TOSHIBA

622Mbps ATM Segmentation and Reassembly Chipset

1998

TC35860F
TC35861F

REVISION 1.0B

DATA BOOK

 Published in January, 1997

This page left blank intentionally

This specification describes an implementation of a high-speed (622Mbps) Segmentation and Reassembly (SAR) chipset for Asynchronous Transfer Mode (ATM) applications from Toshiba America Electronic Components, Inc (TAEC). In this chapter some document-wide conventions are established, the relationship of the SAR chipset to the ATM standards is described, a high level system overview is provided and the SAR chipset features are outlined.

Addressing Conventions

This document uses the **Big Endian** notation for defining buses, registers and data structures. The term **word** is used to refer to a 4 byte quantity unless otherwise noted. For example the Buffer Memory Address/Data bus is 64 bits wide and can be referred to as a **double word**. Bit 31 of a word is the most significant bit. Bit 7 of a byte is the most significant bit. Data structures composed of multiple bytes or words are numbered from **0** - **N** where **0** is the lowest address and **N** is the highest.

Table 1-1 illustrates the Big Endian format and, for contrast, Table 1-2 illustrates the Little Endian format not used by this document.

Table 1-1 Big Endian Byte Order

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Byte 0 (MSB)								Byte 1								Byte 2								Byte 3 (LSB)							
Word N																															

Table 1-2 Little Endian Byte Order

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Byte 3 (MSB)								Byte 2								Byte 1								Byte 0 (LSB)							
Word N																															

Protocol Data Units

The term “PDU” (Protocol Data Unit) as used in this specification refers to a block of data queued for transmission or presented for reception. From a standards point of view the term PDU typically refers to a data block sent between layer-N on one system to layer-N on another. The term “SDU” (Service Data Unit) refers to data

blocks transferred between layer N and layer N+1 or layer N-1. This specification generally uses the term PDU for both references since the main thrust is to describe a block of data without undue focus on the layering abstraction.

ATM Standard/Specification

ATM is a technique to realize the deployment of Broadband Integrated Services Digital Network (B-ISDN) communication network. ATM is specified by several international and national standards bodies. To provide a scalable solution for B-ISDN, ATM utilizes a basic unit of information in the form of a 53 byte ATM cell. The ATM cell forms the link between the ATM related layers; the ATM Adaptation Layer, ATM Layer and the ATM Physical layer (figure 1-1).

In the Higher Layer Protocols the data packets can take several different forms which are transparent to the ATM Adaptation Layer. Several ATM Adaptation Layers have been defined to further encapsulate user data to provide data integrity, etc. The most commonly referred to adaptation layers are AAL1, AAL3/4 and AAL5. Various groups are working on other schemes. Each AAL type is targeted to a class of higher-layer applications. A full 32-bit CRC is maintained on the data for AAL3/4 and AAL5.

The protocol layers of which the SAR chipset is a silicon implementation are the ATM Adaptation Layer and the ATM Layer itself shown in figure 1-1. The ATM SAR chipset bridges the gap between the physical interface and high level communication applications.

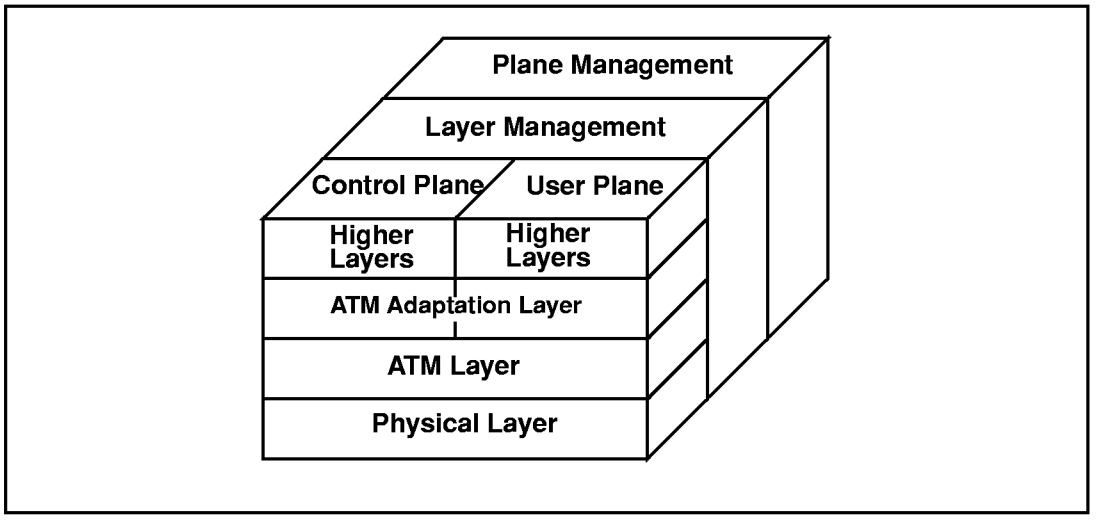


Figure 1-1 B-ISDN Protocol Reference Model

The standard's view of the SAR function in the ATM Network is shown in figure 1-2 within the user's Terminal Equipment (TE). The TE interfaces to the ATM Network at the User Network Interface or UNI (private or public). This is usually illustrated by the Equivalent Terminal Reference Model as shown in I.371 and UNI 3.0. In figure 1-2 we show an extract of that figure concentrating on the SAR functionality.

Note that while the extract shown in figure 1-2 is depicted as unidirectional, the terminal and ATM Layer is in fact bidirectional and incoming data is de-multiplexed, but not shaped, into different traffic destinations.

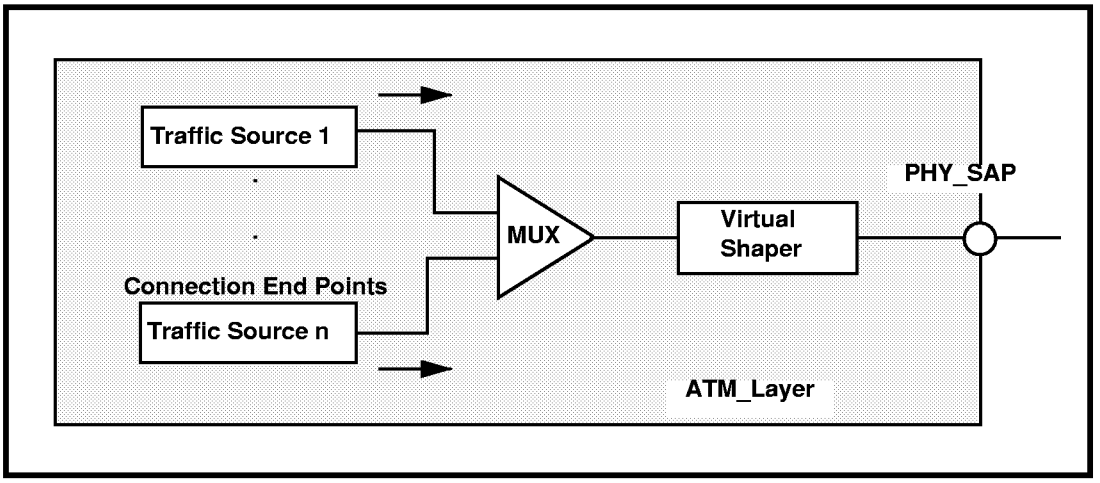


Figure 1-2 SAR Location in the Equivalent Terminal

SAR System Overview

The goal of the SAR chips are to provide an implementation which will handle back to back ATM cell flows at a link rate of 622.080Mbps (i.e., OC-12). The SAR chips are designed to support systems which require high speed, high capacity communication links such as video/data servers, LAN-ATM routers and state of the art workstations.

The Toshiba SAR implementation consist of two devices, a Segmentation device and a Reassembly device. They are referred to herein as the S-chip and R-chip respectively. The term SAR refers to the functionality of the chipset as a whole.

A simplified system block diagram is shown in figure 1-3.

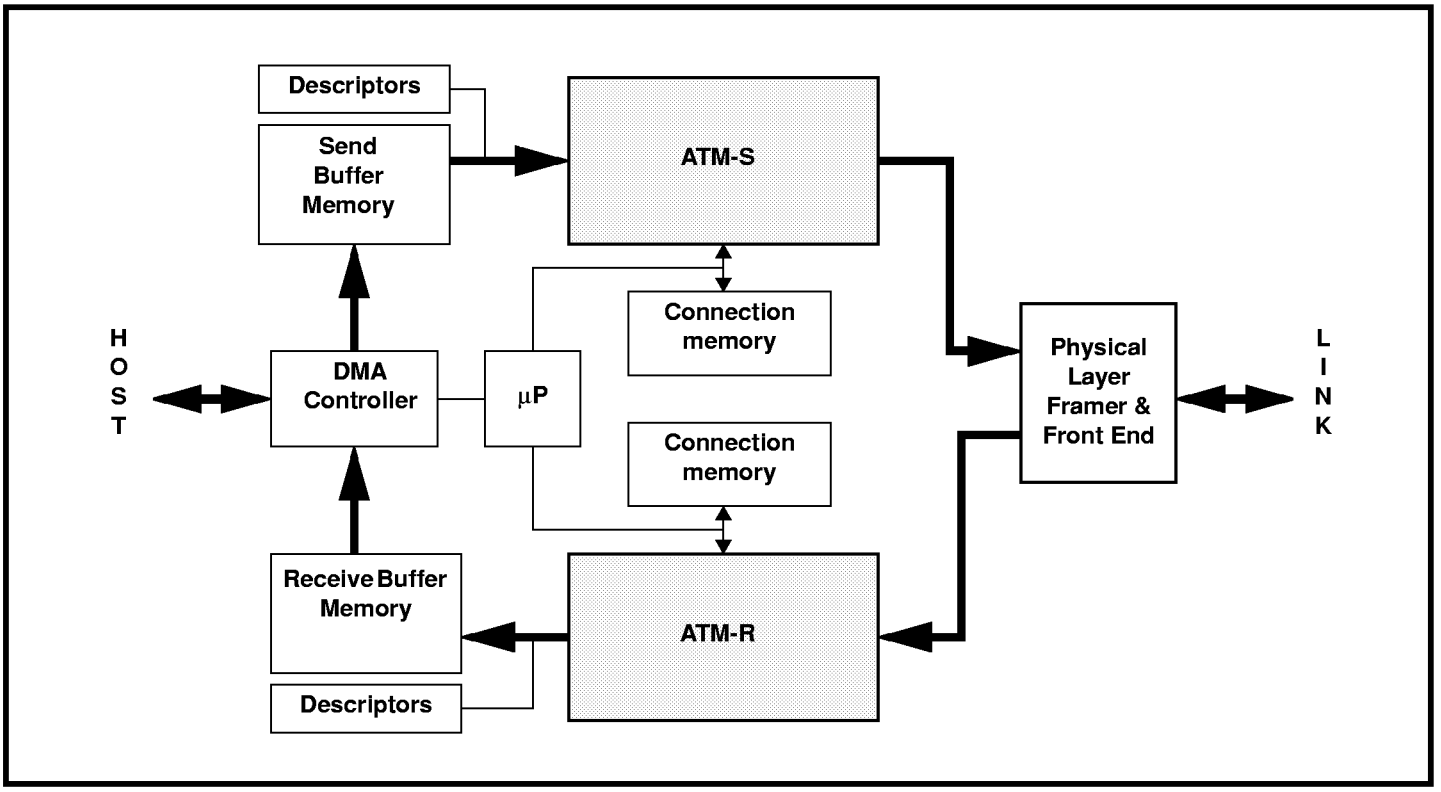


Figure 1-3 Simple System Block Diagram for OC-12 Data Rates

To maintain full data bandwidth at 622Mbps each device uses 3 independent memories: Buffer Memory, Descriptor Memory and Connection Memory. Buffer Memory and Descriptor Memory share a multiplexed address/data bus and can, in some cases, be implemented as a single physical memory if the memory allows pipelined address/data cycles with no start-up latency. The Buffer Memory provides local storage for user data. The Descriptor Memory holds descriptors which create linked lists for the buffers in the Buffer Memory. The Connection memory holds information for the reassembly and segmentation processes on each established Virtual Connection. For 622Mbps operation the data path to Buffer Memory is 64 bits wide.

A system block diagram for 155Mbps (OC-3) is shown in figure 1-4. The Buffer and Descriptor Memories are shown as a single physical memory and may be possible to implement with DRAM. Furthermore, the data path to the Buffer Memory is reduced to 32 bits for 155Mbps operation.

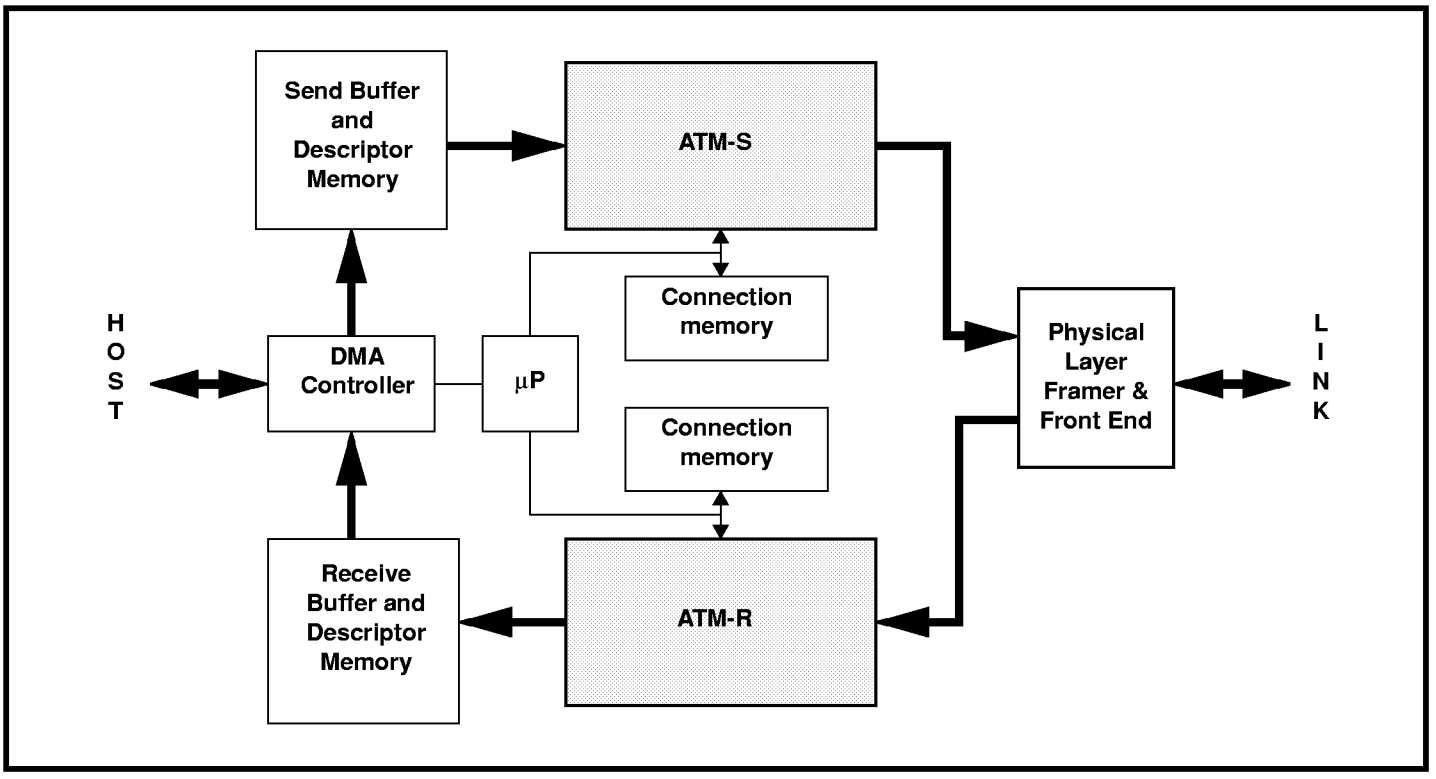


Figure 1-4 Simple System Block Diagram for OC-3 Data Rates

The microprocessor shown initializes the information in the connection memory, exchanges pointers with the SAR (thru the register interface) to transmit and receive PDUs and may also handle higher-layer protocol functions such as call (connection) set-up and call tear-down.

The interface to the ATM Physical Layer matches the industry agreement known as the "UTOPIA" interface. The SAR is designed to be physical media independent. The Physical Layer block shown provides the framing of ATM Cells into the proper transmission media, clock recovery and cell extraction.

Features

Some of the main features of the SAR are:

- Supports full duplex operation of ATM link rates up to 622.080Mbps. This implementation is designed to handle small Protocol Data Units (1-2 cells in size) at the full link rate.
- Supports simultaneous Segmentation and Reassembly of 32,768 Virtual Connections.
- Supports ATM Adaptation layers 3/4 and 5 plus cell modes with and without a canned header. Provides limited AAL 1 support.
- All necessary CRC and Sequence Number, generation and checking, is performed on-chip.

- Error indications for CRC errors, sequence number errors and lost cells.
- Four receive queues, three intended for user data and fourth for management, OAM cells and congestion notification. VCs may also be assigned independently to any of the 3 data queues.
- VCs are traffic-shaped by any one of 16 independent rate generators. Three user-programmable levels of priority are employed in the shaping mechanism.
- The traffic shaping mechanism is programmable and has a high accuracy. The Shaper provides peak rate pacing to handle rate generator contention.
- Supports the Universal Test & Operations PHY Interface for ATM (UTOPIA) as specified by the ATM Forum. The 16-bit interface is used for 622Mbps operation; the 8-bit interface is used for lower speed operation.
- The memory interface for 622Mbps operation is 64 bits wide, for 155Mbps and lower speed operation the memory interface is 32 bits wide.
- Byte Parity on all interfaces.

Reference Documents

The following documents are used for design direction and reference throughout this document:

ATM User-Network Interface Specification Version 3.0, ATM Forum November 1993.

ATM Forum Contribution 93-940, *An ATM PHY Data Path Interface*, December 1993. Presentation of the "UTOPIA Interface".

ITU-T Document TD-XVIII/10, *AAL Type 5, Draft Recommendation text for section 6 of I.363*, 29 January 1993, Geneva.

Functional Block Descriptions

The ATM SAR segments user data packets into ATM cells for transmission and on reception reassembles ATM cell flows into user data packets. In each case the segmentation and reassembly actions are accomplished according to the type of ATM Adaptation Layer employed for each connection.

The TAEC SAR implementation is a two chip solution divided by segmentation and reassembly. A high-level functional description of each chip is provided in the following sections.

Segmentation Device Functional Blocks

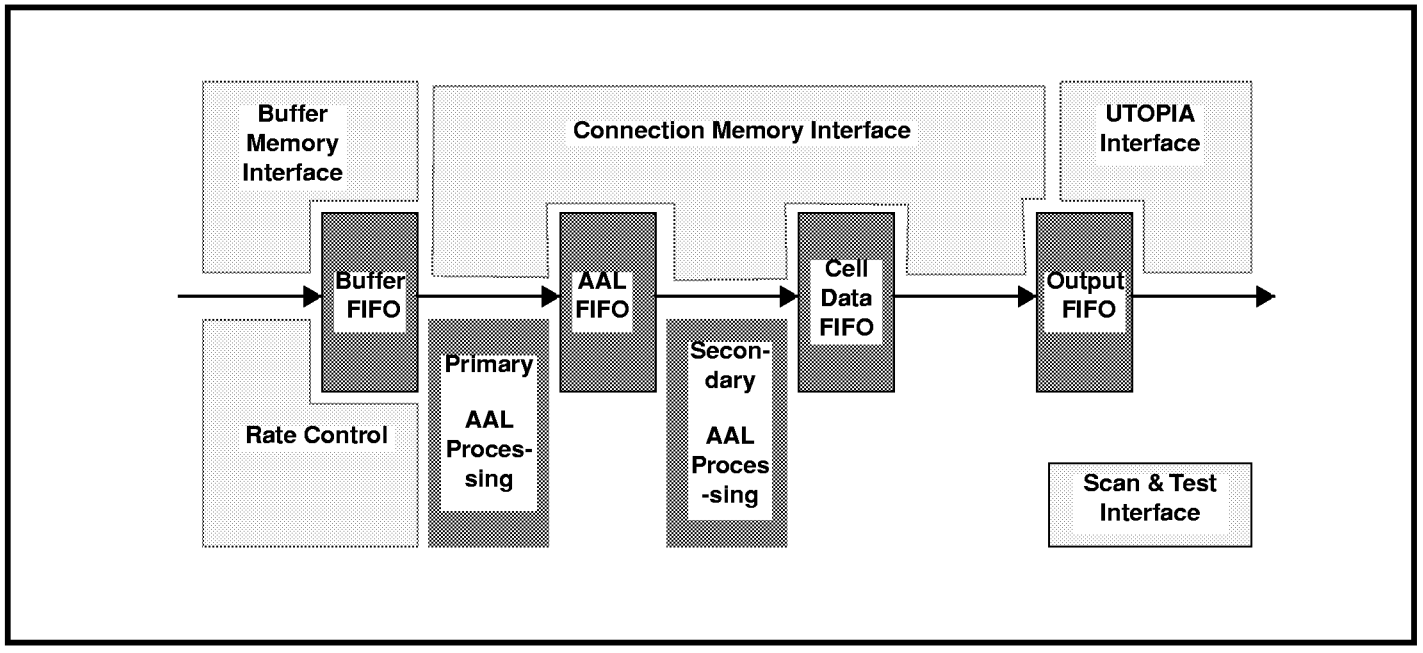


Figure 2-1 Segmentation Block Diagram

A block diagram of the Segmentation device is shown in Figure 2-1, this section describes the functionality of the blocks.

Buffer Memory Interface

The Buffer Memory Interface transfers user data from the external Buffer Memory to the Segmentation device. It includes a state machine that generates a request for the use of Buffer Memory and waits for a grant a signal from an external arbiter. Is also is used to read and write Descriptor Memory. Internally this function reads

control information and writes user data to asynchronous FIFOs that are accessed from a different clock domain (MClk) by other functions in the device. The Buffer Memory function is driven by a separate clock (BmClk) so that the shared memory subsystem can run at a slower rate than most of the internal Segmentation functions.

Connection Memory Interface

The Connections Memory Interface allows the Segmentation device to access a local private memory for data structures which describe all active Virtual Circuits. The device accesses this memory during each cell time using prearranged timeslots. This function also manages the sharing of the Connection Memory data bus with the register interface since register data is transferred on the same pins.

Physical Layer Interface

The Physical Layer Interface provides access to the Physical Layer device assumed to be an OC-12c framer. The interface used is based on the ATM Forum UTOPIA interface. For 622Mbps operation a 16-bit mode is provided. At lower data rates an 8-bit mode is used. This interface is cell-based, i.e., a control signal is checked between cells to determine if another cell can be transmitted across the interface.

Scan and Test Interface

Scan interface is used during manufacturing to serially shift in and out scan test patterns that perform part of the overall manufacturing test. Over 99% of the flip-flops used for normal operation are tested by Scan.

The JTAG interface conforms to the IEEE 1149.1 standard.

Data FIFOs

The cell data path through the Segmentation device consists of a number data FIFOs as described in this subsection. These FIFOs are of varying sizes as required by the state machines which operate on or with the associated data. The connecting data path itself is also of varying width as required at each block.

Buffer FIFO

The Buffer FIFO holds the payload data read from Buffer Memory. Payload data that is split across Buffer Memory blocks is assembled into a single payload in this FIFO. The Buffer FIFO provides rate buffering between the Buffer Memory clock (BmClk) rate and the internal cell processing clock rate (MClk). The FIFO has a 64-bit wide Buffer Memory interface and a 32-bit wide interface to the Primary AAL processing function.

AAL FIFO

The AAL FIFO holds fully assembled cell payloads (48 bytes) after AAL processing. Each payload contains user data or control information or both (depending on the AAL type and state).

Cell Data FIFO

ATM cells are assembled into the Cell Data FIFO. Complete cells are then transferred to the UTOPIA Output FIFO.

Output (UTOPIA) FIFO

The Output FIFO holds completely assembled ATM cells. The Output FIFO also provides rate buffering between the internal cell processing rate and the actual rate at which the UTOPIA Interface transfers cells.

Primary AAL Processing

The Primary AAL Processing block performs the first stage of the pipelined AAL processing which includes the scanning of the incoming data for AAL CRC generation. The CRC result is passed to the Secondary AAL Processing stage.

Secondary AAL Processing

The secondary AAL processing block performs the actual cell assembly, i.e., adds on the ATM cell header and AAL control fields and writes the completed cell to the Output FIFO.

Rate generators

The Rate Generator logic provides 16 independent high-accuracy fractional rate multipliers driven from any of 4 external clock pins. Each rate generator divides down the external clock (by a user-programmable amount) to create "ticks" that initiate the transmission of ATM cells. Each Virtual Connection is then user-programmed to use one of these 16 rates.

Rate queues

There are 16 Rate Queues, one for each rate generator. The Rate Queue logic, activated by one or more rate generator "ticks", determines which connection is allowed to transmit a cell during the current "cell time". It does this by selecting the highest

priority queue and then locating the proper connection in that queue. Connections within a queue are serviced round-robin. The Rate Queue logic follows a linked list of connections that are associated with each of the 16 rate generators. The Rate Queue logic also enforces the priority of the rate generators (which is based on numerical value within each of the 3 user-programmable levels).

Reassembly Device Functional Blocks

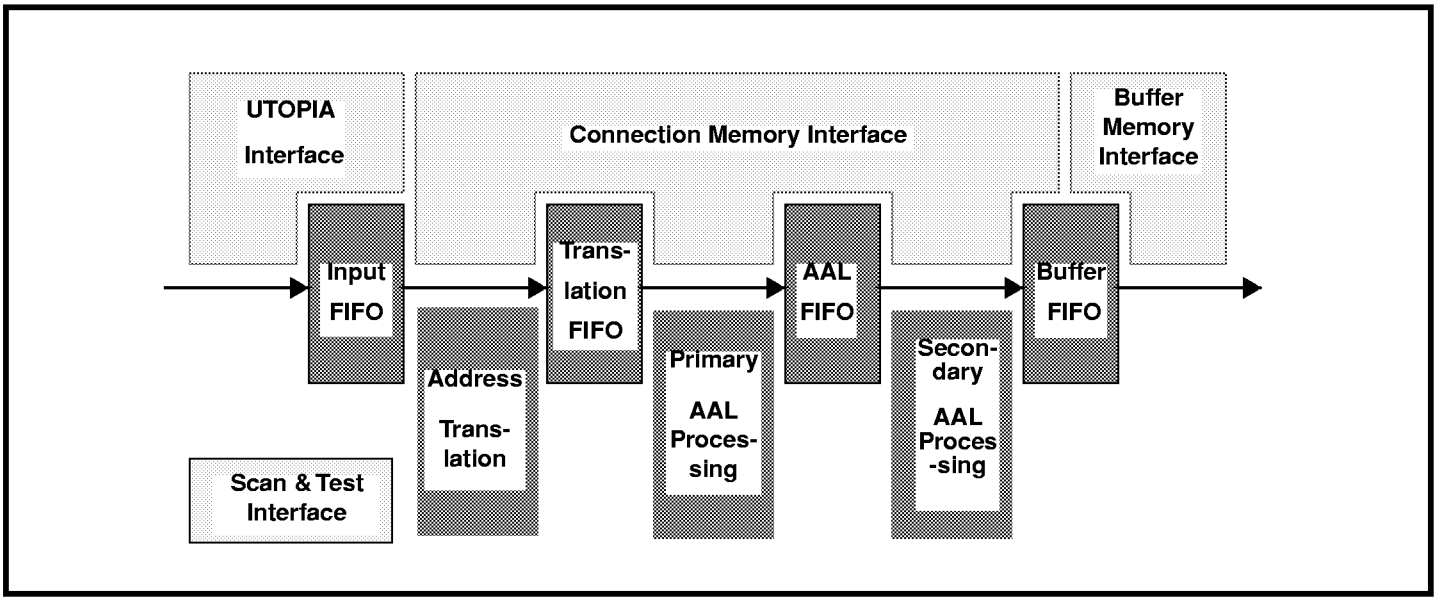


Figure 2-2 Reassembly Block Diagram

A block diagram of the Reassembly device is shown in Figure 2-2, this section describes the functionality of the blocks.

The basic structure is that of a series of data FIFOs which act as a pipeline for the activities carried out by the various state machines.

Buffer Memory (System Bus) Interface

The Buffer Memory Interface transfers user data from the Reassembly device to the external Buffer Memory. It includes a state machine that generates a request for the use of Buffer Memory and waits for a grant a signal from an external arbiter. Is also is used to read and write Descriptor Memory. Internally this function reads control information and user data from asynchronous FIFOs that are accessed from a different clock domain (MClk) by other functions in the device. The Buffer Memory function is driven by a separate clock (BmClk) so that the shared memory subsystem can run at a slower rate than most of the internal Segmentation functions.

Connections Memory Interface

The Connections Memory Interface allows the Reassembly device to access a local private memory for data structures which describe all active Virtual Circuits. The device accesses this memory during each cell time using prearranged timeslots. This function also manages the sharing of the Connection Memory data bus with the register interface since register data is transferred on the same pins.

UTOPIA (Physical Layer) Interface

The Physical Layer Interface provides access to the Physical Layer device assumed to be an OC-12c framer. The interface used is based on the ATM Forum UTOPIA interface. For 622Mbps operation a 16-bit mode is provided. At lower data rates an 8-bit mode is used. This interface is cell-based, i.e., a control signal is checked between cells to determine if another cell can be transmitted across the interface.

Scan and Test Interface

Scan interface is used during manufacturing to serially shift in and out scan test patterns that perform part of the overall manufacturing test. Over 99% of the flip-flops used for normal operation are tested by Scan.

The JTAG interface conforms to the IEEE 1149.1 standard.

Data FIFOs

The cell data path through the Reassembly device consists of a number data FIFOs as described in this subsection. These FIFOs are of varying sizes as required by the state machines which operate on or with the associated data.

Input (UTOPIA) FIFO

The Input FIFO isolates the internal state machines from the various physical data rates which can be used. The FIFO is 16 bits wide.

Translation FIFO

The Translation FIFO buffers the data between the address translation state machine and the AAL processing state machine. It holds one cell and is written by the Address Translation state machine (16 bits wide) and read by the AAL processing state machine (32 bits wide).

AAL FIFO

The AAL FIFO buffers the data between the AAL processing and the AAL disassembly state machines.The AAL FIFO buffers one ATM cell and is 32 bits wide.

Buffer FIFO

The Buffer FIFO holds the payload data destined for Buffer Memory. The FIFO provides rate buffering between the Buffer Memory clock (BmClk) rate and the internal cell processing clock rate (MClk). The FIFO has a 64-bit wide Buffer Memory interface and a 32-bit wide interface to the Secondary AAL processing function. The FIFO holds up to three disassembled ATM cell payloads.

Primary AAL Processing

The Primary AAL Processing block performs an initial scan on received cells and tags the cells with connection information which can be used by subsequent state machines. CRC checks, are also performed at this point.

Secondary AAL Processing

The Secondary AAL Processing block strips the cell of the ATM header and AAL headers/trailers if present. It also compacts the data into the correct byte alignment for the Buffer Memory Interface.

This chapter describes the characteristics and functions of the 188 Segmentation device and the 187 Reassembly device signal pins.

SAR Signals

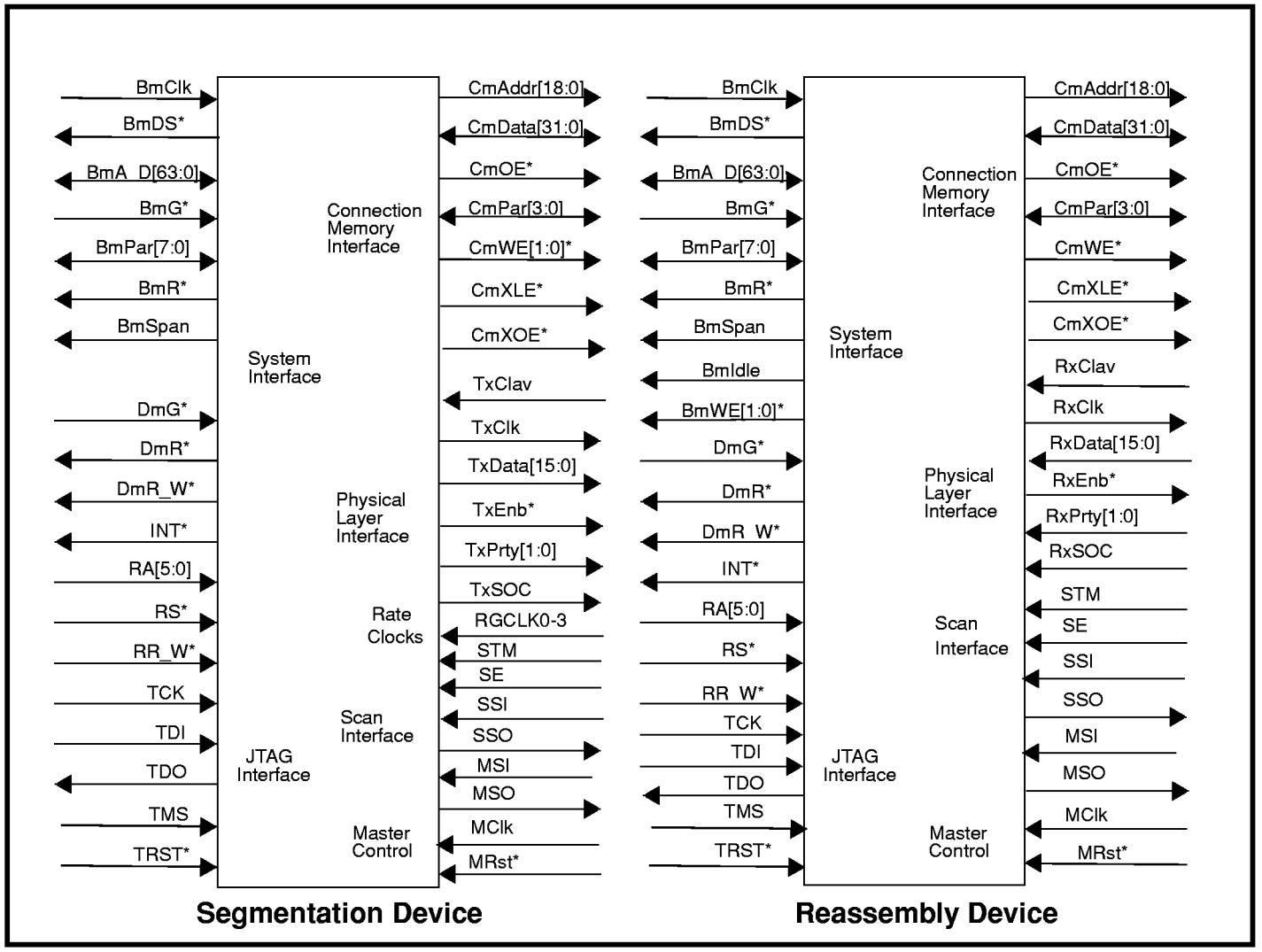


Figure 3-1 Segmentation and Reassembly Device Signals

The following sections describe the pins for both the Segmentation and Reassembly devices. Most pin names are common to both devices although the functionality may differ slightly (primarily in terms of data flow direction). In the latter case the difference between the devices is noted.

The entire design is synchronous (with two clock domains) and is positive-edge triggered. Internally the SAR communicates across the clock domains using asynchronous FIFOs containing synchronizers to handle metastability. The user should not drive both clocks (BmClk and MClk) with separate oscillators of identical or nearly identical value in order to minimize metastability.

Signal characteristics are listed as TTL Input, TTL Output or Tri-State Input/Output. The TTL Inputs are TTL-level compatible CMOS inputs. The outputs are TTL-compatible CMOS rail-to-rail outputs. The Tri-State Input/Outputs act similarly depending on direction.

System Interface Signals

The System Interface serves a dual purpose in the SAR operation. First, it enables the system-level microprocessor access to on-chip registers for operational control and exception handling. Second, it serves the function of a DMA path between the SAR devices and their associated Buffer Memories. A flat memory interface is provided to allow interfacing to various types of memory devices.

Table 3-1 System Interface Signals and Pin Characteristics

Signal Name	Characteristic
BmClk	TTL Input
BmC_S*	TTL Output
BmA/D[63:0]	Tri-State Input/Output
BmG*	TTL Input
BmPar[7:0]	Tri-State Input/Output
BmR*	TTL Output
BmSpan	TTLOutput
BmIdle	TTLOutput
BmWE*	TTL Output
DmG*	TTL Input
DmR*	TTL Output
DmR_W*	Tri-State Output
INT*	TTL Output
RA[5:0]	TTL Input
RS*	TTL Input
RR_W*	TTL Input

BmClk

The Buffer Memory Clock input signal provides clocking for the memory interface logic.

BmDS* (formerly called BmC_S*)

The Buffer Memory Data Strobe output signal is asserted low to indicate the start/continuation of a DMA cycle and is active-low during a DMA burst when data is valid. Input data (to the chip) is latched at every rising edge of the system clock while this signal is asserted for the Segmentation chip. Output data is valid at every rising edge of the system clock while this signal is asserted for the Reassembly chip.

BmA_D[63:0]

The Buffer Memory Address/Data bus is a multiplexed 64-bit bus which provides address and data for both Buffer Memory and Descriptor Memory. Data order is from the high end of the bus to the low. As an example if 8 bytes of data are received or transmitted (that happen to be aligned with the bus width), then the byte-order at the physical layer is BmA_D[63:56], BmA_D[55:48], BmA_D[47:40]....BmA_D[7:0].

For the Segmentation chip the BmA_D pins are driven as:

- BmA_D[63:56] -- Input
- BmA_D[55:32] -- Tri-State Input/Output
- BmA_D[31:16] -- Input
- BmA_D[15:0] -- Tri-State Input/Output
- For the Reassembly chip the BmA_D pins are driven as:
- BmA_D[63:48] -- Output
- BmA_D[47:32] -- Tri-State Input/Output
- BmA_D[31:16] -- Output
- BmA_D[15:0] -- Tri-State Input/Output

622Mbps Mode

- Buffer Memory data (64 bits): BmA_D[63:0].
- Descriptor Memory data (32 bits): BmA_D[31:0].
- Buffer Memory address (22 bits): BmA_D[55:54] are always zero (provided to simplify parity handling). The 22-bit address is BmA_D[53:32]. The address is a double-word (8 byte) address in "wide" mode and a word address in "narrow" mode (i.e. each address references the bus-width of a data cycles).
- Descriptor Memory address (22 bits max): BmA_D[55:54] are always zero (provided to simplify parity handling). The 22-bit address is BmA_D[53:32]; where BmA_D[53:48] is taken from the Descriptor Memory Base Address Register and bits BmA_D[47:32] are used to select a given descriptor. The address is a word (4 byte) address. NOTE: For 622Mbps operation with a separate physical Descriptor Memory -- BmA_D[53:48] can be ignored if the Descriptor Memory Base Address Register is intended to be zero.

155Mbps Mode

- Buffer Memory data (32 bits): BmA_D[63:32].
- Descriptor Memory data (32 bits): BmA_D[63:32]. (NOTE: This difference between modes is due to the need to overlap address/data cycles when running at 622Mbps, yet have only a single 32-bit bus for 155Mbps).

Buffer Memory and Descriptor Memory addresses are the same as the 622Mbps Mode.

BmG*

The Buffer Memory Grant signal (input, active LOW) indicates that access is granted to the chip for Buffer Memory read/write operations. A grant is expected as soon as possible after BmR* is asserted.

BmPar[7:0]

The Buffer Memory Parity Input/Output bus provides odd byte parity signals for the BmA_D bus.

BmR*

The Buffer Memory Request signal (output, active LOW) requests access rights to the Buffer Memory and indicates that an address is present on the BmA_D[53:32] pins. (NOTE: this address must be captured by the external logic during the same clock that BmR* is asserted -- it is not held until BmG*). When cell data falls across a buffer boundary (i.e. a span occurs) BmR* is asserted once for each of the two non-contiguous bursts.

Note: BmR* may or may not be asserted at the same time DmR* is asserted. In this case it is the users option to select which grant to provide (BmG* or DmG*). If both BmR* and DmR* are asserted the user could theoretically grant access to both since the SAR will properly serialize the use of the multiplexed pins and access one memory followed by the other. However, in this case, the board designer may have difficulty determining the memory access pattern and so it is not recommended.

BmSpan

The Buffer Memory Span signal (output, active HIGH) indicates when the current cell-size access cycle will span two Buffer Memory blocks. This signal is provided to optionally allow BmG* to remain active even though BmR* is de-asserted. In this case the arbiter can check for BmSpan (in addition to BmG*) before relinquishing the grant. It is acceptable to ignore this signal and allow normal re-arbitration to occur during a span. BmSpan is only valid at the point that BmR* transitions from asserted to de-asserted.

BmIdle

Indicates that the chipset Buffer Memory interface has gone idle after transferring data to Buffer Memory. This signal can be used to flush the serial-shift-register in implementations that use video RAM to avoid stale data following receive notifications.

BmWE[1:0]*

The Buffer Memory Word Enable output signals indicate validity of the data on BmData[63:32] and BmData[31:0] respectively. This signal is used only by the Reassembly device. The Segmentation device never writes to Buffer Memory.

DmG*

The Descriptor Memory Grant signal (input, active LOW) indicates to the chipset memory interface logic that access is granted for read/write operations to Descriptor Memory.

DmR*

The Descriptor Memory Request signal (output, active LOW) requests chipset access rights to the Descriptor Memory.

DmR_W*

The Descriptor Memory Read/Write input signal indicates whether a read or write access is to be performed on the Descriptor Memory. A write is indicated by a logic zero level and a read by a logic one.

INT*

The Interrupt signal (output, active LOW) indicates an event has occurred that can be determined by reading the Interrupt Status Register. Writing to the Interrupt Status Register de-asserts the INT* pin for at least 3 MClk cycles. The INT* will spring back active if any non-masked interrupt conditions continue to be active after writing to the Interrupt Status Register.

RA[5:0]

The Register Address input bus selects which internal register is accessed when the RS* signal is asserted. Note that the data is provided on the CmData[31:0] bus with use of the CmXLE* and CmXOE* signals.

RS*

The Register Select signal (input, active LOW) initiates the start of a register cycle.

RR_W*

The Register Read/Write input signal indicates whether a read or a write access is to be performed on the chip's internal register set when RS* is asserted. A logic zero level indicates a write and a logic one level indicates a read.

Master Control Signals

The Master Control Signals are a Clock for internal chip use and a Master Reset for initializing the internal registers and state machines.

Table 3-2 Master Control Signals and Pin Characteristics

Signal Name	Characteristic
MClk	TTL Input
MRst*	TTL Input

MClk

The Master Clock input signal is used as the main clock by the Segmentation and Reassembly devices. This clock drives the logic for the UTOPIA interface, the Connection Memory interface and the register interface.

MRst*

The Master Reset input signal provides a chip-wide reset. It resets the logic driven by both BmClk and MClk.

Connection Memory Interface Signals

The Connection Memory Interface signals support the external memory which contains the context information on each virtual circuit. The interface also supports the register datapath (which shares the Connection Memory data bus).

Table 3-3 Control Memory Interface Signals and Pin Characteristics

Signal Name	Characteristic
CmAddr[18:0]	TTL Input/Output
CmData[31:0]	Tri-State Input/Output
CmOE*	TTL Output
CmPar[3:0]	Tri-State Input/Output
CmWE*	TTL Output
CmXLE	TTL Output
CmXOE	TTL Output

CmAddr[18:0]

The Connection Memory Address bus is a 19-bit output bus which accesses up to 2 Megabytes of external Connection Memory (i.e. .5M x 4 bytes). The addresses generated are word (4 byte) addresses.

CmData[31:0]

The Connection Memory Data bus is a 32-bit bidirectional data bus. This data bus is also used to provide a data path for register access using the RS*, RR_W, RA[5:0], CmXLE* and CmXOE* signals.

CmOE*

The Connection Memory Output Enable signal (output, active LOW) strobes the data accesses during read and write operations.

CmPar[3:0]

The Connection Memory Parity bus provides odd byte parity for the CmData bus.

CmWE*

The Connection Memory Write Enable signal (output, active LOW) provides a strobe for write operations. The Segmentation device has CmWE*[1:0] which are half-word write-enables (CmWE*[0] = low half word; CmWE*[1] = high half word). The Reassembly device has only a single CmWE* and writes full words only.

CmXLE*

The Connection Memory External Latch Enable signal (output, active LOW) provides a strobe to latch data into a transceiver during a register read operation. Once this signal has been asserted the register operation is complete from the SAR's perspective.

CmXOE*

The Connection Memory External Output Enable signal (output, active LOW) provides a strobe to output data from a transceiver during a register write operation. Once this signal is asserted the register operation is complete from the SAR's perspective.

Rate Clock Signals

The Rate Clock (Segmentation device only) input signals generate internal clocks for the rate control logic. The signals have the same timing requirements as MClk except that a rate clock (that is asynchronous to MClk) divided by it's prescaler must be less than or equal to MClk/2 (to insure sampling accuracy across clock domains).

Table 3-4 External Clock Signals and Pin Characteristics

Signal Name	Characteristic
RGCLK0-3	TTL Input

RGCLK0-3

The Rate Clock inputs provide four different external clock inputs for use with the rate shaping functions in the Segmentation chip. Each Rate Clock input feeds a 4-bit programmable prescaler which in turn clocks the Rate Generator logic which performs the fractional rate multiplication (using the programmable M/N values).

JTAG Interface Signals

The JTAG interface signals support the IEEE 1149.1 boundary scan interface for board level circuit testing.

Table 3-5 JTAG Interface Signals and Pin Characteristics

Signal Name	Characteristic
TCK	TTL Input
TDI	TTL Input
TDO	TTL Output
TMS	TTL Input
TRST*	TTL Input

TCK

The Test Clock input signal provides clocking to the JTAG boundary scan interface.

TDI

The Test Data In input pin provides an input port for serial data through the boundary scan interface.

TDO

The Test Data Output pin is the output pin for data shifted from TDI through the boundary scan interface.

TMS

The Test Mode Select input signal enables the boundary scan function.

TRST*

The Test Reset input signal resets the JTAG logic and boundary scan interface and is asynchronous to TCK.

Internal Scan Interface Signals

The Scan Interface is used by ATPG equipment as part of the manufacturing process. This interface is normally unused by the designer, but the SE and STM pins must be tied as shown below (and the SSI and MSI pins should not be allowed to float).

Table 3-6 Scan Interface Signals and Pin Characteristics

Signal Name	Characteristic
SE	TTL Input
STM	TTL Input
SSI	TTL Input
SSO	TTL Output
MSI	TTL Input
MSO	TTL Output

SE

The Scan Enable signal (input, active HIGH) enables the shifting of data thru the internal scan chains and disables the normal operation of the chip. For normal (non-ATPG) chip operation SE should be held LOW.

STM

The Scan Test Mode signal (input, active HIGH) is for ATPG use during scan. If STM is set HIGH then the internal chip reset (normally set by MRst* or thru software using the Mode Register) is disabled. This allows the ATPG to prevent "reset" while shifting data thru the internal scan chains. For normal (non-ATPG) chip operation STM should be held LOW.

SSI

The System Scan Input is the serial input for data shifted into the BmClk domain. If SE is set LOW this input is unused (but should not be allowed to float).

SSO

The System Scan Output is the serial output for data shifted from the BmClk domain.

MSI

The Master Scan Input is the serial input for data shifted into the MClk domain. If SE is set LOW this input is unused (but should not be allowed to float).

MSO

The Master Scan Output is the serial output for data shifted from the MClk domain.

Physical Layer Transmit Interface Signals

The Physical Layer Transmit Interface conforms to the ATM Forum UTOPIA interface and are assumed to attach to an ATM Framer. These signals apply only to the Segmentation device. The bus size for 622Mbps is 16 bits wide while at 155Mbps the bus width is 8 bits.

Table 3-7 Physical Layer Transmit Signals and Pin Characteristics

Signal Name	Characteristic
TxClav	TTL Input
TxClav	TTL Output
TxData[15:0]	TTL Output
TxEnb*	TTLOutput
TxPrty[1:0]	TTLOutput
TxSOC	TTLOutput

TxClav

This is a flow control signal (input active HIGH) driven by the Framer and indicates that the PHY FIFO has space for a full cell of data. If this signal is not asserted when the SAR completes the transmission of a cell, the SAR will wait to begin transmitting until the cycle after the SAR sees TXClav asserted again. If TxClav is de-asserted during a cell, the SAR will continue transmitting data until it completes the cell it is currently transmitting.

TxClk

Clock provided by the SAR and originally intended for synchronization of the TxData bus.

IMPORTANT: THIS OUTPUT SIGNAL FROM THE SEGMENTATION CHIP SHOULD NOT BE USED. Instead a typical configuration should feed the board clock that drives MClk to the Framers TxClk input also.

Using this signal (output from Segmentation chip) relaxes the setup time for control and data signals to the framer, but will also produce a failing hold time.

TxData, TxPrty, TxSOC, and TxEnb* signals are driven on the rising edge of MClk and have significant output delay (see AC timing).

Note that the TxClk used by the framer (i.e. MClk) must be 40Mhz for OC-12 operation and at least 20Mhz for OC-3 operation.

TxData[15:0]

Sixteen-bit transmit data driven by the SAR to the Framer. TxData[15] is the MSB. The byte arrangement is big endian. In 155Mbps mode only the low order 8-bits, TxData[7:0], are used. In 622Mbps operation the full 16-bit bus is used.

TxEnb*

This signal (output, active LOW) is asserted by the SAR during cycles when valid data appears on the TxData bus.

TxPrty[1:0]

Odd parity computed on TxData[15:0]. TxPrty[1] is the odd parity of TxData[15:8] and TxPrty[0] is the odd parity of TxData[7:0].

In 155Mbps operation, TxPrty[1] is always high, and TxPrty[0] is the odd parity of TxData[7:0].

TxSOC

This signal (output, active HIGH) is asserted by the SAR to indicate that the start of a new cell is present on the TxData bus.

Physical Layer Receive Interface Signals

The Physical Layer Receive Interface conforms to the ATM Forum UTOPIA interface and are assumed to attach to an ATM Framer. These signals apply only to the Reassembly device. The bus size for 622Mbps is 16 bits wide while at 155Mbps the bus width is 8 bits.

Table 3-8 Physical Layer Receive Signals and Pin Characteristics

Signal Name	Characteristic
RxClav	TTL Input
RxClk	TTL Output
RxData[15:0]	TTL Input
RxEnb*	TTLOutput
RxPrty[1:0]	TTL Input
RxSOC	TTL Input

RxClav

This signal (input, active HIGH) is driven by the Framer and indicates that the Framer has a complete cell of data ready for transfer.

RxClk

Clock provided by the SAR and originally intended for synchronization of the RxData bus.

IMPORTANT: THIS OUTPUT SIGNAL FROM THE SEGMENTATION CHIP SHOULD NOT BE USED. Instead a typical configuration should feed the board clock that drives MClk to the Framers RxClk input also.

Using this signal (output from Reassembly chip) relaxes the hold time for control and data signals from the framer, but is also likely to result in insufficient setup time.

The Framer must provide RxData, RxPrty, RxSOC, and RxClav synchronized to MClk and with setup and hold times as defined in the AC timing.

Note that the RxClk used by the framer (i.e. MClk) must be 40MHz for OC-12 operation and at least 20MHz for OC-3 operation.

RxData[15:0]

Sixteen-bit transmit data driven by the Framer to the Reassembly device. RxData[15] is the MSB. The byte arrangement is Big Endian.

In 155Mbps operation, RxData[15:8] are not used.

RxEnb*

This signal is asserted by the SAR (output, active LOW) to indicate that RxData, RxPrty, and RxSOC will be sampled in the next clock period.

RxPrty[1:0]

Odd parity checked on RxData[15:0]. RxPrty[1] is checked for RxData[15:8] and RxPrty[0] is checked for RxData[7:0].

In 155Mbps operation RxPrty[1] is not used.

RxSOC

This signal (input, active HIGH) is asserted by the Framer to indicate that the start of a new cell is present on the RxData bus.

Pin Diagram

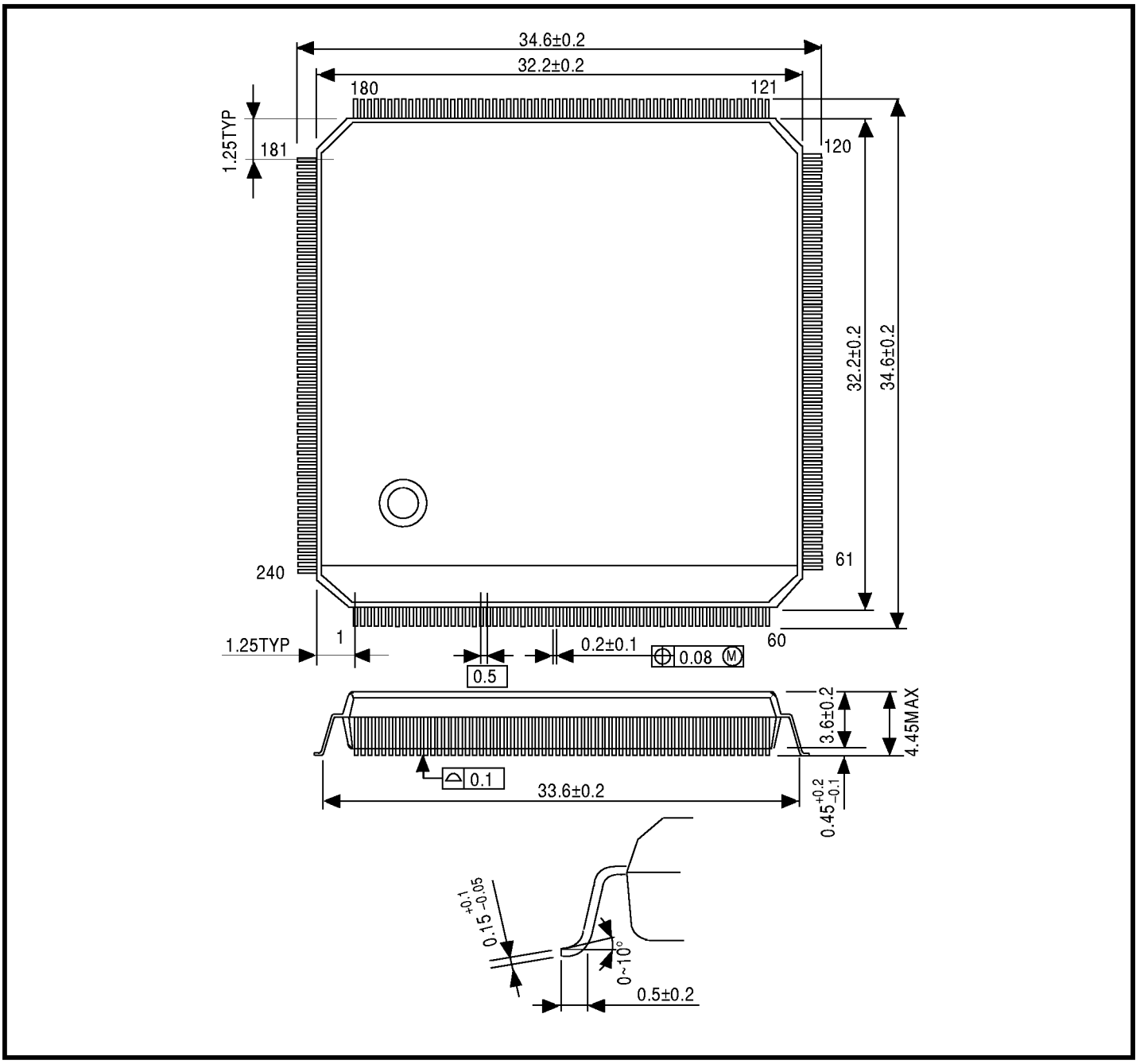


Figure 3-2 QFP240-P-3232 (240-Pin Plastic Package)

Pin Assignments

Reassembly Chip

Table 3-9 Reassembly Chip Pin Assignments

Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name
1	VSS	37	BmA_D[11]	73	VSS
2	BmC_S	38	BmA_D[10]	74	RxPrty[0]
3	BmSpan	39	BmA_D[9]	75	RxData[7]
4	BmIdle	40	BmA_D[8]	76	RxData[6]
5	VSS	41	VSS	77	RxData[5]
6	BmWE[0]	42	VSS	78	RxData[4]
7	BmWE[1]	43	BmPar[0]	79	RxData[3]
8	VSS	44	BmA_D[7]	80	RxData[2]
9	BmPar[3]	45	BmA_D[6]	81	RxData[1]
10	BmA_D[31]	46	BmA_D[5]	82	RxData[0]
11	BmA_D[30]	47	BmA_D[4]	83	VSS
12	BmA_D[29]	48	BmA_D[3]	84	VDD
13	BmA_D[28]	49	BmA_D[2]	85	VSS
14	BmA_D[27]	50	BmA_D[1]	86	RxSOC
15	BmA_D[26]	51	BmA_D[0]	87	RxClav
16	BmA_D[25]	52	VDD	88	RxEnb
17	BmA_D[24]	53	BmClk	89	RxClk
18	VSS	54	VSS	90	VDD
19	VDD	55	SSI	91	VSS
20	VSS	56	SSO	92	CmPar[3]
21	BmPar[2]	57	STM	93	CmData[31]
22	BmA_D[23]	58	MSI	94	CmData[30]
23	BmA_D[22]	59	MSO	95	CmData[29]
24	BmA_D[21]	60	VDD	96	CmData[28]
25	BmA_D[20]	61	VSS	97	CmData[27]
26	BmA_D[19]	62	SE	98	CmData[26]
27	BmA_D[18]	63	RxPrty[1]	99	CmData[25]
28	BmA_D[17]	64	RxData[15]	100	CmData[24]
29	BmA_D[16]	65	RxData[14]	101	VSS
30	VDD	66	RxData[13]	102	VSS
31	VSS	67	RxData[12]	103	CmPar[2]
32	BmPar[1]	68	RxData[11]	104	CmData[23]
33	BmA_D[15]	69	RxData[10]	105	CmData[22]
34	BmA_D[14]	70	RxData[9]	106	CmData[21]
35	BmA_D[13]	71	RxData[8]	107	CmData[20]
36	BmA_D[12]	72	VDD	108	CmData[19]

Table 3-9 Reassembly Chip Pin Assignments (continued)

Pin No.	Pin Name
109	CmData[18]
110	CmData[17]
111	CmData[16]
112	VDD
113	VSS
114	CmPar[1]
115	CmData[15]
116	CmData[14]
117	CmData[13]
118	CmData[12]
119	CmData[11]
120	VDD
121	VSS
122	CmData[10]
123	CmData[9]
124	CmData[8]
125	VSS
126	VSS
127	CmPar[0]
128	CmData[7]
129	CmData[6]
130	CmData[5]
131	CmData[4]
132	CmData[3]
133	CmData[2]
134	CmData[1]
135	CmData[0]
136	VSS
137	VSS
138	CmAddr[18]
139	CmAddr[17]
140	CmAddr[16]
141	VDD
142	CmAddr[15]
143	CmAddr[14]
144	CmAddr[13]

Pin No.	Pin Name
145	CmAddr[12]
146	CmAddr[11]
147	CmAddr[10]
148	CmAddr[9]
149	CmAddr[8]
150	VDD
151	VSS
152	CmAddr[7]
153	CmAddr[6]
154	CmAddr[5]
155	CmAddr[4]
156	CmAddr[3]
157	CmAddr[2]
158	CmAddr[1]
159	CmAddr[0]
160	VDD
161	VSS
162	MRst
163	MClk
164	VSS
165	CmOE
166	CmWE
167	Reserved
168	CmXOE
169	CmXLE
170	VSS
171	RA[5]
172	RA[4]
173	RA[3]
174	RA[2]
175	RA[1]
176	RA[0]
177	RS
178	VSS
179	INT
180	VDD

Pin No.	Pin Name
181	VSS
182	RR_W
183	DmR_W
184	DmR
185	DmG
186	BmR
187	BmG
188	VSS
189	VSS
190	BmPar[7]
191	BmA_D[63]
192	BmA_D[62]
193	BmA_D[61]
194	BmA_D[60]
195	BmA_D[59]
196	BmA_D[58]
197	BmA_D[57]
198	BmA_D[56]
199	VDD
200	VSS
201	BmPar[6]
202	BmA_D[55]
203	BmA_D[54]
204	BmA_D[53]
205	BmA_D[52]
206	BmA_D[51]
207	BmA_D[50]
208	BmA_D[49]
209	BmA_D[48]
210	VDD
211	VSS
212	BmPar[5]
213	BmA_D[47]
214	BmA_D[46]
215	BmA_D[45]
216	BmA_D[44]

Table 3-9 Reassembly Chip Pin Assignments (continued)

Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name
217	BmA_D[43]	225	BmA_D[38]	233	BmA_D[32]
218	BmA_D[42]	226	BmA_D[37]	234	TCK
219	BmA_D[41]	227	BmA_D[36]	235	TDI
220	BmA_D[40]	228	VSS	236	TDO
221	VDD	229	VSS	237	TMS
222	VSS	230	BmA_D[35]	238	TRST
223	BmPar[4]	231	BmA_D[34]	239	VSS
224	BmA_D[39]	232	BmA_D[33]	240	VDD

Segmentation Chip

Table 3-10 Segmentation Chip Pin Assignments

Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name
1	VSS	26	BmA_D[19]	51	BmA_D[0]
2	BmC_S	27	BmA_D[18]	52	VDD
3	BmSpan	28	BmA_D[17]	53	BmClk
4	VDD	29	BmA_D[16]	54	VSS
5	RGCLK0	30	VDD	55	SSI
6	RGCLK1	31	VSS	56	SSO
7	RGCLK2	32	BmPar[1]	57	STM
8	RGCLK3	33	BmA_D[15]	58	MSI
9	BmPar[3]	34	BmA_D[14]	59	MSO
10	BmA_D[31]	35	BmA_D[13]	60	VDD
11	BmA_D[30]	36	BmA_D[12]	61	VSS
12	BmA_D[29]	37	BmA_D[11]	62	SE
13	BmA_D[28]	38	BmA_D[10]	63	TxPrty[1]
14	BmA_D[27]	39	BmA_D[9]	64	TxData[15]
15	BmA_D[26]	40	BmA_D[8]	65	TxData[14]
16	BmA_D[25]	41	VSS	66	TxData[13]
17	BmA_D[24]	42	VSS	67	TxData[12]
18	VSS	43	BmPar[0]	68	TxData[11]
19	VDD	44	BmA_D[7]	69	TxData[10]
20	VDD	45	BmA_D[6]	70	TxData[9]
21	BmPar[2]	46	BmA_D[5]	71	TxData[8]
22	BmA_D[23]	47	BmA_D[4]	72	VDD
23	BmA_D[22]	48	BmA_D[3]	73	VSS
24	BmA_D[21]	49	BmA_D[2]	74	TxPrty[0]
25	BmA_D[20]	50	BmA_D[1]	75	TxData[7]

Table 3-10 Segmentation Chip Pin Assignments (continued)

Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name
76	TxData[6]	115	CmData[15]	154	CmAddr[5]
77	TxData[5]	116	CmData[14]	155	CmAddr[4]
78	TxData[4]	117	CmData[13]	156	CmAddr[3]
79	TxData[3]	118	CmData[12]	157	CmAddr[2]
80	TxData[2]	119	CmData[11]	158	CmAddr[1]
81	TxData[1]	120	VDD	159	CmAddr[0]
82	TxData[0]	121	VSS	160	VDD
83	VSS	122	CmData[10]	161	VSS
84	VDD	123	CmData[9]	162	MRst
85	VSS	124	CmData[8]	163	MClk
86	TxSOC	125	VSS	164	VSS
87	TxClav	126	VSS	165	CmOE
88	TxEnb	127	CmPar[0]	166	CmWE[0]
89	TxClk	128	CmData[7]	167	CmWE[1]
90	VDD	129	CmData[6]	168	CmXOE
91	VSS	130	CmData[5]	169	CmXLE
92	CmPar[3]	131	CmData[4]	170	VSS
93	CmData[31]	132	CmData[3]	171	RA[5]
94	CmData[30]	133	CmData[2]	172	RA[4]
95	CmData[29]	134	CmData[1]	173	RA[3]
96	CmData[28]	135	CmData[0]	174	RA[2]
97	CmData[27]	136	VSS	175	RA[1]
98	CmData[26]	137	VSS	176	RA[0]
99	CmData[25]	138	CmAddr[18]	177	RS
100	CmData[24]	139	CmAddr[17]	178	VSS
101	VSS	140	CmAddr[16]	179	INT
102	VSS	141	VDD	180	VDD
103	CmPar[2]	142	CmAddr[15]	181	VSS
104	CmData[23]	143	CmAddr[14]	182	RR_W
105	CmData[22]	144	CmAddr[13]	183	DmR_W
106	CmData[21]	145	CmAddr[12]	184	DmR
107	CmData[20]	146	CmAddr[11]	185	DmG
108	CmData[19]	147	CmAddr[10]	186	BmR
109	CmData[18]	148	CmAddr[9]	187	BmG
110	CmData[17]	149	CmAddr[8]	188	SS
111	CmData[16]	150	VDD	189	VSS
112	VDD	151	VSS	190	BmPar[7]
113	VSS	152	CmAddr[7]	191	BmA_D[63]
114	CmPar[1]	153	CmAddr[6]	192	BmA_D[62]

Table 3-10 Segmentation Chip Pin Assignments (continued)

Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name
193	BmA_D[61]	209	BmA_D[48]	225	BmA_D[38]
194	BmA_D[60]	210	VDD	226	BmA_D[37]
195	BmA_D[59]	211	VSS	227	BmA_D[36]
196	BmA_D[58]	212	BmPar[5]	228	VSS
197	BmA_D[57]	213	BmA_D[47]	229	VSS
198	BmA_D[56]	214	BmA_D[46]	230	BmA_D[35]
199	VSS	215	BmA_D[45]	231	BmA_D[34]
200	VSS	216	BmA_D[44]	232	BmA_D[33]
201	BmPar[6]	217	BmA_D[43]	233	BmA_D[32]
202	BmA_D[55]	218	BmA_D[42]	234	TCK
203	BmA_D[54]	219	BmA_D[41]	235	TDI
204	BmA_D[53]	220	BmA_D[40]	236	TDO
205	BmA_D[52]	221	VDD	237	TMS
206	BmA_D[51]	222	VSS	238	TRST
207	BmA_D[50]	223	BmPar[4]	239	VSS
208	BmA_D[49]	224	BmA_D[39]	240	VDD

This page left blank intentionally

Overview

The register interface is defined in the following sections. This interface is used to initialize the SAR's operating modes and to transfer pointers for data received and transmitted. The interface includes a set of maskable interrupts and an indirect access mechanism to Connection Memory. The Segmentation device also includes a set of programmable rate generators.

Register Access

Registers are addressed by the RA[5:0] bus. The tables that follow summarize the register attributes. The "Access" column in these tables refer to the following key:

1. Can only write certain bits while "Device Reset" bit is one.
2. This register can only be read or written when the "Tx Enable" bit in the S_MODE register is zero. When "Tx Enable" is one, writing to the register has no affect and the read-back value will is meaningless.
3. Read-back value is from internal working register when "Rx Enable" is active in Mode register. Otherwise read-back value is user-written value.
4. Writing a one to a bit position clears the corresponding condition if it is no longer active.
5. Writing any value to the register resets the register to zero.
6. Can only write while "Transmit Enable" bit in Mode register is zero.

Table 4-1 Segmentation Registers

Name	Symbol	Addr	Bits	Access
Mode	S_MODE	00h	12	R/W(1)
Descriptor Memory Base Address	S_DMBASE	01h	6	R/W(6)
Connection Memory Address	S_CMAR	02h	19	R/W
Connection Memory Data	S_CMDR	03h	32	R/W
Interrupt Status	S_ISR	04h	9	R/W(4)
Interrupt Mask	S_IMR	05h	9	R/W
Free-Pool Tail	S_FREET	06h	16	R/W(2)
Transmit Request (CID)	S_TXRCID	07h	16	R/W
Transmit Request (Head/Tail)	S_TXRHT	08h	32	R/W
Transmit Notification Queue Control	S_TXNQC	09h	32	R/W
Transmit Notification (CID)	S_TXNCID	0ah	16	R
Reserved		0bh		
Reserved		0ch		
Cells Transmitted	S_CELLT	0dh	32	R/W(5)
Transmit Purge	S_TXPUR	0eh	16	R/W(5)
Rate Generator Prescaler	S_RGP	0fh	16	R/W
Rate Generator Control 0	S_RGC0	10h	32	R/W
Rate Generator Control 1	S_RGC1	11h	32	R/W
Rate Generator Control 2	S_RGC2	12h	32	R/W
Rate Generator Control 3	S_RGC3	13h	32	R/W
Rate Generator Control 4	S_RGC4	14h	32	R/W
Rate Generator Control 5	S_RGC5	15h	32	R/W
Rate Generator Control 6	S_RGC6	16h	32	R/W
Rate Generator Control 7	S_RGC7	17h	32	R/W
Rate Generator Control 8	S_RGC8	18h	32	R/W
Rate Generator Control 9	S_RGC9	19h	32	R/W
Rate Generator Control 10	S_RGC10	1ah	32	R/W
Rate Generator Control 11	S_RGC11	1bh	32	R/W
Rate Generator Control 12	S_RGC12	1ch	32	R/W
Rate Generator Control 13	S_RGC13	1dh	32	R/W
Rate Generator Control 14	S_RGC14	1eh	32	R/W
Rate Generator Control 15	S_RGC15	1fh	32	R/W
Rate Generator Divider 0	S_RGD0	20h	28	R/W
Rate Generator Divider 1	S_RGD1	21h	28	R/W
Rate Generator Divider 2	S_RGD2	22h	28	R/W
Rate Generator Divider 3	S_RGD3	23h	28	R/W
Rate Generator Divider 4	S_RGD4	24h	282	R/W
Rate Generator Divider 5	S_RGD5	25h	28	R/W

Table 4-1 Segmentation Registers (continued)

Name	Symbol	Addr	Bits	Access
Rate Generator Divider 6	S_RGD6	26h	28	R/W
Rate Generator Divider 7	S_RGD7	27h	28	R/W
Rate Generator Divider 8	S_RGD8	28h	28	R/W
Rate Generator Divider 9	S_RGD9	29h	28	R/W
Rate Generator Divider 10	S_RGD10	2ah	28	R/W
Rate Generator Divider 11	S_RGD11	2bh	328	R/W
Rate Generator Divider 12	S_RGD12	2ch	28	R/W
Rate Generator Divider 13	S_RGD13	2dh	28	R/W
Rate Generator Divider 14	S_RGD14	2eh	28	R/W
Rate Generator Divider 15	S_RGD15	2fh	28	R/W

Table 4-2 Reassembly Registers

Name	Symbol	Addr	Bits	Access
Mode	R_MODE	00h	16	R/W(1)
Descriptor Memory Base Address	R_DMBASE	01h	6	R/W
Connection Memory Address	R_CMAR	02h	19	R/W
Connection Memory Data	R_CMDR	03h	32	R/W
Interrupt Status	R_ISR	04h	19	R/W(4)
Interrupt Mask	R_IMR	05h	19	R/W
Free-Pool Head	R_FREEH	06h	16	R/W(3)
Receive Purge	R_RXPUR	07h	16	R/W
Address Translation	R_ATR	08h	32	R/W
Cells Received	R_CELLR	09h	32	R/W(5)
Cells Discarded	R_CELLD	0ah	32	R/W(5)
Receive Queue Control 0	R_RQC0	10h	32	R/W
Receive Queue Control 1	R_RQC1	11h	32	R/W
Receive Queue Control 2	R_RQC2	12h	32	R/W
Receive Queue Control 3	R_RQC3	13h	32	R/W
Receive Management CID/Head(que 0)	R_RMCID0	14h	32	R
Reserved		15h		
Receive Notification CID/Head (que 1)	R_RNCID1	16h	32	R
Receive Notification Status/Len (que 1)	R_RNSL1	17h	20	R
Receive Notification CID/Head (que 2)	R_RNCID2	18h	32	R
Receive Notification Status/Len (que 2)	R_RNSL2	19h	20	R
Receive Notification CID/Head (que 3)	R_RNCID3	1ah	32	R
Receive Notification Status/Len (que 3)	R_RNSL3	1bh	20	R

Segmentation Device Registers

The Segmentation registers are set to the following values following a hardware (MRst*) reset.:

S_MODE = 0
S_DMBASE = 0
S_ISR = 0
S_IMR = all ones
S_CMAR = 0
S_TXNQC = 0
S_FREET = 0
S_CELLT = 0
S_TXR = 0
S_TXRCID = 0
S_TXRHT = 0

The following sections define the individual registers.

Mode Register

S_MODE **00h**

Bit	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	Device Reset	Reserved	Buffer Memory Parity Enable	Connection Memory Parity Enable	Wide/Narrow Bus mode	Reserved	Host Buffer Size	

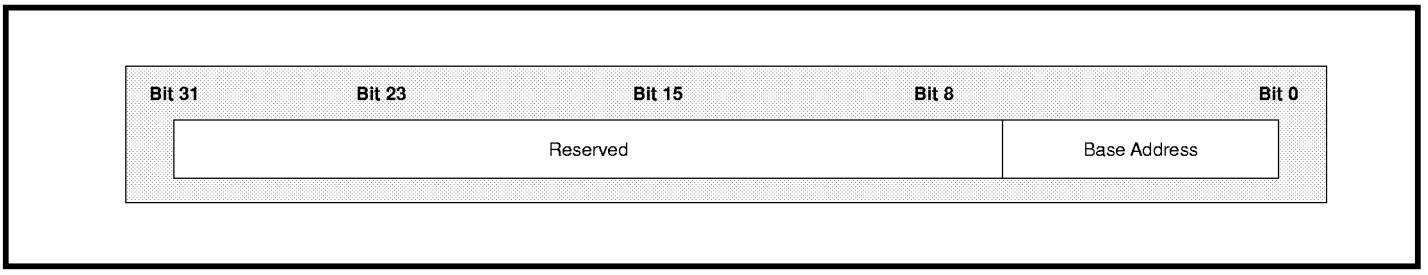
Bit	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	Reserved				Transmit Enable	Reserved (must be set to zero)		

This register is used to configure the Segmentation device. Register bits 6-0 can only be written while the “Device Reset” bit is a one.

Transmit Enable	If set to one, the Segmentation device will transmit queued data. If set to zero the Segmentation device will suspend transmission.
Device Reset	The Segmentation device is reset (except for portions of the register interface) when this bit is set to a one and left on for a minimum of 1µs. The bit must be set to zero to take the device out of reset. The other lower-numbered bits (6-0) in this register can only be changed while this bit is a one. NOTE: This bit does not affect any of the user-programmable registers (i.e. they are not reset or forced to an initial value as is done by a hardware reset thru MRst*).
Buffer Memory Parity Enable	When this bit is set Parity checking is enabled on the Buffer Memory and Descriptor Memory interfaces.
Connection Memory Parity Enable	When this bit is set Parity checking is enabled on the Connection memory interface.
Wide/Narrow Bus Mode	If set to zero, the device will operate in "wide" bus-width mode. In this mode the UTOPIA interface has a 16-bit wide data path and the Buffer Memory data bus is 64 bits wide. If set to zero the device will operate in the "narrow" bus-width mode whereby the UTOPIA interface has an 8-bit wide data path and the Buffer Memory data bus is 32 bits wide.
Host Buffer Size	The size of the buffers in Buffer Memory are set up as follows: b'00' -- 256 byte buffers b'01' -- 512 byte buffers b'10' -- 2048 byte buffers b'11' -- 4096 byte buffers

Descriptor Memory Base Address Register

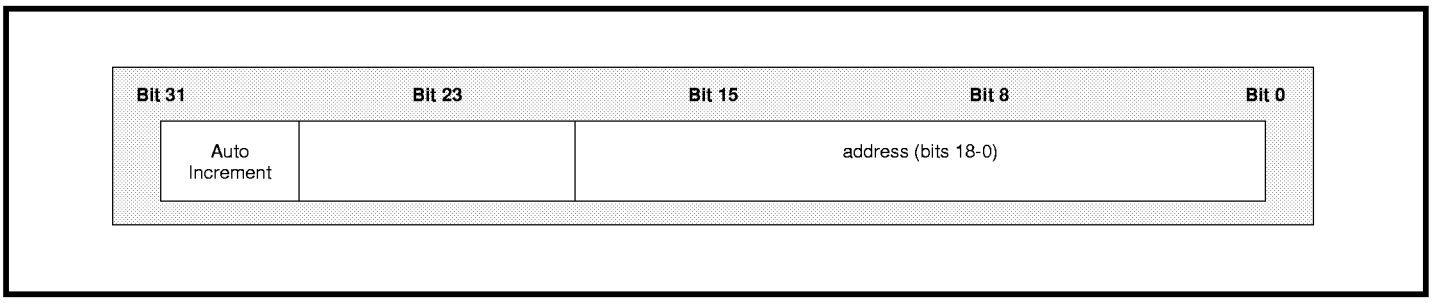
S_DMBASE **01h**



This register defines the high-order 6 bits of the 22-bit Descriptor Memory address generated by the SAR. The 6-bit value placed in this register (register bits 5-0) will be placed on the appropriate BmA/D pins during Descriptor Memory address cycles. If Descriptor Memory & Buffer Memory are separate (as required for 622Mbps operation) then this register would typically be set to zero if used at all (only 16 address bits are required to address all possible descriptors assuming the address space starts at zero). This register allows Buffer Memory & Descriptor Memory to be combined into one physical memory for 155Mbps operation.

Connection Memory Address Register

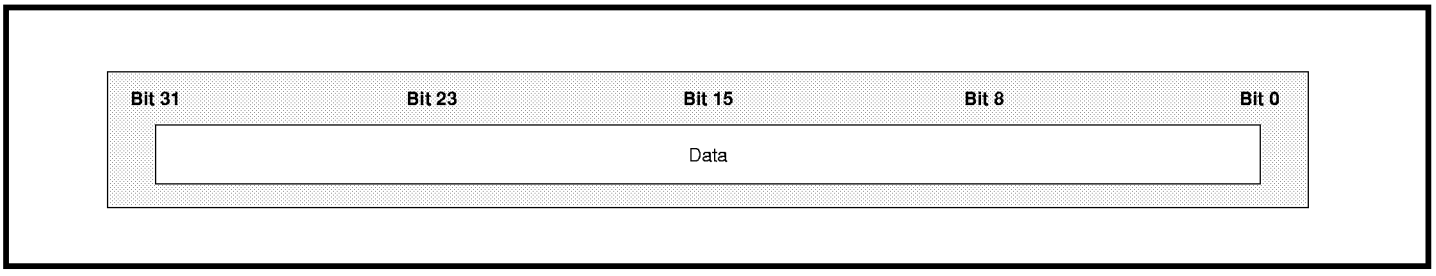
S_CMAR **02h**



This register is used to set addresses when accessing Connection Memory through the S_CMDR register. Bit 31, if set, causes this register to auto-increment each time the S_CMDR register is read or written. Bits 18-0 define the Connection Memory word address (where a word is 4 bytes).

Connection Memory Data Register

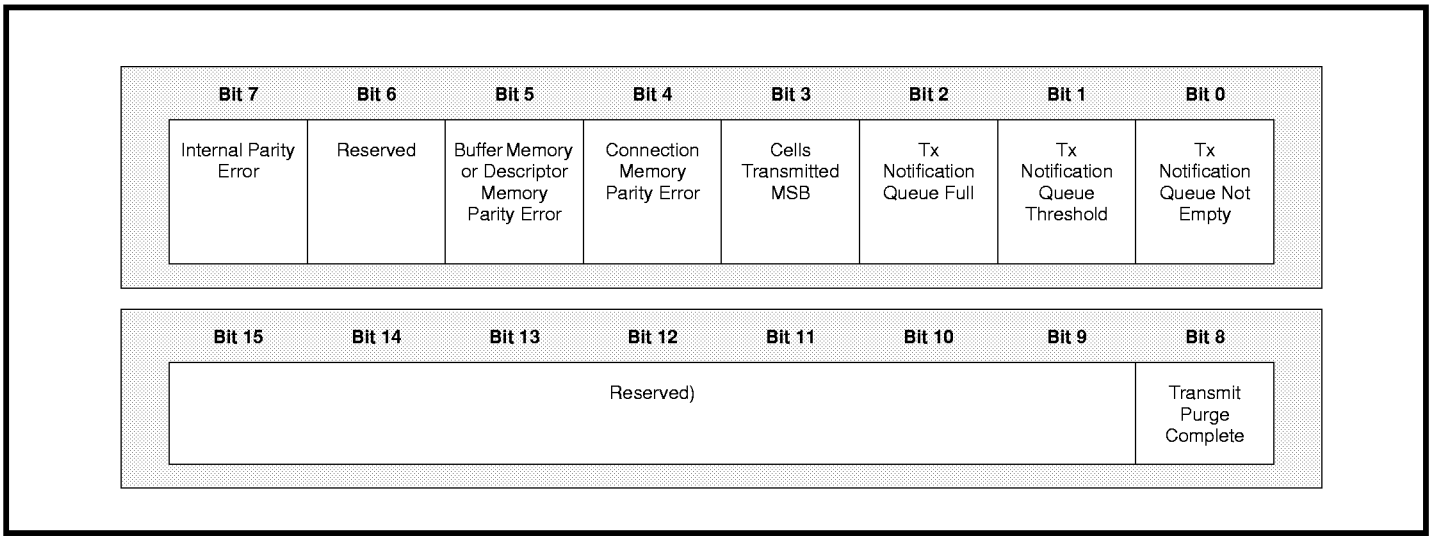
S_CMDR **03h**



This register contains four bytes read from, or written to, the Connection Memory. All Connection Memory accesses are word aligned to 4 bytes. The address used is taken from the S_CMAR register.

Interrupt Status Register

S_ISR **04h**



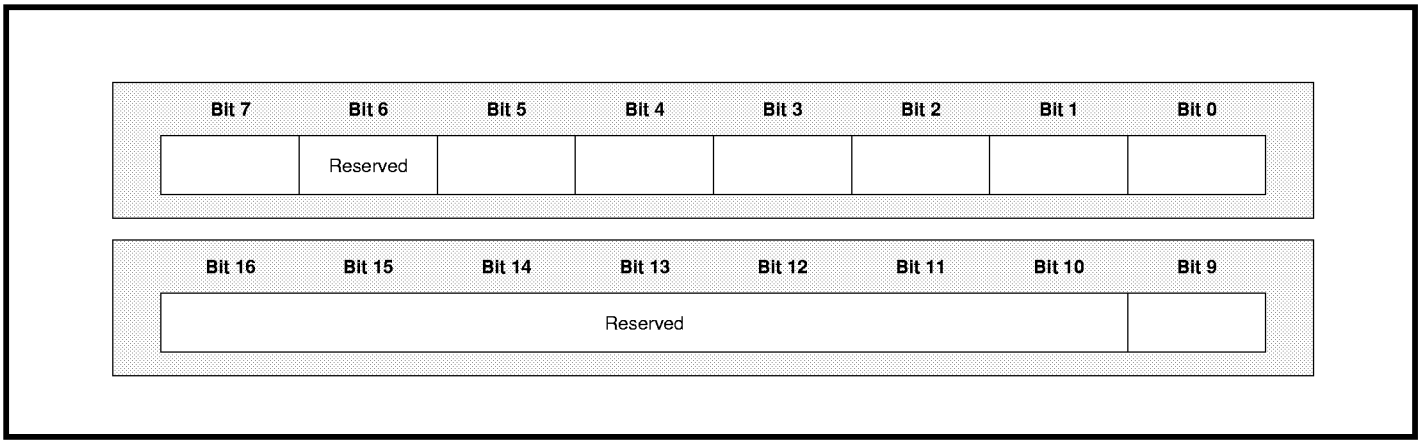
This register contains interrupt status information. Any bit read as "one" indicates that the condition has occurred. Writing to this register clears interrupt conditions. Each bit written as "one" clears the corresponding read-back bit unless the condition which caused the bit to become set is still present at write-back time (for example, if a " Tx Notification Queue Not Empty" condition is posted and the bit is written back as "one" it will not clear because the queue remains "not empty")

IMPORTANT: Parity error interrupt status bits are set regardless of whether the corresponding parity-enable bit is set in the S_MODE register. If parity is disabled in the S_MODE register then the corresponding interrupt bit should remain masked and the bit should be ignored (masked by software) when this register is read. The disabling of parity in the S_MODE register is still important (if one intends to not support parity) because it does cause the chip to internally ignore the error rather than treating it as fatal.

Transmit Purge Complete	If set to one, a previously initiated Transmit Purge (using the S_TXPUR register) has completed and the S_TXPUR register is now available for reuse.
Internal Parity Error	If set to one, an internal parity error has occurred in the Segmentation device. NOTE: The chip must be reset to clear this error if enabled.
Buffer Memory or Descriptor Memory Parity Error	A parity error has been detected at the Buffer Memory or Descriptor Memory interfaces.NOTE: The chip must be reset to clear this error if enabled.
Connection Memory Parity Error	A parity error has been detected on the Connection Memory interface. NOTE: The chip must be reset to clear this error if enabled.
Cells Transmitted MSB	The S_CELLT register has set its most significant bit (but will continue to count). Reading and clearing the S_CELLT register is recommended to avoid overflow if accuracy is desired (overflow of the S_CELLT register will not affect operation of the Segmentation chip).
Tx Notification Queue Full	The Transmit Complete Queue is full or overflowing.
Tx Notification Queue Threshold	The Transmit Complete Queue has exceeded it's notification threshold.
Tx Notification Queue Not Empty	If set to one, the Transmit Notification Queue is not empty.

Interrupt Mask Register

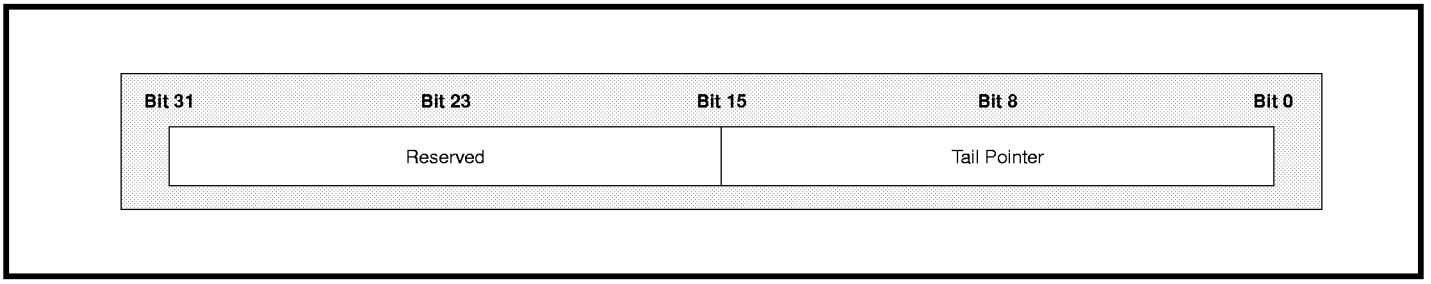
S_IMR **05h**



The S_IMR bit fields correspond to each bit in the S_ISR. Setting a bit to one in the S_IMR masks or prevents the corresponding bit in the S_ISR from generating an interrupt. Masked interrupt bits are still reported in the S_IMR even though they will not cause the interrupt pin to transition. NOTE: The Reassembly device will not report masked interrupts in the R_ISR unlike the Segmentation device.

Free-Pool Tail Register

S_FREET **06h**



Typical applications will initialize the buffer descriptors in Descriptor Memory so that a linked list of free buffers is created with a Head and Tail. This free pool is intended for use by all connections that transmit data. Normally the user maintains the head pointer and uses that to obtain buffers as needed for transmission. The Segmentation device automatically adds buffers onto the tail of this chain after completing the last DMA from each buffer previously queued.

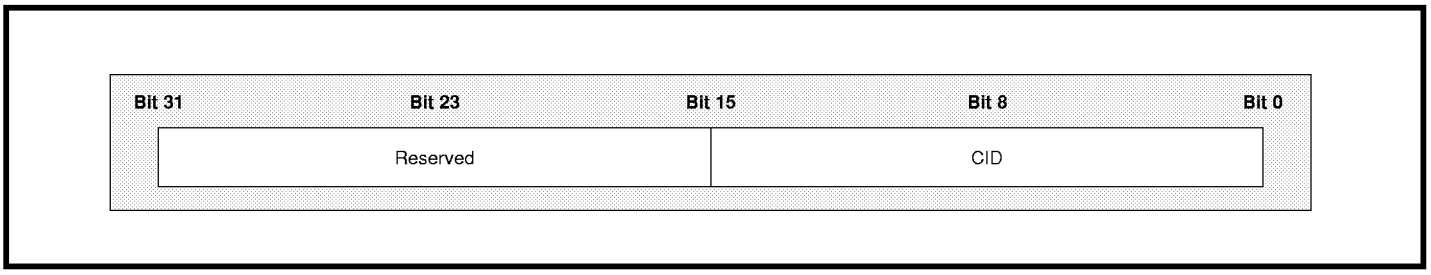
It is assumed that during system initialization the user writes the tail pointer of the list described above into the S_FREET register so that the Segmentation Device has a starting point to return freed buffers as it empties them. Once the "Transmit Enable" bit in the S_MODE register is turned on the user cannot write to this register, rather the device maintains an internal working register which points to the current tail of the chain. This internal register is not accessible to the user.

Since the Segmentation device uses the last buffer in the chain as the point to append newly freed buffers, the user is obliged to avoid acquiring the last buffer in the free pool (i.e. a "get_buffer" routine must consider the pool effectively empty when the head pointer locates a descriptor whose forward pointer is zero (end-of-chain).

The Segmentation device does not require the user to acquire buffers from the head of the list described above in the sense that no checking is done by the device as to the origin of buffers presented for transmission. Regardless of how the user manages free buffers the device will always return consumed buffers in the manner prescribed.

Transmit Request (CID) Register

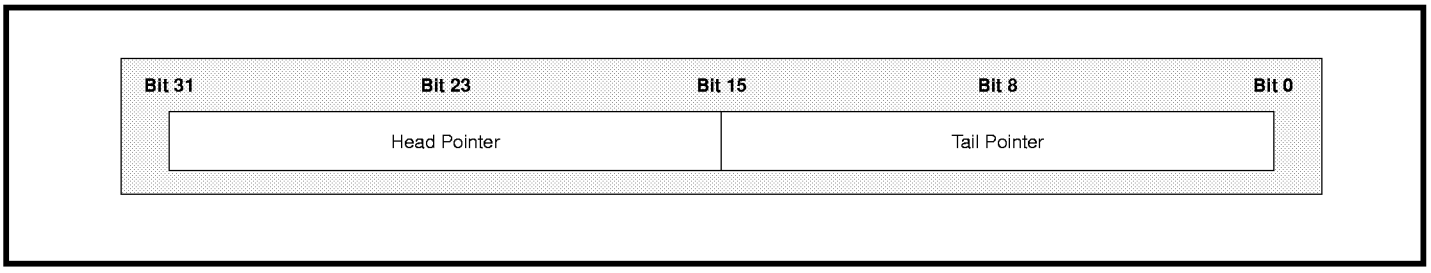
S_TXRCID **07h**



To queue a PDU for transmission two register accesses are made. One access is made to the S_TXRHT register to supply the head and tail pointers which describe the linked list of buffers holding the transmit data. The second access is to the S_TXRCID register which identifies the connection associated with the data. The CID value is an index into the Connection Table (in Connection Memory). The S_TXRHT register must be written first, prior to writing to the S_TXRCID register. The Segmentation device initiates the queuing of the PDU for transmission when S_TXRCID is written.

Transmit Request Head/Tail Register

S_TXRHT **08h**



To queue data for transmission the user must write the head and tail pointer (to a linked list of buffers in Buffer Memory) into this register. The actual queuing operation begins when the S_TXRCID register is written to identify the connection of the outgoing data.

Data queued which does not completely fill the last cell of a PDU is zero-padded. For AAL3/4 and AAL5 the padding is removed on the receiving side transparently. For all other aal types supported (raw, aal0 and aal1) the zero-padding is presented on receive.

For transmitting in RAW (cell) mode, the data format (shown as consecutive bytes) is as follows: pad, pad, H0, H1, H2, H3, pad, pad, payload0, payload1 ... payload47. In this mode only one ATM cell can placed in a buffer.

A PDU must begin on a buffer boundary unless the descriptor is setup to specify a positive offset. Multi-buffer PDU's must have all buffers filled except the last. The

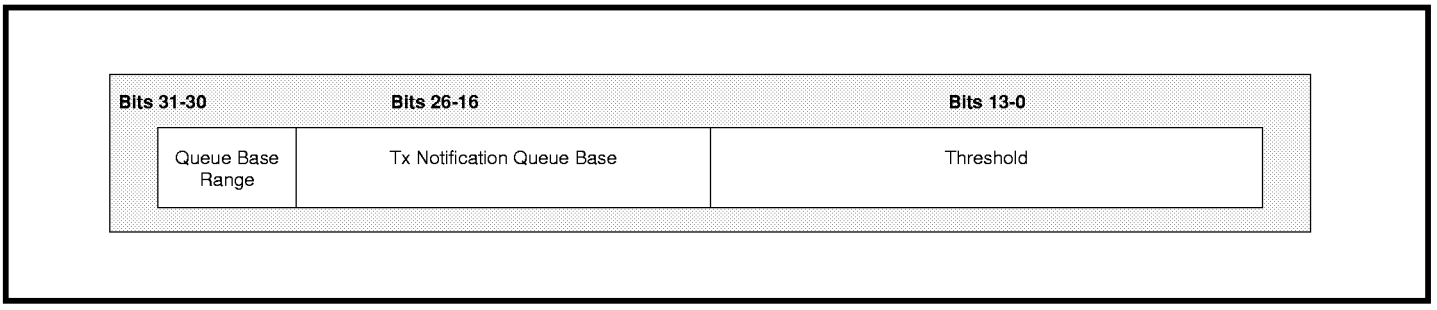
Segmentation device only examines the length field in the descriptor on the last buffer of a PDU, all other buffers are assumed to contain a data length corresponding to the "Host Buffer Size" parameter set in the S_MODE register.

IMPORTANT PERFORMANCE TIP: Multiple pdu's for the same connection can be queued with a single write to this register. The user must insure that the end-of-PDU bit is set in Descriptor Memory for each buffer containing the end of a pdu and the PDU's must be chained together. There is no chipset limitation to the number of PDU's queued in this manner (other than address-space).

Head Pointer	The pointer to the first buffer in a linked-list of buffers which contain data to be transmitted.
Tail Pointer	The pointer to the last buffer in a linked-list of buffers which contain data to be transmitted.

Transmit Notification Queue Control Register

S_TNQC **09h**

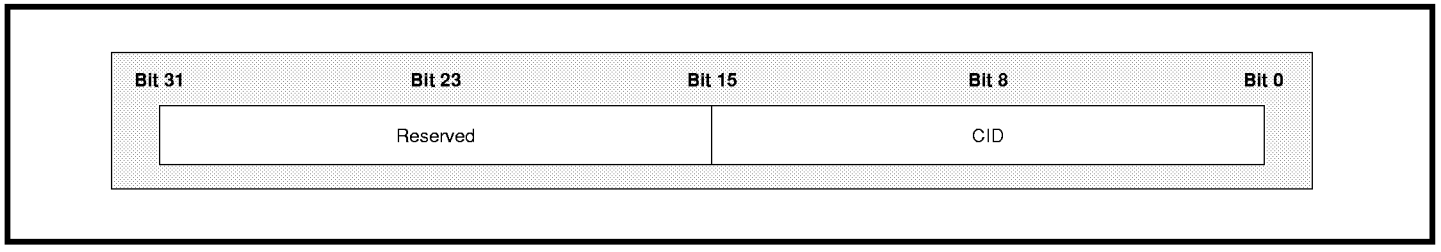


This register is used to specify where in Connection Memory the base of the Transmit Notification Queue will reside. The Segmentation device will completely manage this queue once the control information has been specified. The user is only obliged to determine the appropriate size and location of the queue and insure that no overlay occurs with other user-locatable data structures (e.g. Connection Table Entries).

Queue Base Range	This 2-bit field occupies bits 31-30 and contains an encoded value that defines the number of low order bits used from the "Tx Notification Queue Base" field to form a queue addresses. This base register is used to generate the high order bits when accessing the queue in Connection Memory. The options are: b'00' -- 5-bits from base register; queue size is 16k words (16k notifications) b'01' -- 7-bits from base register; queue size is 4k words (4k notifications) b'10' -- 9-bits from base register; queue size is 1K words (1K notifications) b'11' -- 11-bits from base register; queue size is 256 words (256 notifications)
Tx Notification Queue Base	This field specifies the high-order address bits into Connection Memory for notification queue access. The number of bits used in this field are specified from the preceding Queue Base Range field. The chosen base value should be aligned on the low portion of this field (e.g if an encoded base range of b'10' is selected then 9 bits from this field will be used to form the an address into the queue...and those 9 bits are aligned such that the 2 most significant bits of this field are ignored and the remaining 9 are used in forming an address).
Threshold	This field defines the number of active queue entries that will cause a special threshold-exceeded interrupt to be posted (see the Interrupt Status Register). If the interrupt is permanently masked then this field can be ignored.

Transmit Notification Register

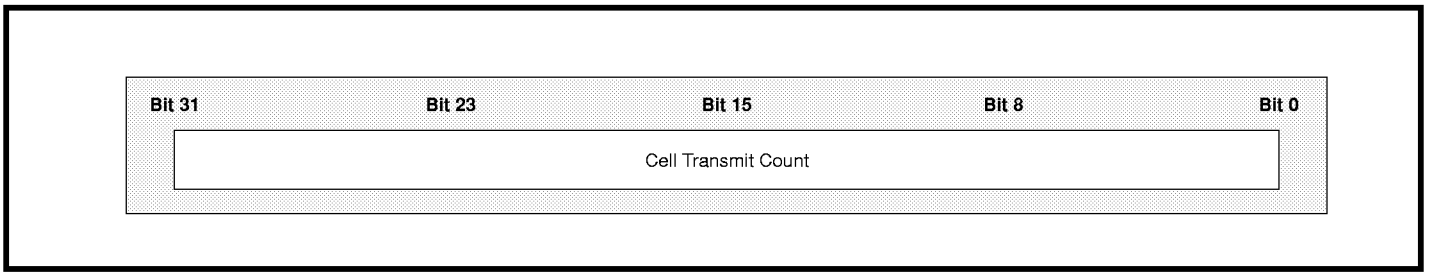
S_TXNCID **0ah**



When queued user data SDUs have been transmitted and the buffer(s) returned onto the free list, notifications are provided on a Transmit Notification Queue kept in Connection Memory. The system may access each element kept on the queue through the S_TXNCID register. A CID value of zero indicates that the queue is empty.

Cells Transmitted Register

S_CELLT **0dh**

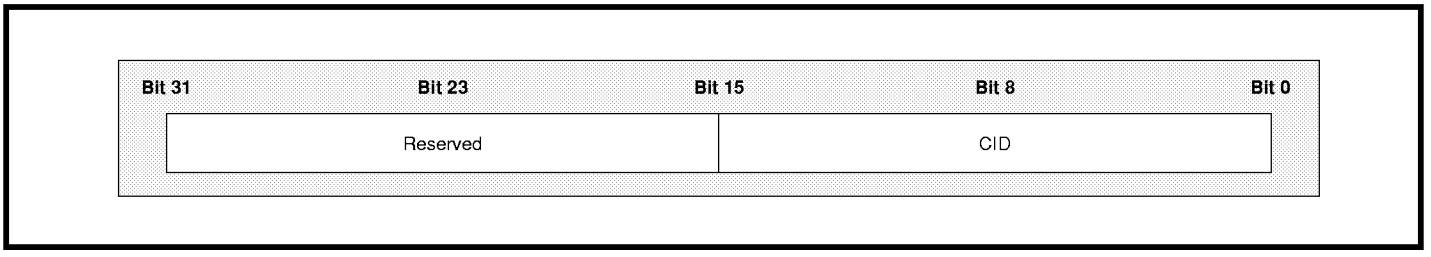


This register is a 32-bit counter that increments each time a cell is transmitted by the "S" device. The counter is unaffected each time it is read. The counter is cleared to zero by attempting to write any value to it.

An interrupt is generated when, and only when, a carry occurs into the most significant bit (which is approximately after receiving 2 billion cells). The external system should respond to the interrupt by reading this register. The register wraps to zero if allowed to overflow.

Transmit Purge Register

S_TXPUR **0eh**

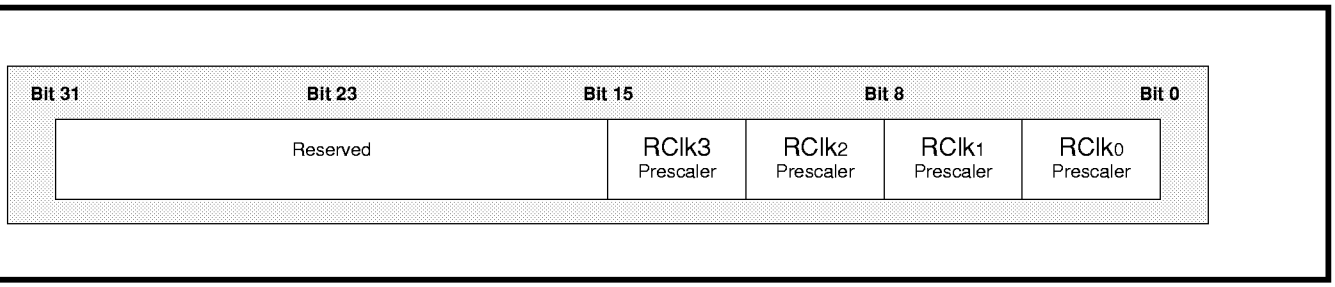


This register is used to abort all transmission of data queued to a selected connection. The aborted buffers are freed as if it had been transmitted normally and a notification is posted for each aborted PDU. This command can be used when a connection is to be terminated, but data is (or may be) queued to it.

The user must provide the CID of the connection to be purged. All PDU(s) queued to the connection will be purged. If a connection is purged that is not being completely shut down, then the user should wait until all previously issued PDUs have been purged before queuing new ones (i.e. wait until the command completes to insure that PDUs queued after the purge are not also aborted).

Rate Generator Prescaler Register

S_RGP **0fh**

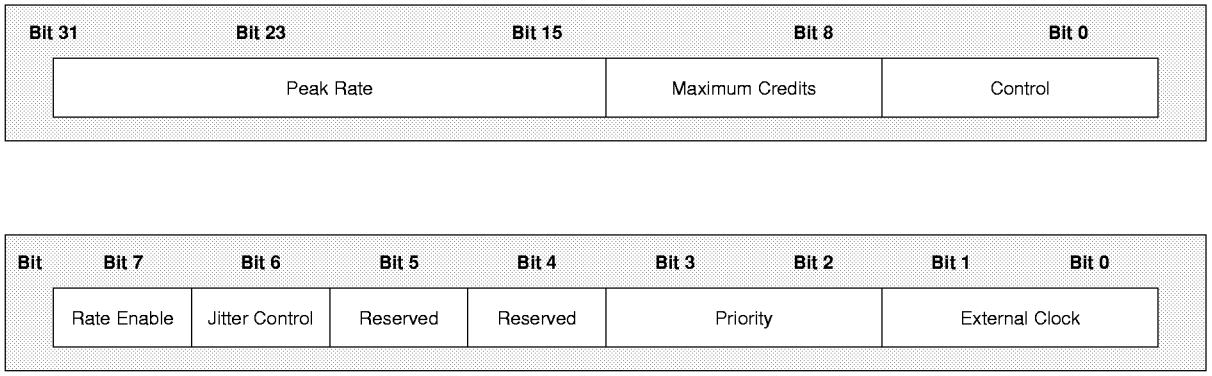


This register configures the 4-bit values used with each prescaler associated with each Rate Clock input pin, RClk[3:0]. Each prescaler divide value is a 2n, where n is four bits.

RClk3 Prescaler	Prescaler value for external clock RClk3
RClk2 Prescaler	Prescaler value for external clock RClk2
RClk1 Prescaler	Prescaler value for external clock RClk1
RClk0 Prescaler	Prescaler value for external clock RClk0

Rate Generator Control Register

S_RGC*i* **10h + <generator #>**

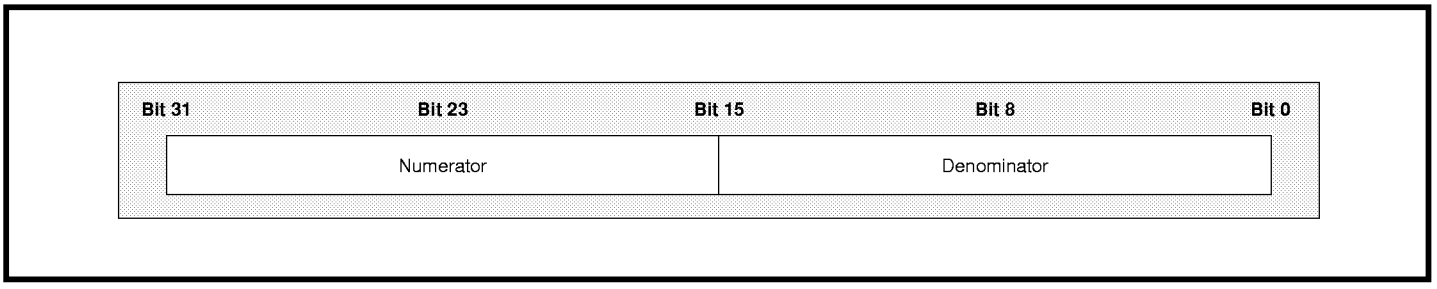


This register is used to configure and enable a Rate Generator. No data will be transmitted until using a particular rate generator until it has been configured.

Peak Rate	The Peak Cell Rate is minimum number of cell-times between consecutive cell transmissions.
Maximum Credits	The number of credits that can be accumulated on this rate generator when preempted by higher priority traffic. The valid values are from 0 to 255. If this field is set to zero, this rate generator will never exceed its configured rate, however, the traffic flow may be disrupted when pre-emptied.
Rate Enable	When set to one the Rate Generator is enabled. If set to zero disabled and although traffic may be queued it will not be transmitted.
Jitter Control	If set to zero, connections that have no data queued will be automatically added/ removed from the linked list of connections depending on whether data is queued or not. This eliminates wasted bandwidth on connections that have no data queued for transmission, but may cause some variation of cell spacing (jitter) to the remaining queued connections. If the option is set to one, connections that have no data queued will not be removed from the linked list of connections regardless of whether data is queued or not. With this option connections which have no data queued consume a cell-time anyway -- eliminating that source of jitter (but not utilizing potential bandwidth).
Priority	Rate generator priority. Valid values are 0, 1 or 2 with zero the highest priority.
External Clock	This field indicates which external clock pin, RClk[3:0] will be used to drive this rate generator. b'00' = RClk0; b'01' = RClk1; b'10' = RClk2; b'11' = RClk3

Rate Generator Divider Register

S_RGD*i* **20h + <generator #>**



This register provides the values to fractionally divide downward the rate clock for this rate generator by a precise amount. The clock which is divided is the pre-scaled external clock chosen for this rate generator (as selected by the corresponding Rate Shaping Configuration register). The rate of this generator is defined as R_i = (Numerator/Denominator) * Rclk/2prescaler where Numerator > Denominator.

Numerator	The 12-bit numerator value (high 4 bits are reserved). If set to zero the rate generator will be disabled (however, the recommended way to disable a rate generator is thru the "Rate Enable" bit in the Rate Generator Control Register).
Denominator	The 12-bit denominator value (high 4 bits are reserved). Do not set to zero.

Reassembly Device Registers

The Reassembly registers are set to the following values following a hardware (MRst*) reset.:

R_ISR = 0
R_IMR = all ones
R_CMAR = 0
R_ATR = 0
R_MODE = 0
R_DMBASE = 0
R_RQC0 = 0
R_RQC1 = 0
R_RQC2 = 0
R_RQC3 = 0
R_FREEH = 0
R_RXPUR = 0

Mode Register

R_MODE **00h**

Bit	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	Device Reset	UTOPIA Parity Enable	Buffer Memory Parity Enable	Connection Memory Parity Enable	Wide/Narrow Bus mode	Rx Length Mode	Host Buffer Size	

Bit	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	Receive Enable	Congestion Enable	F5 OAM Enable	Reserved PTI Enable	ATT Fail Enable	Reserved (must be set to zero)		

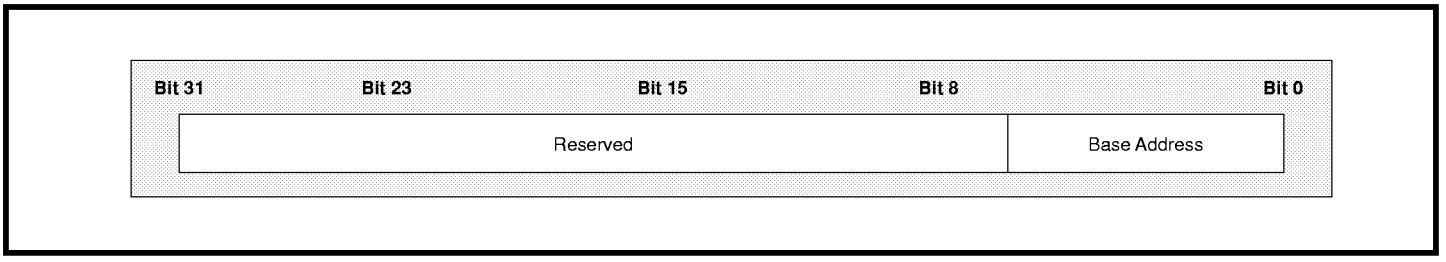
Receive Enable	If this bit is one, received cells will be processed normally and data will be transferred to external memory. If this bit is zero, all incoming cells will be discarded by the Reassembly device. During reset this bit is forced to zero and must be explicitly re-enabled. This allows the processor to setup the receive environment before allowing incoming cells to be processed. This bit takes precedence over other receive options in this register.
Congestion Enable	If this bit is one, received cells that have a PTI field of b'010' or b'011, (indicating congestion notification) will cause notifications to be placed on the Receive Management Queue. If this bit is zero, notifications for congestion will not be created.
F5 OAM Enable	If this bit is one, received cells that have a PTI field of b'100' or b'101, (indicating an F5 OAM cell) will be placed in the Receive Management Queue (regardless of the Virtual Circuit they are received on). If this bit is zero, these cells will be discarded.
Reserved PTI Enable	If this bit is one, received cells that have a PTI field of b'110 or b'111 (reserved PTI in UNI 3.0) will be placed in the Receive Management Queue (regardless of the Virtual Circuit they are received on). If this bit is zero, these cells will be discarded.
ATT Fail Enable	If this bit is one, received cells that fail address translation will be placed in the Receive Exception Queue. If this bit is zero, these cells will be discarded.
Device Reset	The Reassembly device is reset (except for portions of the register interface) when this bit is set to a one and left on for a minimum of 1us. The bit must be set to zero to take the device out of reset. The other lower-numbered bits (6-0) in this register can only be written while this bit is a one. NOTE: This bit does not affect any of the user-programmable registers (i.e. they are not reset or forced to an initial value as is done by a hardware reset thru MRst*).
UTOPIA Parity Enable	When this bit is one parity checking at the UTOPIA interface is enabled.
Buffer Memory Parity Enable	When this bit is set Parity checking is enabled at the Buffer Memory and Descriptor Memory interfaces.
Connection Memory Parity Enable	When this bit is set Parity checking is enabled on the Connection memory interface.
Wide/Narrow Bus Mode	If set to zero, the device will operate in "wide" bus-width mode. In this mode the UTOPIA interface has a 16-bit wide data path and the Buffer Memory data bus is 64 bits wide. If set to zero the device will operate in the "narrow" bus-width mode whereby the UTOPIA interface has an 8-bit wide data path and the Buffer Memory data bus is 32-bits wide.
Rx Length Mode	If set to zero, receive notifications will always require two register accesses and the PDU status/length will be presented in the second register access (the default). If set to one, the status/length will not be presented unless an error occurred and receive notifications will therefore normally require only one register access (which may be important for very-small-pdu performance). NOTE: If an error occurs 2 register accesses are required regardless of the setting of this option (since the error code is presented in the 2nd access). In that case the Receive Notification CID/Head register contains a bit which identifies whether a second read is required for the status/length.

Host Buffer Size — The size of the buffers in Buffer Memory are set up as follows:

b'00' -- 256 byte buffers
b'01' -- 512 byte buffers
b'10' -- 2048 byte buffers
b'11' -- 4096 byte buffers

Descriptor Memory Base Address Register

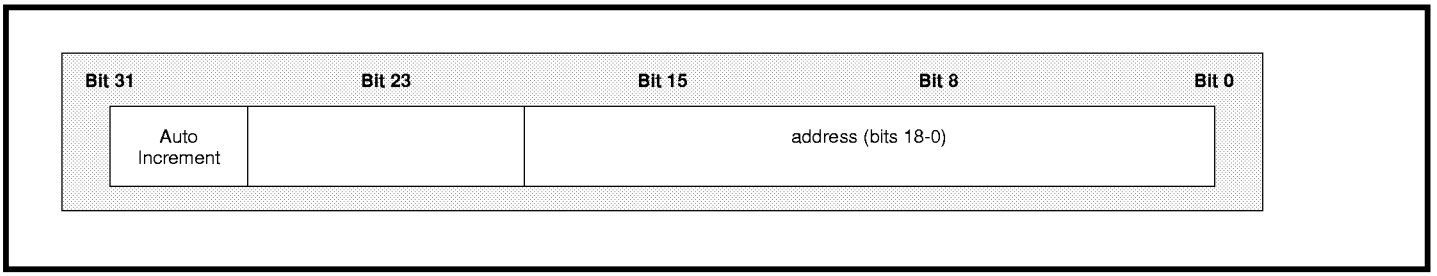
R_DMBASE **01h**



This register defines the high-order 6 bits of the 22-bit Descriptor Memory address generated by the SAR. The 6-bit value placed in this register (register bits 5-0) will be placed on the appropriate BmA/D pins during Descriptor Memory address cycles. If Descriptor Memory & Buffer Memory are separate (as required for 622Mbps operation) then this register would typically be set to zero if used at all (only 16 address bits are required to address all possible descriptors assuming the address space starts at zero). This register allows Buffer Memory & Descriptor Memory to be combined into one physical memory for 155Mbps operation.

Connection Memory Address Register

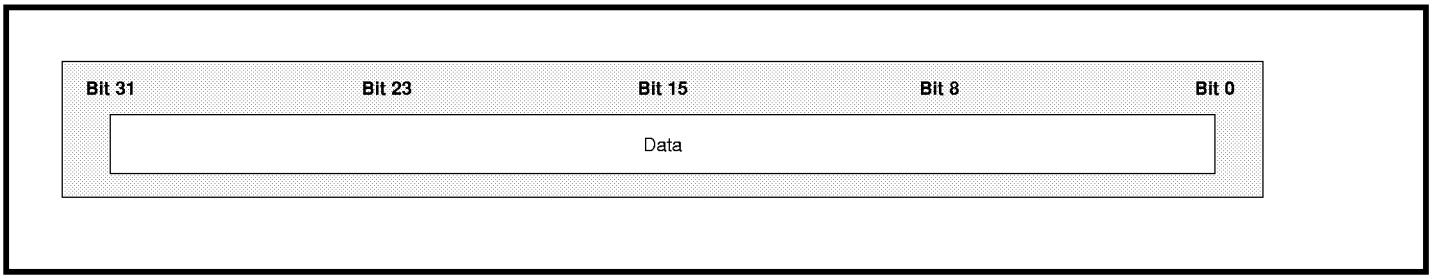
R_CMAR **02h**



This register is used to set addresses when accessing Connection Memory through the R_CMDR register. Bit 31, if set, causes this register to auto-increment each time the R_CMDR register is read or written. Bits 18-0 define the Connection Memory word address (where a word is 4 bytes).

Connection Memory Data Register

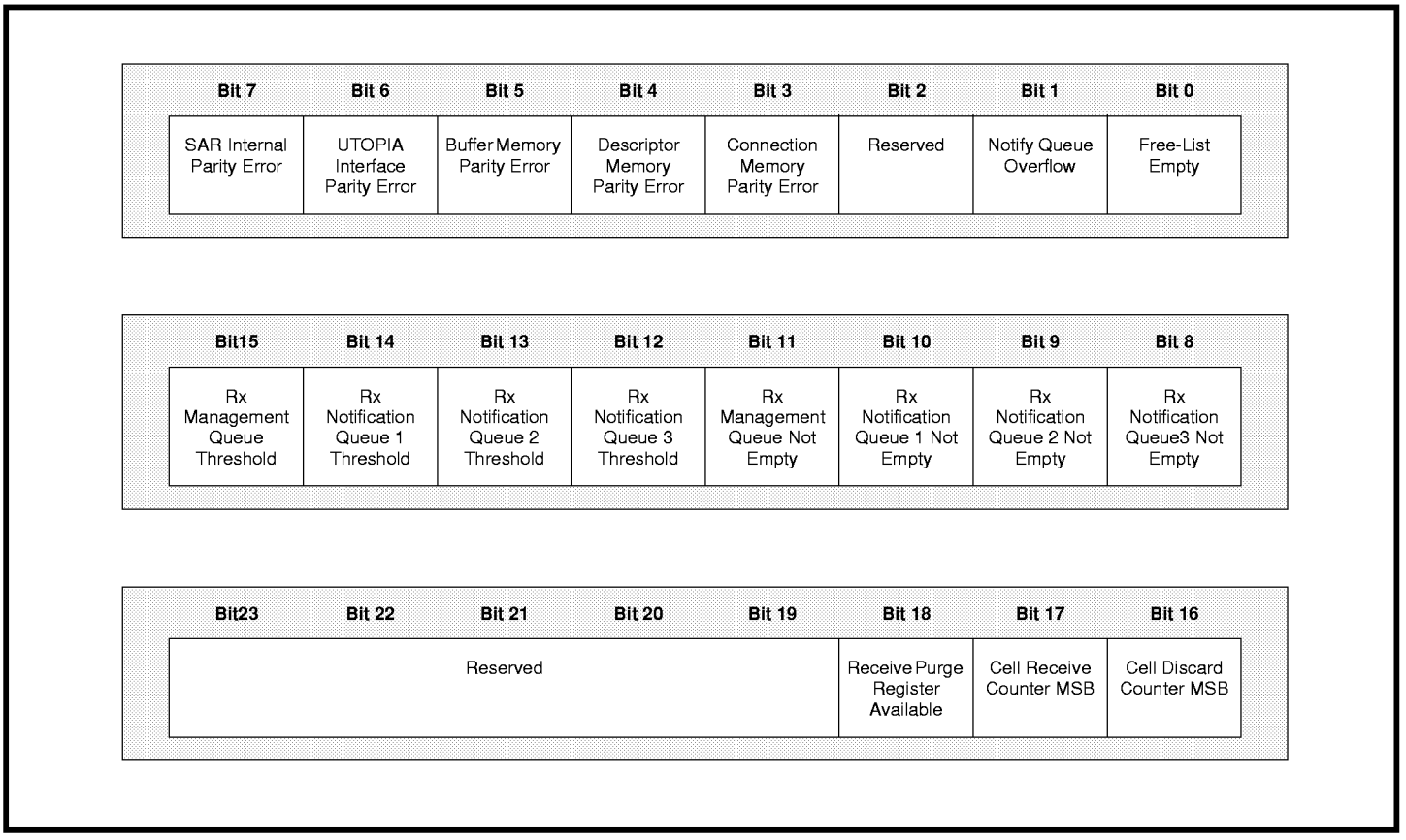
R_CMDR **03h**



This register contains four bytes read from, or written to, the Connection Memory. All Connection Memory accesses are word aligned to 4 bytes. The address used is taken from the R_CMAR register.

Interrupt Status Register

R_ISR **04h**



This register contains interrupt status information. Any bit read as “one” indicates that the condition has occurred. Writing to this register clears interrupt conditions. Each bit written as “one” clears the corresponding read-back bit unless the condition

which caused the bit to become set is still present at write-back time (for example, if a " Rx Notification Queue Not Empty" condition is posted and the bit is written back as "one" it will not clear because the queue remains "not empty").

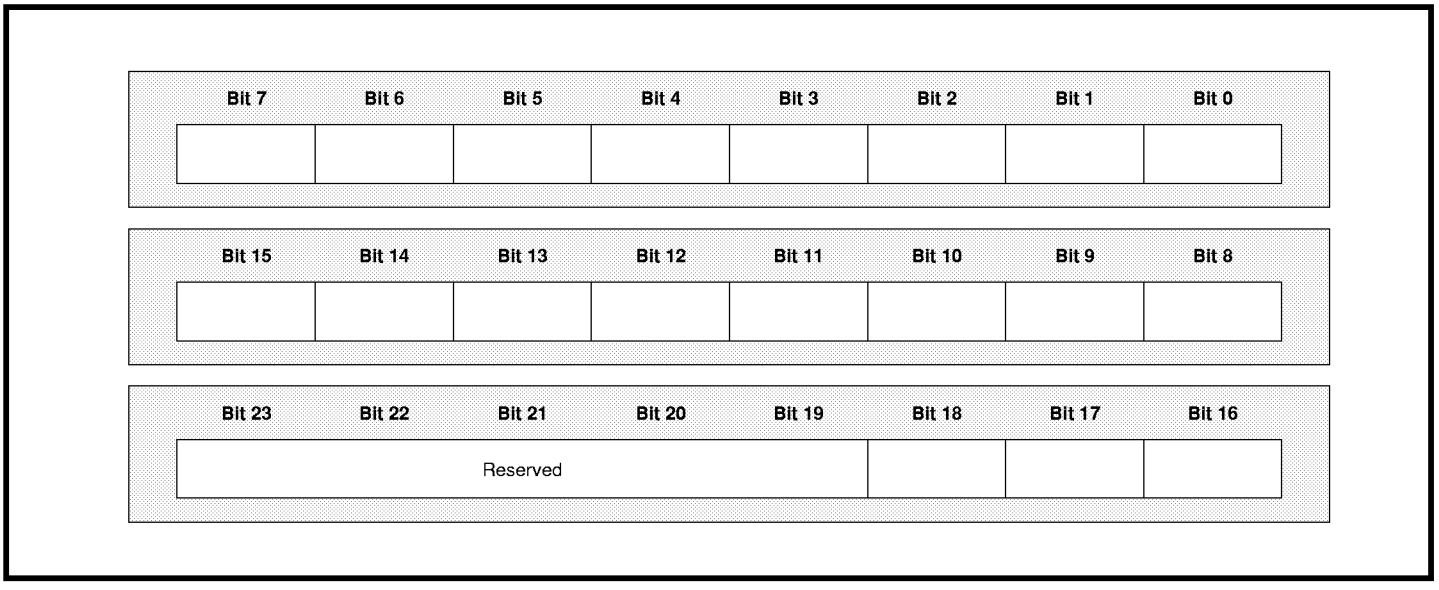
Note: Parity error interrupt bits will be reported if unmasked even if they have been disabled in the R_MODE register. The latter disable only prevents the device from internally treating as a fatal error. Note that the Segmentation device is slightly different in how it handles this situation (see the S_ISR register).

Receive Purge Register Available	If set to one, a previously issued Receive Purge operation has completed to the extent that the Receive Purge register is available for another command. No interrupt is provided to indicate when the command actually completes and in most cases it may appear synchronized to this bit (however in certain unusual cases a Receive Purge can be delayed since it waits for an idle cell-time to perform its work).
Cell Receive Counter	The R_CELLR register has set its most significant bit (but will continue to count). This bit is a reflection of the counter's most significant bit and is cleared by reading the R_CELLR register to extract the current value and cause it to auto-clear
Cell Discard Counter	If set to one, the R_CELLD register has set its most significant bit (but will continue to count). This bit is a reflection of the counter's most significant bit and is cleared by reading the R_CELLD register to extract the current value and cause it to auto-clear
Rx Management Queue Threshold	If set to one, the Receive Management Queue has reached or exceeded it's programmable notification threshold.
Rx Notification Queue 1 Threshold	If set to one, the Receive Notification Queue 1 has reached or exceeded it's programmable notification threshold.
Rx Notification Queue 2 Threshold	If set to one, the Receive Notification Queue 2 has reached or exceeded it's programmable notification threshold.
Rx Notification Queue 3 Threshold	If set to one, the Receive Notification Queue 3 has reached or exceeded it's programmable notification threshold.
Rx Management Queue Not Empty	If set to one, the Receive Management Queue is not empty.
Rx Notification Queue 1 Not Empty	If set to one, the Receive Notification Queue 1 is not empty.
Rx Notification Queue2 Not Empty	If set to one, the Receive Notification Queue2 is not empty.
Rx Notification Queue3 Not Empty	If set to one, the Receive Notification Queue3 is not empty.

SAR Parity Error	If set to one, an internal parity error has occurred within the SAR. NOTE: This error, if enabled, can only be cleared by reset.
UTOPIA Interface Parity Error	If set to one, a parity error has been detected at the PHY interface. NOTE: This error, if enabled, can only be cleared by reset.
Buffer Memory Parity Error	If set to one, a parity error has been detected when reading from Buffer Memory. NOTE: This error, if enabled, can only be cleared by reset.
Descriptor Memory Parity Error	If set to one, a parity error has been detected when reading from Descriptor Memory. NOTE: This error, if enabled, can only be cleared by reset.
Connection Memory Parity Error	If set to one, a parity error has been detected when reading from Connection Memory. NOTE: This error, if enabled, can only be cleared by reset.
Notification Queue Overflow	If set to one, one or more of the notification queues has overflowed. This condition should not normally be allowed to occur by timely handling of notifications. Buffer & data loss will occur on connections that attempt to use the overflowed queue. For typical implementations it is recommended that this condition be treated as fatal.
Free-List Empty	The free list has been consumed by the Reassembly device. The handling of the "Receive Enable" bit is the same as in the preceding field. Buffers should be returned to the free-list before re-enabling.

Interrupt Mask Register

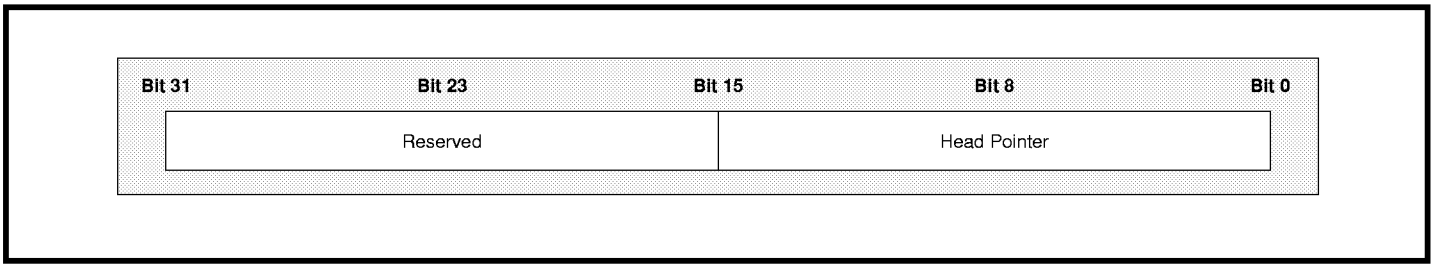
R_IMR **05h**



The R_IMR bits correspond to each bit in the Interrupt Status Register. Setting a bit to one in the R_IMR prevents the bit from being reported in the R_ISR and from generating a transition on the interrupt pin. Note that the S_ISR handles masked interrupts slightly different.

Free-Pool Head Register

R_FREEH **06h**

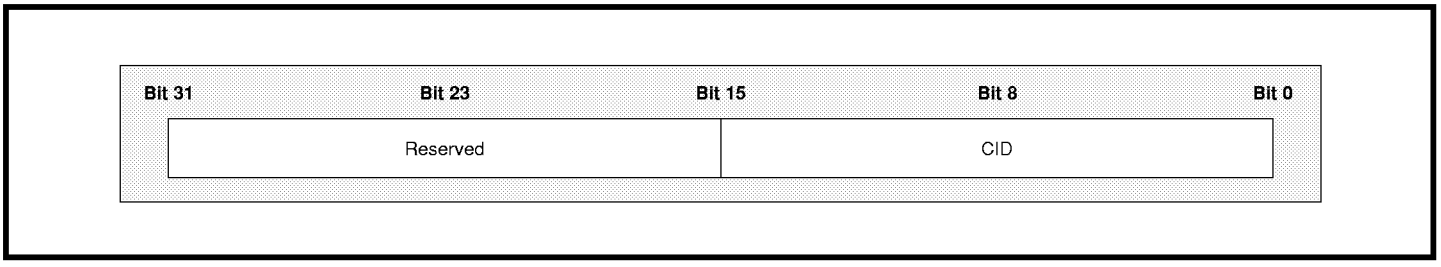


System initialization of the buffer descriptor entries in the Descriptor Memory results in a linked list with a Head and Tail. The Head Pointer value is written via the R_FREEH register so that the Reassembly device can obtain free buffers as required when receiving data. The Local Processor maintains the tail of the buffer chain and adds consumed buffers onto the tail.

This register is written only when the "Rx Enable" bit is off in the R_MCR register. When the "Rx Enable" bit is on, reading the register will provide an indication of where the Reassembly device is in the chain.

Receive Purge Register

R_RXPUR **07h**



This register allows a connection (identified by CID) to be purged of accumulated data even if that data is not complete from an AAL perspective. The purged data will be posted through the normal mechanism (e.g. added to the Receive Notification queue) but may be a partial PDU if issued to a connection that is AAL3/4 or AAL5.

This command could be used as a normal means of extracting data from certain types of Virtual Circuits (but not AAL3/4 or AAL5), and in other cases may be used only as a means of exception processing. For example, with out-of-band signalling a connection could be shut down in a fashion where a partial PDU has been received. In that case it is important issue the purge to reclaim the consumed buffer(s).

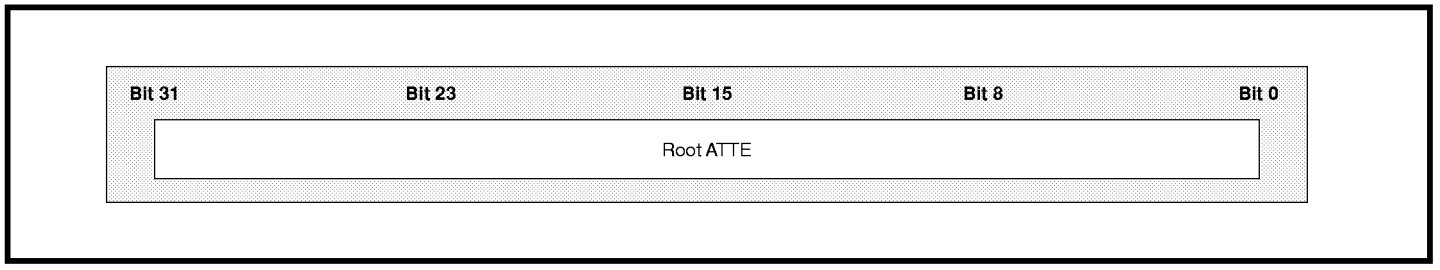
Virtual circuits that use AAL mechanisms to determine "PDU complete" may avoid this command for normal operation.

When the device is reset this register is set to zero. When a non-zero value is written by the external system the command initiates in the Reassembly device and the register cycle is completed. When the command later completes (which could take a few microseconds or more) this register is cleared to zero and a maskable

interrupt is posted to the R_ISR. Attempts to write to the register while non-zero will be ignored. The system should insure the register is zero before initiating a new command (either by reading and checking for zero, or insuring that a completion interrupt was posted since the last command was issued).

Address Translation Register

R_ATR **08h**



This register contains the "root" Address Translation Table Entry (ATTE). This register (in conjunction with the remainder of the Address Translation Table in Connection Memory) allows the Reassembly device to perform a mapping of the ATM address in an incoming ATM cell to a Connection ID (CID). The CID is then used to locate the context information in Connection Memory to be used in handling the cell.

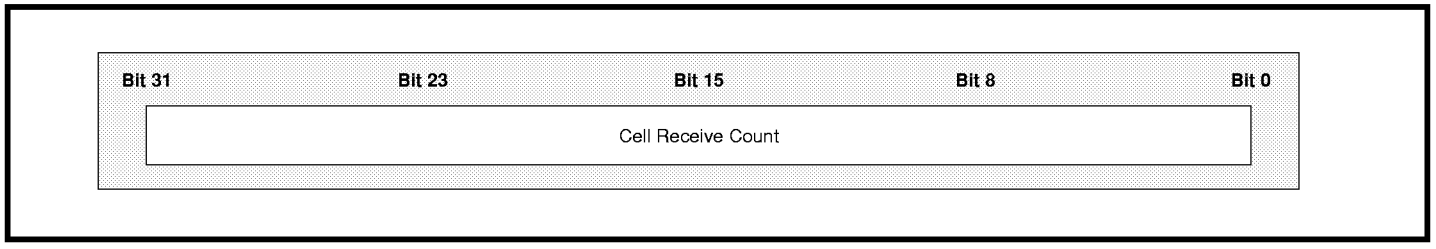
The R_ATR should be initialized prior to setting the Receive Enable bit in the R_MODE register so that incoming cells are handled properly.

The format of this register is the same as all Address Translation Table Entries as defined in Chapter 5. The only difference is that the entry loaded into this register is the "root" entry of the address translation data structure (i.e. it points to the VPI table in Connection Memory). See Chapter 5 for more details.

Note: This register cannot contain a leaf address-translation entry (i.e. it must point to a VPI table in Connection Memory).

Cells Received Register

R_CELLR **09h**



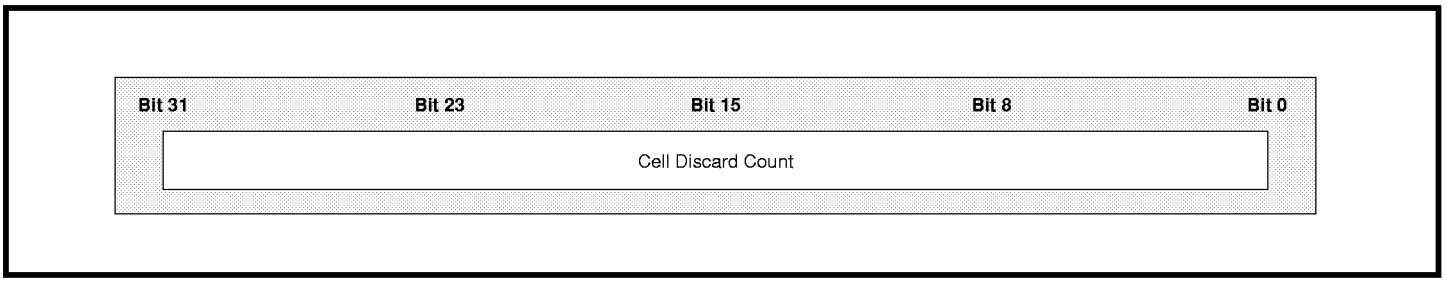
This register is a 32-bit counter that increments each time a cell is received by the Reassembly device unless it must immediately be discarded due to a parity error at the UTOPIA interface. The cell payload is not checked prior to incrementing this counter (this is handled at a point further down the data path by the AAL processing functions).

The counter is unaffected each time it is read. The counter is cleared to zero by attempting to write any value to it.

An interrupt is generated when, and only when, a carry occurs into the most significant bit (which is approximately after receiving 2 billion cells). The external system should respond to the interrupt by reading this register. The register wraps to zero if allowed to overflow.

Cells Discarded Register

R_CELLD **0ah**

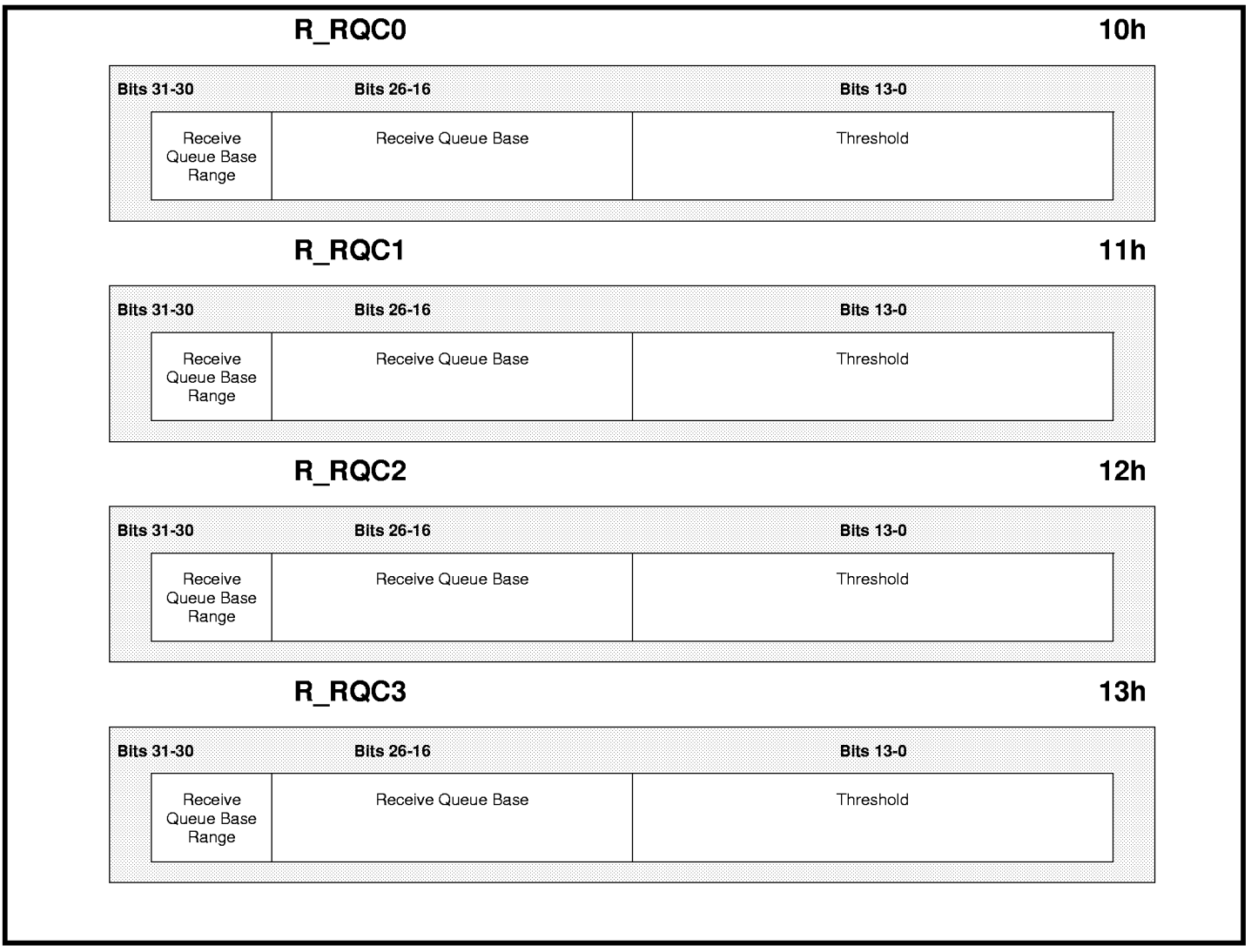


This register is a counter that increments each time a cell is received and discarded by the "R" device due to a parity error at the UTOPIA interface or an address translation error. The cell payload is not checked prior to incrementing this counter (this is handled at a point further down the data path by the AAL processing functions).

The counter is unaffected each time it is read. The counter is cleared to zero by attempting to write any value to it.

An interrupt is generated when, and only when, a carry occurs into the most significant bit (which is approximately after discarding 2 billion cells). The external system should respond to the interrupt by reading this register. The register wraps to zero if allowed to overflow.

Receive Queue Control Registers



These registers are used to specify an address in Connection Memory where the base of a Receive Notification queue will reside. The Reassembly device will completely manage the notification entries within this range. The user is obliged to determine the appropriate size and location of each notification queue and insure that no overlay occurs with other data structures.

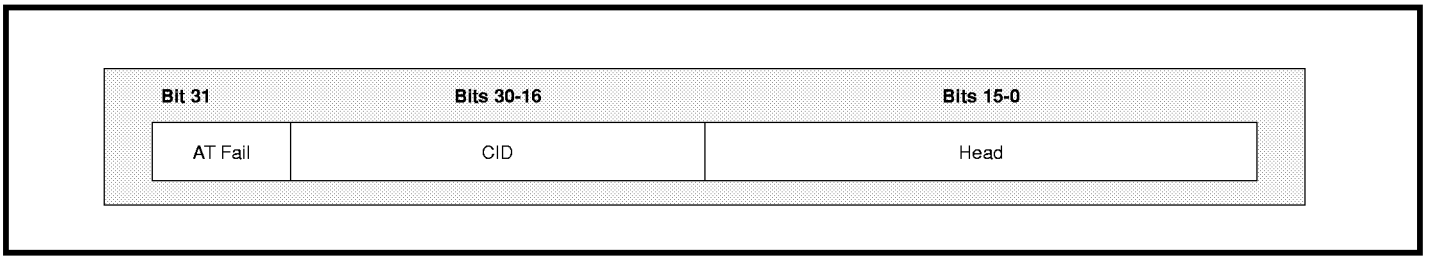
Each notification queue is defined by a Receive Queue Base register and a Receive Queue Size register.

The R_RQC0 defines the base of the Receive Management Queue. R_RQC1 defines the base of the Receive Notification Queue 1 and so on.

Receive Queue Base Range	This 2-bit field occupies bits 31-30 and contains an encoded value that defines the number of low order bits used in the Receive Queue Base register to form a queue addresses. This base register is used to generate the high order bits when accessing a queue in Connection Memory. The options are: b'00' -- 5-bits from base register; queue size is 16k words (8k notifications) b'01' -- 7-bits from base register; queue size is 4k words (2k notifications) b'10' -- 9-bits from base register; queue size is 1k words (512 notifications) b'11' -- 11-bits from base register; queue size is 256 words (128 notifications)
Receive Queue Base	The high-order address bits to a Connection Memory notification queue are generated from this field. The number of low-order bits used in this field are specified from the Receive Queue Base Range field.
Receive Queue Notification Threshold	This field defines the number of active queue entries that will cause a special threshold-exceeded interrupt to be posted (see the Interrupt Status Register). If the interrupt is permanently masked then this field can be ignored. The 3 high order bits of this field are reserved.

Receive Management CID/Head Register

R_RMCID0 **14h**



This register contains the Connection Identifier (Bits 30-16) of a received PDU in Buffer Memory and a Head pointer to the first of one or more buffers that comprise the PDU.

Bit 31 "AT Fail", if set, indicates the notification is for a cell that has failed address translation (see chapter 4 in the Mode Register section for enabling/disabling this option). In this case the CID field should be ignored (since the cell could not be associated with a known connection). If bit 31 is zero, them the notification is for a cell that has passed address translation, but has been directed to this queue based on the chosen programmable options.

If the CID is non-zero and the Head pointer is zero, then this is an "EFCI notification" (EFCI-bit set in ATM cell header) as opposed to the presentation of cell data. The CID field identifies the connection on which the EFCI was seen. The report will only occur if the "Congestion Enable" bit is turned on in the Mode Register (see Chapter 4).

Note: Since the "EFCI notification" is not posted along with data the SAR uses "zero pointer" notation to indicate EFCI (as opposed to assigning a bit and reducing an address space). If enabled the SAR will report all cells with their EFCI bits set as long as the cell is not the end-of-PDU cell. If the cell is an end-of-pdu cell the SAR will attempt to generate both a normal receive-data notification and an "EFCI notification" but will ignore the EFCI in some cases of back-to-back-one-cell-pdu's due to bandwidth limitations. Data notifications, however, are never lost regardless of EFCI presence.

Data presented thru this interface is in "Raw AAL" mode (i.e. unedited ATM cells) in the following consecutive-byte format: H0, H1, H2, H3, H4, pad, payload0, payload1...payload47. The length of each cell presented in this manner (including pad) is 54 decimal. Note also that the H4 byte is framer-dependent. If the framer passes the byte to the SAR as it was received from the physical layer then it is the HEC byte containing the CRC on the ATM cell header.

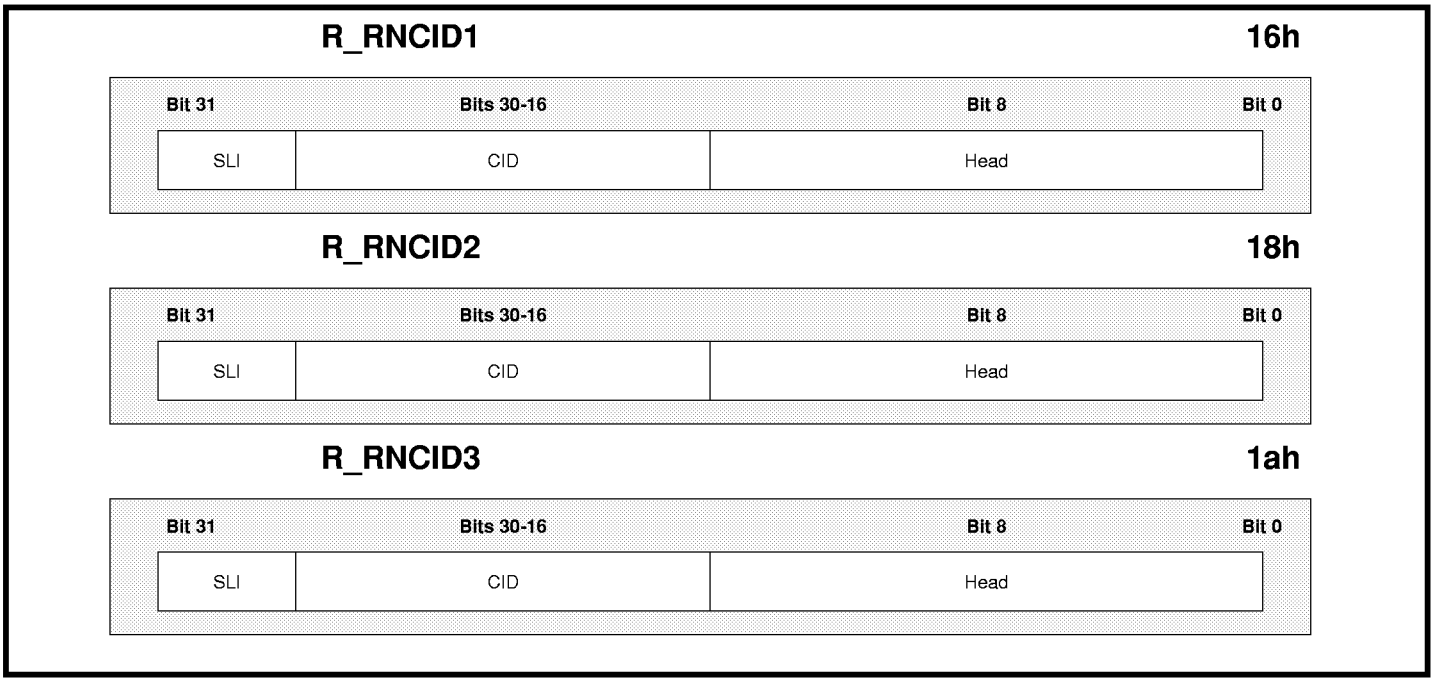
This queue contains status or cells that either are explicitly assigned to it on a VC basis or that are directed to it based on the options selected in the Receive Mode Register. This queue is intended to be used for:

- Congestion notification (EFCI PTI field x1x, see Mode Register).
- OAM F5 flow (PTI field 10x, see Mode Register)
- Reserved PTI field (PTI field 110 or 111, see Mode Register)
- Assigned uses for Signalling or OAM F4 flow based on VCI address.

Note: A CID & AT Fail (bits 31-16) of zero indicates "exception notification queue empty".

Receive Notification CID/Head Registers

There are three pairs of registers associated with each of the three receive data queues. Each pair consists of a Notification CID/Head register and a Notification Status/Length register.



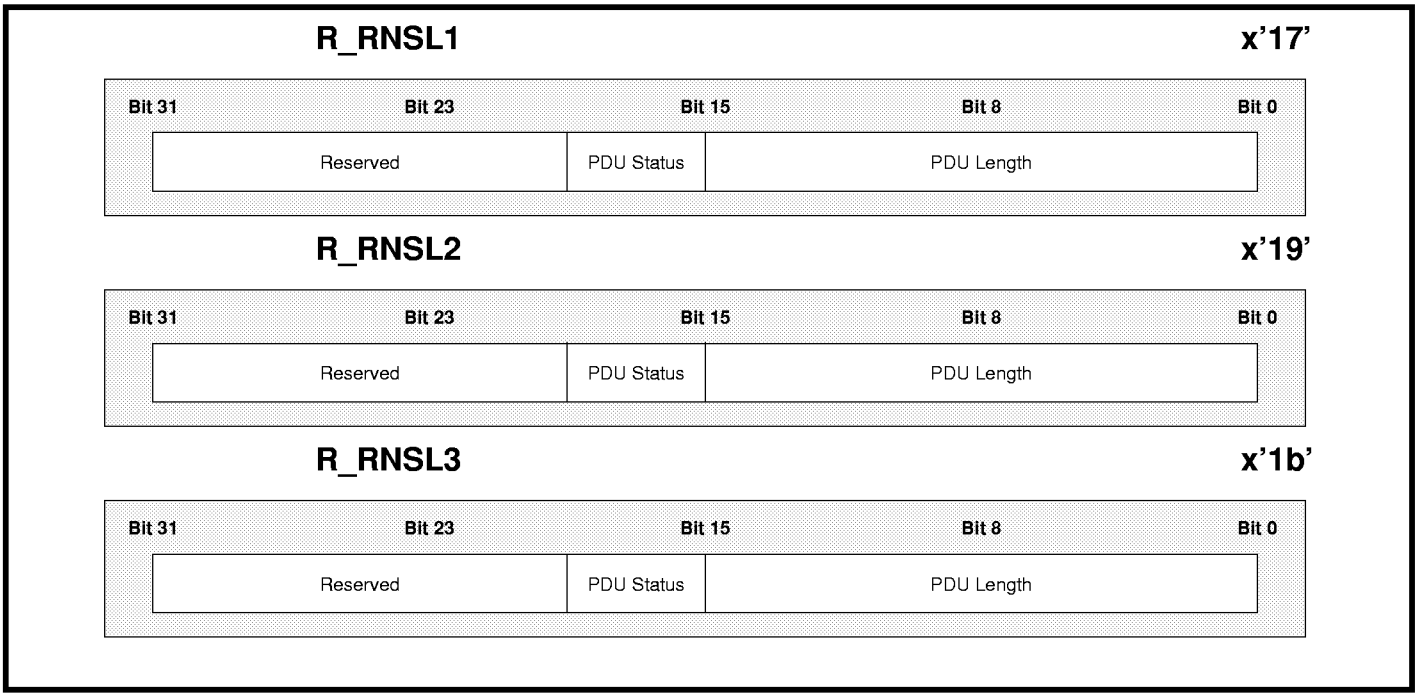
Each register contains the Connection Identifier (CID) that has a complete received SDU in the Buffer Memory. SLI (Bit 31: Status/Length Indicator), if set to one, indicates that the data received has additional status/length information present in the Receive Notification Status/Length Register. The notifications which are queued behind this register are for normally enabled virtual circuits (e.g. the address translation maps to an enabled Virtual Circuit Table Entry).

Connections set up in "Raw AAL" mode (i.e. unedited ATM cells) are presented data in the following consecutive-byte format: H0, H1, H2, H3, H4, pad, payload0, payload1...payload47. The length of each cell presented in this manner (including pad) is 54 decimal. Note also that the H4 byte is framer-dependent. If the framer passes the byte to the SAR as it was received from the physical layer then it is the HEC byte containing the CRC on the ATM cell header.

A CID of zero indicates that the particular notification queue is empty. The interrupt status can also be used to determine if a queue is empty.

Receive Notification Status/Length Registers

There are three pairs of registers associated with each of the three receive data queues. Each pair consists of a Notification CID register and a Notification Data register.



These registers provide status and PDU-length information corresponding to each Receive Notification CID/Head register. There is a register for each of the Receive Notification Queues to which VCCs may be assigned. The system must read this register if the associated ReceiveNotification CID/Head register has bit 31 turned on (indicating that the incoming PDU has status/length information in the notification queue).

Error bit	bit 20
PDU Status	4-bit status of received PDUs. Bit 19 is the error-bit (if set this PDU has an error). Bits 18-16 contain error/exception status as follows: b'000' -- no error/exception b'001' -- aal CRC error b'010' -- aal sequence error b'011' -- aal length error b'100' -- aal header error b'101' -- max count exceeded b'110' -- aborted by receive purge command and no error detected prior to abort (pending errors take precedence). b'111' -- aborted by sender using aal protocol
PDU Length	Byte length of this PDU (equals to the sum of the descriptor lengths).

This page left blank intentionally

This chapter gives the memory layout and format of the user visible data structures. These data structures need to be initialized by the local microprocessor prior to enabling the SAR for operation. Some of the data structures will also be initialized by the local microprocessor each time a virtual circuit is setup or torn down.

Overview

These data structures are found in the memories attached to the Segmentation and Reassembly devices. Each device has three memories attached unless Descriptor Memory and Buffer Memory are combined, in which case the device has two memories attached.

- Buffer Memory (BM) - where buffers of data are stored before transmission and after reception. The Segmentation Buffer Memory (SBM) holds input data for the segmentation device. The Reassembly Buffer Memory (RBM) holds data output by the reassembly device.
- Descriptor Memory (DM) - contains one descriptor for each buffer in the BM. The descriptor contains a "next pointer" field for buffer chaining, a length field, and control information. The Segmentation Descriptor Memory (SDM) contains descriptors describing the SBM. The Reassembly Descriptor Memory (RDM) contains descriptors describing the RBM.
- Connection Memory (CM) - contains the Virtual Circuit Connection Table (VCCT), Notification Queues and Address Translation Tables. The VCCT contains an entry for each virtual circuit. These entries are 32 bytes in size and save the context and configuration of the connection for the SAR. ATM connections are normally full duplex, so each virtual circuit has two CM entries, one in the Segmentation Connection Memory (SCM) and one in the Reassembly Connection Memory (RCM).

Memory Layout

Buffer Memory (BM) Layout

Buffer Memory is divided into equal sized buffers of either 256, 512, 2048 or 4096 bytes. The buffers are referenced using a Buffer Index Number (BIN) that is a maximum of either 13, 14 or 16 bits depending on the selected buffer size. These indexes are converted to a 22-bit long-word (8 bytes) address. The maximum Buffer Memory size is limited by either the number of addressing bits or the use of the 16-bit buffer index (i.e. a maximum of 64K buffers). The following table summarizes the sizes.

Table 5-1 Buffer Memory (BM) Sizes

Buffer Size	Maximum BIN size	Maximum Number of Buffers	Maximum Buffer Memory Size
256B	16 bits	64K	16M
512B	16 bits	64K	32M
2048	14 bits	16K	32M
4096B	13 bits	8K	32M

Table 5-1 shows the maximum number of buffers and maximum buffer memory size for the choices of buffer size. The table shows that for 256 byte buffers the limitation is the 16-bit buffer index. For 512 byte buffers the limits are balanced. For the larger buffer sizes the maximum number of buffers is reduced to accommodate the 22-bit address limitation. Buffer 0 is not used, since address 0 is used as an End-of-Chain (EOC) value.

Within a Buffer Memory, buffers are allocated on aligned addresses. For 256-byte buffers, this means that the last five bits (8 minus 3 since addresses are 2 double words) of a buffer address are always zero. For 4096-byte buffers, the last 9 bits (12-3) of the address are always zero. The high-order 16 bits of the buffer address are used as the Buffer Index Number (BIN).

Descriptor Memory (DM) Layout

Each DM contains one four byte entry corresponding to each buffer in Buffer Memory. Therefore the size of Descriptor Memory required (in bytes) is the chosen number of buffers times 4. The maximum size is a function of the maximum buffers possible as shown in Table 5-1. Table 5-2 shows some examples.

Table 5-2 Descriptor Memory (DM) Sizes

Number of Buffers	Descriptor Memory Size
64K	256KB
32K	128KB
16K	64KB
8K	32KB
4K	16KB

Within a Descriptor Memory descriptors are allocated on aligned word-boundaries. The chipset generates a 16-bit word (4 byte) addresses to Descriptor Memory using bits BmA_D[47:32]. Descriptor 0 is not used, because the corresponding buffer is not used. The Buffer Index Number can be used directly to locate the corresponding descriptor in Descriptor Memory since, by good fortune, the unit of memory access matches the size of an entry.

Connection Memory (CM) Layout

The CMs contain three types of data:

- Notification Queues (NQs) - used to store pending notifications prior to presentation to the user.
- Address Translation Tables (ATTs) - used by Reassembly device to translate VPI-VCI-MID addresses into Connection Identifiers (CIDs).
- Virtual Circuit Connection Table (VCCT) - used to save a connections context and configuration.

The location of these data structures is user assignable within certain addressing constraints. Valid CIDs are from 1 to (32k-1). Each CID is multiplied by 8 to convert to a word address (or 32 to get the byte address as perceived by software). The remaining tables can be located anywhere as long as address alignment requirements are met. Table 5-2 shows the size of Connection Memory required to support various numbers of connections. These sizes do not account for the Address Translation Table or Notification Queues (i.e. reduce the number of connections to accommodate the additional tables or go to the next higher memory size increment.

Table 5-3 Connection Memory (CM) Sizes

Number of VCs	Connection Memory Size
< 32K	1MB
< 16K	512K
< 8K	256K
< 4K	128K

Data Structure Formats

Descriptor Memory Entry (DME)

Each Descriptor Memory Entry (DME) is four bytes and contains the following fields:

- A two byte next Buffer Index Number (BIN).
- A 13-bit length field.
- Two status bits for marking ends of PDUs and controlling notification.

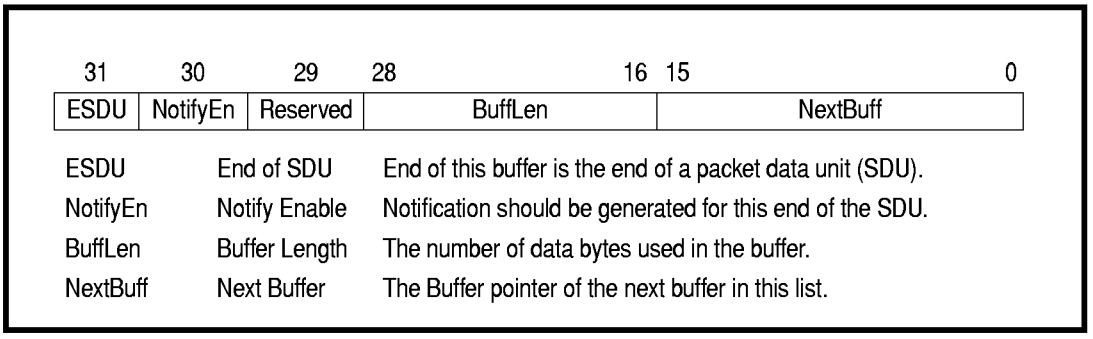


Figure 5-1 Descriptor Memory Entry (DME)

The Notify Enable bit is only used by the segmentation device. The reassembly device always generates a notification at the end of a received PDU.

The Next Buffer field contains a 16-bit pointer index which can be used locate the next buffer in the chain and that buffers descriptor. The value is the actual word address of the next descriptor in the chain unless the R_DMBASE/S_DMBASE register is set to non-zero to cause an offset. The buffer address is the index multiplied by the buffer size (and converted to a double-word address). A value of zero indicates the end of the list.

Buffer Memory and Descriptor Memory can be viewed as separate arrays which share a common 16-bit index. If we want to have 100 buffers then the following "C" code will illustrate the relationship assuming an "unsigned int" is 4 bytes and that we pick 256 for our buffer size:

- typedef struct buf {char data[256];} buf_t;
- buf_t bufferMem[100];
- unsigned int descriptorMem[100];
- for(i=1; i<100; i++) {
- descriptorMem[i] = <value to assign "i"th descriptor>;
- bufferMem[i]= <value to assign to first byte of "i"th buffer>;}

Because zero is used as an "end of chain" value descriptorMem[0] and bufferMem[0] must not be used.

The SAR views the base address of these arrays as zero (although Descriptor Memory can be offset by using the R_DMBASE/S_DMBASE registers). Descriptor Memory is accessed as words (4 byte) which conveniently match the size of a Descriptor -- therefore the SAR will, for example, output the address 3 when chaining to a Descriptor located by NextBuf=3. The corresponding buffer in Buffer Memory (assuming 256-byte buffers) would be addressed by the SAR as 3 * 256 / 8 = 96. The divide by 8 is due to the double-word (8-bytes) Buffer Memory address.

Segmentation DMEs can be set to force the SAR to skip over the start of a buffer when transmitting data allowing that portion of the buffer to be dedicated to non-SAR usage. The range within the buffer that can be skipped depends on the chosen buffer size. The smaller the buffer size the more bits are available to specify the programmable offset. The actual byte-offset is the field value multiplied by two. The following ranges are available:

- 4k buffers: no offset available
- 2k buffers: DME[28] is offset specifier (byte offset is 0 or 2)
- 512-byte buffers: DME[28:26] is offset specifier (byte offset is 0,2,4 or 6)
- 256-byte buffers: DME[28:25] is offset specifier (byte offset is 0,2...30)

Notify Queue Element (NQE) Formats

The segmentation device uses four-byte notification entries, while the reassembly device uses eight-byte notification entries. The format of the entries is not relevant to the user except in their indirect affect on how the SAR maps the data into user registers that post notifications. The SAR manages the notification queues without user intervention (except to initialize their location) and should not be written by the user. For debug purposes it is possible to read the contents of a queue thru the Connection Memory Address/Data registers.

Segmentation Notify Queue Element (SNQE) Format

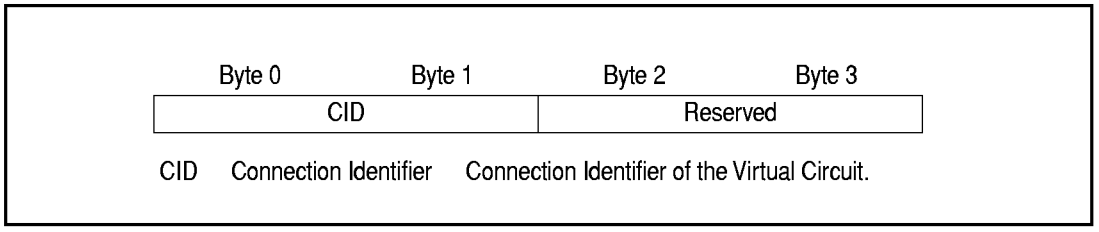


Figure 5-2 Segmentation Notify Queue Element (SNQE)

The segmentation device can be programmed to always notify, never notify, or occasionally notify, by setting or clearing the Notify Enable bit of the last descriptor of a PDU.

Reassembly Notify Queue Element (RNQE) Format

The reassembly device notifies the system at the end of receiving a PDU or a group of PDU's. The reassembly device can be programmed to use different notification queues for different classes /groups of circuits by setting the notify queue number in each Connection Table Entry.

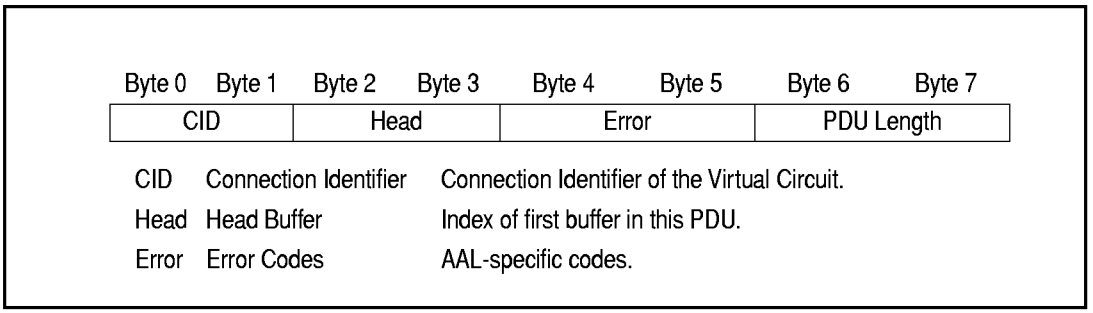


Figure 5-3 Reassembly Notify Queue Element (SNQE)

Address Translation Table Entry (ATTE) Format

The Address Translation Table is composed of entries (ATTE's) used to translate the ATM-cell-header-address into a Connection Identifier (CID). The 10-bit MID field of an AAL 3/4 cell can also be used in the translation.

The translation works by hoping thru tables until a "leaf" entry (cont=0) is reached which contains the sought after CID. The technique is similar to the "page table" scheme used by virtual memory systems.

All tables used in address translation must begin on 32-word boundaries. There is one VPI table, up to 256 VCI tables (if every VPI is "active) and possibly many MID tables if using AAL3/4. The user must insure boundary alignment when initializing the tables using the R_CMAR/R_CMDR registers (e.g. insure that the R_CMAR register is pointing to a 32-word boundary when starting to write any of the tables). The Reassembly device locates the base of each table by multiplying the BaseN value described below by 32 decimal (i.e. appending 5 binary zeroes to BaseN). Again, these are word (4-byte) addresses since Connection Memory is only addressable as words. The programmer may vue this alignment as 128 byte (32 * 4).

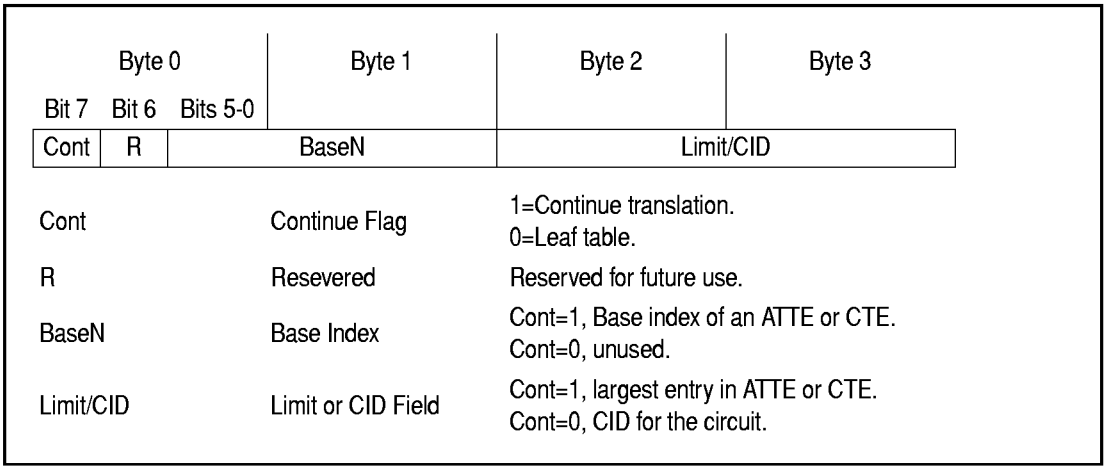


Figure 5-4 Address Translation Table Entry (ATTE)

The continuation flag tells how much translation to do. It is possible to translate any of the following, depending on how the continue flag is set at each level in the tables:

- VPI only (a Virtual Path connection)
- VPI and VCI (a Virtual Channel connection)
- VPI, VCI and MID (an AAL3/4 connection using MIDs)

Connection Table Entries

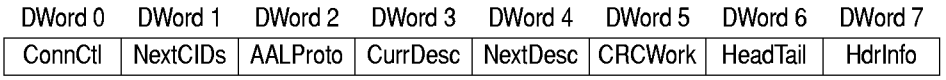
The connection table entries are eight words (32-bytes) and contain two types of fields: circuit setup fields, and working data fields. The local microprocessor must initialize the circuit setup fields and zero the working data fields prior to using the virtual circuit.

A Connection Table Entry is located by it's Connection ID (CID). A CID is a 15-bit index that can efficiently be converted to a Connection Memory address by multiplying by 8 (e.g. by concatenating 3 binary zeroes to the low bits of the CID). CID one is located at Connection Memory word address 8 (i.e. byte address 32). CID zero is not allowed because zero is used as an end-of-chain value. The user can select which CIDs in the range of 1-32K are used if sufficient Connection Memory exists. A typical scheme would be to allocate sequentially from 1 to N, where N is the maximum number of active connections. If Switched Virtual Circuits (SVCs) are used, then CIDs can be reused by re-initializing the Connection Table Entry of the associated CID and, in the case of Reassembly, changing the Address Translation Table if the VPI/VCI/MID has changed

For the Reassembly chip CIDs are selected by the values placed in the "leaf" nodes of the Address Translation Table. For the Segmentation chip the user initializes the chosen CIDs and then uses those values when queuing PDUs for transmission.

Care should be taken to insure that chosen CIDs do not overlap other data structures whose locations are programmable (e.g. for the Reassembly chip the Notification Queues and Address Translation Table; for the Segmentation chip the Transmit Notification Queue).

Segmentation Connection Table Entry (SCTE)



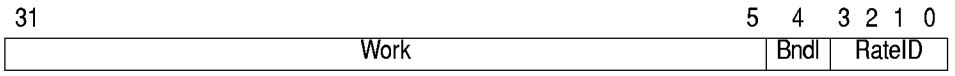
Symbol	Name	Description
ConnCtl	Connection Control	Connection controls for skip count & rate control.
NextCIDs	Next CID Fields	Working area for next entry in rate or bundle list.
AALProto	AAL Protocol Control	AAL type, skip limit, etc.
CurrDesc	Current Descriptor	Working area for current descriptor.
NextDesc	Next Descriptor	Working area for next descriptor.
CRCWork	CRC Work Space	CRC working area for AAL protocols.
HeadTail	Head and Tail Pointers	Working area for Head & Tail of queue.
HdrInfo	Header Information	ATM header information.

Figure 5-5 Segmentation Connection Table Entry (SCTE)

A Segmentation Connection Table Entry must be initialized by the user before PDU's can be queued to the Segmentation device on the associated connection. After initialization these values are under control of the device and should not be written by the microprocessor. The user must initialize the ConnCtl and HdrInfo words as defined below. The other 6 words must be initialized to zero.

Connection Control

(ConnCtl)

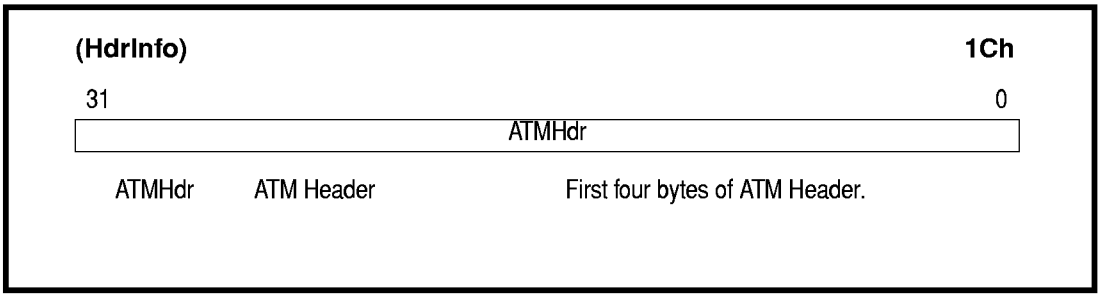


Bndl	Bundle Flag	Set if this circuit is part of a bundle.
RateID	Rate ID	Rate queue index.

Table 5-4 AAL Type Field Encoding

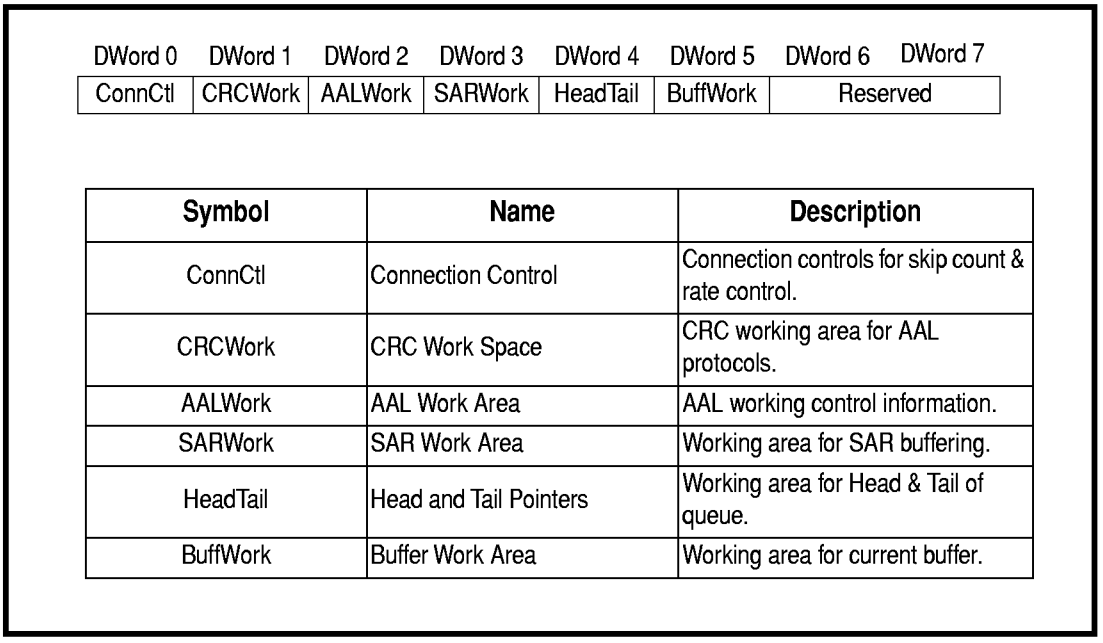
AAL Type Field	Meaning
000b	Null AAL (48-byte payloads)
001b	AAL 1
011b	AAL 3/4
100b	Raw 53-byte ATM cells.
101b	AAL 5

Header Information



The ATM Header is initialized with the GFC,VPI, VCI and PTI fields. The fifth byte of the ATM Header is normally generated by the framer and is output by the Segmentation chip as a pad-byte of zero. If using "raw" AAL mode this field is not used. Instead the entire cell including the header is taken from the transmit buffer.

Reassembly Connection Table Entry (RCTE)



Symbol	Name	Description
ConnCtl	Connection Control	Connection controls for skip count & rate control.
CRCWork	CRC Work Space	CRC working area for AAL protocols.
AALWork	AAL Work Area	AAL working control information.
SARWork	SAR Work Area	Working area for SAR buffering.
HeadTail	Head and Tail Pointers	Working area for Head & Tail of queue.
BuffWork	Buffer Work Area	Working area for current buffer.

Figure 5-6 Reassembly Connection Table Entry (RCTE)

Before activating a connection to receive data (e.g. before setting the Address Translation Table to point to this CID) the RCTE data structure must be initialized by the user. The ConnCtl word should be set to the desired options as defined below. The CRCWork word must be initialized to 0xFFFFFFFF. The remaining words must be initialized to zero. After initialization, these values are under control of the SAR and should not be written by the microprocessor. Words 6 and 7 are reserved, and are not currently used by the Reassembly device. However, for future compatibility it is recommended that these words be initialized to zero. The values that need to be set for the Connection Control word are documented below.

Connection Control

(ConnCtl) **00h**

31 24	23	22 20	19 18	17 16	15 0
CTEFlag	Active	AALType	Work	NotQN	NotLen

CTEFlag	CTE Flag	Set to hex 'EA', to show initialization.
Active	ATT Active	Address translation is active for this VC.
AALType	AAL Type	AAL Type for this VC as shown in Table 5-4.
NotQN	Notify Queue Number	Number of Notification Queue for this VC.
NotLen	Notify Length	Max. number of bytes to receive before notification.

Memory Initialization

Buffer Memory (BM) Initialization

The contents of the buffers in Buffer Memory do not need to be initialized before operation is enabled because these contain no control information, only transmit or receive data. However, the descriptors for these buffers must be set up.

Descriptor Memory (DM) Initialization

Descriptors Memory should be initialized by the user to create a buffer free-chain prior to entering normal operating mode. During normal operation the buffer chains are managed jointly by the SAR and user (the SAR managing one end of the chain and the user managing the other).

Connection Memory (CM) Initialization

A Connection Table Entry must be initialized by the user prior to transmitting or receiving on a connection. This could be a one-time power-on initialization for Permanent Virtual Circuits or as needed for Switched Virtual Circuits.

The Reassembly chip also must have its Address Translation Table initialized at system start-up or dynamically as needed. During normal operation, tables or entries can be added dynamically to reflect newly activated connections. Care should be taken to "hook" in new tables after they are built to avoid unpredictable results.

Notification queues do not need to be initialized since the SAR writes each entry before read-back.

This chapter describes the interfaces between the SAR devices and memory. The interfaces include Buffer Memory, Descriptor Memory and Connection Memory. The SAR register interface is also described.

Buffer Memory

Buffer Memory holds user-data in transit between the SAR and the system. The Segmentation Buffer Memory holds PDUs waiting to be segmented into ATM cells and transmitted to the Framer. The Reassembly Buffer Memory holds reassembled PDUs waiting for transfer to the system.

The SAR provides a generic memory interface that runs in burst mode (i.e. a starting address is provided followed by a variable length data burst). Because the interface operates in burst mode it is easily interfaced to memories (or memory ports) that contain internal address counters such as VRAM, triple-ported VRAM and Synchronous DRAM (SDRAM). Other memory types such as SRAM are readily used by implementing an external address counter.

The maximum size for the Buffer Memory is 32Mbytes (16Mbytes for 256 byte buffers).

Choosing a Memory Type

The board designer is likely to prefer a particular memory type based on cost, density, availability, ease of interfacing, etc. In order for a memory to interface to the SAR it must be capable of running in burst mode at the frequency the user drives BmClk (i.e. it must be capable of doing data cycles in consecutive clock periods in the same direction). The SAR will not perform mixed read/write operations to Buffer Memory.

Another consideration is memory bandwidth. There are 27 clock-periods per *cell-time* if running at 40MHz and 22 clock-periods if running at 33MHz. In order to run at full speed it must be possible for both the SAR and the System to transfer one-cells-worth of user data in the available number of clocks per *cell-time*. It may also be necessary to allocate some bandwidth to the local CPU for network management, connection setup/teardown, etc. A transfer of one cells-worth of data between the SAR and Buffer Memory requires 6 or 7 data cycles (seven only if the data is misaligned to a double-word boundary). This assumes 48 bytes are transferred, the maximum size of a cell payload. The use of AAL Raw Mode adds 1-2 more data cycles.

Note that extra overhead is required when a multi-buffer PDU requires a span from one buffer to the next (see the timing diagrams in the following sections). Other overhead that should be factored in includes address setup and delay (i.e. how many clocks after address setup before the first data cycle?) and overhead for

memory arbitration. It is recommended that the user carefully add up the overhead and accesses to determine if their chosen memory-type allows full speed operation. The distinguishing characteristic between different memory types is likely to be the enforced delay (if any) following address setup (for row transfer, etc.).

Interface Signals

The following signals form the Buffer Memory interface:

- BmR*
- BmG*
- BmA_D[63:0]
- BmPar[7:0]
- BmWE[1:0] (Reassembly device only)
- BmDS*
- BmSpan

Buffer Memory Read Cycle (Segmentation)

The Buffer Memory read cycle is performed by the Segmentation device. This device does not perform a Buffer Memory write since the segmentation process only requires the transfer of user data in the outgoing direction.

Figure 6-1 shows a Buffer Memory read cycle without *span* and is described in the following paragraphs. The cycle is initiated when the SAR asserts the Buffer Memory Request signal BmR* and places the starting burst address on BmA_D[53:32] (BmA_D[55:54] are set to zero for ease of parity checking). The normal action is to latch the address immediately on the next clock edge (although the address will remain valid until the data cycles begin as long as the user does not grant access to Descriptor Memory following a DmR* which could have been asserted at the same time as BmR*).

At some point following BmR* the arbitration logic responds to the request with a Buffer Memory Grant BmG*. The SAR does not enforce or require any specific latency to grant. However, excessive delays to BmG* will introduce jitter into the cell-transmit rate and reduce the available ATM bandwidth (i.e. one or more empty cells will be sent at the PHY level).

The SAR will expect valid data on BmA_D[63:0] on the 3rd positive clock edge following BmG*. The number of data cycles is determined by a combination of the AAL type, the current AAL state and the amount of data remaining in the current PDU (6 cycles maximum for AAL5). If parity is enabled it must also be presented at the same time. The Segmentation device outputs BmDS* to indicate that the data cycle is active. It may be possible to use BmDS* to generate an output-enable to the memory (check the AC timing section to determine if enough setup will exist on the returning data).

IMPORTANT: The board designer may need to rely on the fixed number of clocks between BmG* and BmDS* to determine when to begin incrementing the burst address or to generate an OE to the memory with sufficient setup-time on the data.

The Segmentation device de-asserts BmR* one clock ahead of the edge on which the data is last captured. In the following clock period the arbiter de-asserts the grant.

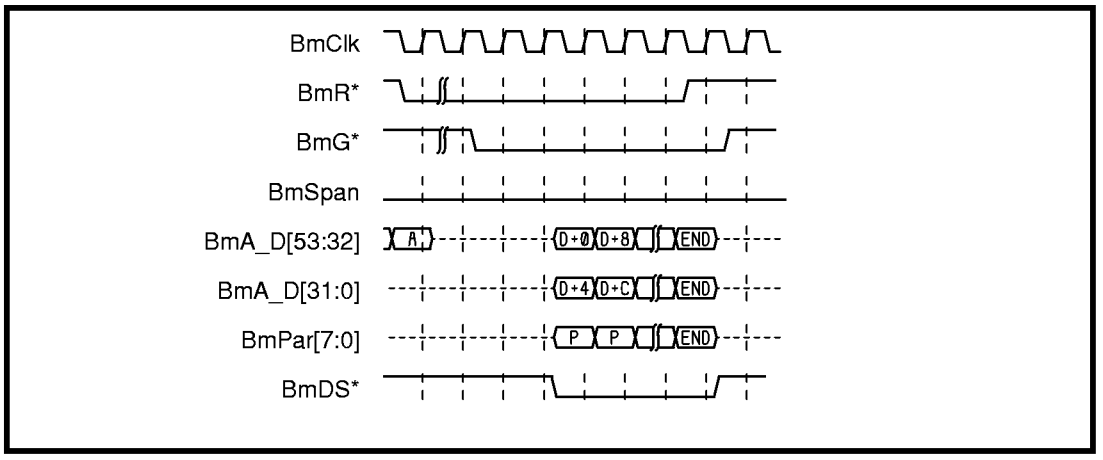


Figure 6-1 622Mbps Buffer Memory Read Cycle without Span

Figure 6-2 shows the *span* case. In this example the first burst is 2 data cycles and the second is shown as 4 or more. For AAL5 the first burst is 2 data cycles and second is 4 unless the end-of-PDU is reached beforehand. The size of each burst is determined by the amount of data required and the amount remaining in the buffer (for example 256/48 provides a remainder of 16 which requires 2 data cycles).

When a buffer is spanned the Segmentation device acquires part of the data for a cell from the end of one buffer and part from the beginning of the next buffer in the chain. In this case the Segmentation device asserts BmSpan one clock after BmR* for the burst immediately preceding the span. This need not be a special case as long as normal arbitration is allowed to occur for each of the 2 bursts. BmR* is de-asserted for 2 clocks following the first burst and then re-asserted again for the second burst. As in the preceding discussion the grant is eventually provided and the second burst begins with the same timing as in the non-span case.

The use of the BmSpan signal is optional. It could be used by the users arbiter to insure that the SAR is given adequate memory bandwidth (i.e. even though the SAR de-asserts BmR* after the first burst -- the BmSpan signal provides a tip-off that a second burst for the same cell is about to follow). It is also possible to design the arbiter so that BmG* is held asserted between the bursts. In that case the SAR will immediately see the grant without the normal delay and there will only be 2 idle data cycles between the first burst and the second. However, there may be insufficient time to setup the address of the second burst. Also, the memory-type used may require more clocks between address setup and data cycles.

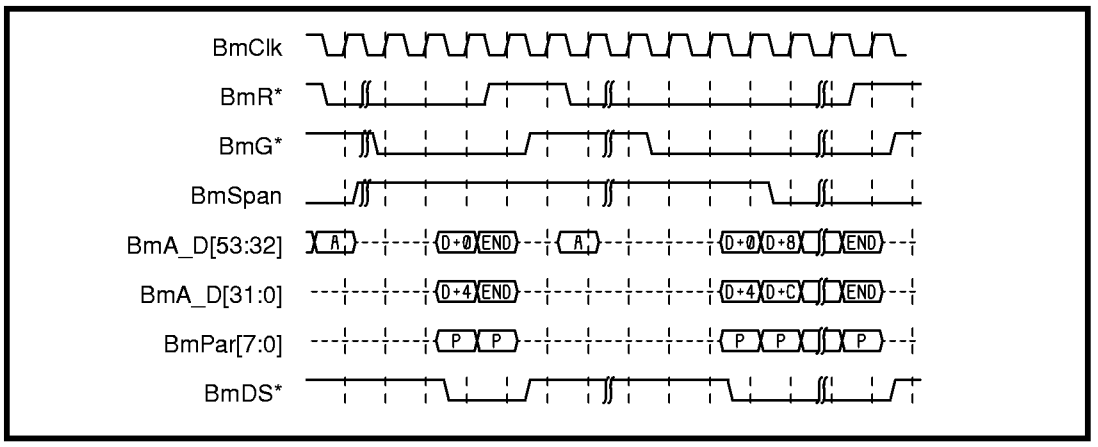


Figure 6-2 622Mbps Buffer Memory Read Cycle with Span.

Figure 6-3 shows the non-span case when operating in the *narrow* mode (e.g. 155Mbps). Since the data bus is half the size it takes 12 cycles to transfer 48 bytes in a single burst (13 if not word-aligned).

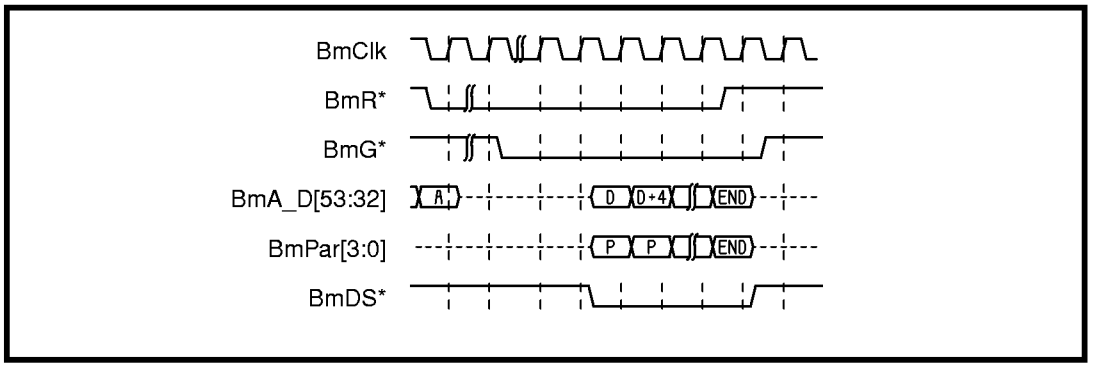


Figure 6-3 155Mbps Buffer Memory Read Cycle without Span

Buffer Memory Write Cycle (Reassembly)

The Buffer Memory write cycle is performed by the Reassembly device. This device does not perform a Buffer Memory read since the reassembly process only requires the transfer of user data into Buffer Memory.

Figure 6-4 shows a Buffer Memory write cycle and is described in the following paragraphs. The cycle is initiated by the Reassembly device placing the required address on BmA_D[53:32] (BmA_D[55:54] are set to zero for ease of parity checking) and asserting BmR*. The arbitration logic grants the request by asserting BmG*. During the 4th clock following BmG* the Reassembly device places the data on BmA_D[63:0] and indicates that the data is valid with BmDS*. On the 5th positive edge following BmG* the data is valid. NOTE THAT THE REASSEMBLY DEVICE REQUIRES 2 CLOCKS MORE THAN THE SEGMENTATION DEVICE.

Most of the interface for Reassembly behaves identical to that for Segmentation as can be seen in the figure. THE MAIN DIFFERENCE IS THAT FOR REASSEM-

BLY THERE ARE TWO ADDITIONAL CLOCKS BETWEEN BmG* AND BmDS*. The data output from the Reassembly device becomes valid during the 4th clock period following BmG* and remains valid past the 5th positive edge following BmG*.

The Reassembly device outputs BmWE[1:0]* to provide the write-enables for the high word (BmA_D[63:32]) and low word (BmA_D[31:0]) on the data bus. In cases where cell data ends on a non-word boundary the Reassembly device stores the misaligned data (in Connection Memory) so that it can be properly recombined with new data at a later time.

IMPORTANT: As with Segmentation the board designer may need to rely on the fixed number of clocks between BmG* and BmDS* to determine when to begin incrementing the burst address. Note that this 'trick" may also be usable on BmWE[1:0] as long as the AAL types used insure that each successive burst starts on a double-word boundary (as is the case with AAL5).

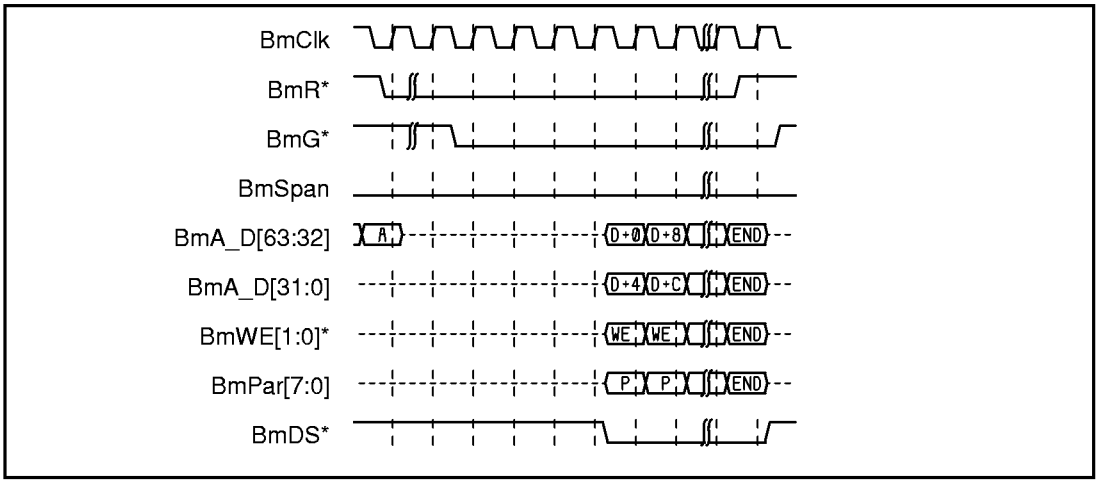


Figure 6-4 622Mbps Buffer Memory Write Cycle without Span

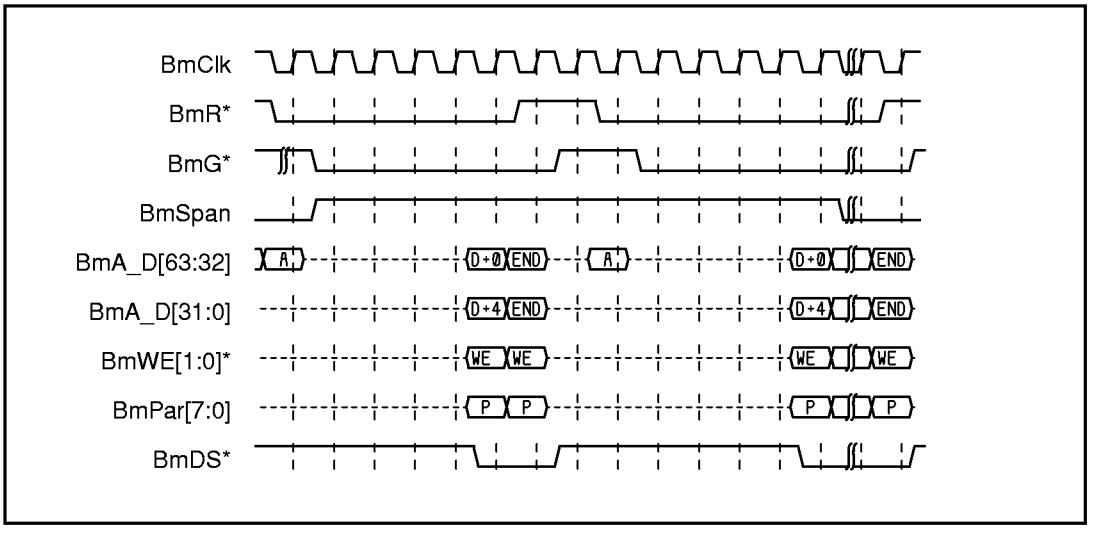


Figure 6-5 622Mbps Buffer Memory Write Cycle with Span

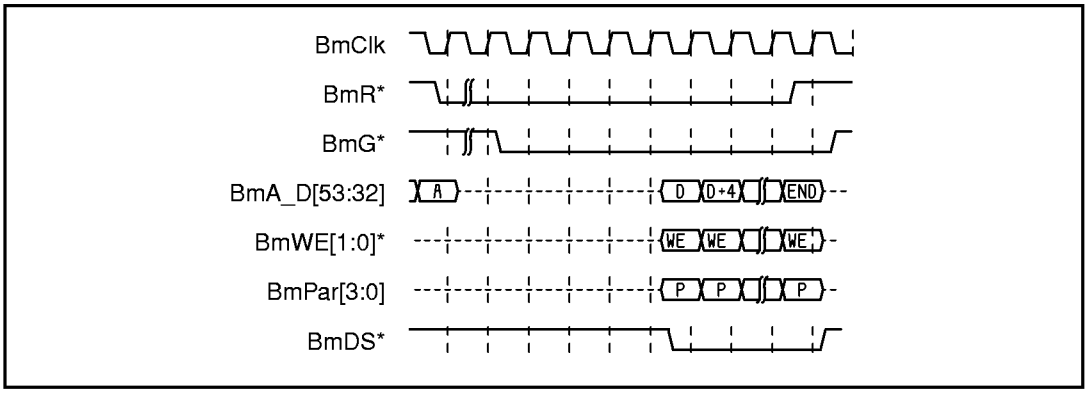


Figure 6-6 155Mbps Buffer Memory Write Cycle Timing Diagram

Buffer Memory External Logic

External logic must be able to arbitrate requests for Buffer Memory between the Host DMA Logic, the SAR devices and the local CPU.

The external logic may also need to perform the following tasks:

- De-multiplex the addresses and provide an address latch/counter.
- Separate Row and Column addresses.
- Provide memory-specific control signals (e.g. RAS*, CAS*, TR*/OE*, ME*/WE*, DSF, and SE* for VRAM).
- Handle memory refresh.
- If a burst-type memory is used the arbiter may need to set up for the burst before granting access to the SAR.

Descriptor Memory

Descriptor Memory, like Buffer Memory, is shared between the SAR, the host DMA logic and the CPU. Descriptor Memory holds *descriptors* which contain control information, DMA lengths and forward pointers. Access to Descriptor Memory must also be arbitrated by external logic.

The following signals form the Descriptor Memory interface:

- DmR*
- DmG*
- DmR_W*
- BmA_D[31:0] for descriptor data (in “wide” mode)
- BmA_D[47:32] for descriptor addresses (in “narrow” mode BmA_D[63:32] is used for descriptor data)
- BmPar[7:0]

Descriptor Memory Read Cycle

Descriptor Memory read cycles are initiated by asserting DmR*. The DmR_W* signal is asserted HIGH and the starting address is placed on BmA_D[47:32]. The external arbiter grants the request by asserting the DmG* signal and the pipelined address/data cycles proceed as shown in Figure 6-5. For "narrow" mode operation (e.g. 155Mbps) alternate address/data cycles occur as shown in Figure 6-6.

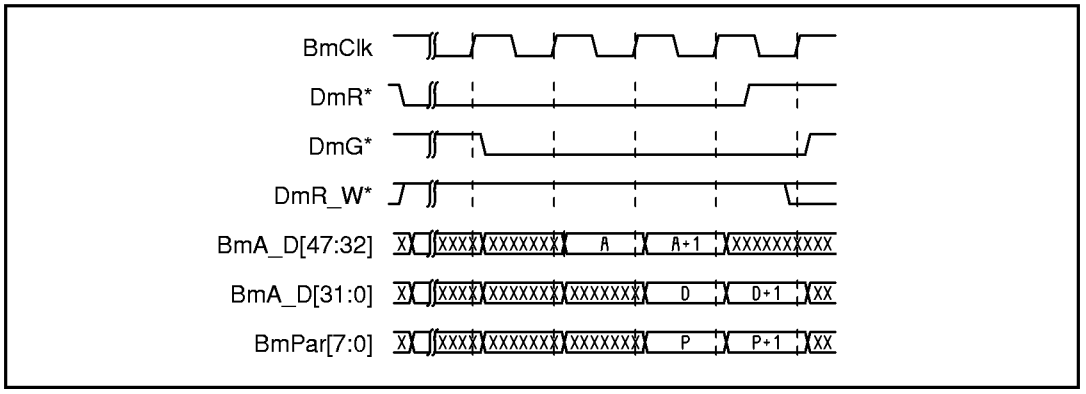


Figure 6-7 622Mbps Descriptor Memory Read Cycle Timing Diagram

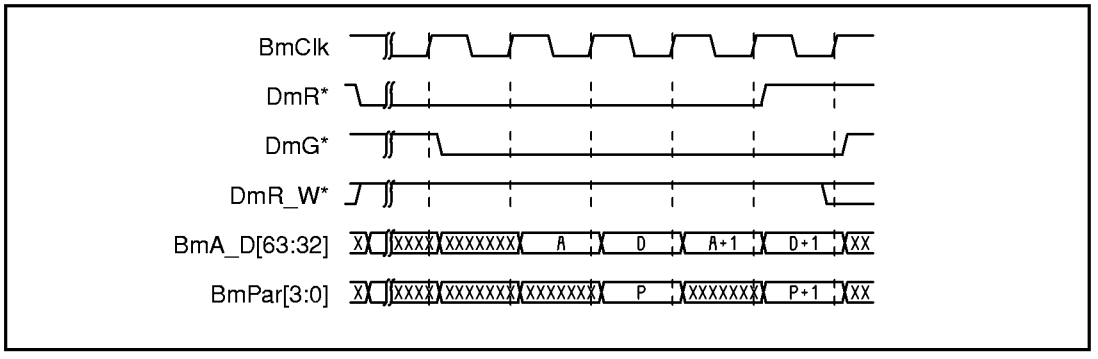


Figure 6-8 155Mbps Descriptor Memory Read Cycle Timing Diagram

Descriptor Memory Write Cycle

Descriptor Memory write cycles are initiated by asserting DmR*. The DmR_W* signal is asserted LOW.For non-multiplexed 622Mbps operation the address is placed on BmA_D[47:32] and the data is placed on Bm[31:0]. For lower speed, multiplexed operation only the address is initially placed on the Buffer Memory bus. When the arbitration logic grants the request by asserting DmG* at the next transition the data is written.

Figures 6-9 and 6-10 show Descriptor Memory write cycles.

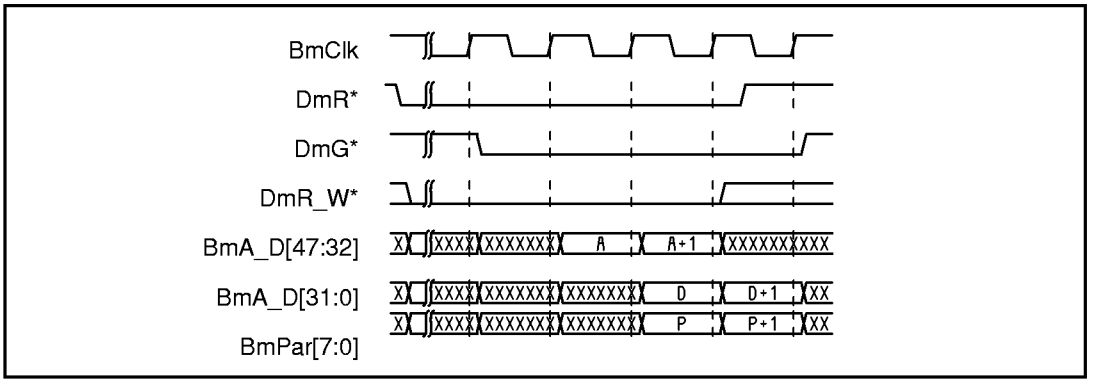


Figure 6-9 622Mbps Descriptor Memory Write Cycle Timing Diagram

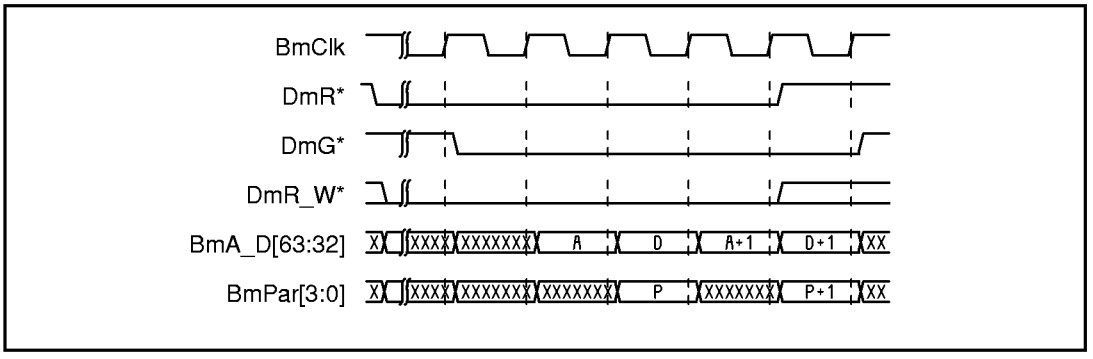


Figure 6-10 155Mbps Descriptor Memory Write Cycle Timing Diagram

Connection Memory

Connection Memory is used to save the context of connections. It also contains the notification queues and, for the Reassembly device, the Address Translation Table. A microprocessor is required to setup Connection memory at initialization time.

The SAR is designed to use SRAM for Connection Memory. Other types of random access memory may be possible in the future as speed improves. Each connection consumes 32 bytes.

The Connection Memory interface is comprised of the following signals:

- CmData[31:0]
- CmPar[3:0]
- CmAddr[18:0]
- CmWE*
- CmOE*

Connection Memory Read Cycle

For a Connection Memory read cycle, the SAR will place the address onto the CmAddr[18:0] bus. During the following cycle CmOE* is asserted and the data is captured by the SAR on the next positive clock edge. The CmWE* signal is de-asserted during the address cycle to signal a read cycle.

Figure 6-11 shows a functional timing diagram for a Connection Memory read cycle.

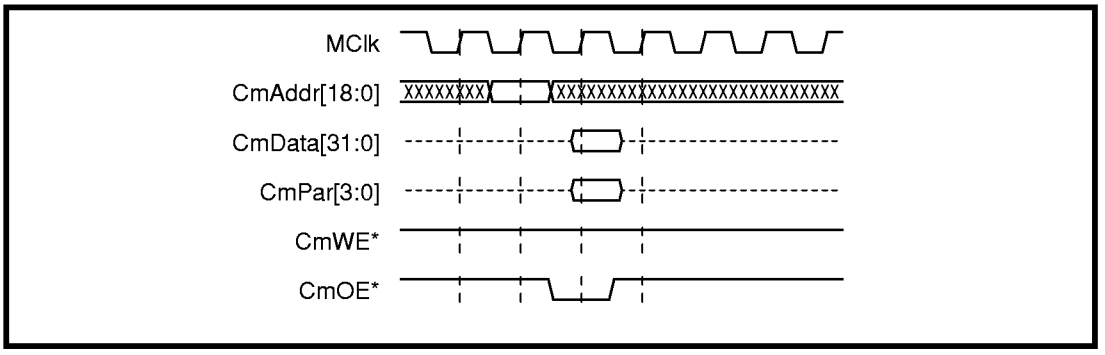


Figure 6-11 Connection Memory Read Cycle Timing Diagram

Connection Memory Write Cycle

For a Connection Memory write cycle, the SAR places the address on CmAddr[18:0] Bus and asserts CmWE*. On the next rising edge of BmClk the data on CmData[31:0] and CmPar[3:0] is written. The CmOE* signal is de-asserted.

Figure 6-12 shows a functional timing diagram for a Connection Memory write cycle.

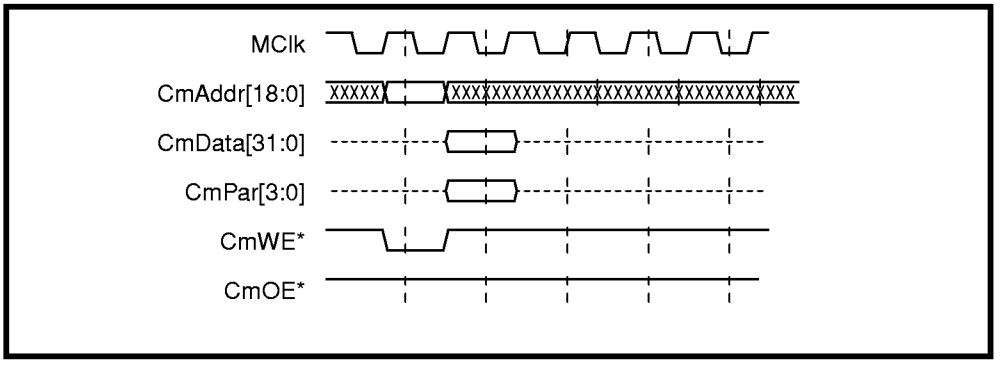


Figure 6-12 Connection Memory Write Cycle Timing Diagram

Register Interface

The SAR device's internal registers are accessed by sharing the Connection Memory datapath pins CmData[31:0]. Bus isolation is required in order to achieve this sharing. The normal approach is to use bus-isolating transceivers with registers. The registered transceivers allow independent clock domains for the CPU (normally BmClk) and the SAR's register interface. They can also help relax timing by removing the propagation delay thru the transceiver from the single-cycle path. The following signals are used to access the register interface:

- RS*
- RR_W*
- RA[5:0]
- CmData[31:0]
- CmXOE*
- CmXLE*

The input signals RS*, RR_W* and RA[5:0] are internally synchronized to MClk before use (and can therefore can be generated from a different clock such as BmClk as long as the signals are held long enough to insure detection by an MClk-based synchronizer).

The output signals CmXOE* and CmXLE* are driven by MClk and the bidirectional data bus CmData[31:0] is accessed using MClk. It is assumed that the side of the transceiver attached to the CmData bus operates strictly in the MClk domain controlled by CmXLE* and CmXOE*.

Register Read Cycle

A Register read cycle is initiated when the SAR samples RS* LOW on the positive edge of MClk. On this same edge RR_W* must be set HIGH and the register address must be placed on RA[5:0]. The SAR signals valid register data on the CmData[31:0] bus by asserting CmXLE* LOW. The exact latency to the data cycle is variable and occurs when a free "slot" is reached in the chips internal Connection Memory state machine (small number of MClk cycles). The CmXLE* signal should be qualified with MClk to insure adequate hold time when latching CmData[31:0]. Once the data is latched into a transceiver and is made available on the CPU-bus-side the register cycle can be terminated to the CPU. As far as the SAR is concerned the register cycle is complete after the CmXLE* LOW pulse.

Figure 6-13 shows a functional timing diagram for a Register Read Cycle.

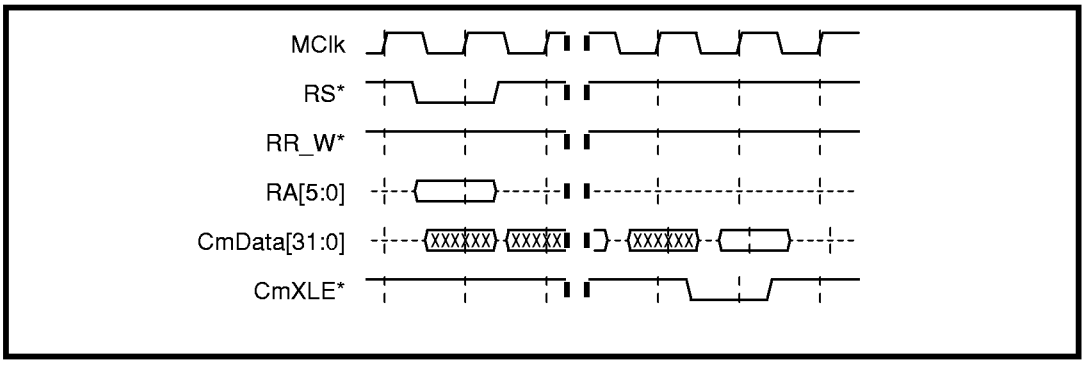


Figure 6-13 Register Read Cycle

Register Write Cycle

A Register write cycle is initiated when the SAR samples RS* LOW on the positive edge of MClk. On this same edge RR_W* must be set LOW and the register address must be placed on RA[5:0]. Also the register data should be output and transferred to the input side of a bus-isolating transceiver. A variable number of clocks later the SAR will reach a free "slot" in it's Connection Memory state machine and will assert CmXOE* LOW. This should turn on the output enable of the transceiver holding the register data to be written to the SAR on CmData[31:0]. On the first positive edge of MClk following CmXOE* LOW the SAR captures the register data and the register cycle is complete from the SAR's perspective.

Note: It is possible to terminate the CPU-side of the register access once the data has been written to a transceiver. This may yield a slight performance boost since the latency to CmXOE* is eliminated. The transceiver logic must be setup so that if the transceiver is "busy" with a "previous write in progress" then subsequent reads or writes are held up (e.g. not started from the SAR's point of view) until the previously started cycle completes on the SAR-side by assertion of CmXOE* LOW. The scheme needs to be transparent to the SAR as it will only accept a new register access after the current one completes.

Figure 6-14 shows a functional timing diagram for a Register Write Cycle.

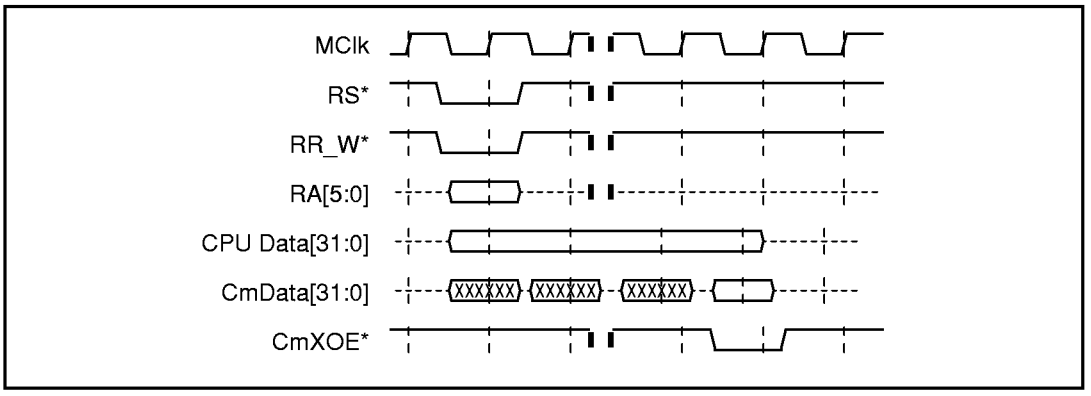


Figure 6-14 Register Write Cycle

The physical interface to the SAR chips is compliant with the Universal Test and Operations PHY Interface for ATM (UTOPIA) standard produced by the ATM Forum. This standard interface allows the SAR chips to connect to many different types of physical interfaces.

Overview of Operation

The physical (PHY) interface is allows parallel data transfer between the PHY device and the SAR. In 622Mbps operation, a 16-bit parallel interface at 40MHz is used. In 155Mbps operation, the interface is 8-bit parallel at 20MHz.

Both the transmit and receive clocks are generated by the SAR. It is the responsibility of the PHY to synchronize its information with these signals. The PHY must accept both transmit and receive clocks from the SAR. Because of this, the PHY layer have rate matching FIFO's.

The UTOPIA standard allows for byte-level and cell-level flow control. The PHY interface on the SAR implements only cell-level flow control. In this flow control scheme, the PHY asserts a signal to the SAR transmit interface indicating that it has sufficient space for an ATM cell. The SAR then transfers a complete cell to the PHY.

Data Formats

622Mbps Data Format

The cell data format for the interface is shown in Table 7-1. The interface is 16 bits wide with Big Endian byte ordering. The first four bytes of the ATM header are passed through the interface in the first two words. The third word is user defined and could be used to pass ATM header check sum (HEC) information. Words 4 through 28 of a cell transfer contain the 48 bytes of ATM cell payload. A complete ATM cell appears on the interface as 54 bytes.

The SAR receive PHY interface normally ignores the two user defined bytes. However, in *raw mode*, these bytes are transferred through the SAR and into buffer memory along with the header and payload.

The SAR transmit PHY interface normally places zeros in the used defined bytes. In *raw mode*, the contents of these bytes are taken from buffer memory along with the header and payload bytes.

Table 7-1 Cell Data Format for 622Mbps Operation

Bit 15	Bit 0
Header 1	Header 2
Header 3	Header 4
UDF 1	UDF2
Payload 1	Payload 2
:	:
:	:
Payload 47	Payload 48

155Mbps Data Format

The cell data format for the interface is shown in Table 7-2. In this case, the interface is byte wide. The first four bytes of the ATM header are passed through the interface in the first four bytes. The fifth byte is user defined and could be used to pass ATM header check sum (HEC) information. Bytes 6 through 52 of the cell transfer contain the 48 bytes of ATM cell payload. A complete ATM cell appears on the interface as 53 bytes.

The SAR receive PHY interface normally ignores the user defined byte. However, in *raw mode*, this bytes is transferred through the SAR and into buffer memory along with the header and payload.

The SAR transmit PHY interface normally places zeros in the used defined byte. In *raw mode*, the contents of this byte is taken from buffer memory along with the header and payload bytes.

Table 7-2 Cell Data Format for 155Mbps Operation

Bit 7 Bit 0
Header 1
Header 2
Header 3
Header 4
UDF
Payload 1
:
Payload 48

Transmit Operation

To begin transmitting, the SAR waits until it clocks **TxClav** on a rising edge of **TxClk**. On the next clock cycle, **TxData, TxSOC,** and **TxEnb*** are asserted. This indicates that the current word is the start of a cell.

On the following cycles, **TxSOC** is deasserted and the remaining cell data is passed across the interface. After a complete cell is passed across the interface, the SAR deasserts **TxEnb*** if it has no more cells to transfer.

If the SAR has another cell to transfer, **TxClav** is checked on the last cycle of the previous cell transfer. If **TxClav** is asserted, the SAR immediately begins transferring the next cell's data. If **TxClav** is not asserted at this time, the SAR deasserts **TxEnb*** and waits to transfer the new cell until it sees **TxClav** on a rising edge of **TxClk**.

The following signals are used:

- TxData[15:0]
- TxEnb*
- TxSOC
- TxClk
- TxClav

A logic diagram of the transmit interface operation is shown in Figure 7-1.

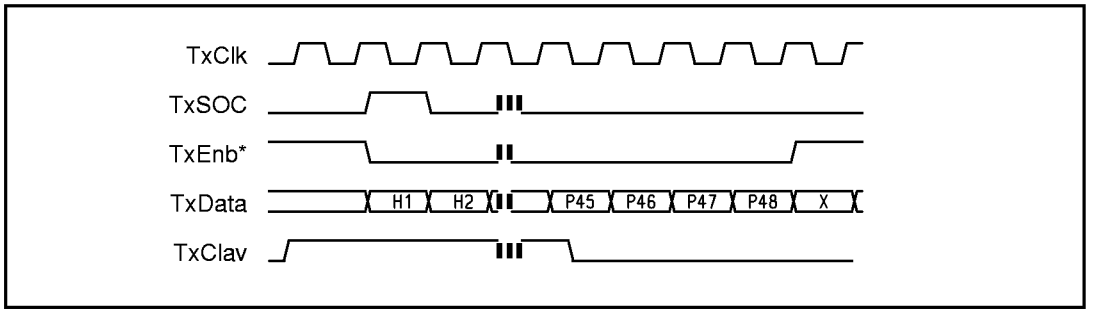


Figure 7-1 Transmit Interface Operation

Receive Operation

In receive operation, the SAR waits until **RxClav** is asserted by the PHY indicating that a complete cell of data is ready to be transferred by the PHY. The SAR then asserts **RxEnb*** to indicate that it will sample **RxSOC** and **RXData** on the next cycle of **RxClk**. Following this cycle, the SAR clocks in data on each cycle until a cell has completed.

The PHY can continue to transfer data to the SAR. At cell boundaries, the PHY must assert **RxSOC** to indicate that the data currently on the interface begins a new ATM cell. If the PHY does not have another cell to transfer to the SAR, it must deassert **RxClav** sometime during the transfer of the current cell.

The following signals are used:

- RxData[15:0]
- RxPrty[1:0]
- RxEnb*
- RxSOC
- RxClk
- RxClav

A logic diagram of the transmit interface operation is shown in Figure 7-2.

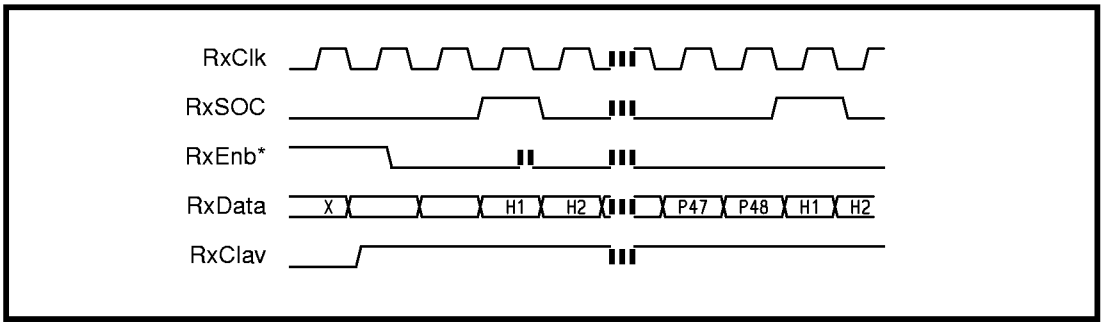


Figure 7-2 Receive Interface Operation

Referenced Documents

This document references the ATM Forum forum standard titled, *UTOPIA, an ATM-PHY Interface Specification, Level 1, Version 2.01*, March 21, 1994.

This chapter deals with the functional operation of the Segmentation device. The sections covered are: Overview, Initialization, AAL functionality, Rate Shaping and Queuing mechanism and Flow Control. The figure below illustrates some of the major components with which the segmentation device works.

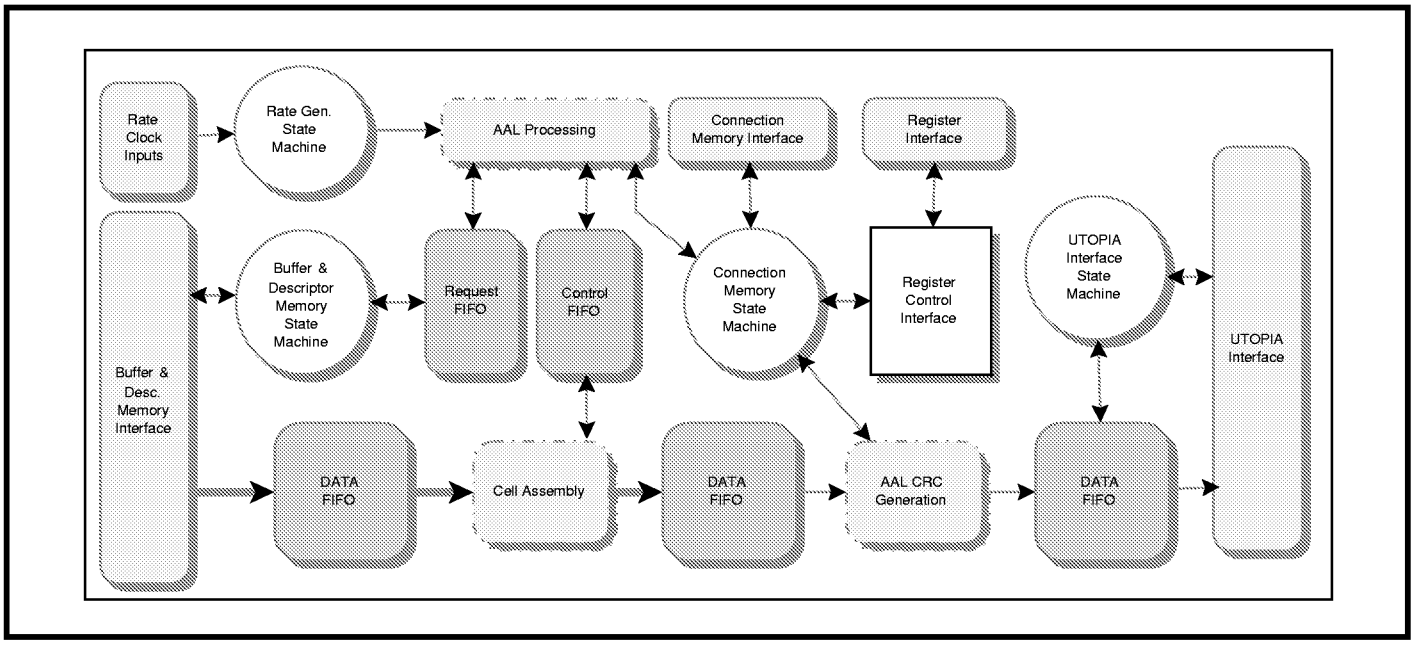


Figure 8-1 Overview of Segmentation Transmit Architecture

Overview of Operation

The function of the Segmentation device is to transfer user data from Buffer Memory where it is stored in a user data PDU format into an ATM cell format to an ATM physical layer. There are many aspects to this data transfer. The primary concerns of the ATM segmentation process are AAL functionality and rate shaping. The first concern deals with the manner in which the user data is to be segmented into ATM cells including integrity protection, sequence numbering, etc. The second concern, rate shaping, provides a mechanism which paces the ATM cell flow to ensure that the source traffic generated does not violate network congestion and flow control expectations. The Segmentation device allows the user to determine on a virtual circuit by virtual circuit basis both the method of AAL segmentation and the traffic shaping or flow control method required.

To perform the segmentation process several different external data structures and internal mechanisms are required.

The external Connection Memory contains a data structure for each Virtual Circuit Connection which is setup and initialized by a microprocessor. The Connection Memory also contains a queue which is used to hold transmit PDU notifications.

The user's data resides in the Buffer Memory. The segmentation device allows user data to exist in noncontiguous buffers.To provide a description of a given user PDU the Descriptor Memory is very closely associated with the buffer memory and through link pointers describes the location and flow of the user PDU.

The internal rate shaping mechanism is clock driven and when activated, segments exactly one ATM cell from each queued Virtual Circuit PDU. When several PDUs are queued for a single Virtual Circuit only one PDU, the oldest queued, is available for segmentation. Each Virtual Circuit is associated with one of 16 possible rate queues by setting the rate control in its Connection Table Entry.

System Unitization

Memory Subsystems

The initialization of the three memory subsystems is discussed in the following section.

The Buffer Memory is logically divided into blocks of 256, 512, 2048 or 4096 bytes. There are no required initializations in this memory since the control structures for it are maintained in Descriptor Memory.

The Descriptor Memory is a separate memory (due to bandwidth and latency requirements at 622Mbps). When the segmentation device is used at lower ATM data rates, this restriction need not apply. Each block of the Buffer Memory, regardless of which size is used, has a corresponding Descriptor in the Descriptor Memory. The Descriptor entries are 32 bits each and are set up to contain lengths and pointers. On initialization the descriptors must be set up to form a single linked list with a head and tail pointer. The head pointer of this list is used by the system to acquire data buffers in the Buffer Memory for transmit PDU's and the tail is used by chip to automatically return consumed buffers after transmission.

Connection Establishment

Before user data can be properly transmitted an ATM connection must be established by setting up a Connection Memory Table Entry (CTE) in Connection Memory. Each established connection has a CTE which contains all the pertinent information for that connection. The systems access to the CTE is done through the use of a Connection Identifier, or CID. The CTEs contain the AAL segmentation type, the rate queue to be used, the VPI/VCI address and other information. These entries are set up by the local microprocessor and are described in the Memory Organization section of chapter 6.

Queuing PDUs for Transmission

After connection setup, when the system needs to transfer a user data PDU the following steps are taken.

The user data is transferred to the Buffer Memory (using the descriptors in Descriptor Memory) from the head of the free list.

The head and tail pointers of the user data and the CID for the desired connection are written to the S_TXRHT and S_TXRCID through a register accesses.

When the user data has been transmitted notification and status is placed on a notification queue in the Connection Memory and the system is notified via an interrupt that the queue is not empty.

Details of the segmentation process and rate shaping are in subsequent sections.

Scheduling a User Data PDU for Transmission

To set up a user data PDU for transmission, first the user data PDU is moved to the local Buffer memory then the buffer pointer information is set up in the Descriptor memory.

The head and tail pointer of the PDU in buffer memory is put in the S_TXRHT register and the Connection IDentifier is put in the S_TXRCID register which queues the PDU for transmission. These actions schedule the PDU for transmission.

AAL Operations

There are several ATM Adaptation Layer (AAL) protocols. The segmentation device supports AAL1, AAL3/4 and AAL5. Also supported are modes which transmit only 48 user bytes with header information from the CTE (payload mode) or 53 user supplied bytes (raw mode).

When a particular Virtual Circuit is selected to have a cell extracted (segmented) from the user data and transmitted the following actions are taken by the AAL processor:

- The pertinent CTE is accessed to obtain all current connection information.
- The Memory interface is requested to begin a data transfer based on the last pointer values in the CTE.
- The payload is assembled depending on the AAL type of the connection. For example, a sequence number may need to be inserted.
- Any update and CRC is performed as the user data is assembled.
- The appropriate ATM cell header is extracted from the CTE to provide VPI/VCI address.
- The cell is assembled complete with the ATM header, payload, and as required sequence number and CRC.
- The cell is placed in the FIFO for the physical connection.
- An updated CTE is placed into Connection Memory.

- Return any freed Descriptors to the tail of the free list.
- Post any required transmit completion notification.

The above description is somewhat rudimentary since many of the actions are pipelined and occur simultaneously. Other actions such as obtaining user data from the Buffer Memory in the case of multiple buffers has not been given.

AAL Processing

The Segmentation Unit supports the segmentation of user data into AAL1, AAL3/4, and AAL5 ATM cell data formats. During the segmentation process partial results and AAL state information for each VCC is stored in the VCCs CTE in the Connection Memory. The data structure maintained for each AAL type is as follows:

AAL Type 1

```
seqNum          3-bit current sequence number
```

AAL Type 3/4

```
lengthOut,      // 16 bits used for length field in
                // tail of cpcs-pdu
seqNumOut,      // 4 bit sequence number.
bTagOut,        // 8 bit beginning tag.
sendStateOut,   // 1 bit state of sender.
```

AAL Type 5

```
lastCrcOut,     // 32 bit crc.
lengthOut,      // 16 bit accumulated pdu length.
```

Null AAL Mode

In Null AAL mode 48 bytes of payload are sent out without any processing. The 4 byte header information is copied from the CTE (the framer computes the HEC).

Raw Cell Mode

In the Raw Cell mode, all 54 bytes of data transferred to the physical interface are read directly from buffer memory. This 54 bytes of data must be 64-byte aligned in buffer memory. The header information in the corresponding CTE is ignored.

Rate Shaping

One of the fundamental traffic concepts of ATM uses the law of large numbers which provides a high level of assurance that a large number of traffic sources may be statistically multiplexed without cell loss due to congestion. To further aid the congestion issue ATM employs a policing mechanism described by the "leaky bucket" algorithm to police VCC cell flows entering an ATM switch.

Some traffic sources are self-shaped due to their characteristics. Other sources, especially data, can be very bursty and could easily result in back to back cell traffic at the link rate.

The Segmentation unit employs 16 independent rate generators. Any given traffic source, or VCC, can be assigned to a particular generator. To guarantee high accuracy over a long period of time there are four external clock sources each with an internal 4-bit prescaler of 2^n. Each rate generator may be driven off of any one of the clock sources. This is shown in following figure.

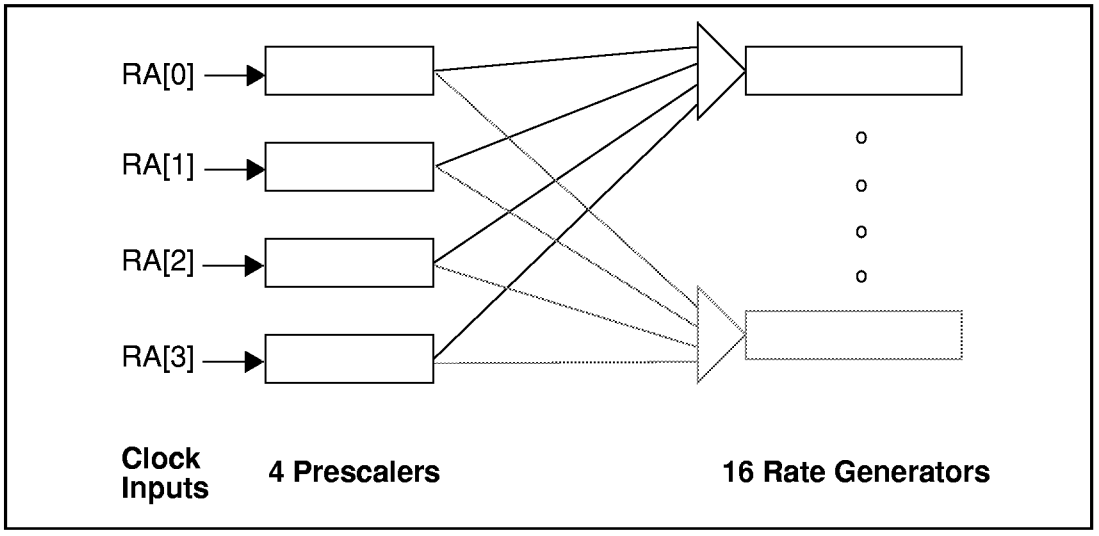


Figure 8-2 Rate Generators and Input Reference Clocks

Associated with each generator is a data queuing mechanism. As user data for a VCC is presented to the Segmentation device it is queued to its assigned rate generator. If there is already PDUs queued for that VCC the new PDU will be queued behind the current PDU(s). Only one PDU per VCC is segmented and transmitted at a time. The assigned rate generator is found by using the user supplied CID from the appropriate CTE in the Connection Memory. The rate generator queues are also maintained in Connection Memory.

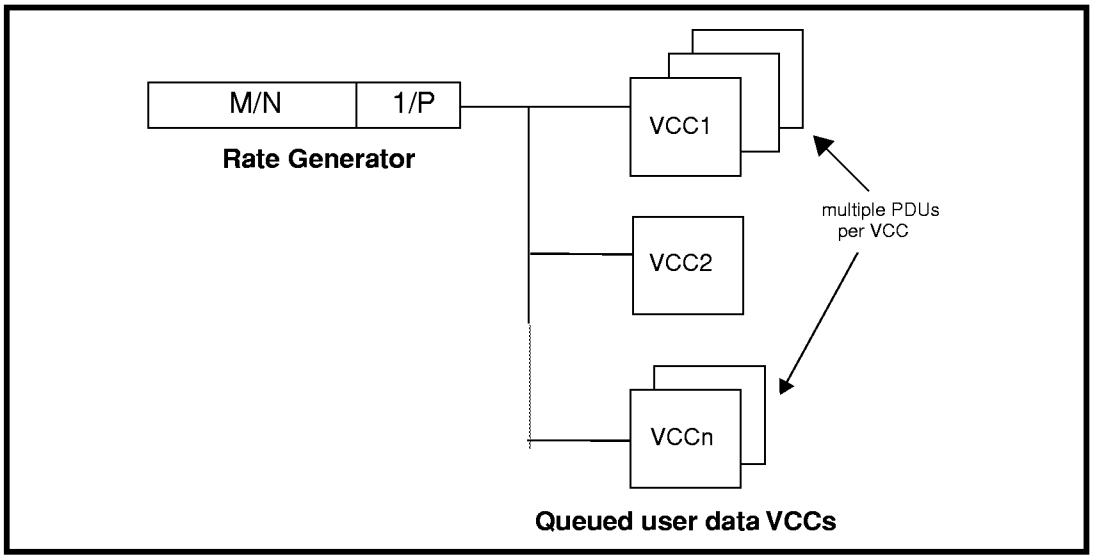


Figure 8-3 Rate Generators and the VCC Data Queues

In addition to the clock source each rate generator is assigned the parameters, M, N and P. The M and N parameters feed a fractional divide logic which guarantees that at every N prescaled cycles of the input clock exactly M cells will be evenly generated as possible from the source data PDU. The P parameter is a peak rate parameter which insures that the minimum inter-cell spacing is maintained.

The following are examples of how the rate generator parameters might be set in order to obtain a specific data rate:

Examples

How should I program the rate generation mechanism to produce 1Mcell/s using a 33MHz reference clock?

1. Program the prescaler to 1/32 and the rate generator to 32/33.

- Byte-wide data synchronized to clock C is deposited into the SAR Buffer Memory. How do I program the rate generation mechanism so that data is transmitted from the SAR at exactly C Bytes/s? Assume that the ATM data is transmitted in AAL5 format using 512-Byte Protocol Data Units (PDUs).

2. Achieving exact synchronization of the ATM data transmission rate with data produced by clock C requires the use of clock C as a rate generator input source. The transmission of a 512-Byte AAL5 PDU requires 11 ATM cells. The data for the 512-Byte PDU is produced in 512 C clocks. Exact synchronization of the ATM stream to clock C is achieved if the rate generator is programmed to 11/512.
3. The same effect can be achieved by programming the prescaler to divide by 32 and programming the rate generator to 11/16.

The 12-bit M/N fractional multiplier circuitry is designed particularly for video server applications where there are seemingly conflicting needs. First, the source traffic is stored and is available in large blocks and could, if unthrottled, fill the link so that congestion control at the Peak Cell Rate is needed. Secondly, the receiving stations typically have limited storage resources and so require the traffic to be fed at a usable rate neither too fast to cause overflow nor too slow to cause breaks in the program material. The input clocks guarantee the accuracy of the mechanism while the M/N logic assures an average cell delivery time.

One of the observed behaviors of rate shaping multiplexing mechanisms is that they inadvertently introduce jitter at the traffic source. The source jitter has two causes. The use of multiple rate generators and the completion of interleaved queued PDUs. The first source of jitter is due to the fact that multiple rate generators will eventually conflict and the higher priority generator will delay the activity of the lower priority one. The second jitter source is that on a given rate generator when some connections complete and are removed from the queue, the structure collapses over the idle connection and the remaining elements have a new, shorter, time inter-relationship.

The latter jitter problem is relatively small and can be easily accounted for but should be recognized. The first jitter source is alleviated to a high degree in the

design of the Segmentation unit by use of the M/N logic. When any rate generator is pre-empted by a higher priority generator it accrues credits. The credits so accrued allow the rate generator to transmit its traffic at a higher rate, not to exceed the 1/P peak rate to catch up.

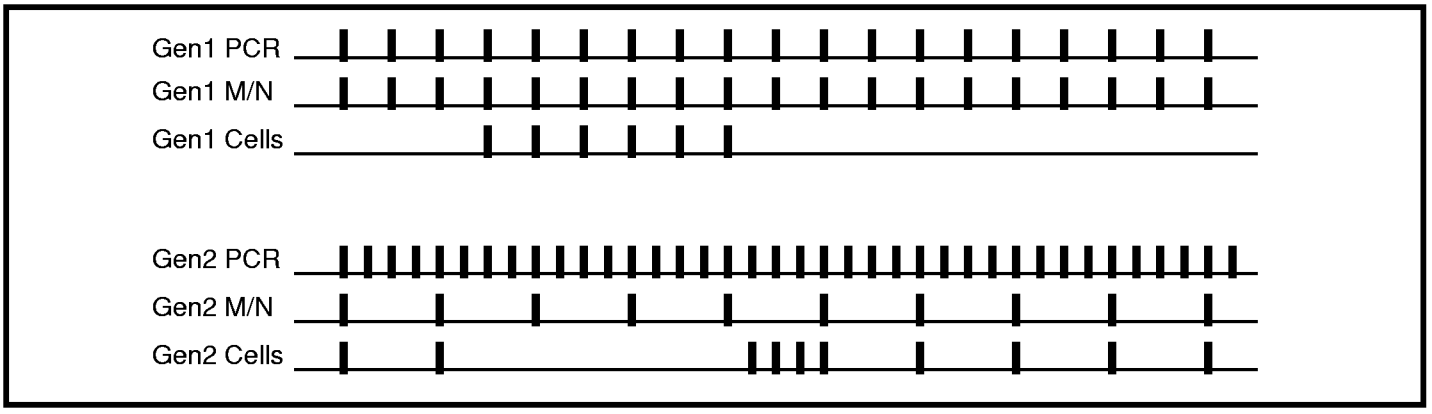


Figure 8-4 Rate Generator Pre-emption and Catchup

In Figure 8-4 two rate generators are operating at different PCR and M/N values. Generator 1 has a higher priority than Generator 2. In this figure the Cell flow of Gen2 was pre-empted by a burst of cells by Gen1. Gen2 was able to "catch up" at its PCR and then resume normal cell generation.

An additional feature may be employed on a VCC by VCC basis to alter the rate of cell production. This feature uses a skip count to alter the rate generator's behavior. Each VCC has an associate "Skip Count" which means that cells will only be generated every Skip Counts worth of rate generator ticks. A Skip Count of one reduces the cell output to1/2, that is every other tick, a Skip Count of three produces cells every three ticks and reduces the cell rate by 1/4 and so forth.

One of the possible uses of the Skip Count Feature would be to use it in conjunction with a rate-based flow control method if the reduction to 1/n+1 provides acceptable granularity.

Virtual Connection Selection

The Queue selection/rate shaping mechanism is free running and forms the basis for determining when a cell of user data will be processed and transmitted.

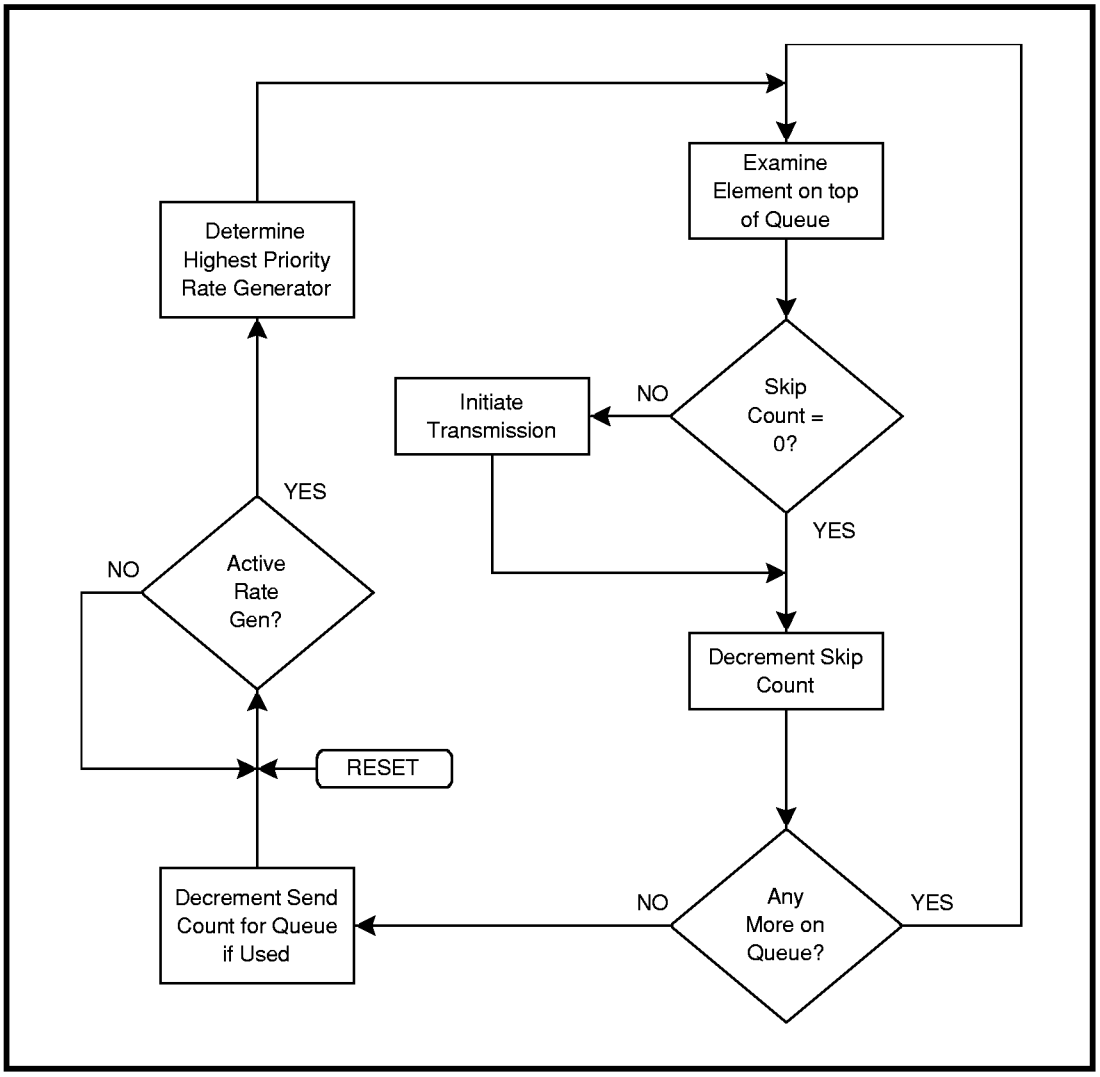


Figure 8-5 Transmission Queue Selection Scheme

1. Determine the highest priority active rate queue that has data queued.
2. Each virtual circuit, or rate queue entry, has a skip count. If the skip count is not zero we decrement it and when the rate queue ticks we don't send data, otherwise the virtual circuit ID is passed on (which schedules the a cell for transmission).If the circuit does not have any data left in buffer memory to be transmitted it is removed from the rate queue.
3. Examine each rate queue entry until all have been serviced.
4. When all circuits in a queue have been serviced, decrement the send credit, if used, of that rate generator and reset the head of the queue.
5. Proceed to the next highest priority active queue and continue as above.

Flow/Congestion Control

The Segmentation Device currently supports a rate based shaping mechanism for Flow/Congestion Control. This mechanism supports the ATM Forum UNI 3.0 specification for traffic management.

The current ATM Forum ABR algorithm has been finalized too late for the current revision of the SAR to include. A future revision is planed that will incorporate full support for it. A limited degree of ABR service is possible using the curent features of the SAR. This entails re-programming a particular rate generator which is used for traffic of a given class of service to a different value on receipt of a Resource Management cell. The Skip Count scheme could be used to implement an easily changeable reduction technique.

Support for a Credit-based scheme is not currently implemented as an accepted standard has not been reached.

This page left blank intentionally

This chapter deals with the functional operation of the Reassembly device. The sections covered are: Overview, Initialization, Address Translation, AAL PDU Assembly, OAM Management and Flow Control. The figure below illustrates some of the major components with which the Reassembly device works.

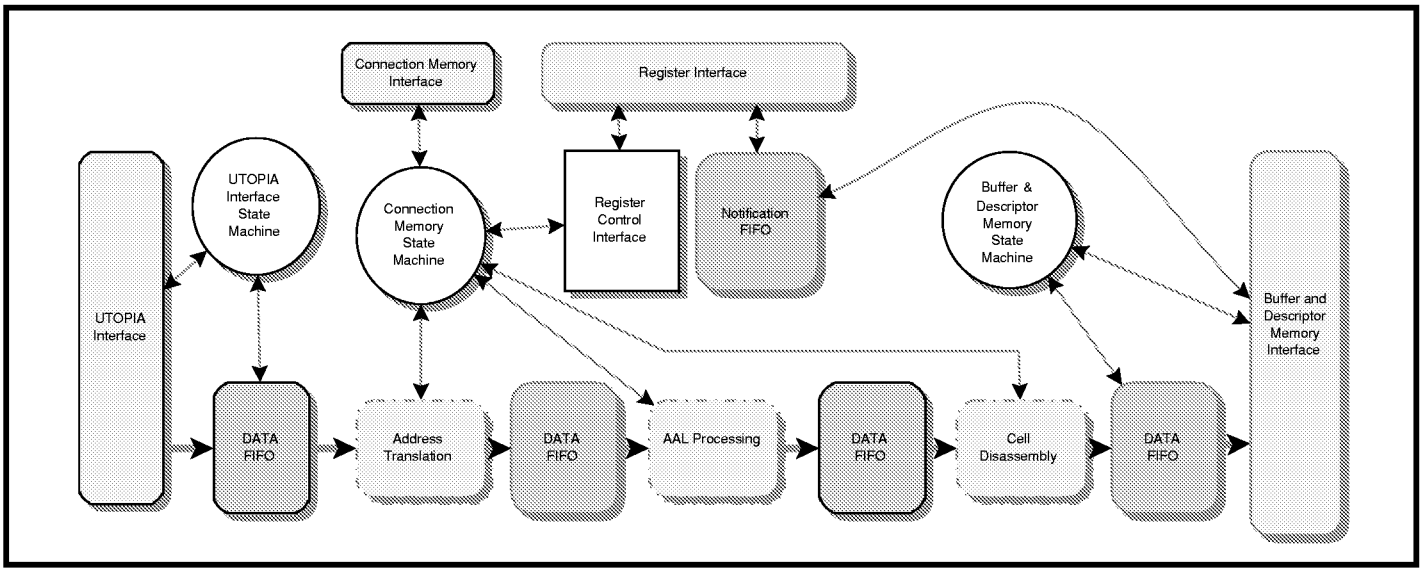


Figure 9-1 Overview of Reassembly Receive Architecture

Overview of Operation

The function of the Reassembly device is to assemble user data PDUs from ATM cells received from the physical layer. The major aspect of this data transfer is AAL functionality which deals with the manner in which the user data is to be reassembled from discontinuous ATM cells including integrity protection, sequence numbering, etc. The Reassembly device allows the user to determine on a virtual circuit by virtual circuit basis both the method of AAL reassembly.

To perform the reassembly process several different external data structures and internal mechanisms are required.

The external Connection Memory contains a data structure for each Virtual Circuit Connection which has been setup by the microprocessor. The Connection Memory also contains a queue which is used to queue notifications to be able to report to the microprocessor the status on the transfer and reassembly of each user PDU.

The user's data resides in the Buffer Memory. The reassembly device allows user data to exist in noncontiguous buffers. To provide a description of a given user PDU the Descriptor Memory is very closely associated with the buffer memory and through link pointers describes the location and flow of the user PDU.

The Connection memory also contains four notification queues. Each VCC may be assigned to have its notifications placed on one of these queues. One of the queues, Receive Queue 0 is the automatic recipient of special notifications such as receipt of F5 OAM cells. This overview is superficial, more details are provided in the following sections.

Initialization

Memory Subsystems

The system initialization dealt with here centers on setting up the structures in the three memory subsystems.

The Buffer Memory is logically divided into blocks of 256, 512, 2048 or 4096 bytes. Only data resides in the Buffer Memory. The structures which define the linkage and usage of the Buffer Memory blocks are contained in the Descriptor Memory.

The Descriptor Memory is a separate memory from the Buffer Memory due to bandwidth and latency requirements of accessing both the data and linkage information at 622Mbps. When the reassembly device is used at lower ATM data rates, this restriction need not apply. The Descriptor Memory is a mirror of the Buffer Memory. That is, each block of the Buffer Memory, regardless of which size is used, has a corresponding Descriptor in the Descriptor Memory. The Descriptor entries are only 32 bits each and contain link pointers. On initialization all of the descriptors must be set up to form a linked list with both head and tail pointers. The head pointer of this list is then used by the Reassembly device to acquire data buffers in the Buffer Memory when user data is received. When the system has processed the received user data the Buffer Memory blocks may be freed and made available for reuse by placing the associated descriptor with an updated link pointer onto the free list tail.

As user data is received the payload is extracted from the ATM cell and transferred to buffer memory. Since the cells for various connections may be received interleaved this process is highly discontinuous and several partial PDUs could exist in the buffer memory. Intermediate results on each VCC cell flow is maintained in the CTEs in Connection Memory. Partial results include CRCs, sequence numbers and current buffer pointer and size of buffer.

Connection Establishment

Before user data can be properly received an ATM connection must be established. Connection establishment is beyond the purview of this document, however, as it pertains to the data structures used by the Reassembly device that connection establishment is best shown in the Connection Memory Table Entry (CTE). In connection memory each established connection has a CTE which contains all the pertinent information for that connection. The systems access to this is done through a Connection Identifier, or CID. The CTEs contain AAL type, buffer pointers and other information. These entries are set up by the local microprocessor and are described in the Memory Organization chapter.

AAL Handling

The Reassembly Unit supports the reassembly of ATM cells of types AAL1, AAL3/4, and AAL5 into user data PDUs. During the reassembly process partial results and AAL state information for each VCC is stored in the VCC's CTE in the Connection Memory. The data structure maintained for each AAL type is as follows:

AAL 1

seqNum	3-bit current sequence number
maxCount	16-bit receive size (bytes) for pdu notification
workCount	16-bit current-receive size

AAL 3/4

length,	// 16 bits PDU length.
recvState,	// 1-bit state of receiver
	// either 1: in process of reassembly
	// or 0: Idle.
seqNum,	// 4-bit sequence number.
bTag,	// 8-bit beginning tag.

AAL 5

lastCrcOut,	// 32-bit crc.
lengthOut,	// 16-bit accumulated pdu length.

Null AAL Mode

Null or Raw AAL cell flows have no pertinent AAL information stored in the associated CTE.

Pipeline Processing

The reassembly device uses a pipeline approach to process received ATM cells. During each cell time the Connection Memory must be accessed for cell treatment information including the AAL treatment type, any current data buffer pointers are extracted along with any previous partial PDU results such as sequence numbers and CRCs. When the payload has been processed the CTE is updated. The Connection Memory bandwidth is almost completely utilized.

The cell processing only takes place on valid VCs, all invalid or unassigned VC data is flushed. The incoming cell arrives in the FIFO of the Reassembly device where the following procedures take place; the 6-byte cell header is stripped off and the VPI/VCI information is used to perform an address translation with an indexed lookup into the Connection Table in Connection Memory to extract a CTE for the cell. The cell is processed depending on the AAL requirements (CS-PDUs, CRCs, and sequence number checking, etc.), the data payload is assembled in the Host Buffer, and finally the Local Processor is notified when completed data traffic needs to be serviced.

When the payload data is transferred to Buffer Memory storage it may have to be concatenated with previously received data. Also, at the beginning of each new PDU and when current Buffer Memory blocks are filled, new Buffer Memory blocks must be allocated for the PDU.

Cell Header Extraction

The first cell processing step is to extract the ATM Cell header. The VPI and VCI fields will be used during the address translation phase. The PTI bits could indicate immediate cell handling information such as F5 cells, etc. The cell header is passed on to the address translation block while the payload data is carried in a parallel FIFO.

Address Translation

The function of address translation is to acquire an index into the Connection Table so that the context information for a given connection can be located. Figure 9-2 shows the indexing method implemented in the VCATT to translate the VPI/VCI fields of the cell header to access a specific Connection Table Entry. The "root" of the VCATT is programmed in the R_ATR register and all other entries of the hierarchical data structure are placed in Connection Memory by the user (typically by writing to Connection Memory with a local processor). All entries in the data structure (including the value programmed into the R_ATR) have the format defined in section chapter 5).

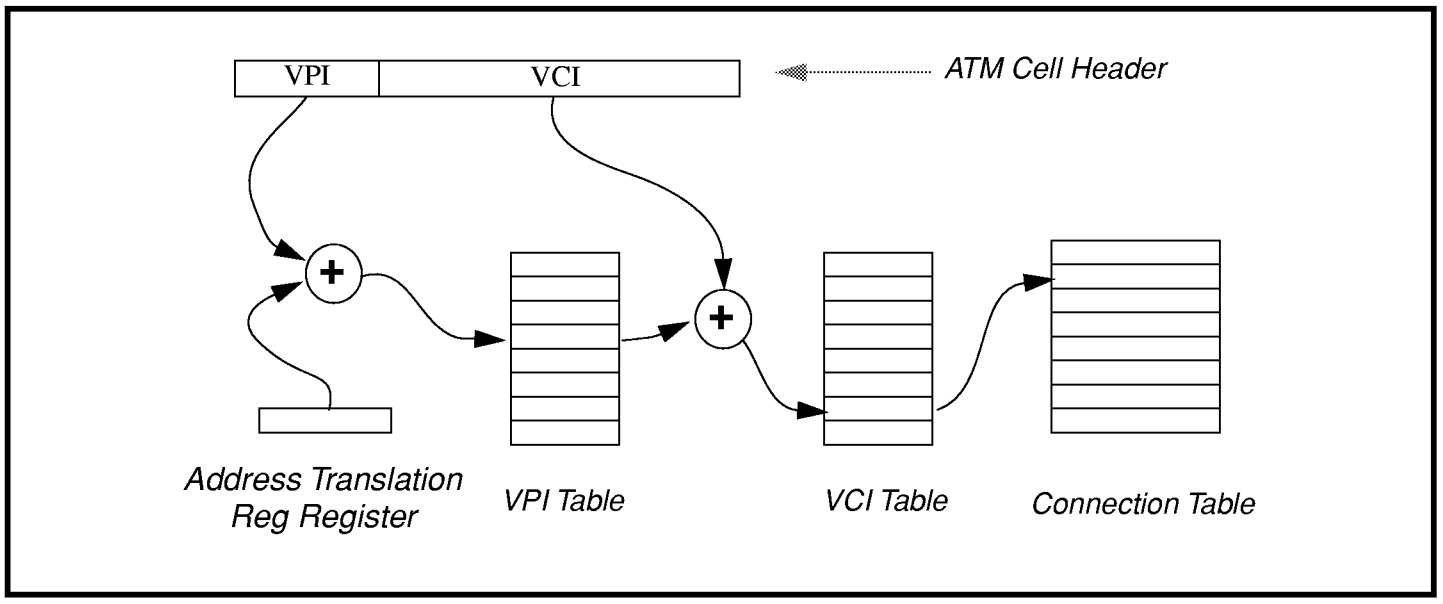


Figure 9-2 VPI-VCI Lookup Translation

In the Figure 9-2 example each VPI table entry contains both a base address to a VCI table and a bound for the size of that table. The VCI table contains the Connection IDs (CIDs) that are being sought. Other configurations are possible such as mapping a VPI directly to a CID. In that case the VCI value would not be used in address translation (but all cells sent to the chosen VPI would be considered as belonging to the same connection including any AAL processing).

Steps in the Figure 9-2 example:

Step 1. The VPI from the Cell Header is compared against the bounding value of the VPI Table as programmed in bits 15-0 of the R_ATR register. If the VPI is greater -- the cell immediately fails address translation. Otherwise the VPI is added to the VPI Table base address (determined by bits 29-16 in the R_ATR register) and used to address a VPI table entry.

Step 2. We assume the Link Continuation bit is set in the VPI table entry and contains the base address of a VCI table. In this case the VCI from the Cell Header is compared against the bounds for the VCI table and if valid then the VCI is combined with the base of the VCI table to access an entry in the VCI table.

Step 3. The VCI table entry (assuming an AAL type other than Type 3/4) contain the CID that corresponds to the VPI/VCI address.

Additional step shown in Figure 9-3 for AAL3/4:

Step 4. Assuming the Link Continuation bit is set the entry contains a pointer to the base of a MID Table. The AAL3/4 MID in the cell is compared to the bound value and if within range the MID is added to the table base address to acquire the CID.

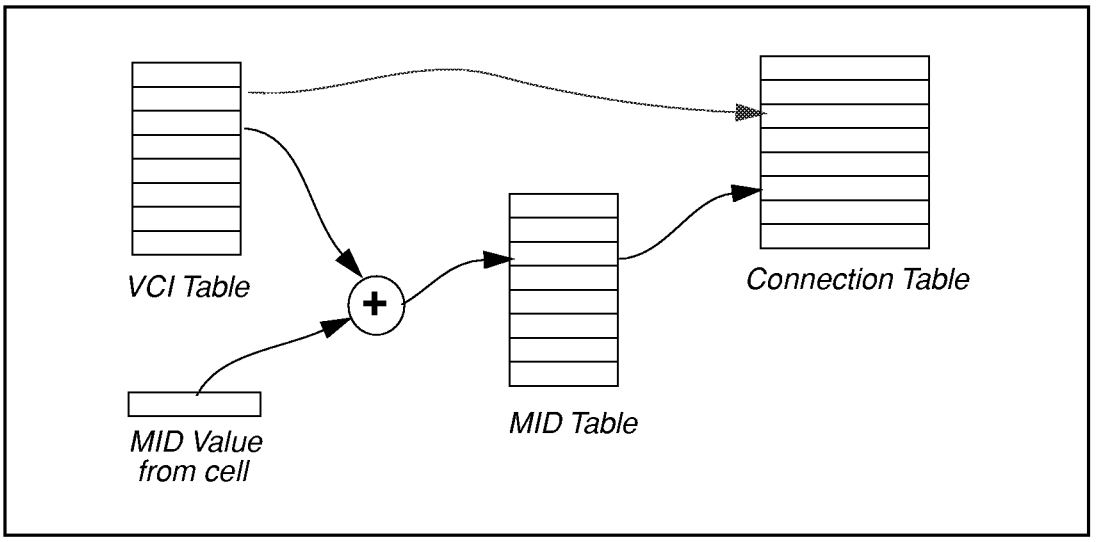


Figure 9-3 VCI or an MID Lookup Translation

Any base/bound violation results in an "address translation failing" cell which is discarded or optionally placed on the Receive Management Queue for debug (see the R_MODE register).

AAL Processing and Buffer Assembly

Upon completion of the AAL processing of each incoming cell, the extracted data needs to be reassembled into data blocks in the Buffer Memory to be passed up to the Host. Buffer Memory, with respect to reassembly, is organized as a series of individual buffer queues, one for each active connection, and a free buffer list.

As described in chapter 5, each of the buffer chains, as well as the free list has a head and a tail pointer associated with it. These pointers reside in the Buffer Index Memory and have a one to one correspondence to an associated block in the Buffer Memory. Each of the pointer entries contains a next pointer and a length of the data in that particular block. The last pointer in each of the free lists contains a null pointer indicating end-of-list. As each cell is processed, it is put into the buffer at the end of its appropriate VCC buffer chain. The length field of the associated pointer is then updated to reflect the addition of the cell to the buffer. When the last byte is written into the current buffer a new buffer is allocated to the buffer chain from the head of the free list, which is stored in a register of the reassembly device. This is done to ensure that there is always a buffer ready and if there is not, the reassembly device notifies the host.

First, the head pointer to the free list is updated to the value contained in the Next Pointer field of its pointer. Then the current pointer location is placed as a pointer in the Next Pointer field of the last buffer in the VCC buffer chain. When all data PDU and information is complete, the host is notified that the particular queue needs to be serviced.

Host PDU Completion and Notification

The Reassembly device supports four receive notification queues which are maintained in the Connection Memory. Each VCC may be assigned to a particular receive queue. One of the queues, RQ0 always receives certain management cell flows such as F5 and Resource Management cells indicated by PTI bit settings. If enabled, Unassigned or Invalid cells are also received on this queue. However, other cell flows may also be assigned to this queue but must be explicitly assigned in the appropriate CTE for that VCC in Connection Memory.

Notification that a receive queue is not empty is accomplished through the use of individual maskable interrupts. The interrupt remains on whenever there is at least one element in the queue waiting to be extracted by external logic.

The Receive Notification Queue size may be configured for 128, 512, 2048, or 8192 notifications. The individual notifications are extracted from the queues via register read accesses.

Management Cell Flow Reception

Special cell flows are reported. The following is a list of special cells which are reported on the Receive Management Queue:

- F5 OAM cells, PTI bits = 100 or 101.
- Resource Management cells, PTI bits = 110.
- Reserved for future use cells, PTI bits = 111.
- F4 OAM cells, VCI address of 3 or 4, regardless of PTI bit settings.
- Unassigned and Invalid cells of VPI/VCI = 0.

Flow Control Management

As of the May 1994 ATM Forum meeting is was proposed that the Resource Management cell be used to carry flow control information regardless of the flow control mechanism finally proposed by the ATM Forum. The Reassembly device supports this proposal by reporting Resource Management cells on the receive Management Queue where they may be interdicted by software.

This page left blank intentionally

This chapter contains the AC & DC Characteristics of the Toshiba TC35860F and TC35681F parts. **THIS INFORMATION IS PRELIMINARY.**

AC Characteristics

All timing parameters given are with respect to the following operating conditions unless otherwise noted:

- V_{DD} = 5.0 Volts +/- 5%
- Ta = 0 to +70°C

Table 10-1 Reassembly Chip Output Timing with loads at 10pF & 50pF

Parameter	Min (load of 10pF)	Max (load of 10pF)	Min (load of 50pF)	Max (load of 50pF)
CmXOE* valid	3	11	4	17
Clock to CmXLE* valid	3	11	4	17
Clock to CmWE* valid	3	10	3	13
Clock to CmOE* valid	4	14	5	16
Clock to CmAddr[18:0] valid	5	16	5	17
Clock to CmData[31:0] valid	4	14	5	18
Clock to CmData[31:0] High-Z	4	14	4	17
Clock to CmPar[3:0] valid	4	18	5	20
Clock to CmPar[3:0] High-Z	4	14	5	17
Clock to DmR_W*	4	13	5	18
Clock to DmR*	4	13	5	18
Clock to INT*	5	18	6	21
Clock to BmR*	4	14	5	18
Clock to BmSpan	4	14	4	17
Clock to BmIdle	4	14	4	17
Clock to BmWE*	4	14	4	17
Clock to BmDS*	4	14	4	17
Clock to BmA_D valid	4	15	5	18
Clock to BmA_D High-Z	4	18	4	18
Clock to BmPar[7:0]	–	–	–	–
Clock to RxEnb	–	20	–	25

Table 10-2 Segmentation Chip Output Timing with loads at 10pF & 50pF

Parameter	Min (load of 10pF)	Max (load of 10pF)	Min (load of 50pF)	Max (load of 50pF)
Clock to CmXOE* valid	4	14	5	18
Clock to CmXLE* valid	5	15	5	19
Clock to CmWE* valid	3	10	3	13
Clock to CmOE* valid	5	16	6	18
Clock to CmAddr[18:0] valid	5	18	6	18
Clock to CmData[31:0] valid	4	17	5	20
Clock to CmData[31:0] High-Z	4	14	5	19
Clock to CmPar[3:0] valid	–	–	5	20
Clock to CmPar[3:0] High-Z	–	–	5	19
Clock to DmR_W*	5	15	6	20
Clock to DmR*	4	12	5	17
Clock to INT*	5	19	7	21
Clock to BmR*	4	13	5	17
Clock to BmSpan	3	11	4	16
Clock to BmIdle	4	14	4	17
Clock to BmDS*	3	11	4	16
Clock to BmA_D[63:0] valid	4	17	5	19
Clock to BmA_D High-Z	5	15	5	18
Clock to TxData[15:0]	4	13	5	18
Clock to TxEnb	4	11	4	17
Clock to TxSOC	4	12	5	17

Table 10-3 Reassembly Chip Input Timing

Parameter	Setup (WCOM)	Hold (WCOM)	Setup (BCOM)	Hold (BCOM)
RxSOC	7	2	2	1
RxData[15:0]	6	4	2	2
RxClav	2	0	1	1
RxPrty[1:0]	–	–	–	–
RS*	1	2	0	1
RA[5:0]	1	2	0	1
RR_W*	1	2	0	1
DmG*	2	1	1	1
BmG*	2	1	1	1
CmPa[3:0]r	3	2	1	1
CmData[31:0]	3	2	1	1
BmA_D[63:0]	1	2	1	1

Table 10-4 Segmentation Chip Input Timing

Parameter	Setup (WCOM)	Hold (WCOM)	Setup (BCOM)	Hold (BCOM)
TxClav	0	4	0	2
RS*	1	3	0	1
RA[5:0]	0	2	0	1
RR_W*	1	2	0	1
DmG*	1	2	0	1
BmG*	1	2	0	1
CmPar[3:0]	–	–	–	–
CmData[31:0]	1	4	0	2
BmA_D[63:0]	2	2	1	1

DC Characteristics (Ta = 0 - 70°C)

Table 10-5 DC Characteristics

Symbol	Parameter	Min.	Typ.	Max.	Unit
V_{DD}	Power Supply Voltage	4.75	5.0	5.25	Volt
V_{IH}	Input High Voltage	2.0	–	V_{DD}+0.5	Volt
V_{IL}	Input Low Voltage	-0.5	–	0.8	Volt
V_{OH}	Output High Voltage I_{OH} = 4mA	2.4	4.75	V_{DD}	Volt
V_{OL}	Output Low Voltage I_{OL} = 4mA	0	0.2	0.4	Volt
I_{IL}	Input Low Current	–	–	TBD	uA
I_{IH}	Input High Current	TBD	–	–	uA
C_{IN}	Input Capacitance	–	–	7	pF
C_{OUT}	Output Capacitance	–	–	7	pF
I_{DDO}	Operating Current	–	–	TBD	mA
I_{DDS}	Standby Current	–	–	TBD	uA

AAL Functions

Readers should consult the ITU-T or ANSI standards for AAL protocol definitions. This section summarizes the supported functions.

AAL0

This mode allows for the segmentation of user data into 48-byte cell payloads. The payloads are filled verbatim from Buffer Memory without adding any control information. The ATM cell header is built, like all other AAL types except "raw", from the template placed in the Connection Table Entry. The incoming 48-byte payloads are transferred to Buffer Memory without alteration. No error checking is performed on the user payloads.

AAL1

The SAR provides the AAL1 Unstructured Data Transfer (UDT) service. This service adds a one-byte control field to outgoing payloads and includes a sequence number. The data portion of the payload is therefore always 47 bytes. One the receive side the sequence number is checked (along with a CRC which protects it) and stripped from the data stream.

In the UDT service no control information is present to delineate PDU boundaries. The data would typically be a continuous stream (e.g. circuit data rather than packet data). However, the notification mechanism of the SAR requires some amount of data contained in one or more buffers to be posted as a notification. The user has two options for AAL1 notification. The first it to program the "maximum receive length" field in the Connection Table Entry. The second is to use the "Receive Purge" command to force all accumulated data on a connection to be posted.

When the user is notified of data received the accompanying status indicates if a sequence error occurred on any of the cells that comprise the data block being posted.

The SAR does not provide the aal1 Structured Data Transfer (SDT) service which uses P-format payloads to track PDU boundaries. It may be possible to implement this service in software.

The SAR does not provide the AAL1 Synchronous Residual Timestamp algorithm.

AAL3/4

The full ITU-T AAL3/4 protocol is implemented by the SAR. This includes MID support and CRC generation/checking. All control fields including CRC are added/removed internally in the SAR and do not reach Buffer Memory.

AAL5

The full ANSI/ITU-T AAL5 protocol is implemented including CRC generation and checking. All control fields including CRC are added/removed internally in the SAR and do not reach Buffer Memory.

AAL Raw

This mode bypasses the AAL protocols and allows raw ATM cells to be transferred to or from Buffer Memory. This mode therefore bypasses the normal segmentation/reassembly function, but allows for diagnostic or special functions that require bypass of the standard protocols. In this mode the ATM cell header is taken directly from Buffer Memory when performing Segmentation (e.g. the entire cell needs to be placed in Buffer Memory, not just higher-layer data).

Introduction

This section describes the general sequences for initialization and data transfer and is intended to help tie together the more detailed software-related information in the other sections.

Chipset Initialization

The Segmentation device and Reassembly devices are initialized with the following procedure:

1. **Reset and configure the devices.**

   The Segmentation and Reassembly devices are reset and configured through the "Master Control Registers" (MCR). The Master Control Register (MCR) is initialized with basic operational parameters of the chipset during the reset process. This includes parity enable/disable, the bus size of the UTOPIA physical layer interface (must be set to 16 bits for 622Mbps operation), and the size of buffers in Buffer Memory (which can be 256, 1024, 2048 or 4096 bytes).

2. **Initialize the receive buffering.**

   The "Free-List Head" register in the Reassembly device is initialized with a pointer to a chain of receive buffers in Buffer Memory. These buffers must be chained together by writing to a corresponding set of descriptors in Descriptor Memory. The last buffer in the chain must have its forward pointer marked as "end-of-chain". The Reassembly device will automatically acquire buffers from the head of this chain as needed to hold virtual circuit data. The local processor must return consumed buffers to the tail of this chain in order to keep the receive function active.

3. **Initialize the transmit buffering.**

   This is accomplished by loading the "Free-List Tail" register in the Segmentation device with a pointer to the tail of a free chain. These buffers must be chained together by writing to a corresponding set of descriptors in Descriptor Memory. As the Segmentation device transmits virtual circuit data it automatically adds consumed (transmitted) buffers to the tail of this chain. The local processor acquires transmit buffers from the head of this chain.

4. **Initialize the Segmentation Rate Generators.**

The rate generators used by the Segmentation device must be initialized prior to transmitting virtual circuit data at the programmable rates. This could occur at initialization time, or at a later time when the need for an available rate generator is determined.

The local processor initializes the Rate Generator Prescaler Register (RGP) to scale down the external clock rates provided to the Segmentation device. The Rate Control Registers (RGD*i* and RGC*i*) are used to configure the rate generators. There are sixteen rate generators and sixteen pairs of Rate Control Registers. They are used to enable the rate generator, set the priority of the rate generator, assign an external clock to the rate generator, and provide the M/N clock-scaling to the rate generator.

5. **Initialize the Address Translation Table for the Reassembly device.**

This is accomplished by creating an initial Address Translation Table in Connection Memory, and then loading the Address Translation Table Register (ATT) with a pointer to the table.

The Address Translation Table determines if received cells are mapped to active Virtual Circuits or are treated as errors (based on the address in the cell header). The table is composed of a set of inter-related data structures that map incoming cells to context information about the virtual circuits they belong to. These data structures are the VPI (Virtual Path Identifier) table, VCI (Virtual Channel Identifier) tables and MID tables which are used to perform the mapping of incoming VPI/VCI/MID addresses into CIDs.

During initialization it is possible to initialize the Address Translation Table so that a select set of addresses are recognized, or so that no addresses are recognized. Implementations that use signalling (such as Q.2931) to setup circuits would typically initialize the Address Translation Table to recognize only those hardwired addresses that are used by the signalling protocol.

The Segmentation device does not use an Address Translation Table since the transmitting side already knows the mapping performed by the table (and therefore uses CIDs directly to identify the circuits).

6. **Initialize the Segmentation Connection Table.**

This table should be initialized so that every entry in the table is marked as "not in use" unless active Virtual Circuits are known at initialization time. When a virtual circuit is desired to be active, the Segmentation Connection Table entry for that circuit must be initialized.

7. **Initialize the Reassembly Connection Table.**

This table should be initialized so that every entry in the table is marked as "not in use" unless virtual circuits are known at initialization time. When a

virtual circuit is later activated, the appropriate Reassembly Connection Table entry is marked as "active" and the Address Translation Table is updated to associate a VPI/VCI/MID address to this entry.

8. **Enable the Reassembly device to receive cells.**

   The Reassembly device must be explicitly enabled to accept incoming cells by writing to the Master Control Register. This ensures that the preceding initializations, along with any other ATM-related initializations, can occur without threat.

9. **Enter a normal data transfer mode.**

   At this point the SAR chipset is ready to support normal data transfer on any enabled virtual circuits and to respond to the enabling or disabling of circuits based on signalling protocols (or other higher-layer software).

Virtual Circuit Setup and Teardown

The ATM network is connection-oriented. This requires that a connection be set up prior to data transfer across a Virtual Circuit. Connections can be Switched or Permanent. The signaling protocol used to set up Switched connections operates at a higher layer than those implemented in the SAR chipset (although the chipset transmits and receives Protocol Data Units that are used for signalling). Permanent virtual circuits could be determined from higher-layer tables.

Once a higher-layer decision has been made to complete the setup or teardown of a circuit, the SAR devices need to be configured to effect this decision (at the ATM layer). This is accomplished by initializing the appropriate data structures in the Connection Memory (Address Translation Table and Connection Table Entry). At the chipset level, circuits are described as "enabled" or "disabled" to distinguish the operations from the higher-layer setup/teardown protocols.

Virtual Circuit Enable

Once a higher-layer decision has been made to complete the setup of a full-duplex Virtual Circuit (through any appropriate means), both the Segmentation and Reassembly devices need to be enabled so that data transfer can occur normally on that circuit. These operations are described in the following sections.

A set of data structures, collectively referred to as the Address Translation Table, are used by the Reassembly device to map incoming cells from the ATM network to their context information (in Connection Memory). This context information, referred to as the Connection Table, can be thought of as an array of information about connections. Each connection uses one unique element in the Connection Table array, referred to as a Connection Table Entry (CTE). An index into the Connection Table, referred to as a Connection ID (CID), provides a unique identifier for a given connection. This represents an address compression on the much larger VPI/VCI/MID address space in

ATM cells (MID only applies when using AAL3/4).

Each of the CTEs is 32 bytes in length and allows for a total of 32K entries in the 2MBytes (maximum) of Connections Memory. A "free" Connection Table Entry needs to be located in the Connection Table of the Segmentation device, and in the Connection Table of the Reassembly device in order to enable a new full-duplex circuit. Normally, the same index (CID) will be chosen in both tables for simplification. The only reason to violate this would be if a significant number of half-duplex circuits are used (since a half-duplex circuit would otherwise waste the entry in the unused direction).

The following 2 sections assume that a "free" CTE has been located and assigned to the new circuit for each of the SAR devices.

Segmentation

The enabling of a virtual circuit for the Segmentation device requires the following initializations after acquiring a CTE:

1. Determine if a rate generator is initialized and available at the rate of the new circuit. If one is not, then locate an unused generator and initialize it to the desired rate of the circuit to be enabled. If a rate generator cannot be acquired, then an "out of resource" condition has occurred and the higher-layer software may chose to tear down the connection.

2. Initialize the Segmentation CTE to a predefined fill pattern.

3. Initialize the user-selectable options in the CTE (such as AAL type and rate generator ID). At this point the Segmentation is ready to accept requests to queue PDUs for transmission.

Reassembly

The enabling of a virtual circuit for the Reassembly device requires the following initializations after acquiring a CTE:

1. Initialize the Reassembly CTE to a predefined pattern.

2. Initialize the user-selectable options in the CTE (such as AAL type).

3. Update the Address Translation Table so that it will map cells received with the VPI/VCI/MID address of this circuit to the newly acquired CTE.

4. Perform any other non-chipset initializations that may be required prior to allowing cells to be received on this circuit.

5. Turn on the "ATT Active" flag in the CTE so that incoming cells will be handled normally rather than be discarded.

Virtual Circuit Disable

The teardown of a Virtual Circuit causes the chipset to cease handling traffic on that circuit. Like circuit setup, the chipset responds to higher-layer decisions to teardown a virtual circuit, but does not participate in the decision making process (other than to support traffic which, transparent to the chipset, contains signaling information). In order to deactivate a virtual circuit, data structures in the Connection Memories for both the Segmentation and Reassembly devices need to be written by the local processor. The following sections describe the required actions.

Segmentation

A virtual circuit is essentially deactivated in the transmitting direction when a higher-layer function decides to cease sending data on that circuit. In order to fully free up the resource used by the chipset, however, the following steps should be performed.

1. If the Connection may have previously queued data still waiting for transmission and this transmission is not desired, issue the "Transmit Purge" command to the connection. This will free all buffers queued to the connection without transmitting them.

2. Mark the entry as "not in use" by the software so that it can be reused at a later time. There is no specific "dissable" bit in the CTE since the software can achieve this by simply ceasing to queue data to the connection.

Reassembly

When a higher-layer decision has been made to deactivate a virtual circuit it is assumed that, in general, no additional traffic will be received on that circuit. However, there may be cases where errors or latencies in the network produce this condition. The reassembly device will either transparently discard, or separately queue, cells received on deactivated circuits (based on a programmable option). The deactivation is achieved by either or both of the following:

1. By turning off the "ATT Active" flag in the CTE. This causes the address translation process to discard cells even though a Connection Table Entry is reached by traversing the Address Translation Table data structure.

2. Modifying the Address Translation Table so that incomming cells to this connection will be automatically discarded (i.e. modify the table so that translation fails before a CTE is reached).

Data Transfer

Data transfer involves sending virtual circuit data to the Segmentation device, or receiving virtual circuit data from the Reassembly device. The following sections summarize the steps to send and receive data.

Segmentation

The following steps send data from a virtual circuit:

1. Determine the CID of the circuit on which the data is to be transmitted.

2. Transfer a block of data to the Buffer Memory. The size of the block depends on the application. The block of transmit data should be DMAed into however many fixed-size buffers in Buffer Memory are required. The buffers used to hold this data are removed from the "head" of the "transmit free list". If the list is empty, an "out of transmit buffer" condition exists. The Segmentation device automatically chains buffers onto the "tail" of the "transmit free list" after consuming their data. The Head and Tail pointers to the transmit data should be held for use in the next step.

3. Write the Head, Tail and CID to Segmentation registers to initiate the internal queueing of the transmit request. The actual transmission of the data from the circuit will occur based on the rate shaping algorithm.

Reassembly

The following steps receive data from a virtual circuit:

1. The Reassembly device will notify the local processor when programmable receive-queueing thresholds have been crossed (for example, a notification queue going from empty to not empty).
2. Read the Head, Tail and CID from Reassembly registers to acquire the incoming data. The data will always be posted in the order it is received for a given circuit.
3. Move the received data from the Buffer Memory to the target device. This is transparent to the Reassembly device, except for the consumption of memory bandwidth.
4. Return the consumed buffers to the "tail" of the "receive free list". This allows the buffers to be reused automatically by the Reassembly device as they are needed to hold cell data.

Error Handling

The following errors are reported by the SAR devices:

- Parity errors are reported in the Interrupt Status Register (ISR) for the PHY interface, Connection Memory, or Buffer Memory.
- AAL-specific error information is presented with each pdu.

Network Management Support

SNMP

The chipset supports SNMP or other network management functions by:

- Providing cell counters. These include cells transmitted, cells received and cells discarded.
- Providing the AAL5 layer over which SNMP or other management traffic can be sent.

OAM

OAM management is performed by the local processor. The chipset facilitates this management by providing the capability to queue OAM cells (using AAL "raw" mode) to the Segmentation device. Additionally, incoming cells can be filtered by the PTI field and certain types can be optionally redirected to a separate receive queue; the "management notification queue". The Reassembly device can also be enabled to report Congestion through this queue.

This page left blank intentionally

This section analyzes the accuracy of the rate generators and also gives suggestions on how to obtain the desired rate.

Rate Generator Accuracy

The rate generators are 12-bit fractional rate multipliers. The rate generator can be programmed to generate cell frequencies that are ratios of 12-bit numbers. The range of these ratios is restricted to 1/4096 to 4096/4096. A simple estimate of the accuracy of the rate generator is that the rate generators are accurate to one part in 4096 or about 256 ppm.

A detailed study of rate generator accuracy was performed. This study computed the step size between successive division ratios in the range of 0.25<=R<=1. The division ratios were restricted to this range because rate generation accuracy is highest in the range approaching a division ratio of one and because dynamic range of division ratios can be handled by the rate prescalers.

The largest step size in the range 0.25<R<1 is near R=0.5. The integer ratios near R=0.5 are R=2048/4096 and R=2047/4095. The step size between these two ratios is 0.0001221001221. If the desired division ratio is exactly between 2048/4096 and 2047/4095 then an error of 122 ppm is generated because one of these discrete ratios must be selected. If the range of division ratios is restricted to 0.25<=R<=1, then this 122 ppm error is the maximum possible error. Another local error maximum of 122 ppm is found between the ratios of 1024/4096 and 1023/4093, however this error is outside of the 0.25<=R<=1 range.

Discrete Rates

To this point the rate generation problem has been considered as a problem of mapping a desired rate onto the discrete set of rates produced by the rate generators. The mapping of this continuous distribution to a discrete set results in a frequency error.

If the rate generation problem is thought about as the selection of a rate from many possible discrete rates, then the selected rate is exact. More than 16 million division ratios of two 12-bit numbers are possible. A resulting ratio of more than one is not permitted, and there are many duplicate ratios, for instance 4/8 and 8/16. If we restrict the range of division ratios to 0.25<=R<=1, more than three million discrete frequencies are possible. These three million rates can be generated exactly.

Recommendations for Improved Rate Generator Accuracy

- Use a very accurate frequency source to drive the rate generators.
- Use the prescalers to restrict the dynamic range of ratios that the rate generators must generate. It should be possible to restrict the range to 0.25<=R<=1. If a larger range is required, two prescalers programmed to different values can be driven from the same rate generator input clock. The outputs of these two prescalers can be used to drive different rate generators.
- Try to make the desired rate generated one of the millions of rates that can be programmed as an exact value.
- Frequency references with staggered frequencies can be used to further enhance accuracy. Using two crystals offset by 61 ppm can produce accuracy of 61 ppm over the range of 0.25<=R<=1.

622 SAR: TC35860F, TC35861F Errata (r1.0)

* Asynchronous clock data corruption (both chips)

The chipset has been shown to cause data-corruption when asynchronous clocks are used for MClk and BmClk. The nature of the corruption is typically the transmission of an extra cell within a PDU. The CRC (for AAL5, for example) does not identify the errored-PDU because the error occurs prior to the CRC computation.

When MClk=BmClk (same oscillator, not just same frequency) or MClk & BmClk are made harmonic to some reasonable higher frequency (e.g. probably not 1Ghz) the problem is avoided.

* S-chip parity

Buffer/Descriptor Memory parity is broken. The user should not enable this parity function in the S_MODE register.

* Rx Notification Queue (R-chip only)

One of the "Receive Notification Queue" sizes (4K words) is broken and cannot be used. This uses the encoded value of b'01' written to the "Receive Queue Base Range" field in the R_RQC0, R_RQC1, R_RQC2 or R_RQC3 registers. The remaining sizes chosen by the encoded values b'00', b'10' and b'11' are operational.

The main result of the bug is potential memory wastage by necessitating a use of the next larger queue (given that the 4K queue was desired).

* Reading R_FREEH register (R-chip only)

The user cannot read R_FREEH register during normal operation (when "Receive Enable" is turned on in mode register). Attempting to do so will return an invalid value or zero. The intent was to allow this register to be read during normal operation to allow the user to determine which free-buffer the Reassembly chip was at (as described in section 4.4.7 of the Rev1.0 specification). The main purpose was as an assist to debug.

The register still has valid read/write access for diagnostic purposes when the "Receive Enable" bit is turned off in the R_MODE register.

* Span signal (both chips)

The Span signal from the Buffer Memory interface is asserted longer than necessary and could waste memory bandwidth if not properly accommodated by the users arbiter. The span signal is asserted for many clocks beyond the end of a DMA burst for a spanned-into buffer. If the users arbiter does not release the grant until the Span signal is de-asserted, significant memory bandwidth is wasted.

Many users may chose to re-arbitrate between span-bursts for reasons not related to this bug. In this case the Span signal is not used and the bug is harmless.

If the Span signal is used -- the users arbiter can release the grant following the 2nd burst of span without regard to the Span signal. The Span would still be used to hold the grant between the bursts. The impact is slightly more logic in the arbiter.

* S TxPurge

In attempting to purge a Virtual Connection from a queue the S attempts a read modify write to Connection Memory Entry 0. However, the read of the RMW is arbitrary (most likely Entry 0 of Connection Memory Address 0) and so the VC's Connection Memory Entry 0 gets clobbered.

Documentation

1. The BmR* signal is de-asserted by the R-chip one clock earlier than shown in the timing diagrams in Chapter 6 of the Rev 1.0 spec. The BmR* timing for the S-chip is correct as shown in the documentation.

2. Multi-buffer PDU's queued to the S chip for transmission should have their length fields set to the "full" value except for the last buffer which may be less. While the SAR requires intermediate buffers to be full -- failing to set the length accordingly can cause malfunction and must be avoided.

3. The Buffer Memory data bus is intended to be referenced as Big Endian on double-words, but this is not clearly spelled out in the Rev1.0 spec. The spec only describes Big Endian on words, not double-words. Byte 0 of BmA_D[63:0] is therefore bits 63:54, not 32:24.

4. The following section in Chapter 4, page 45 is incorrect and could cause incorrect programming of the rate generators. The sentence ending "Numerator > Denominator" is reversed and should say "Numerator less than or equal to Denominator".

5. The R-chip requires at least 10 buffers to be available when "Receive Enable" is tuned on in the R_MODE register. If this condition is not met the R-chip will stop accepting cells at the UTOPIA interface. The user, upon reading back the R_MODE register, will see the "Receive Enable" bit is zero in spite of previous attempts to set the bit.

6. Descriptor Memory address 0 must be set to zero during normal operation of the chipset for both S & R. The SAR chains through the end-of-descriptor value (zero) and then reads address zero. Malfunction will occur if the value read at address zero is not zero. Care should be taken to insure this address is not left with a diagnostic test pattern or allowed for general sfw use.

7. The "Error bit, bit 20" shown in chapter 4, page 45 does not exist. Also, in the "PDU Status" field description, bit 19 does not exist either. Both bits 19 & 20 were meant to refer to the same error bit which was not implemented.

   The user should examine the 3-bit "PDU Status" field (bits 18-16), as defined in that section, to determine if an error has occurred. There is no loss of functionality.

8. Bit 1 (Notify Queue Overflow) of R_ISR does not exist. The "Receive Enable" bit in R_MODE is not affected when rx queues overflow (but cells are discarded that belong to connections using the overflowed queue). The user may chose to use the programmable rx-notification-thresholds to prevent overflow or to treat as an overflow.

9. The system needs to disable writes from the SAR to the upper 16 bits of S Descriptor Memory (endOfPDU and Length).

10. The R_ATR register should define its fields as the same as those defined in the ATTE format section in chapter 5. The table entries for the VPI table are different:

    cont=31, R=30, base=29:14, limit=13:0.

This page left blank intentionally

TAEC Regional Sales Offices

Irvine, CA
TEL: (714) 453-0224
FAX: (714) 453-0125

San Jose, CA
TEL: (408) 456-8900
FAX: (408) 456-8910

Atlanta, GA
TEL: (770) 931-3363
FAX: (770) 931-7602

Chicago, IL
TEL: (847) 945-1500
FAX: (847) 945-1044

Boston, MA
TEL: (617) 224-0074
FAX: (617) 224-1096

Edison, NJ
TEL: (908) 248-8070
FAX: (908) 248-8030

Portland, OR
TEL: (503) 629-0818
FAX: (503) 629-0827

Dallas, TX
TEL: (214) 480-0470
FAX: (214) 235-4114

www.toshiba.com

SP31550197

TOSHIBA

TOSHIBA 622Mbps ATM Segmentation and Reassembly Chipset DATA BOOK

TOSHIBA

622Mbps ATM Segmentation and Reassembly Chipset

1997

TC35860F
TC35861F

REVISION 1.0B

DATA BOOK

TOSHIBA AMERICA ELECTRONIC COMPONENTS, INC.