high to itself.
MRD — If logic high, indicates that the errored transaction was a read and the address of the read is located in the CUR_MSTR_RD_ADDR field. If logic low, indicates a write with the corresponding address located in the CUR_MSTR_WR_ADDR field (read-only).</td></tr>
<tr><td>MAX_BUR_LEN</td><td>Sets the maximum number of words that can be transferred in a single transaction by the SRC PCI bus master. A value of 0 is invalid and can lead to erroneous and unpredictable results. When HRST* input pin is active, this register is loaded with 16 decimal.</td></tr>
<tr><td>CUR_MSTR_WR_ADDR</td><td>Current write target address used by PCI bus master (read only).</td></tr>
<tr><td>CUR_MSTR_RD_ADDR</td><td>Current read target address used by PCI bus master (read only).</td></tr>
</table>

5.0 Electrical and Mechanical Specifications

This chapter describes electrical and mechanical specifications for the Bt8230 such as timing, absolute maximum ratings, DC characteristics and pin assignments.

5.1 Timing

In Bt8230 timing, the clock drives the overall system, but traffic needs drive the coprocessors, which operate asynchronously from each other. With the exception of the PCI interface and the ATM Physical Interface, the internal logic of the Bt8230 SRC is based on the CLK2X input (maximum 66 MHz). This clock is used to generate SYSCLK and CLKD3 outputs. SYSCLK is CLK2X/2 (maximum 33 MHz). The internal logic is sychronous to this clock. It is provided as an output intended for use as the clock for a local processor and its associated logic.

CLKD3 is an asymmetrically divided clock at 1/3 of the frequency of CLK2X. There is no defined skew relationship between CLK2X and CLKD3. This is intended to provide a UTOPIA clock. It can be externally looped back to FRCTRL, the UTOPIA clock input of the ATM Physical Interface, and can be provided to the PHY part.

The PCI interface is clocked asynchronously to the rest of the device through input pin HCLK (33 MHz max). Since this circuitry is clocked independently, internal synchronization is necessary between the PCI section and the remainder of the SRC circuits.

Assuming a 100 pF load, the duty-cycle of SYSCLK will still meet the +/- 2 ns specification for high time and low time in the Bt8230 SRC user data sheet. Since the SYSCLK output was characterized with a 35 pF output load, the SYSCLK output will be delayed 1.7 nsec in the worst case from the waveforms in the specification. For example, PCS* setup to SYSCLK will be 9.7 nsec instead of 8 ns, and PCS* hold time will be -1.7 ns instead of 0 ns.

5.1.1 PCI Bus Interface Timing

All PCI bus interface signals are synchronous to the PCI bus clock, HCLK, except for HRST* and HINT*. Table 5-1 provides the PCI bus interface timing parameters. Figure 5-1 and Figure 5-2 illustrate this timing for input and output, respectively.

Table 5-1. PCI Bus Interface Timing Parameters *(1 of 2)*

Symbol	Parameter	Min	Max	Units
t_{cyc}	HCLK Cycle Time(1)	30	–	ns
t_{high}	HCLK High Time(1)	11	19	ns
t_{low}	HCLK Low Time(1)	11	19	ns
t_{su}	HAD Input Setup Time to HCLK(1)	7	–	ns
	HC/BE Input Setup Time to HCLK(1)	7	–	ns
	HPAR Input Setup Time to HCLK(1)	7	–	ns
	HFRAME* Input Setup Time to HCLK(1)	7	–	ns
	HIRDY* Input Setup Time to HCLK(1)	7	–	ns
	HTRDY* Input Setup Time to HCLK(1)	7	–	ns
	HSTOP* Input Setup Time to HCLK(1)	7	–	ns
	HDEVSEL* Input Setup Time to HCLK(1)	7	–	ns
	HGNT* Input Setup Time to HCLK(1)	10	–	ns
	HIDSEL Input Setup Time to HCLK(1)	7	-	ns
	HPERR* Input Setup Time to HCLK(1)	7	–	ns
t_h	Input Hold Time from HCLK–All Inputs(1)	0	–	ns

Table 5-1. PCI Bus Interface Timing Parameters *(2 of 2)*

Symbol	Parameter	Min	Max	Units
t_{val}	HCLK to HAD Valid Delay(2)	2	11	ns
	HCLK to HC/BE Valid Delay(2)	2	11	ns
	HCLK to HPAR Valid Delay(2)	2	11	ns
	HCLK to HFRAME* Valid Delay(2)	2	11	ns
	HCLK to HIRDY* Valid Delay(2)	2	11	ns
	HCLK to HSTOP* Valid Delay(2)	2	11	ns
	HCLK to HDEVSEL Valid Delay(2)	2	11	ns
	HCLK to HPERR* Valid Delay(2)	2	11	ns
	HCLK to HREQ Valid Delay(2)	2	12	ns
	HCLK to HSERR* Valid Delay(2)	2	11	ns
	HCLK to STAT Valid Delay(3)	2	20	ns
t_{on}	Float to Active Delay—All Three-state Outputs(2)	2	–	ns
t_{off}	Active to Float Delay—All Three-state Outputs(2)	–	28	ns
$t_{rst\text{-}off}$	Reset Active to Output Float Delay	–	40	ns

Notes: (1). See Figure 5-1 for waveforms and definitions.
(2). See Figure 5-2 for waveforms and definitions. The maximum output delays are measured with a 50 pF load and the minimum delays are measured with a 0 pF load.
(3). Applicable when STAT outputs are configured as BOM cell synchronization signals.

Figure 5-1. PCI Bus Input Timing Measurement Conditions

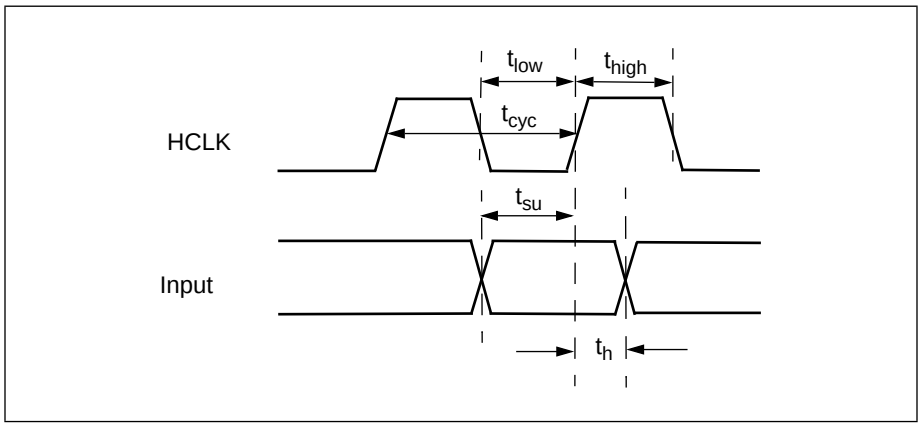
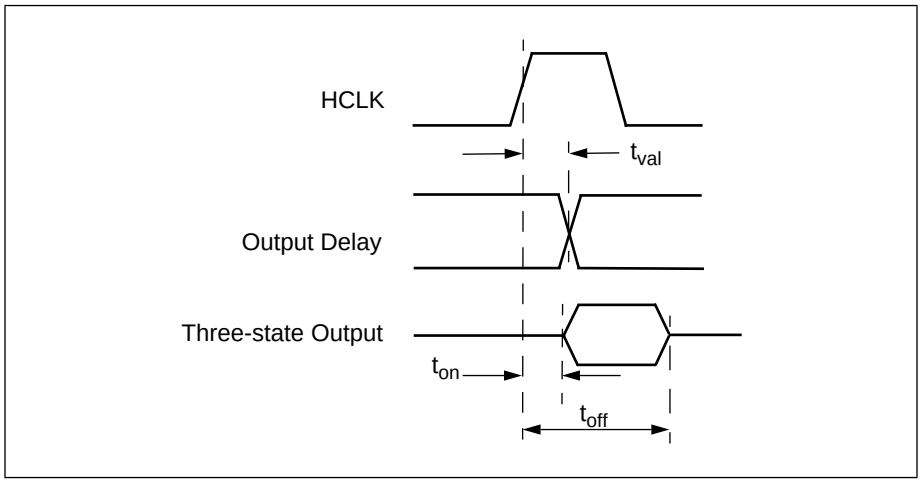


Figure 5-2. PCI Bus Output Timing Measurement Conditions



5.1.2 ATM Physical Interface Timing—UTOPIA and Slave UTOPIA

All ATM physical interface signals are synchronous to the interface clock, FRCTRL, except for TXFLAG* and RXFLAG* in the slave UTOPIA mode. Timing parameters for the UTOPIA interface are provided in Table 5-2. Table 5-3 provides the timing parameters for the slave UTOPIA interface. Input and output timing diagrams for both interfaces are provided in Figure 5-3 and Figure 5-4.

Table 5-2. UTOPIA Interface Timing Parameters

Symbol	Parameter	Min	Max	Units
t_{cyc}	FRCTRL Cycle Time(1)	30	–	ns
t_{high}	FRCTRL High Time(1) (% of t_{cyc})	40	60	%
t_{low}	FRCTRL Low Time(1) (% of t_{cyc})	40	60	%
t_{su}	RXD Input Setup Time to FRCTRL(1)	10	–	ns
	RXPAR Input Setup Time to FRCTRL(1)	10	–	ns
	RXMARK Input Setup Time to FRCTRL(1)	10	–	ns
	RXFLAG* Input Setup Time to FRCTRL(1)	10	–	ns
	TXFLAG* Input Setup Time to FRCTRL(1)	10	–	ns
t_h	RXD Input Hold Time from FRCTRL(1)	1	–	ns
	RXPAR Input Hold Time from FRCTRL(1)	1	–	ns
	RXMARK Input Hold Time from FRCTRL(1)	1	–	ns
	RXFLAG* Input Hold Time from FRCTRL(1)	1	–	ns
	TXFLAG* Input Hold Time from FRCTRL(1)	1	–	ns
t_{val}	FRCTRL to TXD Valid Delay(2)	2	18	ns
	FRCTRL to TXPAR Valid Delay(2)	2	18	ns
	FRCTRL to TXMARK Valid Delay(2)	2	18	ns
	FRCTRL to TXEN* Valid Delay(2)	2	20	ns
	FRCTRL to RXEN* Valid Delay(2)	2	20	ns

Notes: (1). See Figure 5-3 for waveforms and definitions.
(2). See Figure 5-4 for waveforms and definitions. The output delays are measured with a 25 pF load.

Table 5-3. Slave UTOPIA Interface Timing Parameters

Symbol	Parameter	Min	Max	Units
t_{cyc}	FRCTRL Cycle Time(1)	30	–	ns
t_{high}	FRCTRL High Time(1) (% of t_{cyc})	40	60	%
t_{low}	FRCTRL Low Time(1) (% of t_{cyc})	40	60	%
t_{su}	RXD Input Setup Time to FRCTRL(1)	6	–	ns
	RXPAR Input Setup Time to FRCTRL(1)	3	–	ns
	RXMARK Input Setup Time to FRCTRL(1)	8	–	ns
	RXEN* Input Setup Time to FRCTRL(1)	10	–	ns
	TXEN* Input Setup Time to FRCTRL(1)	3	–	ns
t_h	RXD Input Hold Time from FRCTRL(1)	1	–	ns
	RXPAR Input Hold Time from FRCTRL(1)	1	–	ns
	RXMARK Input Hold Time from FRCTRL(1)	1	–	ns
	RXEN* Input Hold Time from FRCTRL(1)	1	–	ns
	TXEN* Input Hold Time from FRCTRL(1)	1	–	ns
t_{val}	FRCTRL to TXD Valid Delay(2)	2	16	ns
	FRCTRL to TXPAR Valid Delay(2)	2	16	ns
	FRCTRL to TXFLAG* Valid Delay(2)	2	20	ns
	FRCTRL to RXFLAG* Valid Delay(2)	2	20	ns
	FRCTRL to TXMARK Valid Delay(2)	2	17	ns

Notes: (1). See Figure 5-3 for waveforms and definitions.
(2). See Figure 5-4 for waveforms and definitions. The output delays are measured with a 25 pF load.

Figure 5-3. UTOPIA and Slave UTOPIA Input Timing Measurement Conditions

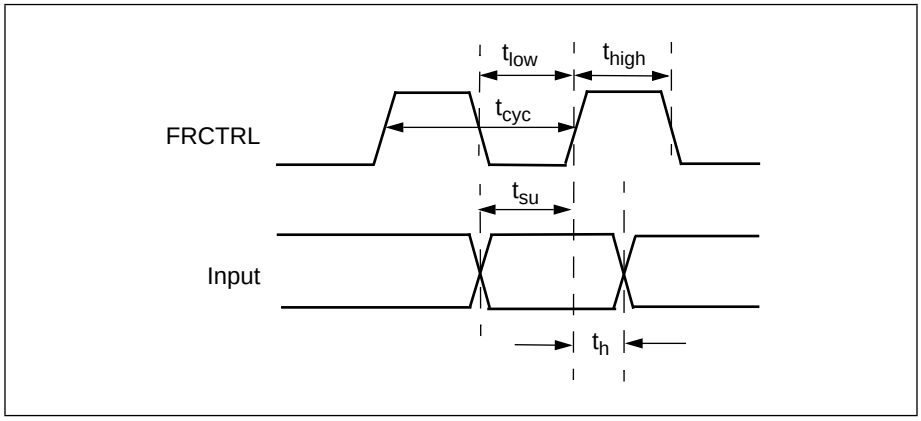
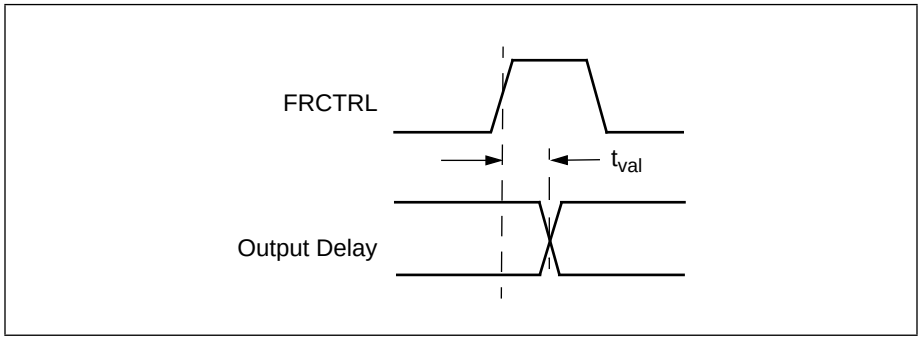


Figure 5-4. UTOPIA and Slave UTOPIA Output Timing Measurement Conditions



5.1.3 ATM Physical Interface Timing—Bt8222 Mode

All ATM physical interface signals are synchronous to the receive strobe, RXEN*, and the transmit strobe, TXEN*. Table 5-4, Figure 5-5, and Figure 5-6 provide the timing parameters and diagrams for the Bt8222 interface.

Table 5-4. Bt8222 Interface Timing Parameters

Symbol	Parameter	Min	Max	Units
t_{cyc}	RXEN*/TXEN* Cycle Time(1)	50	–	ns
t_{high}	RXEN*/TXEN* High Time(1) (% of t_{cyc})	42	58	%
t_{low}	RXEN*/TXEN* Low Time(1) (% of t_{cyc})	42	58	%
t_{su}	RXD Input Setup Time to RXEN*(1)	13	–	ns
	RXPAR Input Setup Time to RXEN*(1)	10	–	ns
	RXMARK Input Setup Time to RXEN*(1)	8	–	ns
	FRCTRL Input Setup Time to RXEN*(1)	7	–	ns
	TXMARK Input Setup Time to TXEN*(1)	3	–	ns
t_h	RXD Input Hold Time from RXEN*(1)	20	–	ns
	RXPAR Input Hold Time from RXEN*(1)	20	–	ns
	RXMARK Input Hold Time from RXEN*(1)	20	–	ns
	FRCTRL Input Hold Time from RXEN*(1)	20	–	ns
	TXMARK Input Hold Time from TXEN*(1)	20	–	ns
t_{val}	TXEN* to TXD Valid Delay(2)	6	36	ns
	TXEN* to TXPAR Valid Delay(2)	6	39	ns
	TXEN* to TXFLAG* Valid Delay(2, 3)	4	21	ns
	RXEN* to RXFLAG* Valid Delay(2)	4	23	ns
t_{on}	Float to Active Delay—All three-state Outputs (TXD/TXPAR)(2)	4	16	ns
t_{off}	Active to Float Delay—All three-state Outputs (TXD/TXPAR)(2)	2	10	ns

Notes: (1). See Figure 5-5 for waveforms and definitions.
(2). See Figure 5-6 for waveforms and definitions. The output delays are measured with a 50 pF load.
(3). TXFLAG* going inactive is asynchronous to TXEN*.

Figure 5-5. Bt8222 Input Timing Measurement Conditions

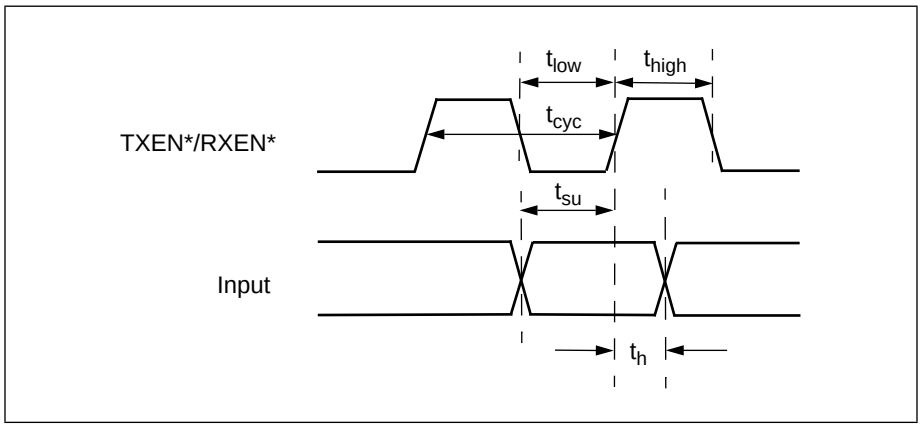
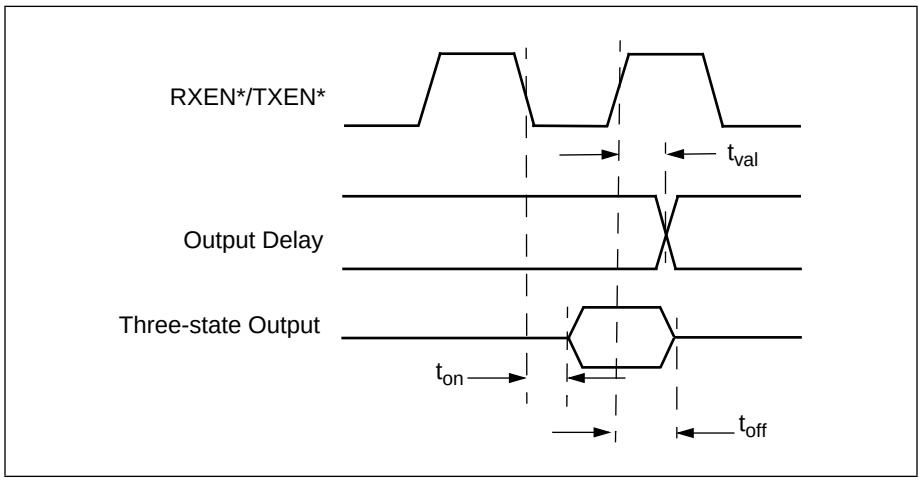


Figure 5-6. Bt8222 Output Timing Measurement Conditions



5.1.4 System Clock Timing

The system clock timing consists of the CLK2X input and the SYSCLK and CLKD3 outputs. The CLK2X input and SYSCLK outputs are used to generate the internal and external system clocks as well as local memory and processor read and write timing. The CLKD3 output can be used as the ATM Physical Interface clock. There is no defined skew relationship between the CLK2X input and SYSCLK and CLKD3 outputs. Table 5-5 specifies the timing parameters of the three clocks. Figure 5-7 and Figure 5-8 illustrate this timing.

Table 5-5. System Clock Timing

Symbol	Parameter	Min	Max	Units
Input Clocks(1)				
t_{ck}	CLK2X Frequency		66	MHz
t_c	CLK2X Period	15.15		ns
t_{cd}	CLK2X Duty Cycle	40	60	%
t_{cr}	CLK2X Rise Time	0	6	ns
t_{cf}	CLK2X Fall Time	0	6	ns
Output Clocks(2)				
t_{sk}	SYSCLK Frequency	$t_{ck}/2$		MHz
t_s	SYSCLK Period	$2t_c$		ns
t_{sh}	SYSCLK High Time	$(t_s/2) – 2$	$(t_s/2) + 2$	ns
t_{sl}	SYSCLK Low Time	$(t_s/2) – 2$	$(t_s/2) + 2$	ns
t_{sr}	SYSCLK Rise Time	1	4	ns
t_{sf}	SYSCLK Fall Time	1	4	ns
t_{dk}	CLKD3 Frequency	$t_{ck}/3$		MHz
t_d	CLKD3 Period	$3t_c$		
t_{dh}	CLKD3 High Time	$(t_d/2) – 2$	$(t_d/2) + 2$	ns
t_{dl}	CLKD3 Low Time	$(t_d/2) – 2$	$(t_d/2) + 2$	ns
t_{dr}	CLKD3 Rise Time	1	4	ns
t_{df}	CLKD3 Fall Time	1	4	ns

Notes: (1). See Figure 5-7 for waveforms and definitions.
(2). See Figure 5-8 for waveforms and definitions. The outputs are measured with a load of 35 pF.

Figure 5-7. Input System Clock Waveform

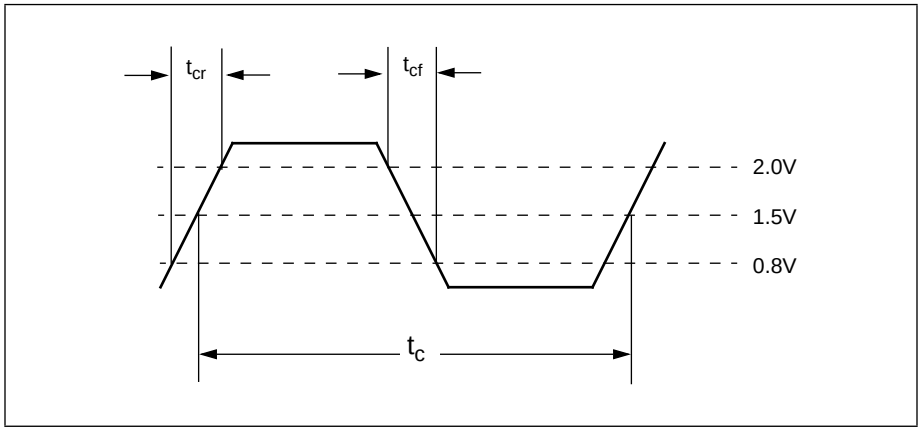
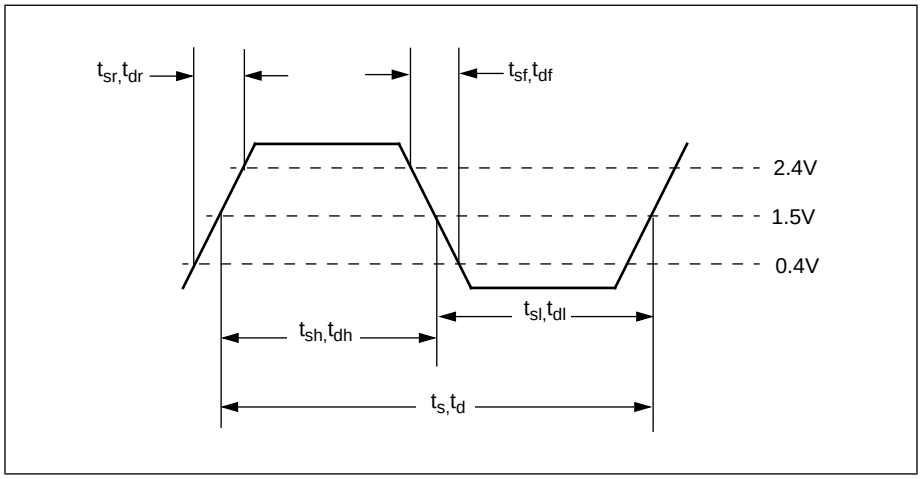


Figure 5-8. Output System Clock Waveform



5.1.5 Bt8230 Memory Interface Timing

Memory access times and other timing requirements are specified at three typical implementations of 1, 2, and 4 banks of by_8 SRAM. Table 5-6 gives the number of loads per bank for the different SRAM organizations. Table 5-7 and Table 5-8 give the capacitive loading used in the timing specifications given for the three typical implementations as applies to product Revisions A/B and C, respectively. Bt8230 Memory Interface Timing is given in Table 5-9 for Revisions A/B and in Table 5-10 for Revision C. Figure 5-9 and Figure 5-10 show read and write timing for Revisions A/B. Figure 5-11 and Figure 5-12 illustrate read and write timing for Revision C. See Section 3.2for details of memory bank organization.

Table 5-6. SRAM Organization Loading Dependencies

Signal	Loads/Bank(1)		
	by_16 SRAM	by_8 SRAM	by_4 SRAM
LADDR[18:0](1)	2	4	8
LDATA[31:0](1)	1	1	1
MWR*(1)	2	N/A	N/A
MOE*(1)	2	4	8
MCSx*(1, 2)	2	4	8
MWEx*(1)	1	1	2

Notes: (1). Typical input loading for SRAM is 7 pF. For exact values, consult the SRAM databook.
(2). Only connected to one bank by definition.

Table 5-7. Bt8230 Rev A and B Local Memory Output Loading Conditions

Signal	4 banks of by_8 SRAM	2 banks of by_8 SRAM	1 bank of by_8 SRAM	Units
Memory Interface Loading(1)				
LADDR[18:0](1)	150	100	50	pF
MOE*	150	100	50	pF
MWR*(2)	—	100	50	pF
LDATA[31:0]	50	35	25	pF
MWE[3:0]*	50	35	25	pF
MCS[3:0]*	50	50	50	pF

Notes: (1). In general, the LADDR loading is the most critical parameter, one bank of by_8 SRAM has the same address loading as 2 banks of by_16 and 1/2 bank of by_4 SRAM. For example, use the timing from the 2 banks of by_8 column for 4 banks of by_16 SRAM, or 1 bank of by_4 SRAM.
(2). For by_16 SRAM, the WE* input has the same loading as the address bus and OE*; therefore, 16 loads specified by the 4 banks of by_8 is not applicable, since the maximum number of by_16 SRAMs supported is 8.

Table 5-8. Bt8230 Rev C Local Memory Output Loading Conditions

Signal	4 banks of by_8 SRAM	2 banks of by_8 SRAM	1 bank of by_8 SRAM	Units
Memory Interface Loading(1)				
LADDR[18:0](1)	150	100	50	pF
MOE*	150	100	50	pF
MWR*(2)	-	50	35	pF
LDATA[31:0]	50	35	25	pF
MWE[3:0]*	50	35	25	pF
MCS[3:0]*	50	35	25	pF

Notes: (1). In general, the LADDR loading is the most critical parameter, one bank of by_8 SRAM has the same address loading as 2 banks of by_16 and 1/2 bank of by_4 SRAM. For example, use the timing from the 2 banks of by_8 column for 4 banks of by_16 SRAM, or 1 bank of by_4 SRAM.
(2). For by_16 SRAM, the WE* input has the same loading as the address bus and OE*; therefore, 16 loads specified by the 4 banks of by_8 is not applicable, since the maximum number of by_16 SRAMs supported is 8.

Table 5-9. Bt8230 Rev A and B Memory Interface Timing *(1 of 2)*

Symbol	Parameter	4 banks of by_8 SRAM		2 banks of by_8 SRAM		1 bank of by_8 SRAM		Units
		Min	Max	Min	Max	Min	Max	
Memory Read Timing								
t_{rc}	READ Cycle Time(2)	T		T		T		ns
t_{adv}	LADDR[18:0] Stable to Required Data Valid Time(1, 2, 3)	T-18		T-15		T-12		ns
t_{csdv}	MCS[3:0]* Low to Required Data Valid Time (1, 2, 3)	T-10		T-10		T-10		ns
t_{oedv}	MOE* Low to Required Data Valid Time(1, 2, 3)	T-23		T-20		T-17		ns
t_{dod}	LDATA Output Disable to MOE* Low(1)	8		8		8		ns
t_{doe}	MOE* High to LDATA Driven by SRC(1)	8		8		8		ns
t_{bers}	MWE[3:0]* Byte Enables Setup to Required Data Valid Time (RAMMODE = 1)(1)	T-10		T-8		T-6		ns
t_{berh}	MWE[3:0]* Byte Enables Hold from Required Data Valid Time (RAMMODE = 1)(1)	10		10		10		ns
Memory Write Timing								
t_{wc}	WRITE Cycle Time(2)	T		T		T		ns
t_{w}	MWR*, MWE[3:0]* Width(2, 4)	T-16	T-14	T-16	T-14	T-16	T-14	ns

Table 5-9. Bt8230 Rev A and B Memory Interface Timing *(2 of 2)*

Symbol	Parameter	4 banks of by_8 SRAM		2 banks of by_8 SRAM		1 bank of by_8 SRAM		Units
		Min	Max	Min	Max	Min	Max	
t_{aws}	LADDR[18:0] Setup to MWR*, MWE[3:0]* Rising(2, 4)	T-11		T-9		T-7		ns
t_{awh}	LADDR[18:0] Hold from MWR*, MWE[3:0]* Rising(2, 4)	2		2		2		ns
t_{cws}	MCS[3:0]* Active to MWR*, MWE[3:0]* Rising(2, 4)	T-4		T-4		T-4		ns
t_{cwh}	MCS[3:0]* Hold from MWE[3:0]* Rising(4)	1		1		1		ns
t_{dws}	LDATA Setup to MWR*, MWE[3:0]* Rising Edge(1, 2, 4)	$1/2t_S$ –3		$1/2t_S$ –3		$1/2t_S$ –3		ns
t_{dwh}	LDATA Hold from MWR*, MWE[3:0]* Rising (1)	2		2		2		ns
t_{bews}	MWE[3:0]* Byte Enables Setup to MWR* Rising, (RAMMODE = 1)(1)	T-6		T-6		T-6		ns
t_{bewh}	MWE[3:0]* Byte Enables Hold from MWR* Rising, (RAMMODE = 1)(1)	4		4		4		ns

Notes: (1). See Figure 5-9 and Table 5-10 for waveforms and definitions.
(2). $T = t_s$ for single cycle memory (no wait states), $T = 2t_s$ for two cycle memory, (one wait state).
(3). Data Required Time is the time at which read data from the SRAM must be stable to meet setup requirements inside the Bt8230 to the internal version of SYSCLK. For two cycle timing, add t_s to t_{dws} time given.
(4). See Figure 5.1.6 and Table 5-10 for waveforms and definitions.

Figure 5-9. Bt8230 Revisions A and B Memory Read Timing

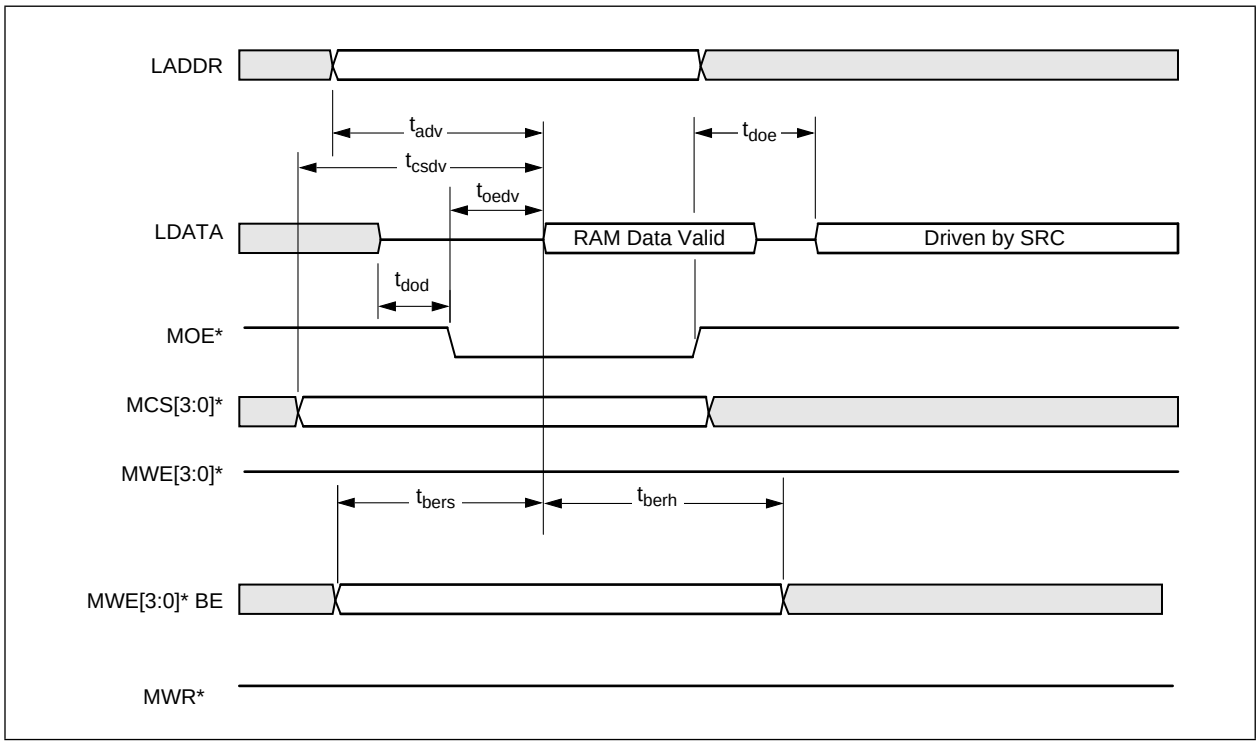


Figure 5-10. Bt8230 Revisions A and B Memory Write Timing

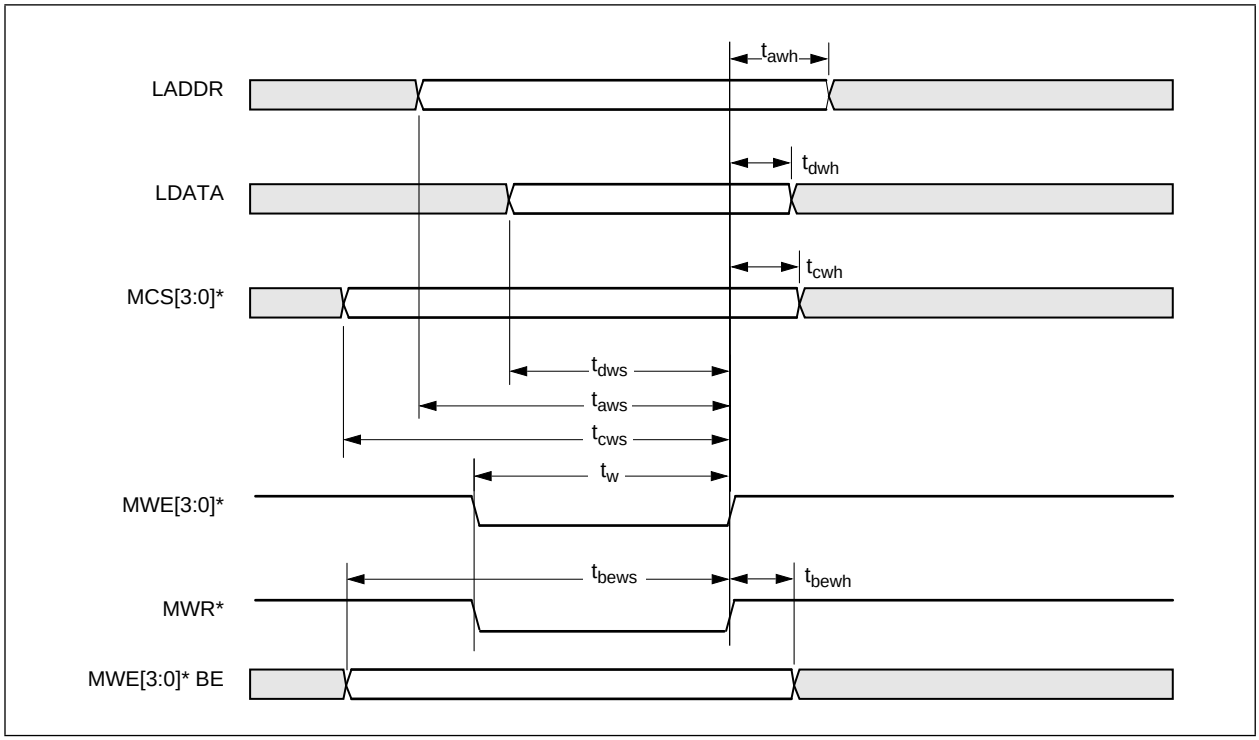


Table 5-10. Bt8230 Rev C Memory Interface Timing

Symbol	Parameter	4 banks of by_8 SRAM		2 banks of by_8 SRAM		1 bank of by_8 SRAM		Units
		Min	Max	Min	Max	Min	Max	
Memory Read Timing								
t_{rc}	READ Cycle Time (2)	T		T		T		ns
t_{aov}	LADDR[18:0] Output Valid (1)	1	10	1	8	1	7	ns
t_{mcsov}	MCS[3:0]* Output Valid (1)	1	10	1	8	1	7	ns
t_{mweov}	MWE* Byte Enables Output Valid (1) (RAMMODE = 1)	1	10	1	8	1	7	ns
t_{moel}	MOE* Low (1)		17		16		15	ns
t_{moeh}	MOE* High (1)		10		8		7	ns
t_{dod}	LDATA Output Disable (1)		7		6		6	ns
t_{dis}	LDATA Input Setup (1)	5		5		5		ns
t_{dh}	LDATA Input Hold (1)	0		0		0		ns
t_{doe}	LDATA Driven by SRC (1)	15		14		14		ns
Memory Write Timing								
t_{wc}	WRITE Cycle Time(2)	T		T		T		ns
t_{w}	MWR*, MWE[3:0]* Width(2, 4)	T/2		T/2		T/2		ns
t_{aov}	LADDR[18:0] Output Valid (4)	1	10	1	8	1	7	ns
t_{mcsov}	MCS[3:0]* Output Valid (4)	1	10	1	8	1	7	ns
t_{mweov}	MWE* Byte Enables Output Valid (1) (RAMMODE = 1)	1	10	1	8	1	7	ns
t_{dov}	LDATA Output Valid (4)	1	10	1	10	1	10	ns
t_{mwel}	MWE[3:0]* Low (4)		16		16		16	ns
t_{mweh}	MWE[3:0]* High (4)		1		1		1	ns
t_{mwrl}	MWR[3:0]* Low (4)		16		16		16	ns
t_{mwrh}	MWR[3:0]* High (4)		1		1		1	ns

Notes: (1). See Figure 5-11 for waveforms.
(2). T = t_s for single cycle memory (no wait states), T = $2t_s$ for two cycle memory, (one wait state).
(3). Insert one clock cycle for two cycle memory (one wait state).
(4). See Figure 5.1.6 for waveforms.
(5). SYSCLK shown for reference only.

Figure 5-11. Bt8230 Revision C Memory Read Timing

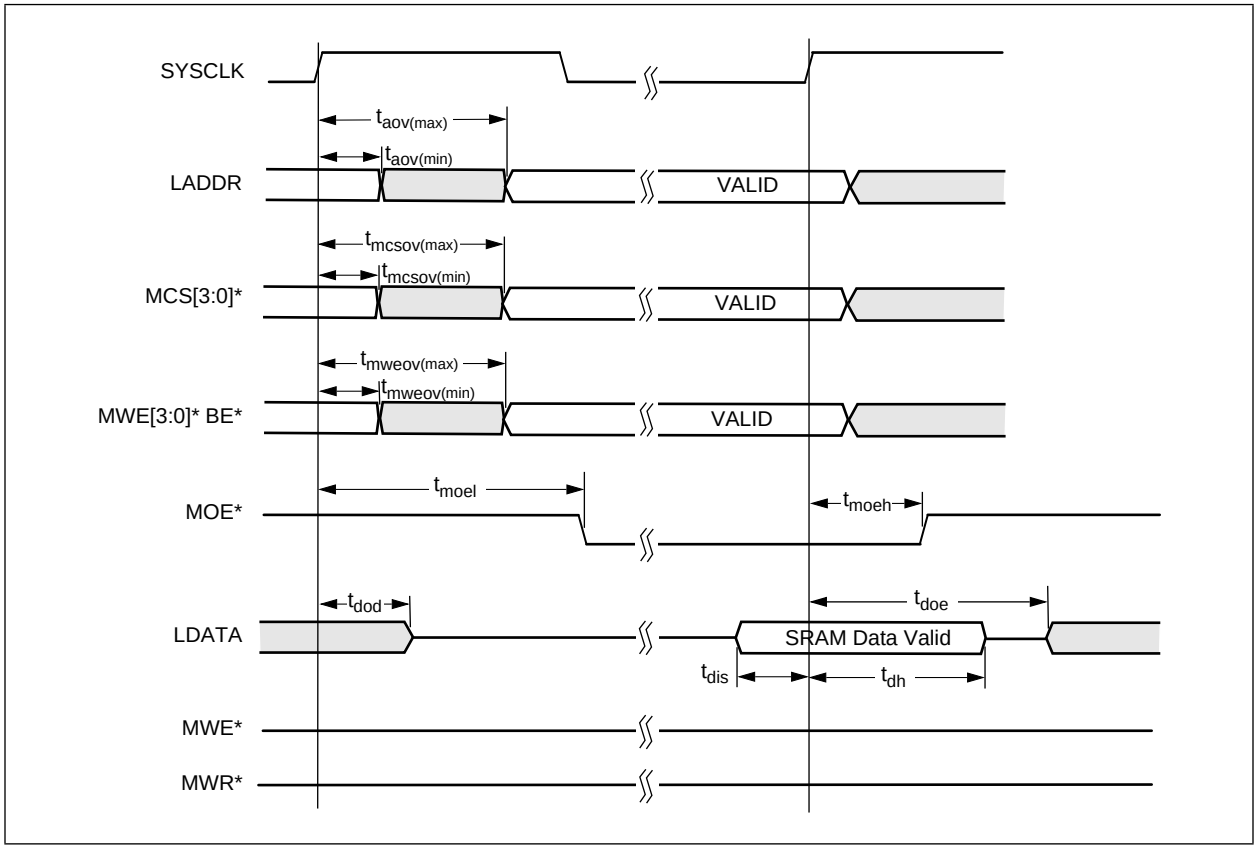
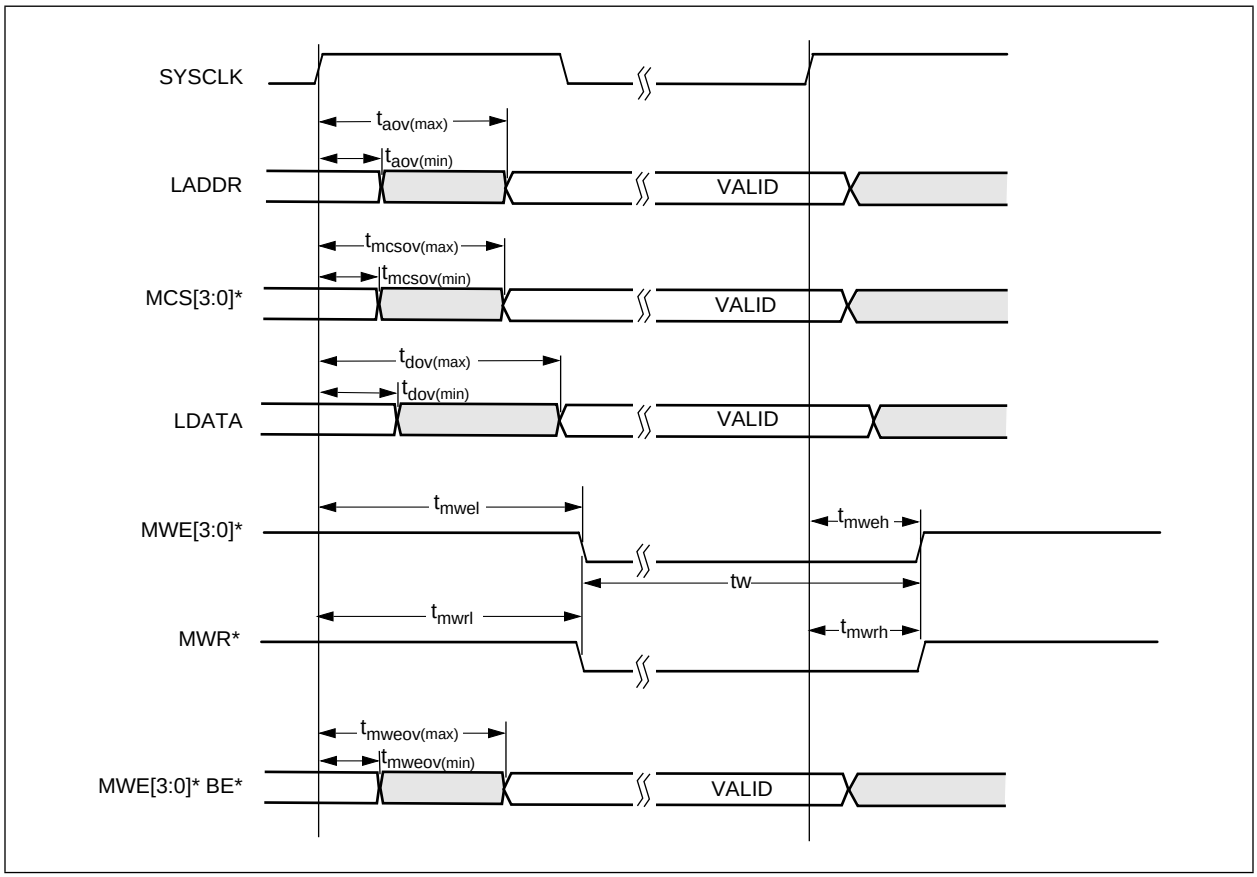


Figure 5-12. Bt8230 Revision C Memory Write Timing



5.1.6 Bt8PHY Interface Timing (standalone Mode)

The standalone mode of operation is entered when the PROCMODE input is at a logic high, indicating that no local processor is present. In this mode, the Bt8230 changes its memory map to include the ATM physical interface device. The interface is fully synchronous to SYSCLK and is designed to interface directly to the Bt8222 ATM Receiver/Transmitter. Timing for Revisions A/B is given in Table 5-11, Figure 5-13, and Figure 5-14. Timing for Revision C is given in Table 5-12, Figure 5-13 and Figure 5-15. See Section 3.3.5for details.

Table 5-11. Bt8230 Rev A and B PHY Interface Timing (PROCMODE = 1)

Symbol	Parameter	Min	Max	Units
	Synchronous Inputs			
t_{is}	PWAIT* Input Setup(1)	17		ns
t_{ih}	PWAIT* Input Hold(1)	0		ns
	Synchronous Outputs			
t_{ov}	PRDY* Output Valid Delay(2)	12		ns
	PAS* Output Valid Delay(2)	12		ns
	PCS* Output Valid Delay(2)	12		ns
	PBLAST* Output Valid Delay(2)	12		ns
	PWNR* Output Valid Delay(2)	12		ns
t_{oh}	PRDY* Output Hold(2)		15	ns
	PAS* Output Hold(2)		15	ns
	PCS* Output Hold(2)		15	ns
	PBLAST* Output Hold(2)		15	ns
	PWNR* Output Hold(2)		15	ns

Notes: (1). See Figure 5-13 for waveforms and definitions.
(2). See Figure 5-14 for waveforms and definitions. The outputs are measured with a load of 35 pF.

Figure 5-13. Revisions A, B and C Synchronous PHY Interface Input Timing

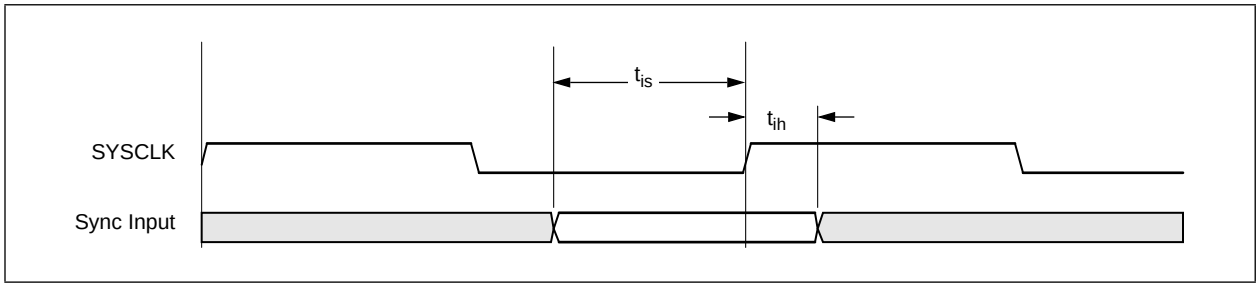


Figure 5-14. Revisions A and B Synchronous PHY Interface Output Timing

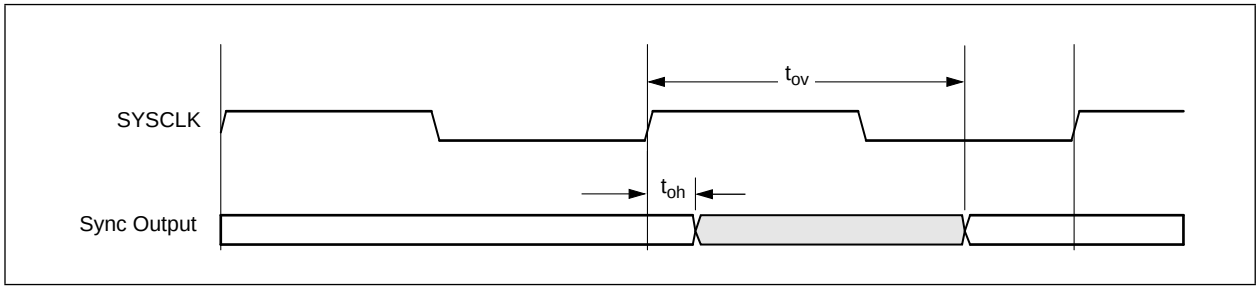
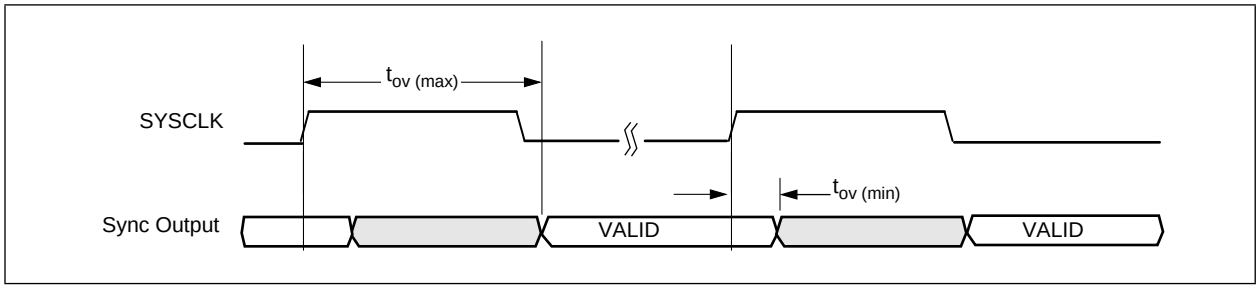


Table 5-12. Revision C PHY Interface Timing (PROCMODE = 1)

Symbol	Parameter	Min	Max	Units
	Synchronous Inputs			
t_{is}	PWAIT* Input Setup(1)	14		ns
	LDATA Input Setup(1)	5		ns
t_{ih}	PWAIT* Input Hold(1)	0		ns
	LDATA Input Hold(1)	0		ns
	Synchronous Outputs			
t_{ov}	PRDY* Output Valid Delay(2)	5	15	ns
	PAS Output Valid Delay(2)	5	15	ns
	PCS* Output Valid Delay(2)	5	15	ns
	PBLAST* Output Valid Delay(2)	5	15	ns
	PWNR Output Valid Delay(2)	5	15	ns
	LDATA* Output Valid Delay(2)	1	10	ns
	LADDR Output Valid Delay(2)	1	10	ns

Notes: (1). See Figure 5-13 for waveforms and definitions.
(2). See Figure 5-15 for waveforms and definitions. The outputs are measured with a load of 35 pF.

Figure 5-15. Revision C Synchronous PHY Interface Output Timing



5.1.7 Local Processor Interface Timing

Timing for the local processor interface can be broken into two sections. The first is the synchronous to SYSCLK interface of processor control signals to and from the Bt8230. The second is the interface to the local SRAM memory and the Bt8230 control and status registers. This interface involves both SRAM and transceiver and buffer timing parameters that are not specified and are left up to the system designer.

All of the synchronous interface signals are inputs to the Bt8230 except for the SYSCLK and the ready output of the Bt8230 (PRDY*). The output loading of these signals is 35 pF for the timing given in Table 5-13 for Revisions A/B and Table 5-14 for Revision C. Memory Interface Timing is given in Table 5-15 for Revisions A/B and Table 5-16 for Revision C. Timing diagrams for Revisions A/B are shown in Figure 5-16 and Figure 5-17. Timing diagrams for Revision C are shown in Figure 5-16 and Figure 5-18. Memory timing diagrams for Revisions A/B are shown in Figure 5-19 and Figure 5-20. Memory timing diagrams for Revision C are shown in Figure 5-21 and Figure 5-22.

Table 5-13. Bt8230 Rev A and B Synchronous Processor Interface Timing

Symbol	Parameter	Min	Max	Units
Synchronous Inputs				
t_{is}	PCS* Input Setup(1)	10		ns
	PAS* Input Setup(1)	10		ns
	PBLAST* Input Setup(1)	10		ns
	PWAIT* Input Setup(1)	10		ns
	PADDR[1:0] Input Setup(1)	10		ns
	PBSEL[1:0] Input Setup(1)	8		ns
	PBE[3:0]* Input Setup(1)	9		ns
t_{ih}	PCS* Input Hold(1)	0		ns
	PAS* Input Hold(1)	0		ns
	PBLAST* Input Hold(1)	0		ns
	PWAIT* Input Hold(1)	0		ns
	PADDR[1:0] Input Hold(1)	0		ns
	PBSEL[1:0] Input Hold(1)	0		ns
	PBE[3:0]* Input Hold(1)	0		ns
Synchronous Outputs				
t_{ov}	PRDY* Output Valid Delay(2)	12		ns
t_{oh}	PRDY* Output Hold(2)		15	ns

Notes: (1). See Figure 5-16 for waveforms and definitions.
(2). See Figure 5-17 for waveforms and definitions. The outputs are measured with a load of 35 pF.

Figure 5-16. Revisions A, B and C Synchronous Local Processor Input Timing

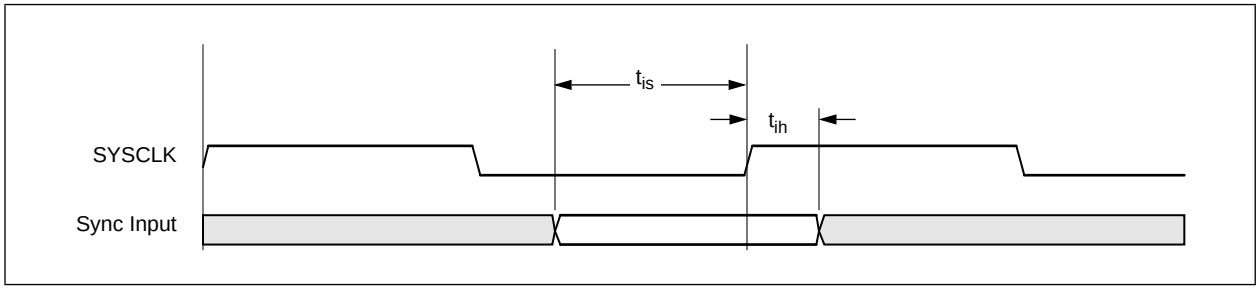


Figure 5-17. Revisions A and B Synchronous Local Processor Output Timing

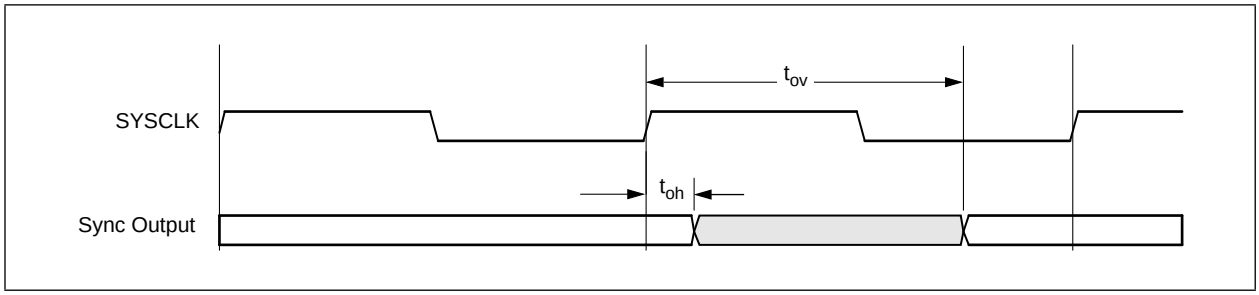


Table 5-14. Revision C Synchronous Processor Interface Timing (1 of 2)

Symbol	Parameter	Min	Max	Units
	Synchronous Inputs			
t_{is}	PCS* Input Setup(1)	8		ns
	PAS* Input Setup(1)	10		ns
	PBLAST* Input Setup(1)	10		ns
	PWAIT* Input Setup(1)	10		ns
	PADDR[1,0] Input Setup(1)	10		ns
	PBSEL[1,0] Input Setup(1)	8		ns
	PBE[3:0]* Input Setup(1)	8		ns
	PWNR Input Setup(1)	10		ns

Table 5-14. Revision C Synchronous Processor Interface Timing *(2 of 2)*

Symbol	Parameter	Min	Max	Units
t_{ih}	PCS* Input Hold(1)	0		ns
	PAS* Input Hold(1)	0		ns
	PBLAST* Input Hold(1)	0		ns
	PWAIT* Input Hold(1)	0		ns
	PADDR[1,0] Input Hold(1)	0		ns
	PBSEL[1,0] Input Hold(1)	0		ns
	PBE[3:0]* Input Hold(1)	0		ns
	PWNR Input Hold(1)	0		ns
Synchronous Outputs				
t_{ov}	PRDY* Output Valid Delay(2)	5	15	ns

Notes: (1). See Figure 5-16 for waveforms and definitions.
(2). See Figure 5-18 for waveforms and definitions. The outputs are measured with a load of 35 pF.

Figure 5-18. Revision C Synchronous Local Processor Output Timing

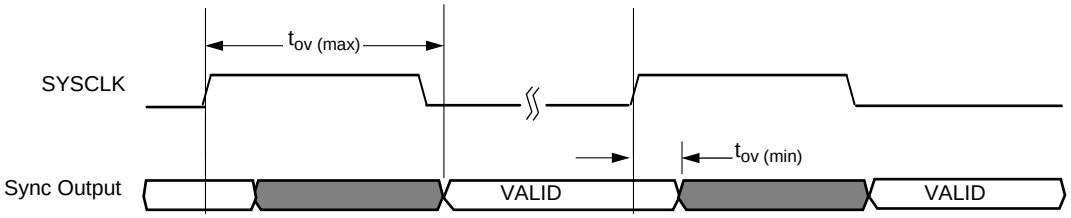


Table 5-15. Bt8230 Rev A and B Local Processor Memory Interface Timing

Symbol	Parameter	Min	Max	Units
t_{pdaenl}	SYSCLK to PDAEN* Low, Bus Recovery Cycle(1)	12	15	ns
t_{pdaenh}	SYSCLK to PDAEN* High, Bus Recovery Cycle(1)	1	4	ns
t_{lod}	LADDR[18:2], LDATA Output Disable to PDAEN* Low, Bus Recovery Cycle(1)	4		ns
t_{loe}	PDAEN* High to LADDR[18:2], LDATA Output Enable, Bus Recovery Cycle(1)	11		ns
t_{MCS}	SYSCLK to MCS*[3:0] Valid(1)	1	4	ns
t_{lav}	SYSCLK to LADDR[1,0] Valid(1, 2)	1	5	ns
t_{OEL}	MOE* Active from SYSCLK(1, 3)	8		ns
t_{OEH}	MOE* Inactive from SYSCLK(1, 3)	1		ns
t_{WL}	MWR*, MWE[3:0] Active from SYSCLK(1, 3)	12	14	ns
t_{WH}	MWR*, MWE[3:0]* Inactive to SYSCLK(1, 3)	1	3	ns
t_{BE}	MWE[3:0]* Byte Enables Valid from SYSCLK (RAMMODE = 1)(1)	1	4	ns

Notes: (1). See Figure 5-19 and Figure 5-20 for waveforms and definitions.
(2). The t_{lav} symbol is valid for second and subsequent accesses during burst transfers; see functional timing diagrams.
(3). In the case of two cycle memory, or when inserting wait states by PWAIT*, MOE*, MWE[3:0]*, and MWR* are extended across 2 or more clock cycles with the same relative timing to SYSCLK, see the functional timing diagrams in Section 3.3.

Figure 5-19. Bt8230 Rev A and B Local Processor Read Timing

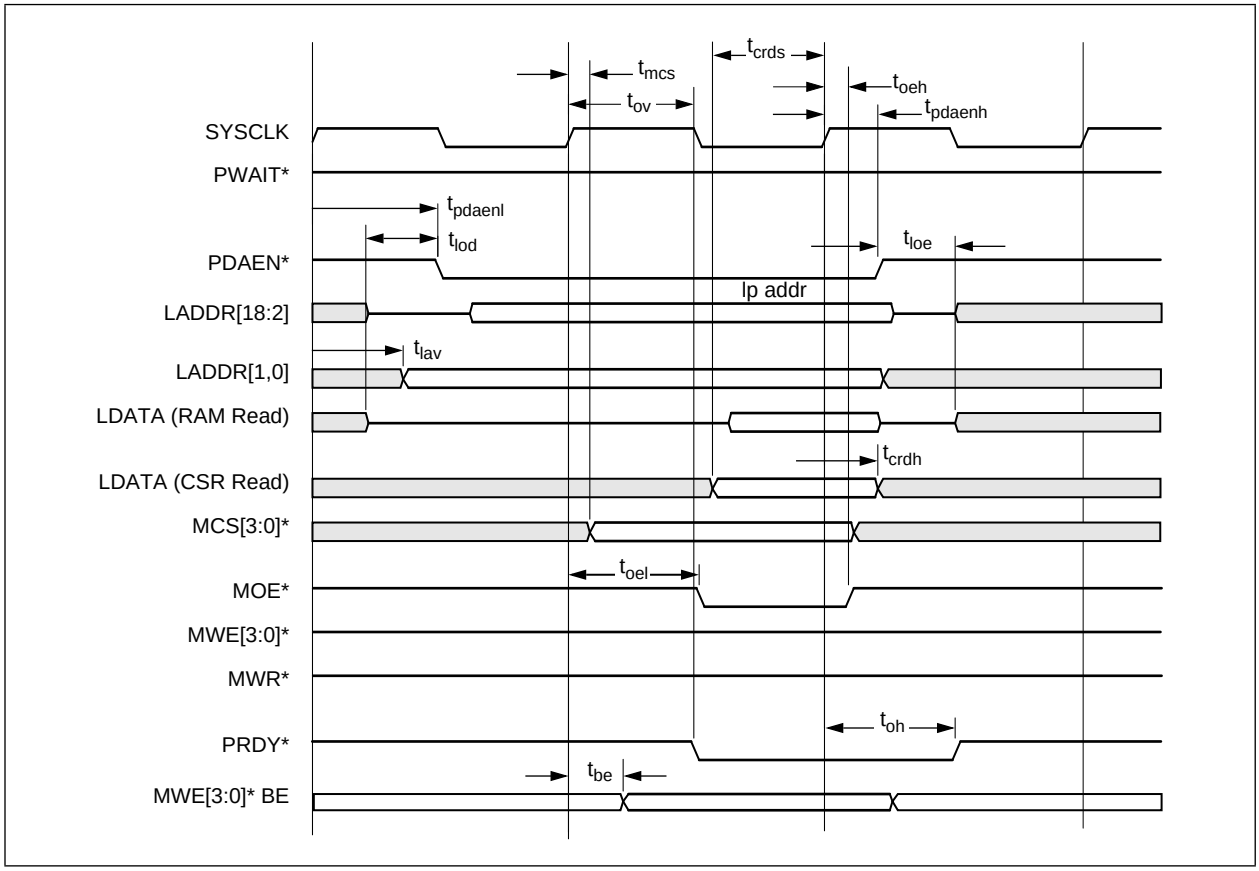


Figure 5-20. Bt8230 Rev A and B Local Processor Write Timing

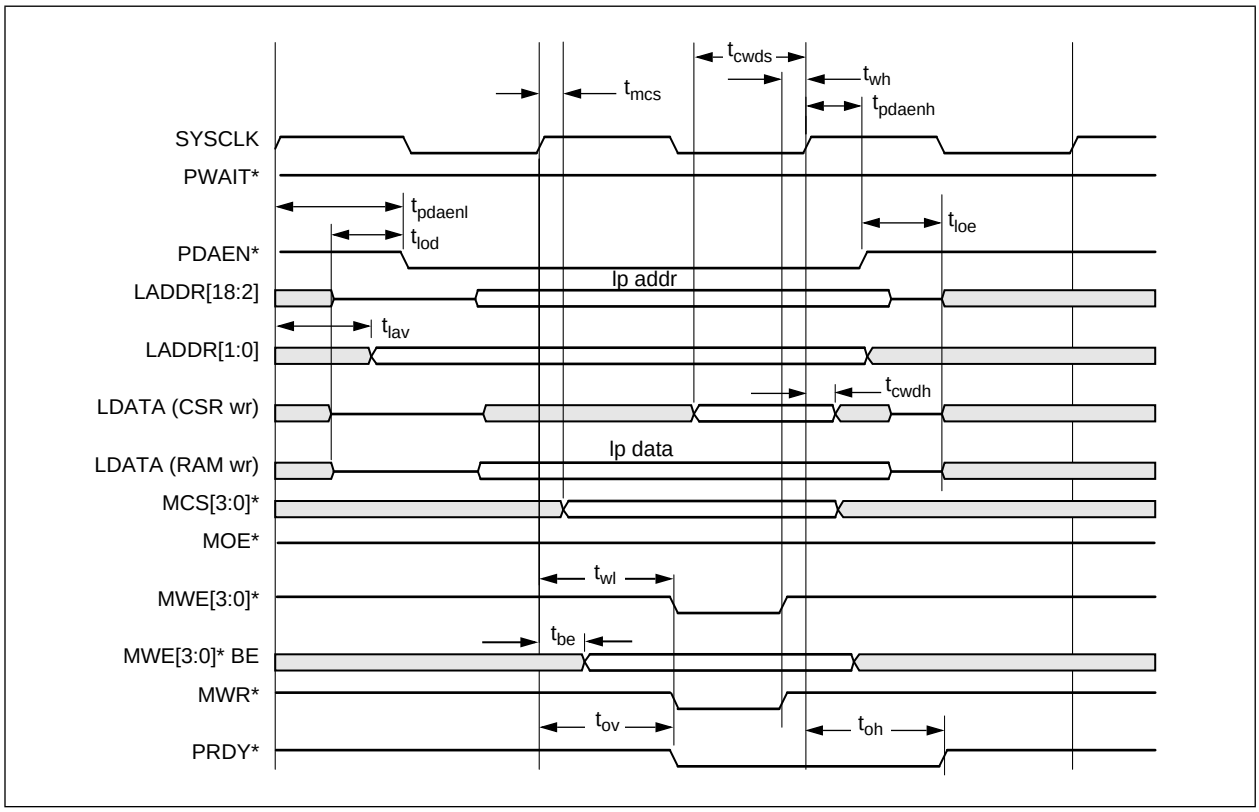


Table 5-16. Bt8230 Revision C Local Processor Memory Interface Timing

Symbol	Parameter	Min	Max	Units
t_{pdaenl}	SYSCLK to PDAEN* Low, Bus Recovery Cycle(1)		12	ns
t_{pdaenh}	SYSCLK to PDAEN* High, Bus Recovery Cycle(1)	2		ns
t_{lod}	LADDR[18:2], LDATA Output Disable to PDAEN* Low, Bus Recovery Cycle(1)	4		ns
t_{loe}	PDAEN* High to LADDR[18:2], LDATA Output Enable, Bus Recovery Cycle(1)	9		ns
t_{mcs}	SYSCLK to MCS*[3:0] Valid(1)	1	6	ns
t_{lav}	SYSCLK to LADDR[1,0] Valid(1, 2)	1	7	ns
t_{oel}	MOE* Active from SYSCLK(1, 3)	8	20	ns
t_{oeh}	MOE* Inactive from SYSCLK(1, 3)	1	10	ns
t_{wl}	MWR*, MWE[3:0] Active from SYSCLK(1, 3)		16	ns
t_{wh}	MWR*, MWE[3:0]* Inactive to SYSCLK(1, 3)		1	ns
t_{be}	MWE[3:0]* Byte Enables Valid from SYSCLK (RAMMODE = 1)(1)	1	7	ns
t_{crd}	CSR Read Data Output Valid	2	20	ns
t_{cwds}	CSR Write Data Setup to SYSCLK	8		ns
t_{cwdh}	CSR Write Data Hold from SYSCLK	0		ns
t_{las}	LADDR Setup to SYSCLK	12		ns

Notes: (1). See Figure 5-21 and Figure 5-22 for waveforms and definitions.
(2). The t_{lav} symbol is valid for second and subsequent accesses during burst transfers; see functional timing diagrams.
(3). In the case of two cycle memory, or when inserting wait states by PWAIT*, MOE*, MWE[3:0]*, and MWR* are extended across 2 or more clock cycles with the same relative timing to SYSCLK, see the functional timing diagrams in Section 3.3.5.

Figure 5-21. Bt8230 Rev C Local Processor Read Timing

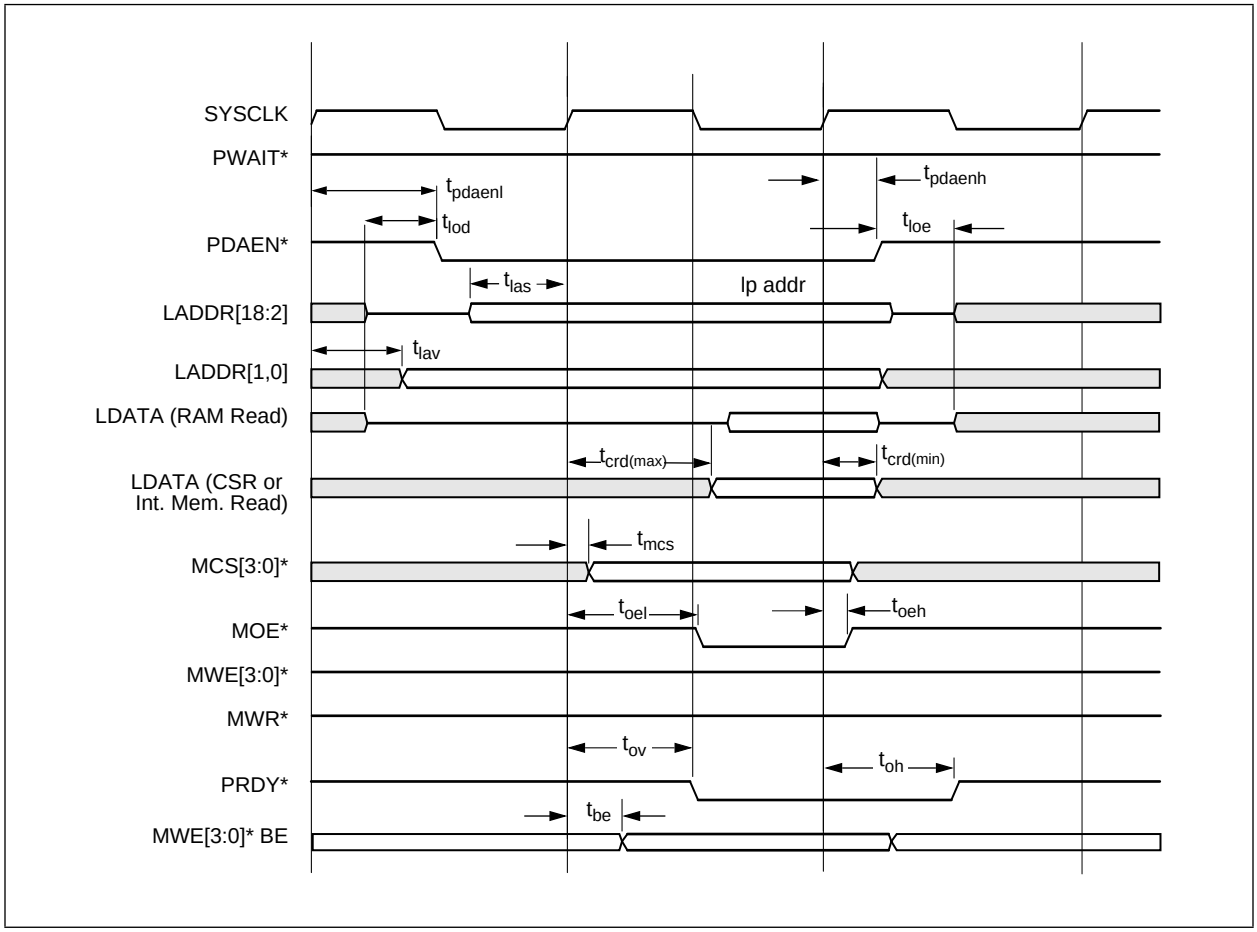
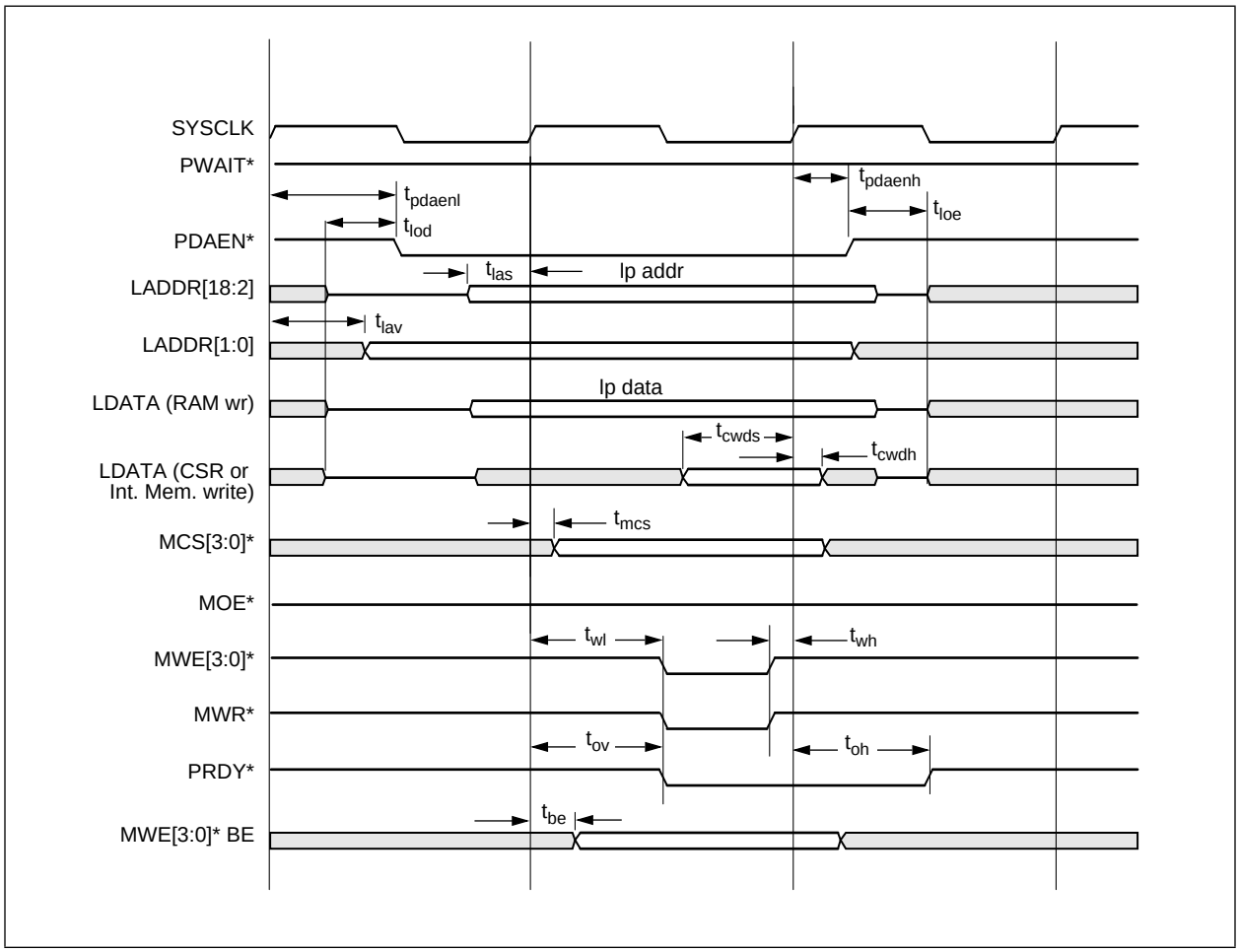


Figure 5-22. Bt8230 Rev C Local Processor Write Timing



5.2 Absolute Maximum Ratings

Stresses above those listed as absolute maximum ratings (see Table 5-17) may cause permanent damage to the device. This is a stress rating only, and functional operation of the device at these or any other conditions above those indicated in other sections of this document is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability. This device should be handled as an ESD-sensitive device. Voltage on any signal pin that exceeds the power supply voltage by more than +0.5 V can induce destructive latchup.

Table 5-17. Absolute Maximum Ratings

Parameter	Value	Unit
Supply Voltage	–0.5 to +6.0	V
Input Voltage	–0.5 to (Vdd + 0.5)	V
Output Voltage	–0.5 to (Vdd + 0.5)	V
Operating Temperature—No Air Flow	–40 to 85	C°
Storage Temperature	–40 to 125	C°
Operating Supply Voltage	4.75 to 5.25	V
Maximum Current @ Max Clock Frequencies	375	mA
MTBF	1.61×10^7	Hours
ESD	2000	V
θJC	6.9°	C/W
θJA (Still Air 0 LFPM)	31°	C/W
θJA (Air Flow 200 LFPM)	26°	C/W

5.3 DC Characteristics

The DC Characteristics of the Bt8230 are given in Table 5-18. The Bt8230 consumes 1.5 W at full processing rate.

Table 5-18. DC Characteristics

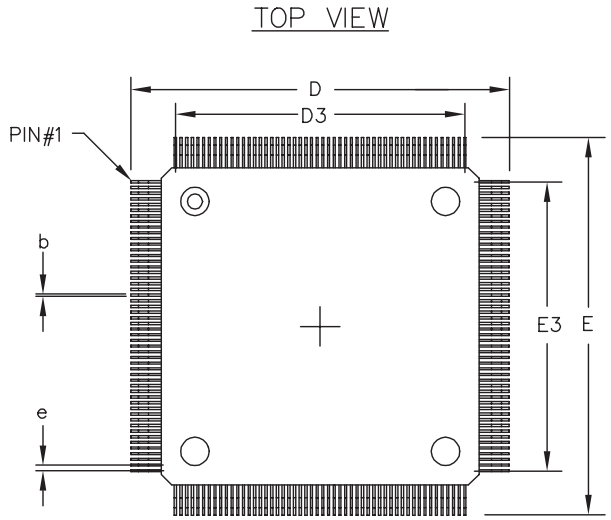
Parameter	Conditions	Min	Max	Units
Output Voltage High	IOH = -0.8 mA	Vdd - 0.1		V
Output Voltage Low	IOL = 0.8 mA		Vss + 0.1	V
Input Voltage High		2.0	Vdd + 0.5	V
Input Voltage Low		-0.5	0.8	V
Input Leakage Current	Vin = Vdd or GND	-10	10	µA
Three-state Output Leakage Current	VOUT = Vdd or GND	-10	10	µA
Input Capacitance			7	pF
Output Capacitance			7	pF

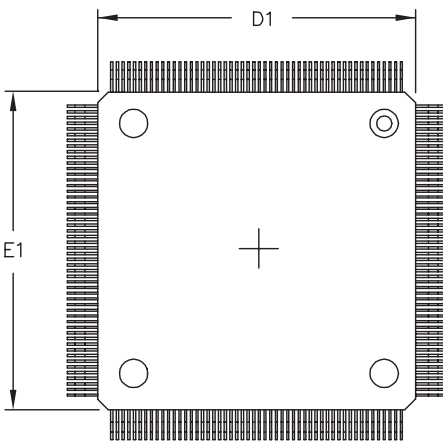
Notes: 1. All outputs are CMOS drive levels and can be used with CMOS or TTL logic.
2. All inputs are TTL compatible.

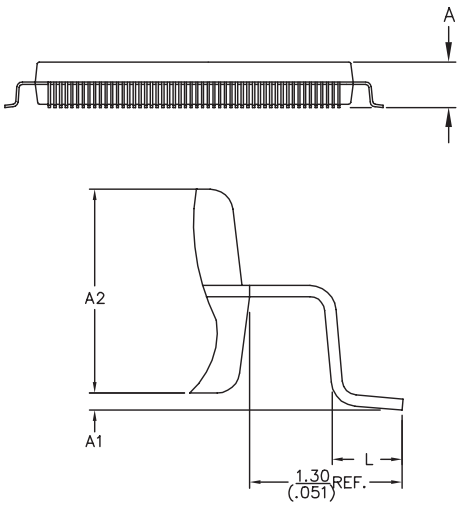
5.4 Mechanical Specifications

The Bt8230 208-pin PQFP package is illustrated in Figure 5-23.

Figure 5-23. 208-Pin Plastic Quad Flat Pack (PQFP)







All Dimensions in Millimeters			
Symbol	MIN	NOM	MAX
A			4.10
A_1	0.25		
A_2	3.20	3.40	3.60
D	30.60 BSC.		
D_1	27.90	28.00	28.10
D_3	25.50 REF.		
E	30.35	30.60	30.85
E_1	27.90	28.00	28.10
E_3	25.50 REF.		
L	0.45	0.60	0.75
e	0.50 BSC.		
b	0.14	0.22	0.30

5.5 Pin Assignments

Figure 5-24 is a pinout configuration of the Bt8230. A pin listing is given in Table 5-19.

Figure 5-24. Bt8230 Pinout Configuration

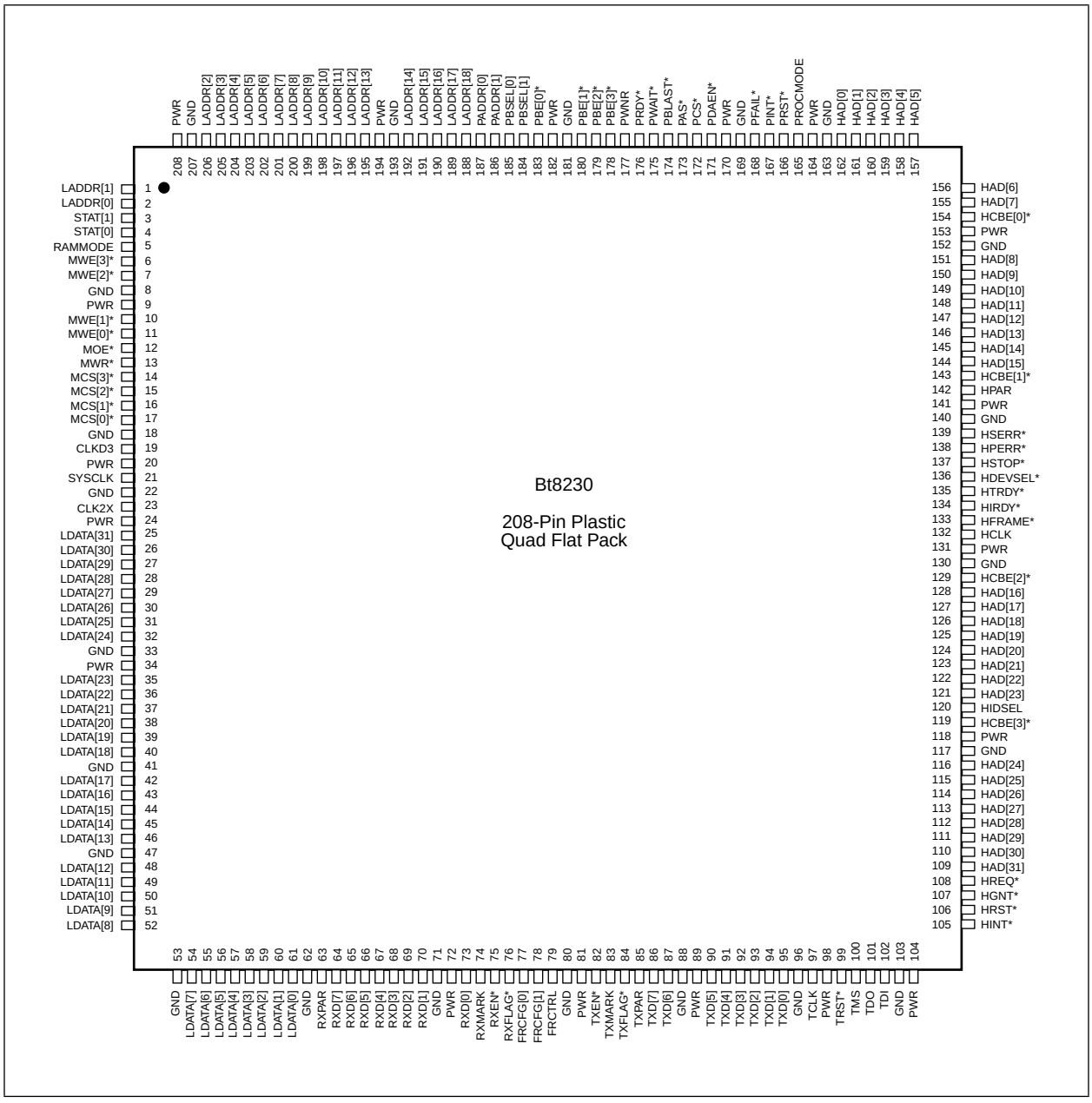


Table 5-19. Pin Descriptions *(1 of 2)*

Pin	Pin Label	I/O	Pin	Pin Label	I/O	Pin	Pin Label	I/O
1	LADDR[1]	I/O	36	LDATA[22]	I/O	71	GND	–
2	LADDR[0]	I/O	37	LDATA[21]	I/O	72	PWR	–
3	STAT[1]	O	38	LDATA[20]	I/O	73	RXD[0]	I
4	STAT[0]	O	39	LDATA[19]	I/O	74	RXMARK	I
5	RAMMODE	I	40	LDATA[18]	I/O	75	RXEN*	I/O
6	MWE[3]*	O	41	GND	–	76	RXFLAG*	I/O
7	MWE[2]*	O	42	LDATA[17]	I/O	77	FRCFG[0]	I
8	GND	–	43	LDATA[16]	I/O	78	FRCFG[1]	I
9	PWR	–	44	LDATA[15]	I/O	79	FRCTRL	I
10	MWE[1]*	O	45	LDATA[14]	I/O	80	GND	–
11	MWE[0]*	O	46	LDATA[13]	I/O	81	PWR	–
12	MOE*	O	47	GND	–	82	TXEN*	I/O
13	MWR*	O	48	LDATA[12]	I/O	83	TXMARK	I/O
14	MCS[3]*	O	49	LDATA[11]	I/O	84	TXFLAG*	I/O
15	MCS[2]*	O	50	LDATA[10]	I/O	85	TXPAR	O
16	MCS[1]*	O	51	LDATA[9]	I/O	86	TXD[7]	O
17	MCS[0]*	O	52	LDATA[8]	I/O	87	TXD[6]	O
18	GND	–	53	GND	–	88	GND	–
19	CLKD3	O	54	LDATA[7]	I/O	89	PWR	–
20	PWR	–	55	LDATA[6]	I/O	90	TXD[5]	O
21	SYSCLK	O	56	LDATA[5]	I/O	91	TXD[4]	O
22	GND	–	57	LDATA[4]	I/O	92	TXD[3]	O
23	CLK2X	I	58	LDATA[3]	I/O	93	TXD[2]	O
24	PWR	–	59	LDATA[2]	I/O	94	TXD[1]	O
25	LDATA[31]	I/O	60	LDATA[1]	I/O	95	TXD[0]	O
26	LDATA[30]	I/O	61	LDATA[0]	I/O	96	GND	–
27	LDATA[29]	I/O	62	GND	–	97	TCLK	I
28	LDATA[28]	I/O	63	RXPAR	I	98	PWR	–
29	LDATA[27]	I/O	64	RXD[7]	I	99	TRST*	I
30	LDATA[26]	I/O	65	RXD[6]	I	100	TMS	I
31	LDATA[25]	I/O	66	RXD[5]	I	101	TDO	O
32	LDATA[24]	I/O	67	RXD[4]	I	102	TDI	I
33	GND	–	68	RXD[3]	I	103	GND	–
34	PWR	–	69	RXD[2]	I	104	PWR	–
35	LDATA[23]	I/O	70	RXD[1]	I	105	HINT*	OD

Table 5-19. Pin Descriptions *(2 of 2)*

Pin	Pin Label	I/O	Pin	Pin Label	I/O	Pin	Pin Label	I/O
106	HRST*	I	141	PWR	–	176	PRDY*	O
107	HGNT*	I	142	HPAR	I/O	177	PWNR	I/O
108	HREQ*	O	143	HCBE[1]*	I/O	178	PBE[3]*	I
109	HAD[31]	I/O	144	HAD[15]	I/O	179	PBE[2]*	I
110	HAD[30]	I/O	145	HAD[14]	I/O	180	PBE[1]*	I
111	HAD[29]	I/O	146	HAD[13]	I/O	181	GND	–
112	HAD[28]	I/O	147	HAD[12]	I/O	182	PWR	–
113	HAD[27]	I/O	148	HAD[11]	I/O	183	PBE[0]*	I
114	HAD[26]	I/O	149	HAD[10]	I/O	184	PBSEL[1]	I
115	HAD[25]	I/O	150	HAD[9]	I/O	185	PBSEL[0]	I
116	HAD[24]	I/O	151	HAD[8]	I/O	186	PADDR[1]	I
117	GND	–	152	GND	–	187	PADDR[0]	I
118	PWR	–	153	PWR	–	188	LADDR[18]	I/O
119	HCBE[3]*	I/O	154	HCBE[0]*	I/O	189	LADDR[17]	I/O
120	HIDSEL	I	155	HAD[7]	I/O	190	LADDR[16]	I/O
121	HAD[23]	I/O	156	HAD[6]	I/O	191	LADDR[15]	I/O
122	HAD[22]	I/O	157	HAD[5]	I/O	192	LADDR[14]	I/O
123	HAD[21]	I/O	158	HAD[4]	I/O	193	GND	–
124	HAD[20]	I/O	159	HAD[3]	I/O	194	PWR	–
125	HAD[19]	I/O	160	HAD[2]	I/O	195	LADDR[13]	I/O
126	HAD[18]	I/O	161	HAD[1]	I/O	196	LADDR[12]	I/O
127	HAD[17]	I/O	162	HAD[0]	I/O	197	LADDR[11]	I/O
128	HAD[16]	I/O	163	GND	–	198	LADDR[10]	I/O
129	HCBE[2]*	I/O	164	PWR	–	199	LADDR[9]	I/O
130	GND	–	165	PROCMODE	I	200	LADDR[8]	I/O
131	PWR	–	166	PRST*	O	201	LADDR[7]	I/O
132	HCLK	I	167	PINT*	OD	202	LADDR[6]	I/O
133	HFRAME*	I/O	168	PFAIL*	I	203	LADDR[5]	I/O
134	HIRDY*	I/O	169	GND	–	204	LADDR[4]	I/O
135	HTRDY*	I/O	170	PWR	–	205	LADDR[3]	I/O
136	HDEVSEL*	I/O	171	PDAEN*	I/O	206	LADDR[2]	I/O
137	HSTOP*	I/O	172	PCS*	I/O	207	GND	–
138	HPERR*	I/O	173	PAS*	I/O	208	PWR	
139	HSERR*	OD	174	PBLAST*	I/O			
140	GND	–	175	PWAIT*	I			

Appendix A

Related Standards

The following is a list of the ITU and ANSI standards relevant to the Bt8230.

- ATM Forum UNI Rev 3.1: User-Network Interface Specification
- ATM Forum Utopia Level 1 Specification
- PCI Rev 2.0: PCI Local Bus Specification
- ANSI T1.627-1993:Broadband ISDN—ATM Layer Functionality and Specification
- ANSI T1.629-1993: Broadband ISDN—ATM Adaptation Layer 3/4 Common Part Functions and Specification
- ANSI T1.635-1994: Broadband ISDN—ATM Adaptation Layer Type 5 Common Part Functions and Specification
- I.363: B-ISDN ATM Adaptation Layer Specification
- I.610, 3/93: B-ISDN Operation and maintenance Principles and Functions
- GR-1248: Generic Requirements for Operation of ATM Network Elements
- IEEE Std 1149.1-1990: IEEE Standard Test Access Port and Boundary Scan Architecture, and Supplement B, dated August 9, 1994 (P1149.1b/D11)

All of the documents listed above can be obtained from the following companies:

Bellcore
Customer Service
8 Corporate Place - Room 3C-183
Piscataway, NJ 08854-4156
1-800-521-CORE

PCI Special Interest Group
P.O. Box 14070
Portland, OR 97214
1-800-433-5177
1-503-797-4207

ATM FORUM
The ATM Forum
303 Vintage Park Drive
Foster City, CA 94404-1138

ANSI
11 West 42nd Street
New York, NY 10036
1-212-642-4900

For ITU documents:
Omnicom
Phillips Business Information
1201 Seven Locks Road, Suite 300
Potomac, MD 20854
1-800 OMNICOM (666-4266)

The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street
New York, NY 10017-2394

Appendix B

Memory Use and Queue Sizing

A system designer using the Bt8230 SRC can optimize memory usage and increase system performance by appropriately sizing the various reassembly and segmentation queues. As a Rockwell-supplied reference design, the Bt8230HPI firmware has many of these sizings built in. The Bt8230 Application Note provides guidelines for selecting queue sizes as a supplement to build-in sizing, or when the Bt8330 HPI is not being used. The guidelines also provide estimates of the memory allocation requirements for all of the Bt8230 queues and data structures residing in local and host memory. The only queue residing in host memory is the SEG Host Transmit Queue; all other queues exist in local memory.

NOTE: These are general guidelines and may not be applicable for all systems or in all situations.

Section 3.4.1, “Memory Setup,” and Section 3.5.1, “Local Memory Setup,” in this datasheet provide detailed information on the purpose and initialization requirements of the queues. Table B-1 lists the register values for queue sizes. Table B-2 and Table B-3 give the minimum and maximum sizes for each data structure region.

SEG Host Transmit Queue

The SEG Host Transmit Queue resides in host memory. Each queue entry is a buffer descriptor indicating that the host processor has a buffer in host memory ready for segmentation. The Bt8230 will process and remove the queue entries upon a write of the SEG_HXMIT (Segmentation Host Transmit) register [0x40]. The Bt8230 can process a queue entry on the order of once per cell slot. Therefore, unless all buffers are very close to one cell in length, the Bt8230 will process many transmit queue entries while transmitting a single buffer.

Example—A 1 kbyte buffer contains approximately 21 cells worth of data (21 * 48 bytes = 1008 bytes). While the Bt8230 segments this buffer, approximately 21 new transmit queue entries would also be processed.

With the exception of unusually small buffers, the Bt8230's transmit queue processing rate is greater than the host software's rate of writing entries. Consequently, strategies for sizing this queue depend primarily on the host interaction with the queue, not on Bt8230 queue processing. Two strategies the host processor might use to manage the transmit queue are **no checking** and **checking**.

No Checking

In this scenario, the number of transmit queue entries is greater than the maximum number of host buffers. This ensures that a transmit queue entry will never be overwritten. When the host processor has a buffer ready, it immediately writes the descriptor to the next slot in the queue. Clearly, this approach can only be used if the host has a known, fixed-size buffer pool and the transmit queue can be sized at least this large.

Checking

If the host's buffer pool can be larger than the transmit queue size, the host processor must invoke some sort of checking to make sure it does not overwrite pending transmit queue entries. In this case, the host processor must verify its ownership of the next slot on the queue prior to writing. One approach is to have the host software keep count of how many entries have been put onto the queue. When this number approaches the queue size, the host processor checks the Entries Pending field (SEG_HPND [bits 29–16]) of the SEG_HXMIT register), to determine how many slots have been processed and freed by the Bt8230. The number of entries now available to the host is equal to the queue size minus the number of pending entries. This number is used as the new queue size, the counter is reset to zero, and the process repeats.

In a checking scenario, the software architecture of the host determines the parameters to consider when sizing the transmit queue. When the host software can handle overflow of the transmit queue by queueing PDU buffers until a transmit queue entry becomes available, the optimal queue size is a function of the host's queueing depth, the driving application's maximum buffer burst size, and the desired frequency of queue checks. If overflow cannot be tolerated but checking is required, the desirable queue size is a function of the driving application's maximum buffer burst size and the frequency or interval of those bursts.

The following worst-case conditions should be met to avoid queue overflow:

- The number of queue entries in a burst is less than the transmit queue size
- The burst interval is greater than the worst-case time for the Bt8230 to process burst entries.
- The worst-case time for the Bt8230 to process burst entries ≈ number of burst entries * number of VCCs * 150 * SYSCLK period.

If the conditions above are violated, the host is adding entries faster than the Bt8230 can remove them, and an overflow condition will occur. This overflow condition occurs when the *entries pending* field of the SEG_HXMIT register is equal to the queue size. The rate at which the processor adds queue entries is based on a function of the application's traffic characteristics, including the number of PDUs, the length of each PDU, and the buffer sizes.

Example—A typical host can monitor the queue's utilization by incrementing a counter for every entry made to the queue. When the counter value approaches the queue size, the host checks the entries pending field and updates the counter with this value. The number of currently available entries is equal to the queue size minus the new counter value. Again, the host repeats the incrementing counter procedure. The queue should be sized large enough so that this reading of the SEG_HXMIT register is limited.

Suppose that the traffic characteristics and host buffer sizes create an average rate of transmit queue additions of 40 entries per 1 millisecond (ms). The Bt8230 will remove transmit queue entries at an average rate of one per cell slot time. The Bt8230 cell slot time translates into approximately 150 * SYSCLK period. For a 30 ns SYSCLK period, 40 queue entries would be removed in roughly (40 * 150 * 30 ns) = 0.18 ms. Since the Bt8230's average rate of processing the queue entries, R_{sar} = 40/0.18 ms, is greater than the host's rate, R_h = 40/1 ms, a minimum queue size (256 entries) should suffice.

However, suppose the host wants to do an occasional burst of 1000 entries to the queue. With a queue size of only 256, this burst would have to be broken up, causing host queueing. Host-initiated PCI reads of the SEG_HXMIT register would be required periodically during the last 744 transmit queue writes. A queue size of 1024 would eliminate the additional queueing and the PCI reads.

The host transmit queue size is fixed by setting the MAXHR field [bits 11, 10] in the SEG_CTRL register [0x20] to the corresponding 2 bits from Table B-1.

Table B–1. Register Field Values for Desired Queue Sizes

Register Field Bits	Number of Queue Entries
00	256
01	1024
10	4096
11	16384

SEG Local Transmit Queue

Each segmentation local transmit queue entry also indicates that a buffer is ready for segmentation. The size of this queue, which resides in local memory, should be determined according to the same guidelines as for the host transmit queue. The size of the local transmit queue is set by the MAXLR field [bits 9, 8] in the SEG_CTRL register according to Table B-1.

Since the local transmit queue resides in local memory, it offers a performance advantage over the host transmit queue. This advantage arises from the reduced latency of a local memory read as opposed to a PCI read, and results in increased segmentation throughput.

Both the Local and Host Transmit Rings can be used to segment host or local data. If a local processor is not being used, all segmentation data should be processed through the local transmit queue. This requires that the host processor write the transmit queue over the PCI bus, instead of the SRC reading the entry across the bus. Since PCI writes are more efficient than reads, this will not adversely affect the PCI bus utilization.

SEG Host Status Queue

Each segmentation host status queue entry indicates to the host processor that a host buffer (in streaming mode) or PDU (in message mode) has been completely segmented. If this queue fills, segmentation halts unless bit SEG_HS_DIS is set to one in the SEG_CRTL register. For optimum performance, the queue should be sized large enough to avoid overflows.

A sizing strategy similar to those discussed for the SEG Host Transmit Queue must be applied. If the host has a fixed-size buffer pool, sizing this queue to be greater than or equal to the number of buffers will guarantee that there are no overflows. If the host has a variable-sized buffer pool, host servicing of the queue must be capable of handling the worst-case queue entry depth in order to avoid overflows. The rate of queue entry processing is a function of the host processor queue service frequency and the number of entries processed per service. The rate of queue additions by the Bt8230 is a function of the number of active VCCs, the number of buffers per PDU, the segmentation mode (streaming or message), and the PDU scheduling. Assuming that the host can process all queued entries with each status queue service, the function is reduced to the equation below. The examples below will help in determining the correct values to plug into this equation:

$$\text{queue size required} = Np * Nb$$

where:

Np = maximum number of PDUs completing segmentation during a host processor status queue service interval
Nb = maximum number of buffers per PDU (in streaming mode)
(in message mode, this = 1)

The number of active VCC connections is an important factor in sizing this queue. In the simplest case, only one active VCC, the rate of status queue entries, is easily approximated. Entries are made sequentially as PDUs are segmented. Overflow of the status queue can be avoided using the smallest queue size, by

matching the average rate of status queue entry additions with the host's average rate of status queue entry processing or deletions. However, as the number of active VCCs increases, the likelihood of bursts of status queue entries increases. This may cause the queue to overflow, thereby halting segmentation. (See Examples 1 and 2 below).

Example 1: One Active VCC—Suppose there is one active VCC and that the average segmentation completion rate is 10 buffers every 0.5 ms. If the host polls and services the status queue once every 5 ms, it could process roughly 100 buffers each time. A minimum queue size of 256 entries would be adequate. If the host polled and serviced the status queue once every 25 ms and processed roughly 500 buffers each time, a queue size of 1024 would suffice. If the host software is interrupt-driven rather than polled, the queue entries are processed as they are added, and a minimum queue size would suffice.

Example 2: 500 Active VCCs—Now suppose that there are 500 active VCCs instead of one, and the same average rate of buffer completion of 10 buffers every 0.5 ms still exists. Due to the asynchronous nature of multiple VCCs, there is a potential peak rate of buffer completion much greater than the average. In the extreme case of two-cell PDUs, half of the cells transmitted will precipitate a status queue entry. Therefore, out of 600 successive cells, 300 would result in status queue writes. At an STS-3c line rate, this would occur in 2.74 µs/cell * 600 cells = 1.64 ms. If this occurred with a queue size of 256 and a host polling interval of 5 ms, it is likely that segmentation would stall until the host processed the status queue. Increasing the queue size to the next largest size, 1024, would eliminate the stall and increase system performance. If the host was interrupt-driven, it might be able to keep up sufficiently with the burst to operate with a minimum queue size of 256. This might not be the case if the host incurs large latency times with its interrupt service routine.

The host status queue size is fixed by setting the MAXHS field in the SEG_CTRL register to the corresponding two bits from Table B-1.

SEG Local Status Queue

Each segmentation local status queue entry indicates that a local memory buffer has been completely segmented. The size of this queue is determined according to the same guidelines given for the SEG Host Status Queue, but is related to PDUs segmented from local memory. The local status queue size is fixed by setting the MAXLS field in the SEG_CTRL register according to Table B-1.

SEG Reserved Area

The segmentation coprocessor requires this space in local memory to store internal state information. Two 2-bit fields in the SEG_CTRL register, MAXI and MAXPND, are used to define the size of this space according to the following equation:

$$\text{reserved space required} = (MAXI_{size} + MAXPND_{size}) * 4 \text{ bytes}$$

$MAXI_{size}$ and $MAXPND_{size}$ are encoded according to Table B-1. $MAXI_{size}$ dictates the maximum intercell interval for segmentation PDUs. In other words, it determines the slowest possible cell rate for all rate-based VCCs. Specifically, $1/MAXI_{size}$ is the minimum cell rate. Using Table B-1, the smallest $MAXI_{size}$ which allows the desired minimum cell rate should be chosen. $MAXPND_{size}$ dictates how many active VCCs are allowed. As a general guideline, set $MAXPND_{size}$ according to the maximum number of simultaneously segmenting VBR VCCs.

SEG VCC Table

A VCC table entry is required for each active VCC. Therefore, allocate this block of local memory according to the following equation:

VCC table space = maximum number of active VCC connections * 32 bytes.

Depending on the method of OAM traffic generation, additional VCC Table entries may be required. If OAM cells are generated using the overwrite capabilities of the segmentation buffer descriptor VCC_CTRL field (WR_PTI and WR_VCI), then no additional VCC Table entries are needed. If more flexible OAM insertion is needed, a separate OAM VCC can be created. This requires additional VCC Table entries. The number of active VCCs is the number of user data channels plus the number of distinct OAM VCC Table entries.

SEG Buffer Descriptors (Host and Local)

Before a buffer is segmented, buffer descriptors are added to the transmit queue by the controlling processor and copied to the buffer descriptor chain in local memory space by the Bt8230. The host writes the transmit queue entries in ascending order and restarts at the beginning of the queue after the end of the queue space is reached. Once the Bt8230 copies a transmit queue entry to a buffer descriptor, that transmit queue entry can be reused by the host. It may be reused before the corresponding segmentation buffer has been processed by the Bt8230. The operation of the local transmit queue is analogous to the host transmit queue.

Since the transmit queue entries may be reused while the buffer descriptor is still allocated, a potential need exists for many more buffer descriptors than transmit queue entries. The ratio between the transmit queue size and the number of buffer descriptors depends on the desired buffering location in the system. For example, if bursts of PDUs consisting of several large buffers are common, and the host cannot buffer the descriptor information, the buffer descriptor space may have to be quite large compared to the transmit queue size. On the other hand, if buffering is done by the processor (e.g., the processor uses a fixed-size buffer pool), then this space may be sized the same as the transmit queue. In terms of VCCs and a fixed-size buffer pool, the number of descriptors required is proportional to the number of VCCs multiplied by the number of buffers allotted to each VCC.

SEG PM Table

Each active channel with PM (Performance Monitoring) enabled requires a 4-word PM table entry. This space resides at the top of the SEG Host Status Queue in local memory. The maximum number of entries is 128.

RSM Host Status Queue

An entry in the RSM Host Status Queue indicates that a buffer of reassembled data in host memory is complete or that a reassembly error has occurred. Under normal conditions the reassembly coprocessor will halt when the queue is full. The strategies for sizing this queue are similar to those discussed for sizing the SEG Host Status Queue. For simplicity, assume the Bt8230 is in loopback mode, and RSM buffers are the same size as SEG buffers. In this case, the examples applied to the SEG Host Status Queue are directly applicable to RSM Host Status Queue sizing. Therefore the following equation is valid:

$$\text{queue size required} = Np * Nb$$

where:

Np = maximum number of PDUs completely reassembled in a host processor status queue service interval

Nb = maximum number of buffers per PDU in streaming mode (in message mode, Nb = 1)

The RSM Host Status Queue size is determined by the HS _SIZE field [bits 5,4] in the RSM_CTRL register [0x70] according to Table B-1.

RSM Local Status Queue

An entry in the RSM Local Status Queue indicates that a buffer of reassembled data in local memory is ready or that a reassembly error condition has occurred. Size this queue with the same guidelines provided for the Host Status Queue section on page 199, but related to PDUs reassembled in local memory. The RSM Local Status Queue size is determined by the LS_SIZE field [bits 1,0] in the RSM_CTRL register [0x70] according to Table B-1.

RSM Host Free Buffer Queue

Each entry in the RSM Host Free Buffer Queue points to a buffer in host memory. These buffers are used by the SRC to store reassembly data. If no buffers are available, the reassembly machine will halt operation and data will be dropped.

To avoid this condition, the free buffer queue should be sized using the guidelines for the RSM Host Status Queue in streaming mode, as discussed. The HF_SIZE field in the RSM_CTRL register assigns a size to this queue.

RSM Local Free Buffer Queue

Each entry in the RSM Local Free Buffer Queue points to a buffer in local memory. This queue should be sized using the guidelines from the RSM Local Status Queue in streaming mode, as discussed above. The LF_SIZE field in the RSM_CTRL register assigns a size to this queue.

RSM Hash Table

A system design trade-off exists in the choice of size for the RSM Hash Table. A small table, relative to the number of VCCs, will result in long hash bucket searches and possibly reduced performance. A relatively large table will use more memory. A good trade-off between memory and performance is to size the hash table so that the maximum hash bucket search is two entries.

This appendix provides a C program to assist in the sizing of the hash table. (See Hash Table Sizing C Program on page 203.) It calculates the minimum hash table for a given list of up to 16 k VPI/VCI pairs and maximum hash bucket chain depth.

The size of the hash table is determined by the TAB_SIZE[3:0] field in the RSM_CTRL register. This encoded value assigns a size between one and 16 k entries of 4 bytes each. It does not include the reserved region at the top of the table. The reserved region contains 4 words for error counters and from 0 to 124 words for performance monitoring.

RSM Hash Bucket

A hash bucket is required for each VCC, and in AAL3/4 for each MID of a given VCC. Therefore, allocate local memory using this formula, counting MIDs as connections:

Hash Bucket space = maximum number of connections * 20 bytes

Additional hash buckets may have to be created to handle OAM traffic. The exact number for this use is highly dependent on the reassembly OAM handling method chosen.

RSM VCC Table

A VCC table entry maintains the state information for an entire CPCS-PDU. Therefore, one entry is required for each active VPI/VCI and MID. Size this space according to the maximum number of active RSM connections.

RSM VCC Table Space=maximum number of active RSM connections * 32 bytes.

Memory Requirements

Table B-2 and Table B-3 give the minimum and maximum sizes for each data structure region.

Table B–2. Local Memory Requirements

Memory Descripiton	Minimum Size, Bytes	Maximum Size, Bytes	Increment Formula(1)
SEG Host Status Queue	(256 * 8) = 2 k	(16 k * 8 + 127 * 32) = 132 k	(Table 1 Value) * 8 + (# of PM VCCs) * 32
SEG Local Status Queue(2)	(256 * 8) = 2 k	(16 k * 8) = 128 k	(Table 1 Value) * 8
Reserved SEG. Area	(256 + 256) * 4 = 2 k	(16 k + 16 K) * 4 = 128 k	(Table 1 Value + Table 1 Value) *4
SEG VCC Table	(1 VCC) = 32	(64 k * 32) = 2 M	(Table 1 Value) * 32
SEG Buffer Descriptors (Host and Local)	(1 Descriptor) = 16	User Defined	(# Descriptors) * 16
SEG Local Transmit Queue(2)	(256 * 16) = 4 k	(16 k * 16) = 256 k	(Table 1 Value) * 16
RSM Host Status Queue	(256 * 16) = 4 k	(16 k * 16) = 256 k	(Table 1 Value) * 16
RSM Local Status Queue(2)	(256 * 16) = 4 k	(16 k * 16) = 256 k	(Table 1 Value) * 16
RSM Host Free Buffer Queue	(256 * 8) = 2 k	(16 k * 8) = 128 k	(Table 1 Value) * 8
RSM Local Free Buffer Queue(2)	(256 * 8) = 2 k	(16 k* 8) = 128 k	(Table 1 Value) * 8
Hash Table	20	(16 k* 4 + 127 * 4) = 65 k	(2 ^ n) * 4 + 4 + (# of PM VCCs) * 4(3)
Hash Bucket	(1 VCC) = 20	(16 k* 20) = 320 k	(# of VCCs) * 20
RSM VCC Table	(1 VCC) = 32	(16 k* 32) = 512 k	(# of VCCs) * 32
RSM/SEG Queue	2 k	2 k	N/A

Notes: (1). See the Bt8230 datasheet Segmentation and Reassembly sections for more detail.
(2). Minimum size of these local data structures could be zero if all processing is done by the host processor.
(3). In the Increment Formula, n = an integer from 0 to 14.

Table B–3. Host Memory Requirements

Memory Descripiton	Minimum Size, Bytes	Maximum Size, Bytes	Increment Formula
SEG Host Transmit Queue	(256 * 16) = 4 k	(16 k* 16) = 256 k	(Table 1 Value) * 16

Hash Table Sizing C Program

The following is a C program which can be used to assist in sizing the hash table.

```
/*********************************************************
        *****
 * *
 * hsh_tbl_sz.c *
 * *
 * Calculate the smallest hash table size for a given
       VPI/VCI list and max *
 * hash bucket chain depth. *
 * *
 * hsh_tbl_sz(fp,max_length) *
 * *
 *        fp          file pointer for input *
 *        max_length  maximum hash bucket chain depth *
 * *
 * Output: *
 *        returns hash table size *
 * *
 * Written: JRG 5/95*
*********************************************************
        ****/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define DEBUG 1

long int hsh_tbl_sz(char * fname, float max_length);

main(int argc, char *argv[])
{
  hsh_tbl_sz(argv[1], atoi(argv[2]));
}

long int hsh_tbl_sz(char * fname, float max_length)
```

```
{
    typedef struct {unsigned int vpi, vci;} VCC;
    VCC *header;
    VCC *bs_hdr;
int i, j, k, hash_total, hash_match;
    float table_size, vcc_total;
    unsigned int endoffile, tbl_sz_ok, tbl_sz_found, mask,
        temp, hash;
    unsigned int *chain;
    unsigned int *phash;
    FILE *fp;
header = (VCC *)malloc(1024 * sizeof(VCC));
    chain = (unsigned int *)calloc(1024, sizeof(unsigned
        int));
    phash = (unsigned int *)calloc(1024,  sizeof(unsigned
        int));
    if (!phash) {printf("\n not enough memory for vcc_total
        %d!!",vcc_total); exit(-1);}
   if (!header) {printf("\n not enough memory for vcc_total
        %d!!",vcc_total); exit(-1);}
    if (!chain) {printf("\n not enough memory for vcc_total
        %d!!",vcc_total); exit(-1);}
    fp = fopen(fname, "r");
    /* read in VPI/VCI list from file to array */
    bs_hdr = header;
    vcc_total = 0;
    endoffile = 0;
    while (!endoffile) {
        if(fscanf(fp,"%X",&temp) == EOF) endoffile = 1;
else {
  header->vpi = temp >> 12;
  header->vci = temp << 20;
  header->vci = header->vci >> 20;
          #ifdef DEBUG
     printf("read header #%0f from file:
        %x%x\n",vcc_total,header->vpi,header->vci);
          #endif
  header++;
  vcc_total++;
       }
    }
```

```
    /* set starting table size */
    tbl_sz_found = 0;
    hash_total = 0;
    table_size = vcc_total / max_length;
    if (table_size == 1)       {mask = 0x0L; tbl_sz_found =
        1;}
    else if (table_size <= 2)  mask = 0x1L;
    else if (table_size <= 4)  mask = 0x3L;
    else if (table_size <= 8)  mask = 0x7L;
    else if (table_size <= 16) mask = 0xfL;
    else if (table_size <= 32) mask = 0x1fL;
    else if (table_size <= 64) mask = 0x3fL;
    else if (table_size <= 128) mask = 0x7fL;
    else if (table_size <= 256) mask = 0xffL;
    else if (table_size <= 512) mask = 0x1ffL;
    else if (table_size <= 1024) mask = 0x3ffL;
    else if (table_size <= 2048) mask = 0x7ffL;
    else if (table_size <= 4096) mask = 0xfffL;
    else {mask = 0x3fffL; tbl_sz_found = 1;}
    /* hash VPI/VCI and add to chain */
    while (!tbl_sz_found) {
      i = 0;
      header = bs_hdr;
      tbl_sz_ok = 1;
      while ((i < vcc_total) && (tbl_sz_ok)) {
        hash = (header->vpi ^ header->vci) & mask;
        #ifdef DEBUG
   printf("header: %x%x hashed to %x with mask %x\n",header-
        >vpi,header->vci,hash,mask);
        #endif
j = 0;
hash_match = 0;
if (hash_total == 0) {
   phash[0] = hash;
   hash_total = 1;
}
while ((j < hash_total) && !hash_match) {
    if (hash == phash[j]) {
       hash_match = 1;
       chain[j]++;
```

```
      if (chain[j] > max_length) {
 if (mask == 0x3fffL) {
    printf("\n EXIT: unable to satisfy chain depth rqrmnt
         w/largest table size\n");
    /* printf("\n EXIT: chain depth: %d, max specified:
         %f\n",chain[j],max_length);*/
    exit(-1);
 }
 mask = mask*2 + 1;
 tbl_sz_ok = 0;
                #ifdef DEBUG
    printf("\n chain length exceeded max, bumping
         table_size, new mask: %x\n",mask);
                #endif
 /* clear chain counts*/
 for (k=0;k < vcc_total;k++) chain[k] = 0;
 hash_total = 0;
      }
    }
    else j++;
}
if (!hash_match) {
  hash_total++;
  phash[hash_total] = hash;
}
i++;
header++;
     }
     if (tbl_sz_ok)
        tbl_sz_found = 1;
    }

    /* convert mask to table_size and display */
    printf("minimum table_size meeting maximum chain depth
         requirement = %d\n",mask+1);

  }
```

Appendix C

PCI Bus Interface Performance

This appendix describes performance of the Bt8230's PCI Local Bus Interface using the Logic Modeling Corporation (LMC) PCI Model Regression Test Suite. Rockwell has labelled the PCI bus interface in the Bt8230 the Bus Interface Unit (BIU).

The performance of the BIU is, in general, dependent on a number of system parameters. The following assumptions are made when predicting performance:

1. A 33 MHz PCI bus clock and a 50 MHz local device clock are used. Frequency and phase differences between the two clocks have a material and wholly unpredictable effect on the BIU latency and throughput, due to the synchronization required when crossing clock domains. Thus, a particular PCI/device clock relationship must be established before attempting to measure performance.
2. When the BIU is acting as a bus master, the addressed target responds in one PCI clock cycle. If the target delays the assertion of either DEVSEL* or TRDY* during a bus transaction, this delay must be factored into the latency and throughput calculations.
3. The Fast Back-to-Back Enable bit [COMMAND field, FB_EN bit] in the PCI Configuration Register is enabled.
4. All addresses are aligned to a word boundary, whether the BIU is acting as a master or a slave. Also, all transfer sizes are multiples of 4 bytes (to make full use of the 32-bit data bus).

   If the DMA controller supplies an unaligned address to the PCI BIU master, the burst transaction will be broken up into unaligned and word-aligned data phases. This will result in a maximum of two additional data phases beyond that required for transferring the same amount of word-aligned data. The maximum number of data phases that are required by the PCI BIU master for a transaction, as a function of the number of bytes of the actual data that are transferred by the DMA controller, is, therefore, derived by dividing the number of bytes by 4 and adding 2.

   If the address presented by the external bus master on the PCI Address/Data (AD) bus is unaligned (the AD[1:0] bits are non-zero during the address phase), the PCI BIU slave logic transfers a single word and then signals a target disconnect. This causes the external master to break up the non-standard burst into a sequence of single-word transfers.

5 PCI bus arbitration delay is taken to be zero.

If the PCI bus arbiter parks the bus with the master that initiated the last transaction, the arbitration delay for that master is zero for subsequent transactions. If the arbiter does not perform bus parking or if the bus was parked at a different master, the additional delay required to obtain access to the bus must be factored into the performance calculations.

6 When the PCI BIU is acting as a target, the downstream register/memory access unit is capable of accepting (for writes) or returning (for reads) data words with a single PCI clock cycle.

The PCI BIU target can access both on-chip registers and local memory. The access to the on-chip registers is always done in one clock cycle by the register/memory access unit. However, all host reads from local memory and host writes of greater than 3 words will incur an additional delay that is equal to the time required by the register/memory access unit to be given access to the local memory.

7 The PCI BIU master is assumed to not be busy at the time that the transaction is begun by the DMA controller.

If this is not the case, then the present PCI BIU master transaction will have to be completed prior to the start of the new transaction. This maximum latency is approximately 35 PCI clock cycles, assuming that the PCI BIU bus master has full use of the PCI bus and that the master was busy doing both read and write transactions. However, under these conditions, the initial bus latency listed in the table below can be decreased by the number of PCI clock cycles required for clock synchronization between the PCI clock and SYSCLK inside the PCI BIU (because these cycles will occur in parallel with the data phase of the previous transaction on the PCI bus).

NOTE: The master can deassert IRDY# in the middle of a burst access. If during a master write, the FIFO goes empty, IRDY# will be deasserted. During a master read, the FIFO will never go full, therefore IRDY# will never be deasserted in this case.

The SRC never generates target abort as a slave. As a master, the SRC responds to a target abort by another slave.

Master Mode

Using the above assumptions, the predicted performance of the BIU master is given in Table C-1:

Table C–1. Bt8230 PCI Bus Interface Unit (BIU) Master Performance

Transaction	Max. Latency	Max. Sustained Bandwidth
Burst Reads	10	1 cycle/word
Burst Writes	7	1 cycle/word

The latency in this context refers to the time (in PCI clock cycles) between the initiation of a read or write transfer by the DMA Controller, and the time that the first read data word is returned to the DMA controller, or the first write data word is latched off the PCI bus.

Bus Master Read Access

If the bus master read access is examined in greater detail, the initial maximum latency of 10 PCI clock cycles comes from the following:

- Cycles 1-2: cycles required for the DMA to write the address and length of the transfer into the PCI BIU internal master read command FIFO.
- Cycles 3-4: cycles required for synchronization between SYSCLK-PCI clock interface, master state machine change, and loading of the AD bus output register internal to the PCI BIU interface.
- Cycle 5: the PCI bus address phase.
- Cycle 6: the first cycle of the PCI data phase. The PCI bus specification demands that the first data phase on a read transaction be a turn-around cycle; the AD bus is placed into high impedance, and the target is required to deassert TRDY*.
- Cycle 7: the PCI bus data phase completes.
- Cycles 8-9: the PCI read data is registered internally, is synchronized across the PCI clock-SYSCLK interface, and is made available to the DMA read controller.
- Cycle 10: DMA read controller can start reading data from master read data FIFO.

Bus Master Write Access

If the bus master write access is examined in greater detail, the initial maximum latency of 7 PCI clock cycles comes from the following:

- Cycles 1-2: cycles required for the DMA to write the command, address, and first data word into the PCI BIU internal master write FIFO.
- Cycles 3-4: cycles required for synchronization between SYSCLK-PCI clock interface and master state machine change. Master waits for two data elements to be in the Master Write FIFO (MWF FIFO) if transfer size is greater than 4 bytes.
- Cycle 5: loading of the AD bus output register internal to the PCI BIU interface.
- Cycle 6: the PCI bus address phase.
- Cycle 7: the PCI bus data phase.

Slave Mode

The predicted performance of the bus interface slave unit is given in Table C-2:

Table C–2. Bt8230 PCI BIU Slave Performance

Transaction	Max. Latency	Max. Sustained Bandwidth
Single-word reads	7	
Single-word writes	2	
Burst Reads	7	1 cycle/word
Burst Writes	2	1.5 cycles/word

In the table above, the latencies for single-word read and write cycles give the worst-case times (in PCI clock cycles) between the address phase of a read or write and the corresponding data phase. In the case of burst reads or writes, they provide the maximum delay between consecutive data phases. The sustained bandwidth figures assume that a long sequence of reads or writes is being performed. Therefore, the address, idle and turnaround phases can be neglected.

Bus Slave Read Access

If the bus slave read access is examined in greater detail, the initial maximum latency of 7 comes from the following:

- Cycle 1: Address phase on the PCI bus.
- Cycles 2-3: read command and address synchronized across the PCI clock-SYSCLK interface to the register/memory access unit.
- Cycle 4: 1 cycle for the register/memory access unit to return the data and write the data in the PCI BIU slave internal read FIFO.
- Cycles 5-6: synchronization across SYSCLK-PCI clock interface and loading of the data in the AD bus register internal to the PCI BIU.
- Cycle 7: first read dataword output on the PCI bus; TRDY* asserted by PCI BIU.

The slave burst read sustained bandwidth of 1 PCI clock cycle per dataword arises from the fact that the register/memory access unit will prefetch data, until either the upstream PCI bus master signals an end to the read transaction, or until the slave internal read FIFO becomes full.

Bus Slave Write Access

If the bus slave write access is examined in greater detail, the initial maximum latency of 2 comes from the following:

- Cycle 1: Address phase on the PCI bus.
- Cycle 2: data phase on PCI bus. The PCI BIU asserts TRDY* on this cycle since its internal write data FIFO can accept the data.

The slave burst write sustained bandwidth is 1.5 PCI clock cycles per dataword as opposed to one cycle because flow control is required once the internal write data FIFO begins to fill. Increasing the size of the Slave Write FIFO (SWF FIFO) improves the sustained slave write memory bandwidth towards the theoretical maximum of 1.0 cycles/word.

Performance Simulation

The following values were derived from a simulation using the LMC PCI Model Regression Test Suite.

Performance results for AAL5 full duplex; one buffer per CPCS-PDU.
Line rate is 356,124 cells/sec.
PDU size is payload only (does not include 8 byte trailer).

SEG and RSM Status Queues and RSM Free Buffer Queue processed every buffer.

Improved (software) queue processing removes extra PCI transaction per queue (for owner bit check).

PDU Size 32	
Maximum throughput	356124 pdu/sec
Reassembly throughput	314465 pdu/sec
Segmentation throughput	209643 pdu/sec
PCI usage	92%
With improved queue processing	90%
Local memory usage	81%
PDU Size 40	
Maximum throughput	356124 pdu/sec
Reassembly throughput	314465 pdu/sec
Segmentation throughput	207039 pdu/sec
PCI usage	92%
With improved queue processing	90%
Local memory usage	81%
PDU Size 88	
Maximum throughput	178062 pdu/sec
Reassembly throughput	178062 pdu/sec
Segmentation throughput	143678 pdu/sec
PCI usage	75%
With improved queue processing	71%
Local memory usage	72%
PDU Size 135	
Maximum throughput	118708 pdu/sec
Reassembly throughput	118708 pdu/sec
Segmentation throughput	105820 pdu/sec
PCI usage	59%
With improved queue processing	55%
Local memory usage	66%
PDU Size 184	
Maximum throughput	89031 pdu/sec
Reassembly throughput	89031 pdu/sec
Segmentation throughput	87719 pdu/sec
PCI usage	50%
With improved queue processing	43%
Local memory usage	60%

PDU Size 232	
Maximum throughput	71224 pdu/sec
Reassembly throughput	71224 pdu/sec
Segmentation throughput	71224 pdu/sec
PCI usage	46%
With improved queue processing	40%
Local memory usage	58%
PDU Size 280	
Maximum throughput	59354 pdu/sec
Reassembly throughput	59354 pdu/sec
Segmentation throughput	59354 pdu/sec
PCI usage	44%
With improved queue processing	34%
Local memory usage	54%
PDU Size 328	
Maximum throughput	50874 pdu/sec
Reassembly throughput	50874 pdu/sec
Segmentation throughput	50874 pdu/sec
PCI usage	43%
With improved queue processing	33%
Local memory usage	52%

Appendix D

Boundary Scan

The Bt8230 supports boundary scan testing conforming to IEEE standard 1149.1-1990 and Supplement B, 1994. This appendix is intended to assist the customer in developing boundary scan tests for printed circuit boards and systems that use the Bt8230. It is assumed that the reader is familiar with boundary scan terminology.

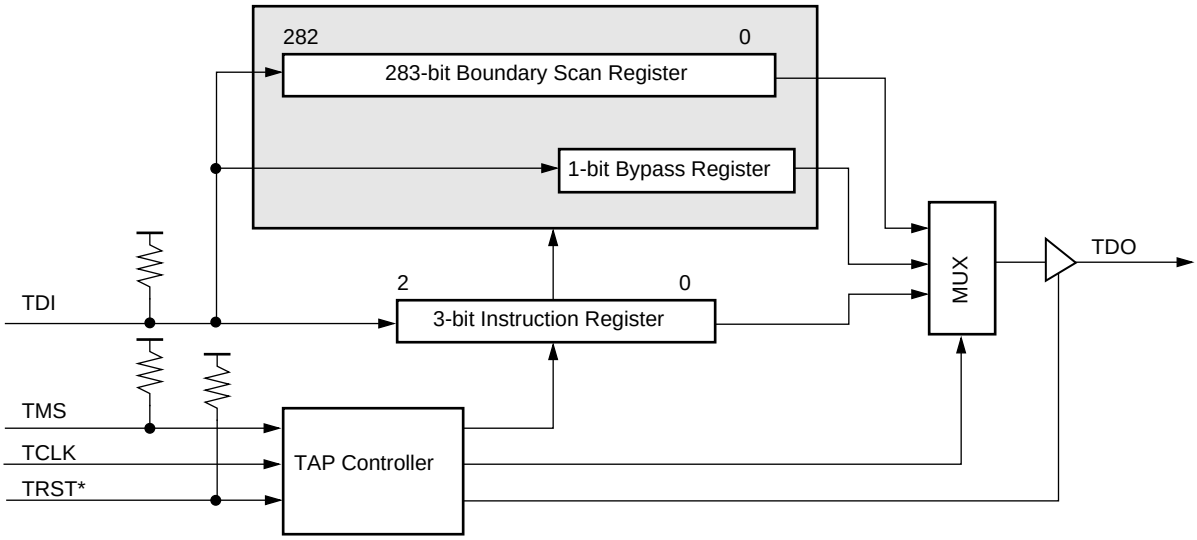
The Boundary Scan section of the Bt8230 provides access to all external I/O signals of the device for board and system-level testing. This circuitry also conforms to IEEE Std 1149.1-1990. The boundary scan test logic is accessed through five dedicated pins on the Bt8230 (see Table D-1).

Table D–1. Boundary Scan Signals

Pin Name	Signal Name	I/O	Definition
TRST*	Test Logic Reset	In	When at a logic low, this signal asynchronously resets the boundary scan test circuitry and puts the test controller into the reset state. This state allows normal system operation.
TCLK	Test Clock	In	Test clocking is generated externally by the system board or by the tester. TCLK can be stopped in either the high state or the low state.
TMS	Test Mode Select	In	Decoded to control test operations.
TDO	Serial Test Data Output	Out	Outputs serial test pattern data.
TDI	Serial Test Data Input	In	Input for serial test pattern data.

As shown in Figure D–1, The test circuitry includes the Boundary Scan Register, a BYPASS Register, an Instruction Register, and the Tap Access Port (TAP) controller.

Figure D–1. Test Circuitry Block Diagram



Note: If the boundary scan circuitry of the Bt8230 is not being used, tie TCLK and TRST* to ground.

Instruction Register

The Instruction Register (IR) is a 3-bit register with no parity. When the boundary scan circuitry is reset, the IR is loaded with the binary value 111 which is equivalent to the BYPASS Instruction value. The Capture-IR binary value is 001 and is shifted out TDO as the IR is loaded.

The eight instructions include three IEEE 1149.1 mandatory public instructions (BYPASS, EXTEST, and SAMPLE/PRELOAD) and five private instructions for manufacturing use only. Bit-0 (LSB) is shifted into the instruction register first. Refer to Table D–2, below:

Table D–2. IEEE Std. 1149.1 Instructions

Bit 2	Bit 1	Bit 0	Instruction	Register Accessed
0	0	0	EXTEST	Boundary Scan
0	0	1	RSTHIGH - Private	—
0	1	0	SAMPLE/PRELOAD	Boundary Scan
0	1	1	TMWFIFO - Private	—
1	0	0	TMRFIFO - Private	—
1	0	1	TSEGFIFO - Private	—
1	1	0	TRSMFIFO - Private	—
1	1	1	BYPASS	Bypass

BYPASS Register

The BYPASS Register is a 1-bit shift register used to pass TDI data to TDO. This is done to facilitate the testing of other devices in the scan path without having to shift the data patterns through the complete Boundary Scan Register of the Bt8230.

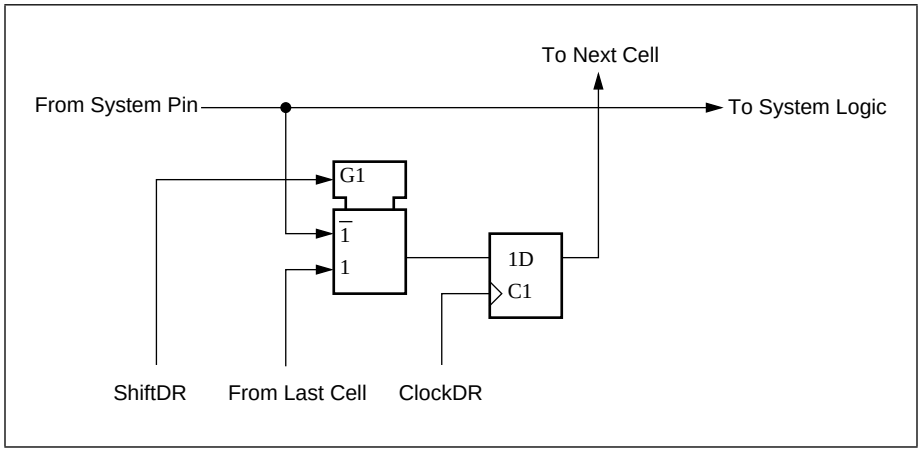
Boundary Scan Register

The Boundary Scan Register consists of three different types of registers: Input-Observe, Output-Control, and Output-Both. These registers correspond to IEEE 1149.1-1990 attributes and definitions.

Input-Observe

(IEEE 1149.1-1990 STD_1149_1_1990 Standard Boundary Cell name BC_4) This cell captures an input value (Figure D-2).

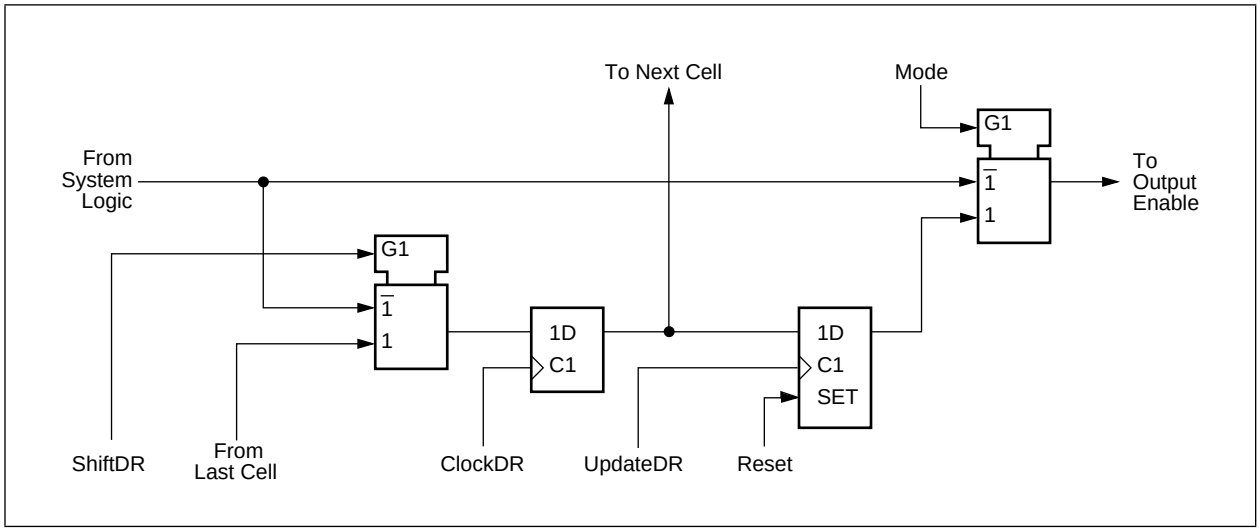
Figure D–2. Input-Observe Cell Diagram



Output-Control

(IEEE 1149.1-1990 STD_1149_1_1990 Standard Boundary Cell name BC_1) This cell passes the system logic controlled output enable to the output I/O cell or controls the output enable of the output I/O cell during EXTEST. Reset*, which is controlled by the TAP controller, will set the Update Register to a high state because the Bt8230 uses active low enables for the I/O cells (Figure D-3).

Figure D–3. Output-Control Cell Diagram



Output-Both

(IEEE 1149.1-1990 STD_1149_1_1990 Standard Boundary Cell name BC_1) This cell passes the system logic controlled output to the output I/O cell or controls the output of the output I/O cell during EXTEST (Table D-3).

Table D–3. Pin Descriptions *(1 of 5)*

Cell	Related Pin Name	Cell	Controlling Cell	Cell	Related Pin Name	Cell	Controlling Cell
1	LADDR2	output3	32	32	*	control	
2	LADDR2	input		33	LADDR18	output3	32
3	LADDR3	output3	32	34	LADDR18	input	
4	LADDR3	input		35	PADDR0	input	
5	LADDR4	output3	32	36	PADDR1	input	
6	LADDR4	input		37	PBSEL0	input	
7	LADDR5	output3	32	38	PBSEL1	input	
8	LADDR5	input		39	PBE0_NEG	input	
9	LADDR6	output3	32	40	PBE1_NEG	input	
10	LADDR6	input		41	PBE2_NEG	input	
11	LADDR7	output3	32	42	PBE3_NEG	input	
12	LADDR7	input		43	PWNR	output3	51
13	LADDR8	output3	32	44	PWNR	input	
14	LADDR8	input		45	PRDY_NEG	output3	283
15	LADDR9	output3	32	46	PWAIT_NEG	input	
16	LADDR9	input		47	PBLAST_NEG	output3	51
17	LADDR10	output3	32	48	PBLAST_NEG	input	
18	LADDR10	input		49	PAS_NEG	output3	51
19	LADDR11	output3	32	50	PAS_NEG	input	
20	LADDR11	input		51	*	control	
21	LADDR12	output3	32	52	PCS_NEG	output3	51
22	LADDR12	input		53	PCS_NEG	input	
23	LADDR13	output3	32	54	*	control	
24	LADDR14	output3	32	55	PDAEN_NEG	output3	54
25	LADDR14	input		56	PDAEN_NEG	input	
26	LADDR15	output3	32	57	PFAIL_NEG	input	
27	LADDR15	input		58	PINT_NEG	output2-OD	58
28	LADDR16	output3	32	59	PRST_NEG	output2	
29	LADDR16	input		60	PROCMODE	input	
30	LADDR17	output3	32	61	HAD0	output3	154
31	LADDR17	input		62	HAD0	input	

Table D–3. Pin Descriptions *(2 of 5)*

Cell	Related Pin Name	Cell	Controlling Cell
63	HAD1	output3	154
64	HAD1	input	
65	HAD2	output3	154
66	HAD2	input	
67	HAD3	output3	154
68	HAD3	input	
69	HAD4	output3	154
70	HAD4	input	
71	HAD5	output3	154
72	HAD5	input	
73	HAD6	output3	154
74	HAD6	input	
75	HAD7	output3	154
76	HAD7	input	
77	HCBE0_NEG	output3	137
78	HCBE0_NEG	input	
79	HAD8	output3	154
80	HAD8	input	
81	HAD9	output3	154
82	HAD9	input	
83	HAD10	output3	154
84	HAD10	input	
85	HAD11	output3	154
86	HAD11	input	
87	HAD12	output3	154
88	HAD12	input	
89	HAD13	output3	154
90	HAD13	input	
91	HAD14	output3	154
92	HAD14	input	
93	HAD15	output3	154
94	HAD15	input	
95	HCBE1_NEG	output3	137
96	HCBE1_NEG	input	
97	*	control	
98	HPAR	output3	97
99	HPAR	input	
100	HSERR_NEG	output2-OD	100
101	*	control	
102	HPERR_NEG	output3	101
103	HPERR_NEG	input	
104	HSTOP_NEG	output3	108
105	HSTOP_NEG	input	
106	HDEVSEL_NEG	output3	108
107	HDEVSEL_NEG	input	
108	*	control	
109	HTRDY_NEG	output3	108
110	HTRDY_NEG	input	
111	*	control	
112	HIRDY_NEG	output3	111
113	HIRDY_NEG	input	
114	*	control	
115	HFRAME_NEG	output3	114
116	HFRAME_NEG	input	
117	HCLK	input	
118	HCBE2_NEG	output3	137
119	HCBE2_NEG	input	
120	HAD16	output3	154
121	HAD16	input	
122	HAD17	output3	154
123	HAD17	input	
124	HAD18	output3	154
125	HAD18	input	
126	HAD19	output3	154
127	HAD19	input	
128	HAD20	output3	154
129	HAD20	input	
130	HAD21	output3	154

Table D–3. Pin Descriptions *(3 of 5)*

Cell	Related Pin Name	Cell	Controlling Cell	Cell	Related Pin Name	Cell	Controlling Cell
131	HAD21	input		165	TXD3	output3	170
132	132	HAD22	output3	166	TXD4	output3	170
133	133	HAD22	input	167	TXD5	output3	170
134	134	HAD23	output3	168	TXD6	output3	170
135	135	HAD23	input	169	TXD7	output3	170
136	136	HIDSEL	input	170	*	control	
137	137	*	control	171	TXPAR	output3	170
138	138	HCBE3_NEG	output3	172	*	control	
139	139	HCBE3_NEG	input	173	TXFLAG_NEG	output3	172
140	140	HAD24	output3	174	TXFLAG_NEG	input	
141	141	HAD24	input	175	*	control	
142	142	HAD25	output3	176	TXMARK	output3	175
143	143	HAD25	input	177	TXMARK	input	
144	144	HAD26	output3	178	*	control	
145	145	HAD26	input	179	TXEN_NEG	output3	178
146	146	HAD27	output3	180	TXEN_NEG	input	
147	147	HAD27	input	181	FRCTRL	input	
148	148	HAD28	output3	182	FRCFG1	input	
149	149	HAD28	input	183	FRCFG0	input	
150	150	HAD29	output3	184	*	control	
151	151	HAD29	input	185	RXFLAG_NEG	output3	184
152	152	HAD30	output3	186	RXFLAG_NEG	input	
153	153	HAD30	input	187	*	control	
154	154	*	control	188	RXEN_NEG	output3	187
155	155	HAD31	output3	189	RXEN_NEG	input	
156	156	HAD31	input	190	RXMARK	input	
157	157	*	control	191	RXD0	input	
158	158	HREQ_NEG	output3	192	RXD1	input	
159	159	HGNT_NEG	input	193	RXD2	input	
160	160	HRST_NEG	input	194	RXD3	input	
161	161	HINT_NEG	output2-OD	195	RXD4	input	
162	162	TXD0	output3	196	RXD5	input	
163	163	TXD1	output3	197	RXD6	input	
164	164	TXD2	output3	198	RXD7	input	

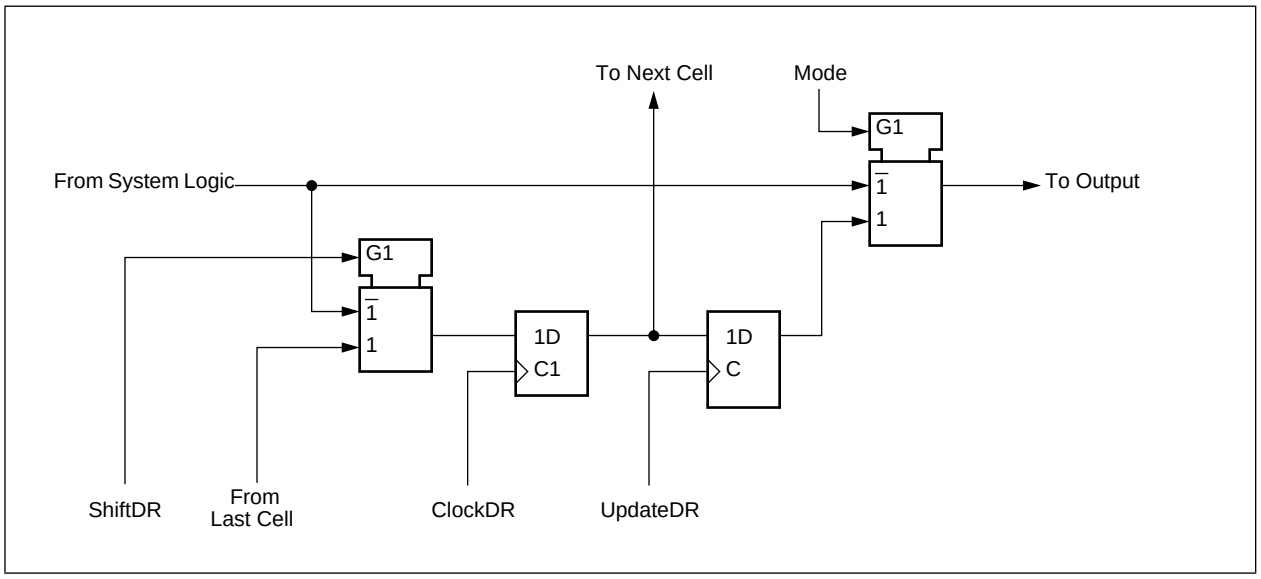
Table D–3. Pin Descriptions *(4 of 5)*

Cell	Related Pin Name	Cell	Controlling Cell	Cell	Related Pin Name	Cell	Controllin g Cell
199	RXPAR	input		233	LDATA16	input	
200	LDATA0	output3	262	234	LDATA17	output3	262
201	LDATA0	input		235	LDATA17	input	
202	LDATA1	output3	262	236	LDATA18	output3	262
203	LDATA1	input		237	LDATA18	input	
204	LDATA2	output3	262	238	LDATA19	output3	262
205	LDATA2	input		239	LDATA19	input	
206	LDATA3	output3	262	240	LDATA20	output3	262
207	LDATA3	input		241	LDATA20	input	
208	LDATA4	output3	262	242	LDATA21	output3	262
209	LDATA4	input		243	LDATA21	input	
210	LDATA5	output3	262	244	LDATA22	output3	262
211	LDATA5	input		245	LDATA22	input	
212	LDATA6	output3	262	246	LDATA23	output3	262
213	LDATA6	input		247	LDATA23	input	
214	RXPAR	input		248	LDATA24	output3	262
215	LDATA0	output3	262	249	LDATA24	input	
216	LDATA7	output3	262	250	LDATA25	output3	262
217	LDATA7	input		251	LDATA25	input	
218	LDATA8	output3	262	252	LDATA26	output3	262
219	LDATA8	input		253	LDATA26	input	
220	LDATA9	output3	262	254	LDATA27	output3	262
221	LDATA9	input		255	LDATA27	input	
222	LDATA10	output3	262	256	LDATA28	output3	262
223	LDATA10	input		257	LDATA28	input	
224	LDATA11	output3	262	258	LDATA29	output3	262
225	LDATA11	input		259	LDATA29	input	
226	LDATA12	output3	262	260	LDATA30	output3	262
227	LDATA12	input		261	LDATA30	input	
228	LDATA13	output3	262	262	*	control	
229	LDATA13	input		263	LDATA31	output3	262
230	LDATA14	output3	262	264	LDATA31	input	
231	LDATA14	input		265	CLK2X	input	
232	LDATA15	output3	262	266	SYSCLK	output3	283

Table D–3. Pin Descriptions *(5 of 5)*

Cell	Related Pin Name	Cell	Controlling Cell	Cell	Related Pin Name	Cell	Controlling Cell
267	CLKD3	output3	283	276	MWE2_NEG	output3	283
268	MCS0_NEG	output3	283	277	MWE3_NEG	output3	283
269	MCS1_NEG	output3	283	278	RAMMODE	input	
270	MCS2_NEG	output3	283	279	STAT0	output3	283
271	MCS3_NEG	output3	283	280	STAT1	output3	283
272	MWR_NEG	output3	283	281	LADDR0	output3	283
273	MOE_NEG	output3	283	282	LADDR1	output3	283
274	MWE0_NEG	output3	283	283	*	control	
275	MWE1_NEG	output3	283				

Figure D–4. Output-Control Cell Diagram



Boundary Scan Register Cells

Table D-3 defines the Boundary Scan Register cells.

Cell 0 is closest to TDO in the chain.

Cell Type definitions:

- Output3 = Output-Both (either the output cell of a bidirectional output or a tri-state output)
- Output2 = Output-Both (bi-state output)
- Output2-OD = Output-Both (Open Drain type Output)
- Input = Input-Observe
- Control = Output-Control

All controlling cells put their respective output cell into the inactive state with a value of 1.

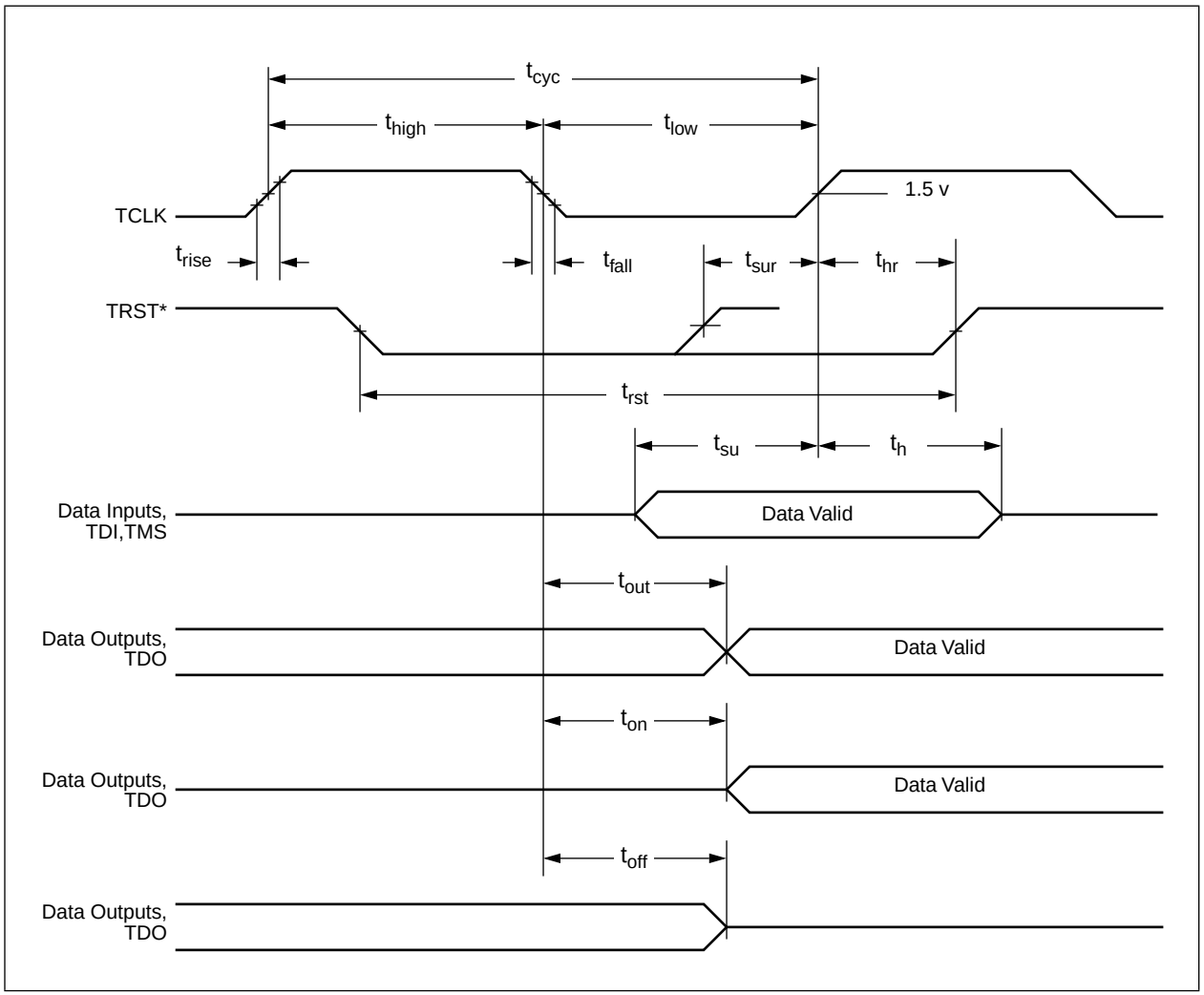
Electrical Characteristics

This section describes the electrical characteristics for boundary scan. Table D-4 provides timing specifications and Figure D-5 shows a timing diagram.

Table D–4. Timing Specifications

Symbol	Description	Min	Max	Unit
	TCLK Frequency	0	5	MHz
t_{cyc}	TCLK Cycle	200	–	ns
t_{high}	TCLK High Pulse Width	90	–	ns
t_{low}	TCLK Low Pulse Width	90	–	ns
t_{rise}	TCLK Rise	0	10	ns
t_{fall}	TCLK Fall	0	10	ns
t_{sur}	TRST* Setup to TCLK Rising Edge(1)	1	–	ns
t_{hr}	TRST* Hold after TCLK Rising Edge(1)	1	–	ns
t_{rst}	TRST* Active	1 t_{cyc}	–	
t_{su}	TDI, TMS, and Data Inputs Setup	20	–	ns
t_{h}	TDI, TMS, and Data Inputs Hold	80	–	ns
t_{out}	TDO and Data Outputs Valid	10	80	ns
t_{on}	TDO and Data Outputs Float to Valid	10	80	ns
t_{off}	TDO and Data Outputs Valid to Float	10	80	ns
Notes: (1). TRST* going “inactive” timing only required if TMS = 0 at the rising edge of TCLK.				

Table D–5. Timing Diagram



Boundary Scan Description Language (BSDL) File

To obtain an electronic copy of this file, contact Rockwell Applications Engineer at 1-800-854-8099.

```
-- BSDL File created/edited by AT&T BSD Editor
--
--BSDE:Revision: 2.0
--BSDE:Description: ATM Segmentation and Reassembly Controller - SAR

entity BT8230 is

generic (PHYSICAL_PIN_MAP : string := "PQFP_208" );

port (
        CLK2X: in bit;
        CLKD3: out bit;
        FRCFG0: in bit;
        FRCFG1: in bit;
        FRCTRL: in bit;
        HAD0: inout bit;
        HAD1: inout bit;
        HAD10: inout bit;
        HAD11: inout bit;
        HAD12: inout bit;
        HAD13: inout bit;
        HAD14: inout bit;
        HAD15: inout bit;
        HAD16: inout bit;
        HAD17: inout bit;
        HAD18: inout bit;
        HAD19: inout bit;
        HAD2: inout bit;
        HAD20: inout bit;
        HAD21: inout bit;
        HAD22: inout bit;
        HAD23: inout bit;
        HAD24: inout bit;
        HAD25: inout bit;
        HAD26: inout bit;
        HAD27: inout bit;
        HAD28: inout bit;
        HAD29: inout bit;
        HAD3: inout bit;
        HAD30: inout bit;
        HAD31: inout bit;
        HAD4: inout bit;
        HAD5: inout bit;
```

HAD6: inout bit;
HAD7: inout bit;
HAD8: inout bit;
HAD9: inout bit;
HCBE0_NEG: inout bit;
HCBE1_NEG: inout bit;
HCBE2_NEG: inout bit;
HCBE3_NEG: inout bit;
HCLK: in bit;
HDEVSEL_NEG: inout bit;
HFRAME_NEG: inout bit;
HGNT_NEG: in bit;
HIDSEL: in bit;
HINT_NEG: out bit;
HIRDY_NEG: inout bit;
HPAR: inout bit;
HPERR_NEG: inout bit;
HREQ_NEG: out bit;
HRST_NEG: in bit;
HSERR_NEG: out bit;
HSTOP_NEG: inout bit;
HTRDY_NEG: inout bit;
LADDR0: out bit;
LADDR1: out bit;
LADDR10: inout bit;
LADDR11: inout bit;
LADDR12: inout bit;
LADDR13: inout bit;
LADDR14: inout bit;
LADDR15: inout bit;
LADDR16: inout bit;
LADDR17: inout bit;
LADDR18: inout bit;
LADDR2: inout bit;
LADDR3: inout bit;
LADDR4: inout bit;
LADDR5: inout bit;
LADDR6: inout bit;
LADDR7: inout bit;
LADDR8: inout bit;
LADDR9: inout bit;
LDATA0: inout bit;
LDATA1: inout bit;
LDATA10: inout bit;
LDATA11: inout bit;
LDATA12: inout bit;
LDATA13: inout bit;
LDATA14: inout bit;
LDATA15: inout bit;
LDATA16: inout bit;
LDATA17: inout bit;

LDATA18: inout bit;
LDATA19: inout bit;
LDATA2: inout bit;
LDATA20: inout bit;
LDATA21: inout bit;
LDATA22: inout bit;
LDATA23: inout bit;
LDATA24: inout bit;
LDATA25: inout bit;
LDATA26: inout bit;
LDATA27: inout bit;
LDATA28: inout bit;
LDATA29: inout bit;
LDATA3: inout bit;
LDATA30: inout bit;
LDATA31: inout bit;
LDATA4: inout bit;
LDATA5: inout bit;
LDATA6: inout bit;
LDATA7: inout bit;
LDATA8: inout bit;
LDATA9: inout bit;
MCS0_NEG: out bit;
MCS1_NEG: out bit;
MCS2_NEG: out bit;
MCS3_NEG: out bit;
MOE_NEG: out bit;
MWE0_NEG: out bit;
MWE1_NEG: out bit;
MWE2_NEG: out bit;
MWE3_NEG: out bit;
MWR_NEG: out bit;
PADDR0: in bit;
PADDR1: in bit;
PAS_NEG: inout bit;
PBE0_NEG: in bit;
PBE1_NEG: in bit;
PBE2_NEG: in bit;
PBE3_NEG: in bit;
PBLAST_NEG: inout bit;
PBSEL0: in bit;
PBSEL1: in bit;
PCS_NEG: inout bit;
PDAEN_NEG: inout bit;
PFAIL_NEG: in bit;
PINT_NEG: out bit;
PRDY_NEG: out bit;
PROCMODE: in bit;
PRST_NEG: out bit;
PWAIT_NEG: in bit;
PWNR: inout bit;

```
        RAMMODE: in bit;
        RXD0: in bit;
        RXD1: in bit;
        RXD2: in bit;
        RXD3: in bit;
        RXD4: in bit;
        RXD5: in bit;
        RXD6: in bit;
        RXD7: in bit;
        RXEN_NEG: inout bit;
        RXFLAG_NEG: inout bit;
        RXMARK: in bit;
        RXPAR: in bit;
        STAT0: out bit;
        STAT1: out bit;
        SYSCLK: out bit;
        TCLK: in bit;
        TDI: in bit;
        TDO: out bit;
        TMS: in bit;
        TRST_NEG: in bit;
        TXD0: out bit;
        TXD1: out bit;
        TXD2: out bit;
        TXD3: out bit;
        TXD4: out bit;
        TXD5: out bit;
        TXD6: out bit;
        TXD7: out bit;
        TXEN_NEG: inout bit;
        TXFLAG_NEG: inout bit;
        TXMARK: inout bit;
        TXPAR: out bit;
        VDD: linkage bit;
        VSS: linkage bit
);

use STD_1149_1_1990.all;

attribute PIN_MAP of BT8230 : entity is PHYSICAL_PIN_MAP;

constant PQFP_208: PIN_MAP_STRING:=
        "CLK2X:23," &
        "CLKD3:19," &
        "FRCFG0:77," &
        "FRCFG1:78," &
        "FRCTRL:79," &
        "HAD0:162," &
        "HAD1:161," &
        "HAD10:149," &
        "HAD11:148," &
        "HAD12:147," &
```

"HAD13:146," &
"HAD14:145," &
"HAD15:144," &
"HAD16:128," &
"HAD17:127," &
"HAD18:126," &
"HAD19:125," &
"HAD2:160," &
"HAD20:124," &
"HAD21:123," &
"HAD22:122," &
"HAD23:121," &
"HAD24:116," &
"HAD25:115," &
"HAD26:114," &
"HAD27:113," &
"HAD28:112," &
"HAD29:111," &
"HAD3:159," &
"HAD30:110," &
"HAD31:109," &
"HAD4:158," &
"HAD5:157," &
"HAD6:156," &
"HAD7:155," &
"HAD8:151," &
"HAD9:150," &
"HCBE0_NEG:154," &
"HCBE1_NEG:143," &
"HCBE2_NEG:129," &
"HCBE3_NEG:119," &
"HCLK:132," &
"HDEVSEL_NEG:136," &
"HFRAME_NEG:133," &
"HGNT_NEG:107," &
"HIDSEL:120," &
"HINT_NEG:105," &
"HIRDY_NEG:134," &
"HPAR:142," &
"HPERR_NEG:138," &
"HREQ_NEG:108," &
"HRST_NEG:106," &
"HSERR_NEG:139," &
"HSTOP_NEG:137," &
"HTRDY_NEG:135," &
"LADDR0:2," &
"LADDR1:1," &
"LADDR10:198," &
"LADDR11:197," &
"LADDR12:196," &
"LADDR13:195," &

"LADDR14:192," &
"LADDR15:191," &
"LADDR16:190," &
"LADDR17:189," &
"LADDR18:188," &
"LADDR2:206," &
"LADDR3:205," &
"LADDR4:204," &
"LADDR5:203," &
"LADDR6:202," &
"LADDR7:201," &
"LADDR8:200," &
"LADDR9:199," &
"LDATA0:61," &
"LDATA1:60," &
"LDATA10:50," &
"LDATA11:49," &
"LDATA12:48," &
"LDATA13:46," &
"LDATA14:45," &
"LDATA15:44," &
"LDATA16:43," &
"LDATA17:42," &
"LDATA18:40," &
"LDATA19:39," &
"LDATA2:59," &
"LDATA20:38," &
"LDATA21:37," &
"LDATA22:36," &
"LDATA23:35," &
"LDATA24:32," &
"LDATA25:31," &
"LDATA26:30," &
"LDATA27:29," &
"LDATA28:28," &
"LDATA29:27," &
"LDATA3:58," &
"LDATA30:26," &
"LDATA31:25," &
"LDATA4:57," &
"LDATA5:56," &
"LDATA6:55," &
"LDATA7:54," &
"LDATA8:52," &
"LDATA9:51," &
"MCS0_NEG:17," &
"MCS1_NEG:16," &
"MCS2_NEG:15," &
"MCS3_NEG:14," &
"MOE_NEG:12," &
"MWE0_NEG:11," &

"MWE1_NEG:10," &
"MWE2_NEG:7," &
"MWE3_NEG:6," &
"MWR_NEG:13," &
"PADDR0:187," &
"PADDR1:186," &
"PAS_NEG:173," &
"PBE0_NEG:183," &
"PBE1_NEG:180," &
"PBE2_NEG:179," &
"PBE3_NEG:178," &
"PBLAST_NEG:174," &
"PBSEL0:185," &
"PBSEL1:184," &
"PCS_NEG:172," &
"PDAEN_NEG:171," &
"PFAIL_NEG:168," &
"PINT_NEG:167," &
"PRDY_NEG:176," &
"PROCMODE:165," &
"PRST_NEG:166," &
"PWAIT_NEG:175," &
"PWNR:177," &
"RAMMODE:5," &
"RXD0:73," &
"RXD1:70," &
"RXD2:69," &
"RXD3:68," &
"RXD4:67," &
"RXD5:66," &
"RXD6:65," &
"RXD7:64," &
"RXEN_NEG:75," &
"RXFLAG_NEG:76," &
"RXMARK:74," &
"RXPAR:63," &
"STAT0:4," &
"STAT1:3," &
"SYSCLK:21," &
"TCLK:97," &
"TDI:102," &
"TDO:101," &
"TMS:100," &
"TRST_NEG:99," &
"TXD0:95," &
"TXD1:94," &
"TXD2:93," &
"TXD3:92," &
"TXD4:91," &
"TXD5:90," &
"TXD6:87," &

```
        "TXD7:86," &
        "TXEN_NEG:82," &
        "TXFLAG_NEG:84," &
        "TXMARK:83," &
        "TXPAR:85," &
        "VDD:104," &
        "VSS:103";

    attribute TAP_SCAN_IN    of TDI : signal is true;
    attribute TAP_SCAN_OUT   of TDO : signal is true;
    attribute TAP_SCAN_MODE  of TMS : signal is true;
    attribute TAP_SCAN_CLOCK of TCLK : signal is (1.00e+06, BOTH);
    attribute TAP_SCAN_RESET  of TRST_NEG : signal is true;

    attribute INSTRUCTION_LENGTH of BT8230 : entity is 3;

    attribute INSTRUCTION_OPCODE of BT8230 : entity is
        "BYPASS ( 111)," &
        "EXTEST ( 000)," &
        "RSTHIGH ( 001)," &
        "SAMPLE ( 010)," &
        "TMRFIFO ( 100)," &
        "TMWFIFO ( 011)," &
        "TRSMFIFO ( 110)," &
        "TSEGFIFO ( 101)" ;

    attribute INSTRUCTION_CAPTURE of BT8230 : entity is "001";

    attribute INSTRUCTION_PRIVATE of BT8230 : entity is
        " RSTHIGH, TMRFIFO, TMWFIFO, TRSMFIFO, TSEGFIFO";

    attribute REGISTER_ACCESS of BT8230 : entity is
        "BYPASS ( BYPASS)," &
          "BOUNDARY ( EXTEST, RSTHIGH, SAMPLE, TMRFIFO, TMW-
FIFO, TRSMFIFO," &
            " TSEGFIFO)";

    attribute BOUNDARY_CELLS of BT8230 : entity is
        " BC_1, BC_4";

    attribute BOUNDARY_LENGTH of BT8230 : entity is 284;

    attribute BOUNDARY_REGISTER of BT8230 : entity is
        "   0 (BC_1, LADDR2, output3, X, 32, 1, Z)," &
        "   1 (BC_4, LADDR2, input, X)," &
        "   2 (BC_1, LADDR3, output3, X, 32, 1, Z)," &
        "   3 (BC_4, LADDR3, input, X)," &
        "   4 (BC_1, LADDR4, output3, X, 32, 1, Z)," &
        "   5 (BC_4, LADDR4, input, X)," &
        "   6 (BC_1, LADDR5, output3, X, 32, 1, Z)," &
        "   7 (BC_4, LADDR5, input, X)," &
        "   8 (BC_1, LADDR6, output3, X, 32, 1, Z)," &
        "   9 (BC_4, LADDR6, input, X)," &
        "  10 (BC_1, LADDR7, output3, X, 32, 1, Z)," &
        "  11 (BC_4, LADDR7, input, X)," &
```

```
" 12 (BC_1, LADDR8, output3, X, 32, 1, Z)," &
" 13 (BC_4, LADDR8, input, X)," &
" 14 (BC_1, LADDR9, output3, X, 32, 1, Z)," &
" 15 (BC_4, LADDR9, input, X)," &
" 16 (BC_1, LADDR10, output3, X, 32, 1, Z)," &
" 17 (BC_4, LADDR10, input, X)," &
" 18 (BC_1, LADDR11, output3, X, 32, 1, Z)," &
" 19 (BC_4, LADDR11, input, X)," &
" 20 (BC_1, LADDR12, output3, X, 32, 1, Z)," &
" 21 (BC_4, LADDR12, input, X)," &
" 22 (BC_1, LADDR13, output3, X, 32, 1, Z)," &
" 23 (BC_4, LADDR13, input, X)," &
" 24 (BC_1, LADDR14, output3, X, 32, 1, Z)," &
" 25 (BC_4, LADDR14, input, X)," &
" 26 (BC_1, LADDR15, output3, X, 32, 1, Z)," &
" 27 (BC_4, LADDR15, input, X)," &
" 28 (BC_1, LADDR16, output3, X, 32, 1, Z)," &
" 29 (BC_4, LADDR16, input, X)," &
" 30 (BC_1, LADDR17, output3, X, 32, 1, Z)," &
" 31 (BC_4, LADDR17, input, X)," &
" 32 (BC_1, *, control, 1)," &
" 33 (BC_1, LADDR18, output3, X, 32, 1, Z)," &
" 34 (BC_4, LADDR18, input, X)," &
" 35 (BC_4, PADDR0, input, X)," &
" 36 (BC_4, PADDR1, input, X)," &
" 37 (BC_4, PBSEL0, input, X)," &
" 38 (BC_4, PBSEL1, input, X)," &
" 39 (BC_4, PBE0_NEG, input, X)," &
" 40 (BC_4, PBE1_NEG, input, X)," &
" 41 (BC_4, PBE2_NEG, input, X)," &
" 42 (BC_4, PBE3_NEG, input, X)," &
" 43 (BC_1, PWNR, output3, X, 51, 1, Z)," &
" 44 (BC_4, PWNR, input, X)," &
" 45 (BC_1, PRDY_NEG, output3, X, 283, 1, Z)," &
" 46 (BC_4, PWAIT_NEG, input, X)," &
" 47 (BC_1, PBLAST_NEG, output3, X, 51, 1, Z)," &
" 48 (BC_4, PBLAST_NEG, input, X)," &
" 49 (BC_1, PAS_NEG, output3, X, 51, 1, Z)," &
" 50 (BC_4, PAS_NEG, input, X)," &
" 51 (BC_1, *, control, 1)," &
" 52 (BC_1, PCS_NEG, output3, X, 51, 1, Z)," &
" 53 (BC_4, PCS_NEG, input, X)," &
" 54 (BC_1, *, control, 1)," &
" 55 (BC_1, PDAEN_NEG, output3, X, 54, 1, Z)," &
" 56 (BC_4, PDAEN_NEG, input, X)," &
" 57 (BC_4, PFAIL_NEG, input, X)," &
" 58 (BC_1, PINT_NEG, output2, 1, 58, 1, Weak1)," &
" 59 (BC_1, PRST_NEG, output3, X, 283, 1, Z)," &
" 60 (BC_4, PROCMODE, input, X)," &
" 61 (BC_1, HAD0, output3, X, 154, 1, Z)," &
" 62 (BC_4, HAD0, input, X)," &
```

```
"  63 (BC_1, HAD1, output3, X, 154, 1, Z)," &
"  64 (BC_4, HAD1, input, X)," &
"  65 (BC_1, HAD2, output3, X, 154, 1, Z)," &
"  66 (BC_4, HAD2, input, X)," &
"  67 (BC_1, HAD3, output3, X, 154, 1, Z)," &
"  68 (BC_4, HAD3, input, X)," &
"  69 (BC_1, HAD4, output3, X, 154, 1, Z)," &
"  70 (BC_4, HAD4, input, X)," &
"  71 (BC_1, HAD5, output3, X, 154, 1, Z)," &
"  72 (BC_4, HAD5, input, X)," &
"  73 (BC_1, HAD6, output3, X, 154, 1, Z)," &
"  74 (BC_4, HAD6, input, X)," &
"  75 (BC_1, HAD7, output3, X, 154, 1, Z)," &
"  76 (BC_4, HAD7, input, X)," &
"  77 (BC_1, HCBE0_NEG, output3, X, 137, 1, Z)," &
"  78 (BC_4, HCBE0_NEG, input, X)," &
"  79 (BC_1, HAD8, output3, X, 154, 1, Z)," &
"  80 (BC_4, HAD8, input, X)," &
"  81 (BC_1, HAD9, output3, X, 154, 1, Z)," &
"  82 (BC_4, HAD9, input, X)," &
"  83 (BC_1, HAD10, output3, X, 154, 1, Z)," &
"  84 (BC_4, HAD10, input, X)," &
"  85 (BC_1, HAD11, output3, X, 154, 1, Z)," &
"  86 (BC_4, HAD11, input, X)," &
"  87 (BC_1, HAD12, output3, X, 154, 1, Z)," &
"  88 (BC_4, HAD12, input, X)," &
"  89 (BC_1, HAD13, output3, X, 154, 1, Z)," &
"  90 (BC_4, HAD13, input, X)," &
"  91 (BC_1, HAD14, output3, X, 154, 1, Z)," &
"  92 (BC_4, HAD14, input, X)," &
"  93 (BC_1, HAD15, output3, X, 154, 1, Z)," &
"  94 (BC_4, HAD15, input, X)," &
"  95 (BC_1, HCBE1_NEG, output3, X, 137, 1, Z)," &
"  96 (BC_4, HCBE1_NEG, input, X)," &
"  97 (BC_1, *, control, 1)," &
"  98 (BC_1, HPAR, output3, X, 97, 1, Z)," &
"  99 (BC_4, HPAR, input, X)," &
" 100 (BC_1, HSERR_NEG, output2, 1, 100, 1, Weak1)," &
" 101 (BC_1, *, control, 1)," &
" 102 (BC_1, HPERR_NEG, output3, X, 101, 1, Z)," &
" 103 (BC_4, HPERR_NEG, input, X)," &
" 104 (BC_1, HSTOP_NEG, output3, X, 108, 1, Z)," &
" 105 (BC_4, HSTOP_NEG, input, X)," &
" 106 (BC_1, HDEVSEL_NEG, output3, X, 108, 1, Z)," &
" 107 (BC_4, HDEVSEL_NEG, input, X)," &
" 108 (BC_1, *, control, 1)," &
" 109 (BC_1, HTRDY_NEG, output3, X, 108, 1, Z)," &
" 110 (BC_4, HTRDY_NEG, input, X)," &
" 111 (BC_1, *, control, 1)," &
" 112 (BC_1, HIRDY_NEG, output3, X, 111, 1, Z)," &
" 113 (BC_4, HIRDY_NEG, input, X)," &
```

```
" 114 (BC_1, *, control, 1)," &
" 115 (BC_1, HFRAME_NEG, output3, X, 114, 1, Z)," &
" 116 (BC_4, HFRAME_NEG, input, X)," &
" 117 (BC_4, HCLK, input, X)," &
" 118 (BC_1, HCBE2_NEG, output3, X, 137, 1, Z)," &
" 119 (BC_4, HCBE2_NEG, input, X)," &
" 120 (BC_1, HAD16, output3, X, 154, 1, Z)," &
" 121 (BC_4, HAD16, input, X)," &
" 122 (BC_1, HAD17, output3, X, 154, 1, Z)," &
" 123 (BC_4, HAD17, input, X)," &
" 124 (BC_1, HAD18, output3, X, 154, 1, Z)," &
" 125 (BC_4, HAD18, input, X)," &
" 126 (BC_1, HAD19, output3, X, 154, 1, Z)," &
" 127 (BC_4, HAD19, input, X)," &
" 128 (BC_1, HAD20, output3, X, 154, 1, Z)," &
" 129 (BC_4, HAD20, input, X)," &
" 130 (BC_1, HAD21, output3, X, 154, 1, Z)," &
" 131 (BC_4, HAD21, input, X)," &
" 132 (BC_1, HAD22, output3, X, 154, 1, Z)," &
" 133 (BC_4, HAD22, input, X)," &
" 134 (BC_1, HAD23, output3, X, 154, 1, Z)," &
" 135 (BC_4, HAD23, input, X)," &
" 136 (BC_4, HIDSEL, input, X)," &
" 137 (BC_1, *, control, 1)," &
" 138 (BC_1, HCBE3_NEG, output3, X, 137, 1, Z)," &
" 139 (BC_4, HCBE3_NEG, input, X)," &
" 140 (BC_1, HAD24, output3, X, 154, 1, Z)," &
" 141 (BC_4, HAD24, input, X)," &
" 142 (BC_1, HAD25, output3, X, 154, 1, Z)," &
" 143 (BC_4, HAD25, input, X)," &
" 144 (BC_1, HAD26, output3, X, 154, 1, Z)," &
" 145 (BC_4, HAD26, input, X)," &
" 146 (BC_1, HAD27, output3, X, 154, 1, Z)," &
" 147 (BC_4, HAD27, input, X)," &
" 148 (BC_1, HAD28, output3, X, 154, 1, Z)," &
" 149 (BC_4, HAD28, input, X)," &
" 150 (BC_1, HAD29, output3, X, 154, 1, Z)," &
" 151 (BC_4, HAD29, input, X)," &
" 152 (BC_1, HAD30, output3, X, 154, 1, Z)," &
" 153 (BC_4, HAD30, input, X)," &
" 154 (BC_1, *, control, 1)," &
" 155 (BC_1, HAD31, output3, X, 154, 1, Z)," &
" 156 (BC_4, HAD31, input, X)," &
" 157 (BC_1, *, control, 1)," &
" 158 (BC_1, HREQ_NEG, output3, X, 157, 1, Z)," &
" 159 (BC_4, HGNT_NEG, input, X)," &
" 160 (BC_4, HRST_NEG, input, X)," &
" 161 (BC_1, HINT_NEG, output2, 1, 161, 1, Weak1)," &
" 162 (BC_1, TXD0, output3, X, 170, 1, Z)," &
" 163 (BC_1, TXD1, output3, X, 170, 1, Z)," &
" 164 (BC_1, TXD2, output3, X, 170, 1, Z)," &
```

```
" 165 (BC_1, TXD3, output3, X, 170, 1, Z)," &
" 166 (BC_1, TXD4, output3, X, 170, 1, Z)," &
" 167 (BC_1, TXD5, output3, X, 170, 1, Z)," &
" 168 (BC_1, TXD6, output3, X, 170, 1, Z)," &
" 169 (BC_1, TXD7, output3, X, 170, 1, Z)," &
" 170 (BC_1, *, control, 1)," &
" 171 (BC_1, TXPAR, output3, X, 170, 1, Z)," &
" 172 (BC_1, *, control, 1)," &
" 173 (BC_1, TXFLAG_NEG, output3, X, 172, 1, Z)," &
" 174 (BC_4, TXFLAG_NEG, input, X)," &
" 175 (BC_1, *, control, 1)," &
" 176 (BC_1, TXMARK, output3, X, 175, 1, Z)," &
" 177 (BC_4, TXMARK, input, X)," &
" 178 (BC_1, *, control, 1)," &
" 179 (BC_1, TXEN_NEG, output3, X, 178, 1, Z)," &
" 180 (BC_4, TXEN_NEG, input, X)," &
" 181 (BC_4, FRCTRL, input, X)," &
" 182 (BC_4, FRCFG1, input, X)," &
" 183 (BC_4, FRCFG0, input, X)," &
" 184 (BC_1, *, control, 1)," &
" 185 (BC_1, RXFLAG_NEG, output3, X, 184, 1, Z)," &
" 186 (BC_4, RXFLAG_NEG, input, X)," &
" 187 (BC_1, *, control, 1)," &
" 188 (BC_1, RXEN_NEG, output3, X, 187, 1, Z)," &
" 189 (BC_4, RXEN_NEG, input, X)," &
" 190 (BC_4, RXMARK, input, X)," &
" 191 (BC_4, RXD0, input, X)," &
" 192 (BC_4, RXD1, input, X)," &
" 193 (BC_4, RXD2, input, X)," &
" 194 (BC_4, RXD3, input, X)," &
" 195 (BC_4, RXD4, input, X)," &
" 196 (BC_4, RXD5, input, X)," &
" 197 (BC_4, RXD6, input, X)," &
" 198 (BC_4, RXD7, input, X)," &
" 199 (BC_4, RXPAR, input, X)," &
" 200 (BC_1, LDATA0, output3, X, 262, 1, Z)," &
" 201 (BC_4, LDATA0, input, X)," &
" 202 (BC_1, LDATA1, output3, X, 262, 1, Z)," &
" 203 (BC_4, LDATA1, input, X)," &
" 204 (BC_1, LDATA2, output3, X, 262, 1, Z)," &
" 205 (BC_4, LDATA2, input, X)," &
" 206 (BC_1, LDATA3, output3, X, 262, 1, Z)," &
" 207 (BC_4, LDATA3, input, X)," &
" 208 (BC_1, LDATA4, output3, X, 262, 1, Z)," &
" 209 (BC_4, LDATA4, input, X)," &
" 210 (BC_1, LDATA5, output3, X, 262, 1, Z)," &
" 211 (BC_4, LDATA5, input, X)," &
" 212 (BC_1, LDATA6, output3, X, 262, 1, Z)," &
" 213 (BC_4, LDATA6, input, X)," &
" 214 (BC_1, LDATA7, output3, X, 262, 1, Z)," &
" 215 (BC_4, LDATA7, input, X)," &
```

```
" 216 (BC_1, LDATA8, output3, X, 262, 1, Z)," &
" 217 (BC_4, LDATA8, input, X)," &
" 218 (BC_1, LDATA9, output3, X, 262, 1, Z)," &
" 219 (BC_4, LDATA9, input, X)," &
" 220 (BC_1, LDATA10, output3, X, 262, 1, Z)," &
" 221 (BC_4, LDATA10, input, X)," &
" 222 (BC_1, LDATA11, output3, X, 262, 1, Z)," &
" 223 (BC_4, LDATA11, input, X)," &
" 224 (BC_1, LDATA12, output3, X, 262, 1, Z)," &
" 225 (BC_4, LDATA12, input, X)," &
" 226 (BC_1, LDATA13, output3, X, 262, 1, Z)," &
" 227 (BC_4, LDATA13, input, X)," &
" 228 (BC_1, LDATA14, output3, X, 262, 1, Z)," &
" 229 (BC_4, LDATA14, input, X)," &
" 230 (BC_1, LDATA15, output3, X, 262, 1, Z)," &
" 231 (BC_4, LDATA15, input, X)," &
" 232 (BC_1, LDATA16, output3, X, 262, 1, Z)," &
" 233 (BC_4, LDATA16, input, X)," &
" 234 (BC_1, LDATA17, output3, X, 262, 1, Z)," &
" 235 (BC_4, LDATA17, input, X)," &
" 236 (BC_1, LDATA18, output3, X, 262, 1, Z)," &
" 237 (BC_4, LDATA18, input, X)," &
" 238 (BC_1, LDATA19, output3, X, 262, 1, Z)," &
" 239 (BC_4, LDATA19, input, X)," &
" 240 (BC_1, LDATA20, output3, X, 262, 1, Z)," &
" 241 (BC_4, LDATA20, input, X)," &
" 242 (BC_1, LDATA21, output3, X, 262, 1, Z)," &
" 243 (BC_4, LDATA21, input, X)," &
" 244 (BC_1, LDATA22, output3, X, 262, 1, Z)," &
" 245 (BC_4, LDATA22, input, X)," &
" 246 (BC_1, LDATA23, output3, X, 262, 1, Z)," &
" 247 (BC_4, LDATA23, input, X)," &
" 248 (BC_1, LDATA24, output3, X, 262, 1, Z)," &
" 249 (BC_4, LDATA24, input, X)," &
" 250 (BC_1, LDATA25, output3, X, 262, 1, Z)," &
" 251 (BC_4, LDATA25, input, X)," &
" 252 (BC_1, LDATA26, output3, X, 262, 1, Z)," &
" 253 (BC_4, LDATA26, input, X)," &
" 254 (BC_1, LDATA27, output3, X, 262, 1, Z)," &
" 255 (BC_4, LDATA27, input, X)," &
" 256 (BC_1, LDATA28, output3, X, 262, 1, Z)," &
" 257 (BC_4, LDATA28, input, X)," &
" 258 (BC_1, LDATA29, output3, X, 262, 1, Z)," &
" 259 (BC_4, LDATA29, input, X)," &
" 260 (BC_1, LDATA30, output3, X, 262, 1, Z)," &
" 261 (BC_4, LDATA30, input, X)," &
" 262 (BC_1, *, control, 1)," &
" 263 (BC_1, LDATA31, output3, X, 262, 1, Z)," &
" 264 (BC_4, LDATA31, input, X)," &
" 265 (BC_4, CLK2X, input, X)," &
" 266 (BC_1, SYSCLK, output3, X, 283, 1, Z)," &
```

```
" 267 (BC_1, CLKD3, output3, X, 283, 1, Z)," &
" 268 (BC_1, MCS0_NEG, output3, X, 283, 1, Z)," &
" 269 (BC_1, MCS1_NEG, output3, X, 283, 1, Z)," &
" 270 (BC_1, MCS2_NEG, output3, X, 283, 1, Z)," &
" 271 (BC_1, MCS3_NEG, output3, X, 283, 1, Z)," &
" 272 (BC_1, MWR_NEG, output3, X, 283, 1, Z)," &
" 273 (BC_1, MOE_NEG, output3, X, 283, 1, Z)," &
" 274 (BC_1, MWE0_NEG, output3, X, 283, 1, Z)," &
" 275 (BC_1, MWE1_NEG, output3, X, 283, 1, Z)," &
" 276 (BC_1, MWE2_NEG, output3, X, 283, 1, Z)," &
" 277 (BC_1, MWE3_NEG, output3, X, 283, 1, Z)," &
" 278 (BC_4, RAMMODE, input, X)," &
" 279 (BC_1, STAT0, output3, X, 283, 1, Z)," &
" 280 (BC_1, STAT1, output3, X, 283, 1, Z)," &
" 281 (BC_1, LADDR0, output3, X, 283, 1, Z)," &
" 282 (BC_1, LADDR1, output3, X, 283, 1, Z)," &
" 283 (BC_1, *, control, 1)";

end BT8230;
```

Revision History

Revision	Date	Description
D	1995	Describes Bt8230 Revision A.
E	7/97	Includes new functionality up to Bt8230 Revision C. Clarifies operation, updates timing, replaces ABR with UBR. Adds information on hashing, PCI performance, memory allocation, FIFOs, monitoring internal status, and firewall condition. Adds Architecture Overview chapter.

Please mail or fax this form to:
Rockwell Semiconductor Systems
Manager, Technical Publications Dept.
(619) 597–4338

READER RESPONSE PAGE

We welcome your evaluation of this publication. Your comments and suggestions will help us make improvements to all current and future documents. Please rate the following characteristics of the publication you received:

Poor Excellent

1. Overall usefulness of the publication.
❍ ❍ ❍ ❍ ❍ ❍
2. Organization of the publication.
❍ ❍ ❍ ❍ ❍ ❍
3. Completeness and thoroughness of the material.
❍ ❍ ❍ ❍ ❍ ❍
4. Clarity and accuracy of the material.
❍ ❍ ❍ ❍ ❍ ❍
5. Usefulness of the diagrams and illustrations.
❍ ❍ ❍ ❍ ❍ ❍
6. Quantity of diagrams and Illustrations.
❍ ❍ ❍ ❍ ❍ ❍
7. Ease of finding specific information.
❍ ❍ ❍ ❍ ❍ ❍
8. Publication page layout and format.
❍ ❍ ❍ ❍ ❍ ❍

9. I received the publication(s): (Please mark one).
❑ Within one week after ordering it
❑ Within two weeks after ordering it
❑ Within three or more weeks after ordering it

10. The publication delivery time was: (Please mark one).
❑ Earlier than I needed for my needs
❑ Just in time for my needs
❑ Too late for my needs

11. In what format(s) would you prefer to receive future documents? (Check all that apply.)
❑ Printed books
❑ Internet/PDF delivery
❑ CD-ROM
❑ Other: _____

12. How would you rank the importance of the following publication features? (1=Very important)
____ Publication design and layout
____ Accuracy of content
____ Detailed diagrams and illustrations
____ Receiving the publication in a timely manner

13. We would like to hear your thoughts on how we could improve this publication. Please write your comments or suggestions below:

_____

_____

_____

Thank you for taking the time to evaluate this publication.

Document title:_____

Revision No.: _____

Your Name: _____

Title: _____

Company: _____

E-mail: _____

Phone: _____

Because Communication Matters™

URL Address
www.nb.rockwell.com

E-mail Address
literature@nb.rockwell.com

For more information:
Call 1-800-854-8099

International information:
Call 1-714-221-6966

Headquarters
Rockwell Semiconductor Systems Inc.
4311 Jamboree Road, P.O. Box C
Newport Beach, CA 92658-8902
Phone: (714) 221-4600
Fax: (714) 221-6375

European Headquarters
Rockwell Semiconductor Systems S.A.R.L.
Les Taissounieres B1
1680 Route des Dolines BP 283
06905 Sophia Antipolis Cedex
France
Phone: 00.33.4.93.00.33.35
Fax: 00.33.4.93.00.33.03

Asia Pacific Headquarters
1, Kallang Sector, #07-04/06
Kolam Ayer Industrial Park
Singapore, 1334
Phone: 011-65-841-3801
Fax: 011-65-841-3802

US Southwest
Phone: (714) 222-9119
Fax: (714) 222-0620

US Los Angeles
Phone: (805) 376-0559
Fax: (805) 376-8180

US South Central
Phone: (972) 479-9310
Fax: (972) 479-9317

US Southeast
Phone: (770) 393-1630
Fax: (770) 395-1419

US Florida/South America
Phone: (813) 799-8406
Fax: (813) 799-8306

US Northwest
Phone: (408) 249-9696
Fax: (408) 249-6518

US North Central
Phone: (630) 773-3454
Fax: (630) 773-3907

US Northeast
Phone: (508) 692-7660
Fax: (508) 692-8185

Europe Mediterranean
Phone: (39-2) 93179911
Fax: (39-2) 93179913

Europe North
Phone: (44-1) 344 486444
Fax: (44-1) 344 486555

Europe South
Phone: (33-1) 49 06 39 80
Fax: (33-1) 49 06 39 90

Europe Central
Phone: (49-89) 829-1320
Fax: (49-89) 834-2734

Australia
Phone: (61-2) 9805 5555
Direct Fax: (61-2) 9888 9372

Hong Kong
Phone: (852) 2 827-0181
Fax: (852) 2 827-6488

Japan
Phone: (81-3) 5371 1551
Fax: (81-3) 5371 1501

Korea
Phone: (82-2) 565-2880
Fax: (82-2) 565-1440

Singapore
Phone: (65) 733-2511
Fax: (65) 733-0635

Taiwan
Phone: (886-2) 720-0282
Fax: (886-2) 757-6760

L8230_E
Printed in USA

Rockwell Semiconductor Systems