(i)chip™SAR+

Users Guide

Document No. UGICHIP-10-00-E00

Release Date: August 2000

NOTE

See Appendix D for Regulatory Statements/Conditions that affect the operation of this product.
The CE Declaration of Conformity can be found at the Interphase Web site:
http://www.iphase.com**.**

Copyright Notice

Interphase Corporation
13800 Senlac
Dallas, Texas 75234

Phone: (214) 654-5000
Fax: (214) 654-5500

Disclaimer

Information in this manual supersedes any preliminary specifications, preliminary data sheets, and prior versions of this manual. While every effort has been made to ensure the accuracy of this manual, Interphase Corporation assumes no liability resulting from omissions, or from the use of information obtained from this manual. Interphase Corporation reserves the right to revise this manual without obligation to notify any person of such revision. Information available after the printing of this manual will be in one or more Read Me First documents. Each product shipment includes all current Read Me First documents. All current Read Me First documents are also available on our web site.

THIS MANUAL IS PROVIDED "AS IS." The user assumes all risk when making changes to the product and its software using the instructions in this manual and using any software provided with this Board Development Kit and Interphase's warranty obligations cease once uses makes any changes or modifications of the Product software. Be further advised that any changes to the software may render any and all regulatory certifications null and void and that it is user's sole responsibility to secure all necessary regulatory certifications required to use or resell such Product. INTERPHASE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING THOSE OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL INTERPHASE BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Trademark Acknowledgments

Interphase® and the Interphase logo are registered trademarks, (*i*)chip™, SynWatch™, FibreView™, ENTIA™, PowerSAN™, SlotOptomizer™, *i*WARE™, *i*NAV™, and *i*SPAN™ are trademarks of Interphase Corporation.

All other trademarks are the property of their respective manufacturers.

Assistance

Product Purchased from Reseller

Contact the reseller or distributor if

- You need ordering, service or any technical assistance.
- You received a damaged, incomplete or incorrect product.

Product Purchased Directly from Interphase Corporation

Contact Interphase Corporation directly for assistance with this, or any other Interphase Corporation product. Please have your purchase order and serial numbers ready.

Customer Service

United States:	Telephone:	(214) 654-5666
	Fax:	(214) 654-5506
	E-Mail:	intouch@iphase.com
Europe:	Telephone:	+ 33 (0) 1 41 15 44 00
	Fax:	+ 33 (0) 1 41 15 12 13

World Wide Web

http://www.iphase.com

END-USER LICENSE AGREEMENT
FOR INTERPHASE CORPORATION SOFTWARE

IMPORTANT NOTICE TO USER–READ CAREFULLY

THIS END-USER LICENSE AGREEMENT FOR INTERPHASE CORPORATION SOFTWARE ("AGREEMENT") IS A LEGAL AGREEMENT BETWEEN YOU (EITHER AN INDIVIDUAL OR SINGLE ENTITY) AND INTERPHASE CORPORATION FOR THE SOFTWARE PRODUCTS ENCLOSED HEREIN WHICH INCLUDES COMPUTER SOFTWARE AND PRINTED MATERIALS ("SOFTWARE"). BY INSTALLING, COPYING, OR OTHERWISE USING THE ENCLOSED SOFTWARE, YOU AGREE TO BE BOUND BY THE TERMS OF THIS AGREEMENT. IF YOU DO NOT AGREE TO THE TERMS AND CONDITIONS OF THIS AGREEMENT, PROMPTLY RETURN, WITHIN THIRTY DAYS, THE UNUSED SOFTWARE TO THE PLACE FROM WHICH YOU OBTAINED IT FOR A FULL REFUND.

The Software is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties. The Software is licensed, not sold.

Grant of License: You are granted a personal license to install and use the Software on a single computer solely for internal use and to make one copy of the Software in machine readable form solely for backup purposes.

Restrictions on Use: You may not reverse engineer, decompile, or disassemble the Software. You may not distribute copies of the Software to others or electronically transfer the Software from one computer to another over a network. You may not use the Software from multiple locations of a multi-user or networked system at any time. You may not use this software on any product for which it was not intended. You may not use this software on any non-Interphase product. LICENSEE MAY NOT RENT, LEASE, LOAN, OR RESELL THE SOFTWARE OR ANY PART THEREOF.

Ownership of Software: Interphase or its vendors retain all title to the Software, and all copies thereof, and no title to the Software, or any intellectual property in the Software, is being transferred.

Software Transfer: You may permanently transfer all of your rights under this Agreement, provided you retain no copies, you transfer all the Software, and the recipient agrees to the terms of this Agreement.

Limited Warranty: Interphase Corporation ("Seller") warrants that (i) the hardware provided to Buyer ("Products") shall, at the F.O.B. point, be free from defects in materials and workmanship for a period of one (1) year from the date of shipment to Buyer; (ii) the software and/or firmware associated with or embedded in the Products shall comply with the applicable specifications for a period of six (6) months from the date of shipment to Buyer; and (iii) its services will, when performed, be of good quality. Defective and nonconforming Products and software must be held for Seller's inspection and returned at Seller's request, freight prepaid, to the original F.O.B. point.

Upon Buyer's submission of a claim in accordance with Seller's Return and Repair Policy, Seller will, at its option either (i) repair or replace the nonconforming Product; (ii) correct or replace the software/firmware; (iii) rework the nonconforming services; or (iv) refund an equitable portion of the purchase price attributable to such nonconforming Products, software, or services. Seller shall not be liable for the cost of removal or installation of products or any unauthorized warranty work, nor shall Seller be responsible for any transportation costs, unless expressly authorized in writing by Seller. This warranty does not cover damage to the Product resulting from accident, disaster, misuse, negligence, improper maintenance, or modification or repair of the Product other than by Seller. Any Products or software replaced by Seller will become the property of Seller.

REMEDIES AND EXCLUSIONS. THE SOLE LIABILITY OF SELLER AND BUYER'S SOLE REMEDY FOR BREACH OF THESE WARRANTIES SHALL BE LIMITED TO REPAIR OR REPLACEMENT OF THE PRODUCTS OR CORRECTION OF THAT PART OF THE SOFTWARE, WHICH FAILS TO CONFORM TO THESE WARRANTIES. EXCEPT AS EXPRESSLY STATED HEREIN, AND EXCEPT AS TO TITLE, THERE ARE NO OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE, IN CONNECTION WITH OR ARISING OUT OF ANY PRODUCT OR SOFTWARE PROVIDED TO BUYER.

IN NO EVENT SHALL SELLER HAVE ANY LIABILITY FOR INDIRECT, INCIDENTAL, SPECIAL OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, ARISING OUT OF THESE WARRANTIES, INCLUDING BUT NOT LIMITED TO LOSS OF ANTICIPATED PROFITS, LOSS OF DATA, USE OR GOODWILL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. (IC-199, 1/97)

Limitation of Liability: NEITHER INTERPHASE NOR ITS LICENSORS SHALL BE LIABLE FOR ANY GENERAL, INDIRECT, CONSEQUENTIAL, INCIDENTAL, OR OTHER DAMAGES ARISING OUT OF THIS AGREEMENT EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Confidentiality: The Software is copyrighted and contains proprietary and confidential trade secret information of Interphase and its vendors. Licensee agrees to maintain the Software in confidence and not to disclose the Software to any third party without the express written consent of Interphase. Licensee further agrees to take all reasonable precautions to prevent access to the Software by unauthorized persons.

Termination: Without prejudice to any other rights, Interphase may terminate this Agreement if you fail to comply with any term or condition of the Agreement. In such event you must destroy the Software together with all copies, updates, or modifications thereof.

Export: You agree to comply with all export and re-export restrictions and regulations of the U.S. Department of Commerce or other applicable U.S. agency. You must not transfer the Software to a prohibited country or otherwise violate any such restrictions or regulations.

U.S. Government Restricted Rights: Use, duplication, or disclosure of the Software to or by the U.S. Government is subject to restrictions as set forth in the applicable U.S. federal procurement regulations covering commercial/restricted rights software. You are responsible for complying with the notice requirements contained in such regulations.

General: You acknowledge that you have read and understand this Agreement, and by installing and using the Software you agree to be bound by the terms and conditions herein. You further agree that this is the complete and exclusive Agreement between Interphase and yourself. No variation of the terms of this Agreement or any different terms will be enforceable against Interphase unless agreed to in writing by Interphase and yourself. The validity of this Agreement and the rights, obligations, and relations of the parties hereunder shall be determined under the substantive laws of the State of Texas. If any provision of this Agreement is held invalid, illegal, or unenforceable, the remaining provisions shall in no way be affected or impaired thereby. All rights in the Software not specifically granted in this Agreement are reserved by Interphase.

Contents

CHAPTER 6 Segmentation Engine Software Reference

APPENDIX B Setup for CBR Scheduling

APPENDIX C ABR Theory of Operation

APPENDIX D Regulatory Statements

Introduction 1

Overview

This chapter provides information about the (i)chipSAR+ product features and overall functionality. The remaining chapters of this guide discuss the information in this introduction in detail.

(i)chipSAR+ Product Features

The (i)chipSAR+ is equipped with the following features:

- PCI 2.1 compliant
- Handling of CBR (Constant Bit Rate), ABR (Available Bit Rate), and UBR (Unspecified Bit Rate) simultaneous traffic types
- ABR implemented in hardware
- Segmentation and reassembly that can be user-defined as on-board or off-board
- 32-bit 0 wait state Master for up to 132Mbyte/s burst DMA rate
- 32-bit Slave
- Expansion ROM support
- Support for 4K VC connections, enhanced CBR traffic, and early packet discard
- Configuration space and IEEE Address initialized from external serial EEPROM
- 256-byte Receive FIFO and 256-byte Transmit FIFO for DMA Master

Overall Functionality

The (i)chipSAR+ is an ATM NIC (Network Interface Card) controller that provides all of the required functionality, from PCI bus interface to Utopia interface, to segment packets from system memory and to reassemble packets into system memory.

The (i)chipSAR+ supports segmentation and reassembly using Adaptation Layer 5[1] or a Null Adaptation Layer.

At the ATM layer, the (i)chipSAR+ provides for either Constant Bit Rate (CBR) service, Available Bit Rate (ABR) service, or Unspecified Bit Rate (UBR) service.

- The CBR service allows for precise frequency and spacing of cells regardless of the amount of other CBR, ABR, or UBR traffic present.
- The ABR service supports the full set of required and recommended behaviors of the ATM Traffic Management 4.0 specification.

1. T1S1.5/92-010 “Broadband ISDN—ATM Adaptation Layer 5 Common Part Functionality and Specification,” May 1992.

- The UBR service allows for a continuous granularity of rates, from voice grade rates up to OC-3 rates.

Once set up, all of these services have the same user interface.

The (i)chipSAR+ uses external memory for data buffers and for control structure storage. This feature enables the chip to be used with large, powerful memories for server applications (with up to 4K virtual channels) or with small, inexpensive memories for desktop applications (with low numbers of virtual channels).

A high-performance 155 Mbps NIC using the (i)chipSAR+ consists of the (i)chipSAR+, memory, and a front end device as shown in Figure 1-1:

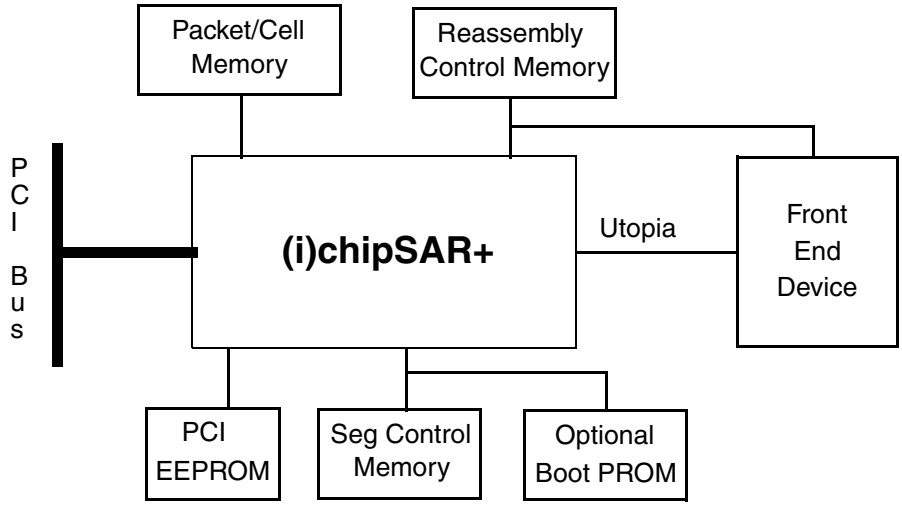

Figure 1-1. High-Performance (i)chipSAR+ NIC Block Diagram

A 25 Mbps NIC using the (i)chipSAR+ consists of the (i)chipSAR+, memory, and a front end device as shown in Figure 1-2:

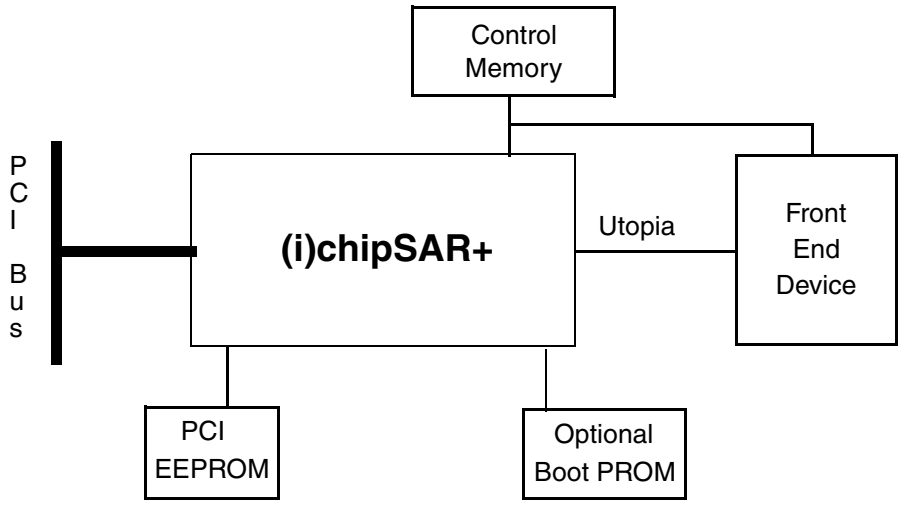

Figure 1-2. 25 Mbps (i)chipSAR+ NIC Block Diagram

Figure 1-3 is a high level block diagram of the (i)chipSAR+ itself:

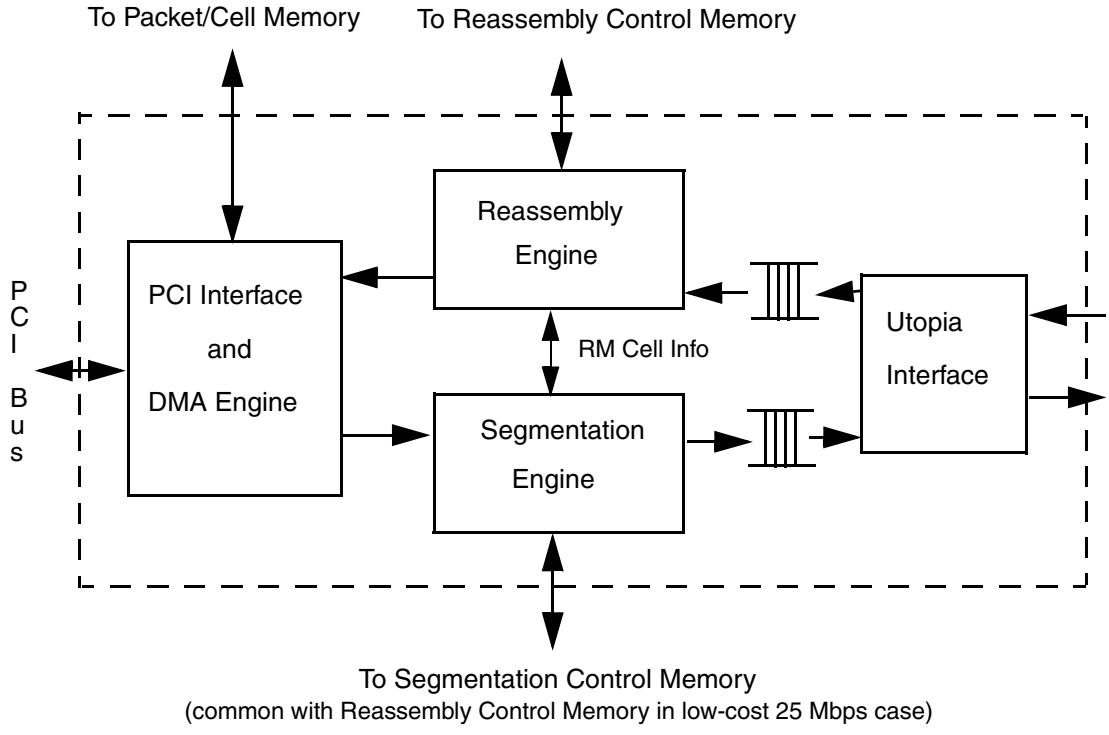

Figure 1-3. High Level Block Diagram

Before either segmentation or reassembly can occur, system software loads the appropriate virtual channel (VC) data structures into the Segmentation Control Memory and the Reassembly Control Memory. Information for these VC data structures is obtained either through provisioning (in the case of Permanent Virtual Circuits) or signalling (in the case of Switched Virtual Circuits).

Segmentation

The (i)chipSAR+ provides two modes of segmentation:

- On-board segmentation
- Off-board segmentation

On-Board Segmentation

For on-board segmentation, system software loads a Packet Descriptor inside the Segmentation Control Memory, along with a DMA data structure in system memory. The Packet Descriptor contains information about the internal buffer for the data, along with specifics about the segmentation of the packet. The DMA data structure contains information about the size and location of the buffer(s) in system memory. A single packet might be spread across several buffers in system memory.

After Packet Descriptor and DMA data structure information is loaded, system software kicks the (i)chipSAR+ to perform the tasks. No more system software intervention is required to send the packet. The (i)chipSAR+, as DMA Master, fetches the packet and loads it into an internal buffer in the Packet/Cell Memory. After the data is loaded in the internal buffer, the Segmentation Engine begins to segment the data. The Segmentation Engine automatically breaks the packet into 48-byte payloads and automatically appends the correct 5-byte header for each payload. The (i)chipSAR+ computes checksums for the headers and CRC-32 for the AAL5 packets.

Off-Board Segmentation

With off-board segmentation, the buffer descriptors in Segmentation Control Memory first point to buffers in system memory. System software then kicks the (i)chipSAR+, and the (i)chipSAR+ begins to schedule and segment the packet. No more system software intervention is required.

As cells are scheduled, the (i)chipSAR+ DMAs the cells from system memory. It automatically appends the correct 5-byte header for each cell (including header checksum) and computes CRC-32 for AAL5 packets.

Segmentation Completion

After the packet is completely segmented, the (i)chipSAR+ updates a Transmit Complete Queue in the Segmentation Control Memory. System software can use this queue as a Free Buffer queue.

Simultaneous Segmentations

The (i)chipSAR+ supports multiple simultaneous segmentations. With simultaneous segmentations, cells from different VCs can be interspersed. This enables each VC to meet its traffic contract. The number of simultaneous segmentations is limited only by the size of the internal buffers and the size of the Packet/Cell Memory.

Reassembly

The (i)chipSAR+ provides two modes of reassembly:

- On-board reassembly
- Off-board reassembly (Cell FIFO mode)

On-Board Reassembly

With on-board reassembly, system software preloads a Free Descriptor Queue in the Reassembly Control Memory. In this mode, buffer descriptors point to buffers in the Packet/Cell Memory. As the (i)chipSAR+ begins to receive cells for a packet, it reassembles the cells in the appropriate buffer in the Packet/Cell Memory. It checks for header checksums, AAL5 CRC-32, and other error conditions.

When the packet reassembly is completed, the (i)chipSAR+ updates the Packet Complete Queue. If the Packet Complete Queue goes from empty to non-empty, and if the (i)chipSAR+ is programmed to do so, the (i)chipSAR+ will also generate an interrupt over the PCI Bus. System software responds by loading the appropriate DMA control structure and kicking the (i)chipSAR+ to DMA the contents of the internal buffer into system memory.

Cell FIFO Mode (Off-Board Reassembly)

With Cell FIFO mode, the buffer descriptors in Reassembly Control Memory point to buffers in system memory. As the (i)chipSAR+ begins to receive cells for a packet, it queues the cells in the Packet/Cell Memory and automatically DMAs them over the PCI Bus to their appropriate buffer. Header checksums, AAL5 CRC-32, and other error conditions are checked.

Upon completion of the packet, the Packet Complete Queue is updated, and an interrupt might be generated over the PCI Bus. A DMA step is not needed because the data was reassembled directly into the host buffer.

Receiving Raw Cells and OAM F5 Cells

Besides having the two AAL5 reassembly modes, the (i)chipSAR+ also can receive raw cells and place them in a Raw Cell Queue. It places both the ATM header and cell payload in the Raw Cell Queue.

The (i)chipSAR+ also handles OAM F5 cells. On the transmit side, the (i)chipSAR+ can send OAM F5 cells on existing VCs with the correct PTI field and with the (i)chipSAR+-generated CRC-10. On the receive side, the (i)chipSAR+ detects OAM F5 cells and automatically sends them to the Raw Cell Queue. In both the transmit and receive cases, OAM F5 cells can occur anytime during segmentation or reassembly of a packet.

ABR in Hardware

An important new feature in the (i)chipSAR+ is that the (i)chipSAR+ implements ABR service in hardware in accordance with the ATM Forum Traffic Management 4.0 specification. The (i)chipSAR+ hardware implements all of the required and recommended behaviors of the ABR specification. System software is required only to load the ABR parameters at VC setup. Hardware performs all of the RM cell handling and all of the rate adjustments automatically. Once the VC is set up, ABR is completely transparent to system software.

Interrupts

An attractive feature of the (i)chipSAR+ is its natural tendency to limit the number of interrupts when the system cannot process received packets quickly enough. For example, with isolated packet reception in on-board reassembly mode, an interrupt may be generated when the packet is received and again when the packet is DMA'ed to the system memory.

However, if multiple packets arrive before the system can process them, only a single reassembly interrupt is asserted when the Packet Complete Queue goes from empty to non-empty. The system can then process several packets under the single interrupt. Likewise, several separate DMAs can be specified before a DMA interrupt is asserted. Single interrupts for multiple packets allow the system to completely process multiple packets per interrupt when heavily loaded.

Packet/Cell Buffer and PCI Bus Interface Functional Description

2

Overview

This chapter describes the interface between the (i)chipSAR+ and the PCI bus. It explains how the Packet/Cell Buffer of the (i)chipSAR+ handles transmit data and receive data and describes the (i)chipSAR+ PCI Bus Interface functionality.

Packet/Cell Buffer

The Packet/Cell Buffer of the (i)chipSAR+ provides intermediate data storage capability for both transmit data and receive data.

For transmit data, packets are DMA'ed into the Packet/Cell Buffer from system memory. This allows efficient, large, power-of-2 PCI Bus bursts (up to 128-byte bursts) to move the transmit packet. Because the entire packet is moved at once, it also can use memory look-ahead technology that might be present in the system.

For receive data, the (i)chipSAR+ provides two reassembly modes: on-board mode and off-board (Cell FIFO) mode.

On-board reassembly mode allows the same highly efficient PCI Bus use for receive data as is allowed for transmit data.

With Cell FIFO mode, the Packet/Cell Buffer is treated as a cell FIFO, with cells being transferred over the PCI Bus for reassembly in system buffers. While Cell FIFO mode does not provide the same degree of PCI Bus efficiency as on-board reassembly, it does allow more simultaneous reassemblies, based on the size of system memory being used for reassembly.

The Packet/Cell Buffer is designed for a single bank of four 32Kx8 SRAMS, or one or two banks of 128Kx8 SRAMS. This design allows for a buffer of either 128 Kbytes, 512 Kbytes, or 1 Mbytes. This buffer supports full-speed transmits and receives at a 155-Mbit rate.

The (i)chipSAR+ is a DMA Master in the movement of packet data into and out of system memory.

DMA of Transmit Data

System software is responsible for placing transmit data in a system memory buffer that is accessible by the (i)chipSAR+ DMA engine. The (i)chipSAR+ allows for buffer chaining of system-resident transmit buffers.

A system-resident circular queue of Descriptor List Entries (DLEs) supplies the (i)chipSAR+ information about transmit DMA buffers. The DLEs give the (i)chipSAR+ information such as the system address of the transmit buffer, the local address of data in the Packet/Cell Buffer, and whether the buffer is chained to another buffer.

To send a packet, the host does the following:

- Updates a buffer descriptor structure in Segmentation Control Memory for the Segmentation Engine
- Builds the DLE in system memory
- Starts the (i)chipSAR+ by performing a single access to an (i)chipSAR+ transaction counter register

The (i)chipSAR+ then performs the packet transfer over the ATM connection with no further host intervention.

See *Chapter 5, PCI Bus Interface Software Reference*, for detailed information about transmit buffer characteristics, DLE structures, and the transaction counter. See *Chapter 6, Segmentation Engine Software Reference*, for detailed information about the Buffer Descriptor used by the Segmentation Engine.

DMA of Receive Data

As mentioned previously, the (i)chipSAR+ provides two modes of reassembling received data:

- On-board reassembly
- Cell FIFO Mode (Off-board reassembly)

On-Board Reassembly

With on-board reassembly, received packets are reassembled in buffers in the Packet/Cell Buffer and then DMA'ed to system buffers after reassembly.

With this mode, the system is notified (through an interrupt or polling) when packets have completed reassembly on board. The system then performs accesses to the (i)chipSAR+ to determine the buffer descriptor number(s) of reassembled packets. The system uses this information to build DLE(s) to DMA the packets into system memory. These DLEs provide address, chaining, and other applicable information similarly to the DLEs used for transmit data.

On-board reassembly operations have the following characteristics:

- Separate system-resident circular queues exist for transmit data and receive data.
- Arbitration priorities between transmit DMAs and receive DMAs are programmable.
- The (i)chipSAR+ places buffer descriptor numbers of completed packets in a Packet Complete Queue. The queue is resident in Reassembly Control Memory.
- The (i)chipSAR+ interrupts (if the interrupt is enabled) when the Packet Complete Queue goes from empty to non-empty. This function allows interrupts to be minimized if the system cannot immediately handle the packet.

  If subsequent packets are completed before the system handles the initial packets, the subsequent packets will not cause further interrupts.

- Interrupts from completion of the DMA operation can be programmed on a DLE-by-DLE basis. This feature allows multiple packets to be DMA'ed into the system memory before a completion interrupt occurs.
- Chaining is allowed on system-resident receive buffers.
- When the DMA process is complete, system software returns the now freed local buffers to a Free Buffer Queue in Reassembly Control Memory.

Chapter 5, PCI Bus Interface Software Reference, and *Chapter 7, Reassembly Engine Software Reference*, provide further details about on-board reassembly.

Cell FIFO Mode (Off-board Reassembly)

Off-board reassembly is referred to as *Cell FIFO mode*. Cell FIFO mode allows the (i)chipSAR+ to use system memory for reassembly while still providing the bus latency tolerance of an intermediate buffer. Using system memory for reassembly allows more simultaneous reassemblies than are physically possible with on-board reassembly mode. The Cell FIFO is a simple fixed-size circular queue; it provides a mechanism for data consistency in system memory. The Cell FIFO can hold 256 cells.

After initialization, the Cell FIFO requires no more system maintenance. The (i)chipSAR+ maintains the FIFO itself.

From the system interface point of view, the major differences between on-board reassembly and Cell FIFO mode are:

- Cell FIFO mode does not use DLEs for receive data, whereas on-board reassembly mode does.
- For Cell FIFO mode, control structures in Reassembly Control Memory point to system memory addresses, whereas in on-board reassembly mode, control structures point to Packet/Cell Buffer addresses.

In Cell FIFO mode, when the (i)chipSAR+ notifies the system that reassembly of a packet(s) is complete, the packet(s) has already been reassembled in system memory. When the system has consumed the data in the system-resident buffer, it returns the data to the Free Buffer Queue in the (i)chipSAR+ Reassembly Control Memory.

When the (i)chipSAR+ Reassembly Engine notifies the system that reassembly is complete, the last cell of a packet has arrived and the cell has been placed in the Cell FIFO. If the Cell FIFO contains a sufficient number of entries (meaning that the system PCI Bus has gotten behind), the system can act on the interrupt before the last cell(s) actually arrive at the system reassembly buffer. A mechanism on the (i)chipSAR+ enables the host to set a *mark* on the Cell FIFO, and the (i)chipSAR+ notifies the host when cells up to the mark have been DMA'ed. This notification to the host might be either an interrupt or a register setting that can be polled.

Note that buffer chaining of system-resident receive buffers is not allowed in Cell FIFO mode.

Chapter 5, PCI Bus Interface Software Reference, and *Chapter 7, Reassembly Engine Software Reference*, provide further details about Cell FIFO mode reassembly.

PCI Bus Interface Functionality

The (i)chipSAR+ PCI Bus Interface generates all PCI signals in accordance with PCI Specification Revision 2.1.

PCI 2.1 Features Supported

The (i)chipSAR+ supports the following PCI features:

- Compliance with PCI Specification Revision 2.1
- Meets PCI Compliance Checklist
- 32-bit 0 wait state Master for up to 132 Mbyte/sec burst DMA rate
- 32-bit Slave
- Expansion ROM supported
- Configuration space and IEEE address initialized from external serial EEPROM
- 256-byte Receive FIFO and 256-byte Transmit FIFO for DMA Master

PCI Bus Cycles Supported

The following table shows the PCI bus cycles supported by the (i)chipSAR+ and the supported modes of each bus cycle. The (i)chipSAR+ ignores bus cycles marked with an asterisk:

Table 2-1. PCI Bus Cycles Supported

C/$\overline{BE}$[3:0]	Bus Cycle Type	Mode Supported
0000	Interrupt Ack	*
0001	Special Cycle	*
0010	I/O Read	*
0011	I/O Write	*
0100	Reserved	*
0101	Reserved	*
0110	Memory Read	Master/Target
0111	Memory Write	Master/Target
1000	Reserved	*
1001	Reserved	*
1010	Configuration Read	Target
1011	Configuration Write	Target
1100	Mem Read Multiple	Master

Table 2-1. PCI Bus Cycles Supported (cont)

C/$\overline{BE}$[3:0]	Bus Cycle Type	Mode Supported
1101	Dual Address Cycle	Master
1110	Memory Read Line	Master
1111	Memory Write and Invalidate	Master

PCI Bus Slave Functionality

As a slave, the (i)chipSAR+ responds to Configuration and Memory cycles. The (i)chipSAR+ does not respond to I/O cycles. The (i)chipSAR+ does not do PCI slave bursting. If a burst is attempted to the (i)chipSAR+, the (i)chip terminates the cycle after the first data transfer.

The (i)chipSAR+ Bus slave interface provides the following features:

- Supports memory and configuration cycles
- Supports optional Expansion ROM
- Supports write posting
- Loads configuration and other registers from external serial EEPROM
- Supports single access only, no bursting supported

PCI Bus Master Functionality

As a PCI bus master, the (i)chipSAR+ moves data between PCI memory and internal FIFOs and registers. The Master State machine supports bursts of the following transfer lengths:

- 128, 64, 32, 16, 8, and 4 bytes for transmits
- 128, 64, 48, 32, 16, 8, and 4 bytes for on-board reassembly receives
- 48, 32, 16, 8, and 4 bytes for Cell FIFO mode receives

If there are fewer than 4 bytes to transfer, the (i)chipSAR+ performs a single access with the appropriate byte enables turned off. The maximum burst size used is based on the burst enable bits in the Control Register and the starting address of the burst.

The (i)chipSAR+ must be aligned to the burst size it is attempting. For example, to perform a 64-byte burst, the least significant 6 bits of the starting address must equal zero. The (i)chipSAR+ holds requests active between bursts for up to 128 bytes of data.

The (i)chipSAR+ PCI Bus master interface has the following features:

- Generates memory cycles only
- Allows receive burst sizes up to 48 bytes for Cell FIFO mode
- Allows receive burst sizes up to 128 bytes for on-board reassembly mode
- Allows transmit burst size up to 128 bytes
- Supports Memory Read Line

- Supports Memory Read Multiple
- Supports Memory Write and Invalidate
- Provides zero wait states
- Supports 64-bit addressing (using a page register to generate the upper 32 bits)

Chapter 5, PCI Bus Interface Software Reference, provides further details about configuration and control register programming.

Segmentation Engine Functional Description 3

Overview

This chapter describes the features and functionality of the (i)chipSAR+ Segmentation Engine. It provides information about how the (i)chipSAR+ supports the AAL5 layer, the Null Adaptation Layer, and the ATM layer. It also describes the (i)chipSAR+ data transmission mechanisms and basic segmentation operation.

Segmentation Engine Features

The (i)chipSAR+ Segmentation Engine has the following features:

- Fully compliant support for the CBR, UBR, and ABR traffic types defined in the ATM Forum Traffic Management 4.0 specification
- Hardware implementation of ABR service, including generation and handling of RM Cells and rate computations
- Same data interface used for support of CBR, UBR, and ABR traffic types
- Precise rate and spacing control for CBR traffic type, even with other CBR, UBR, and ABR traffic segmenting simultaneously
- Continuous spectrum of rates from sub-voice to OC-3
- Support for either on-board or off-board segmentation
- Memory size options for low-cost/low-connection-count 25 Mbps client applications or for high-performance/high-connection-count 155 Mbps server applications
- Support for B-ISDN standard ATM Adaptation Layer AAL5
- Selectable CRC generation and checking on either a per-connection basis or a per-packet basis
- Header checksum generation and checking
- Programmable support for up to 4K virtual circuits
- Simultaneous segmentation of multiple packets on different VCs and queuing of multiple packets per VC
- Support for transmission of OAM F4 and F5 cells using the Null Adaptation Layer
- CRC-10 generation and checking for OAM F4 and F5 cells

ATM Adaptation Layer 5 (AAL5)

This section presents a brief overview of the AAL5 Layer and discusses how the (i)chipSAR+ supports it.

The ATM AAL5 layer is split into three sublayers:

- The *Service Specific Convergence Sublayer (SSCS)* adds service features such as assured transfer (error correction by retransmission). SSCS can also be a null sublayer for unassured transfer (error detection but no correction).
- The *Common Part Convergence Sublayer (CPCS)* performs error detection and control functions at the frame level.
- The *Segmentation and Reassembly Sublayer (SAR)* performs the functions necessary to segment frames into ATM cells and reassemble a multiplexed stream of ATM cells back into their original frames.

Figure 3-1 illustrates the ATM AAL5 sublayers.

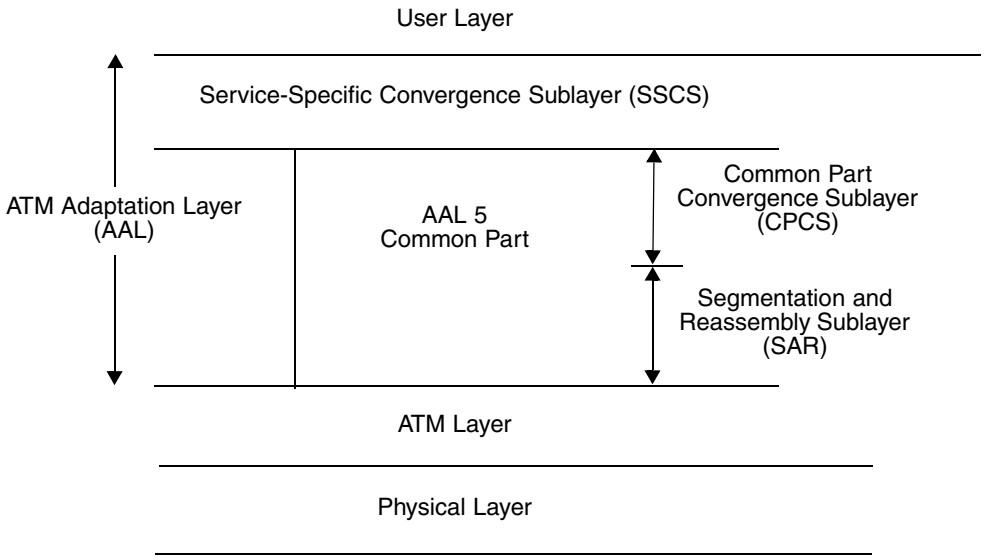

Figure 3-1. Protocol Model for AAL Type 5

Figure 3-2 shows the structure of the CPCS_PDU for AAL 5.

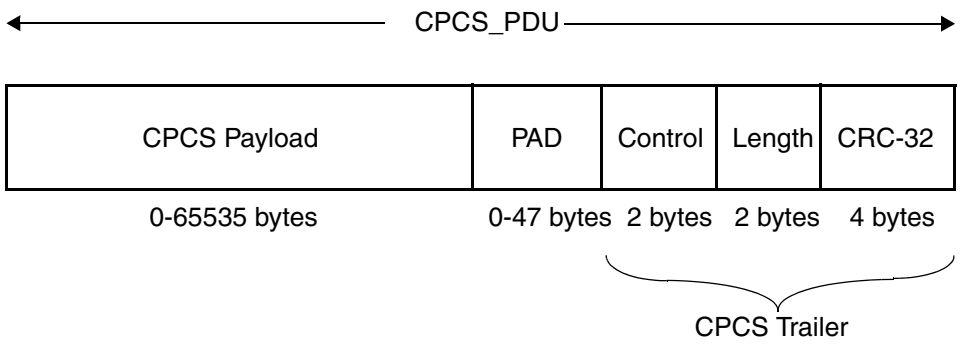

Figure 3-2. Structure of the AAL 5 CPCS_PDU

This packet structure is built by system software and accepted for transmission by the (i)chipSAR+. This is also the structure written to system memory by the reassembly side of the (i)chipSAR+. The fields in this packet structure are as follows:

- The CPCS-Payload field is a variable-length field containing 0 to 65,535 bytes of CPCS-user information.
- The PAD field is used to align the CPCS-PDU to a 48-byte boundary. The PAD field can contain any value, and it is included in the CRC-32 calculation.
- The Control field is reserved for future use.
- The Length field indicates the length in bytes of the Payload field.
- The CRC-32 field contains the result of the CRC-32 calculation performed over the entire CPCS-PDU, as specified in the ANSI draft standard. (This CRC-32 computation is the same as in the ANSI FDDI, IEEE 802.5, and the ANSI Fiber Channel standards.)

You can program the (i)chipSAR+ to either automatically calculate and insert the CRC-32 or to pass the field unchanged from the system. If the (i)chipSAR+ calculates the CRC-32, the system must include four dummy bytes at the end of the packet. The dummy bytes will be replaced by the calculated CRC-32.

Figure 3-3 illustrates the segmentation process for AAL 5.

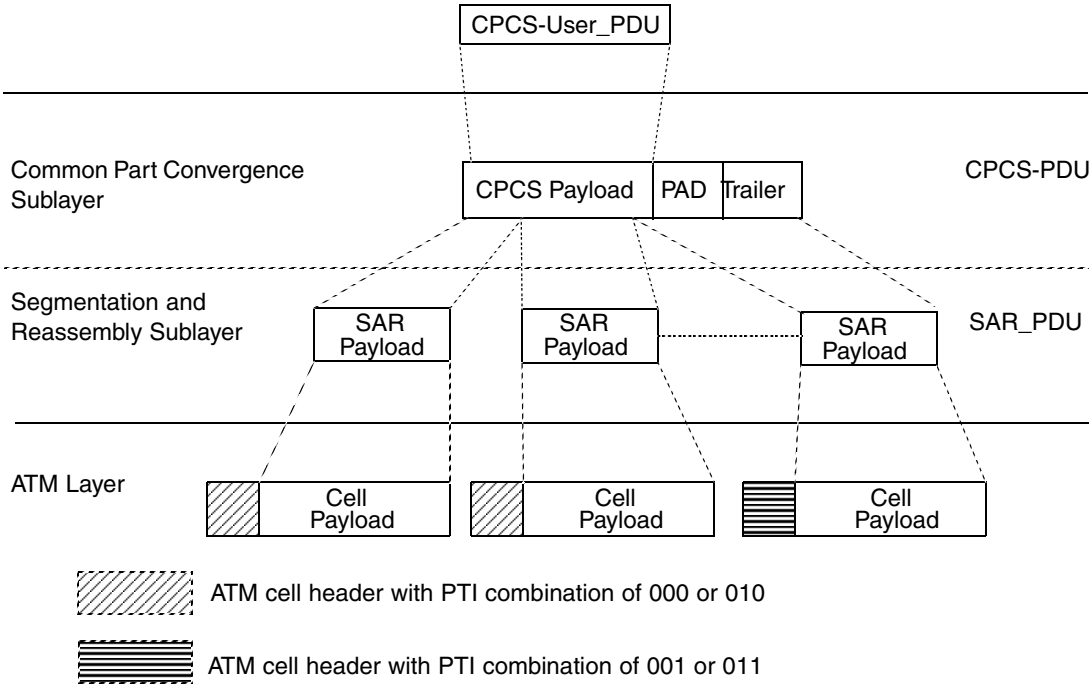

Figure 3-3. Segmentation Process for AAL 5

The CPCS_PDU is segmented into 48-byte segments, with each segment forming a SAR_PDU. No SAR_PDUs are partially filled because the CPCS_PDU is aligned to a 48-byte boundary. No SAR header or trailer is on the SAR_PDU. Each SAR_PDU forms the 48-byte payload of an ATM cell.

Each ATM cell has a five-byte cell header. The 3-bit PTI field within the header identifies whether the cell is the last cell of an AAL5 packet. A PTI combination of 000 or 010 specifies that this is not the last cell of an AAL5 packet. A PTI combination of 001 of 011 specifies that this is the last cell of an AAL5 packet.

The Segmentation Engine of the (i)chipSAR+ can segment ATM AAL5 CPCS_PDUs into AAL5 SAR_PDUs automatically without any per cell user interaction.

Null Adaptation Layer

The (i)chipSAR+ also supports a Null Adaptation Layer. This layer can be used for OAM cells or raw cells, or for support of other adaptation layers implemented in software.

Figure 3-4 illustrates the segmentation process for the Null Adaptation Layer.

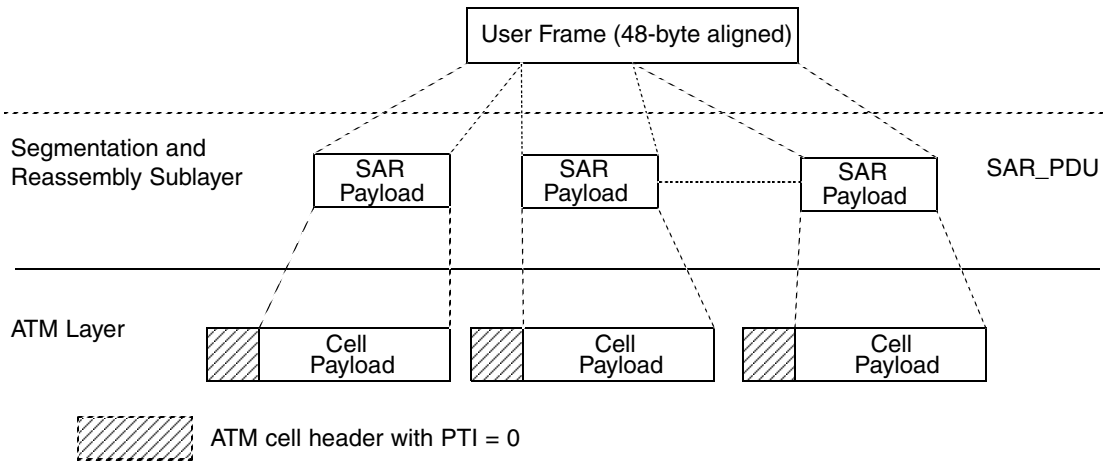

Figure 3-4. Segmentation Process for the Null Adaptation Layer

For Null Adaptation Layer support, the following should be turned off:

- End of message indication
- CRC-32 generation of the (i)chipSAR+ Segmentation Engine

The discussion of the End of Message Enable bit in *Descriptor Mode Word Bits* on page 96 of *Chapter 6, Segmentation Engine Software Reference*, explains how to turn off the end of message indication. The discussion of the Append 32-bit CRC bit in the *Descriptor Mode Word Bits* section explains how to turn off the CRC-32 generation.

Note that OAM cells also have a CRC-10 field at the end of the cell payload. The (i)chipSAR+ can be programmed to insert the CRC-10 field on OAM cells.

ATM Layer

Figure 3-5 shows the format of the ATM cell at the User Network Interface (UNI) of an ATM network, as defined by the CCITT for broadband ISDN[1].

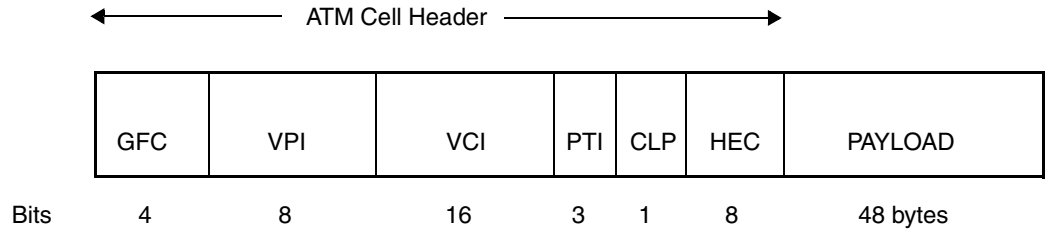

Figure 3-5. Structure of an ATM Cell at the User Network Interface

1. CCITT Recommendations I.150, I.361 (Geneva, June 1992).

The fields in the ATM cell header are as follows:

- The Generic Flow Control (GFC) field ensures fair and efficient access between multiple devices sharing a single UNI.
- A label space of 24 bits is divided into two fields:
  - 8-bit Virtual Path Identifier (VPI), which allows a group of virtual connections, called a virtual path, to be identified
  - 16-bit Virtual Channel Identifier (VCI), which identifies the individual virtual connections within each virtual path
- The Payload Type (PTI) field distinguishes user information and network information. For user information cells, the payload type field carries a single-bit ATM user-to-user identification. ATM cells with a payload type field that indicates network information can be inserted by the (i)chipSAR+.
- The Cell Loss Priority (CLP) field permits two priorities of cell to be defined where the network can discard low priority cells under congestion conditions.
- The Header Error Check (HEC) field provides an 8-bit cyclic redundancy check on the contents of the cell header.

The (i)chipSAR+ maintains a connection table with a copy of the four ATM cell header bytes for each virtual connection (the HEC field is computed for each cell, and therefore is not stored in the connection table). The (i)chipSAR+ can thus insert any pattern of bits into the cell header, and can therefore originate cells with any VPI or VCI.

ATM Layer Service Classes and Cell Scheduling

The (i)chipSAR+ incorporates a flexible and robust cell scheduling mechanism that directly handles Constant Bit Rate (CBR), Available Bit Rate (ABR), and Unspecified Bit Rate (UBR) service classes.

For all supported traffic classes, the mechanism for presenting transmit and receive data to the (i)chipSAR+ is the same.

Constant Bit Rate (CBR) Service Class

The CBR service class provides guaranteed bandwidth and guaranteed cell spacing. Oversubscription is not possible because peak rates are guaranteed.

The (i)chipSAR+ features a separate CBR Scheduling Table in Segmentation Control Memory. This separate scheduling table ensures that no other traffic class can interfere with CBR scheduling, regardless of the number of VCs and the amount of traffic occurring with the other traffic classes.

The CBR Scheduling Table for the (i)chipSAR+ is a simple circular table with a programmable number of entries. Each entry contains either a CBR VC index or a null VC index. Sequential entries in the table are checked every transmit cell time. If a non-null VC index is in an entry when it is checked, that VC segments a cell during that cell time. If a null VC index is in an entry when it is checked, other service classes can use that cell slot.

Figure 3-6 illustrates the CBR scheduling table at a high level.

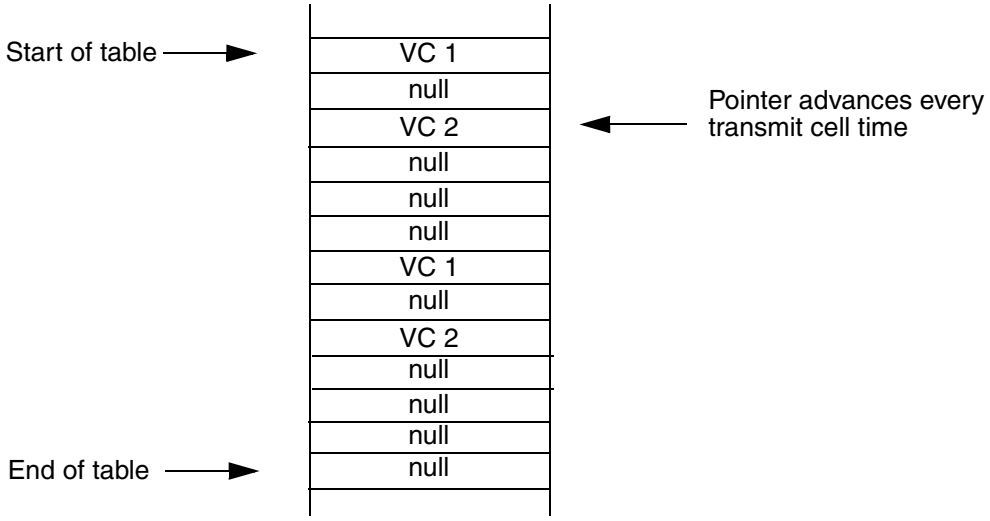

Figure 3-6. CBR Scheduling Table

System software loads the CBR Scheduling Table before the CBR VC is used. Since system software controls the size of the table and the placement of VC indices within the table, system software also controls the exact rate and the exact cell spacing of the cells.

The CBR service class can be completely disabled if it is not needed. Disabling the service class eliminates the requirement for a CBR Scheduling Table in memory.

Also, each VC within the CBR Scheduling Table has its own enable bit within its VC Table structure. This means that a VC can be resident in the table without actually having any data to segment. This capability enables system software to preload the CBR Scheduling Table at initialization, providing that system software knows the rates and cell spacing requirements of CBR VCs in advance.

Appendix B, Setup for CBR Scheduling, provides a more detailed explanation of the algorithm used to load VC indices for particular rates and cell spacings.

Available Bit Rate (ABR) Service Class

The (i)chipSAR+ implements the full ABR specification of ATM Forum Traffic Management 4.0 specification in hardware, including the generation and handling of Resource Management (RM) cells.

ABR Service Parameters

The (i)chipSAR+ uses the following ABR service parameters:

- PCR (Peak Cell Rate)
- MCR (Minimum Cell Rate)

- ICR (Initial Cell Rate)
- AIR (Additive Increase Rate)—based on the signalled RIF
- NRM (Maximum number of cells between forward RM cells)
- MRM (Minimum number of cells between forward RM cells for slow rates)
- RDF (Rate Decrease Factor)
- ACR (Allowed Cell Rate)—set to ICR at VC setup
- CRM (Missing RM Cell Count)
- ADTF (Time permitted between forward RM cells before the rate is reduced to ICR)
- TRM (Upper bound on the time between forward RM cells for an active source)
- CDF (Cutoff Decrease Factor)

These parameters are in effect on a per VC basis, and therefore can be different for each VC used. The ABR parameters are set up at VC setup. The host does not have to handle these parameters after the VC is set up.

ABR VC Table Entries on page 104 of *Chapter 6, Segmentation Engine Software Reference*, provides information about programming ABR parameters in the Segmentation Engine. *ABR VC Table* on page 166 of *Chapter 7, Reassembly Engine Software Reference*, provides information about programming ABR parameters in the Reassembly Engine.

Forward RM Cell Structure

Table 3-1 lists the fields of forward RM cells that the (i)chipSAR+ generates and specifies the information that goes into each field.

Table 3-1. Forward RM Cell Field Descriptions

FIELD	OCTET	BIT(s)	DESCRIPTION	VALUE
Header	1–5	All	ATM Header	Header fields initialized on a per VC basis at VC setup PTI = 110 CLP = 0 if in rate or CLP = 1 if out of rate
ID	6	All	Protocol Identifier	Initialized for all VCs at chip initialization
DIR	7	8	Direction	Initialized for all VCs at chip initialization—nominally 0
BN	7	7	BECN Cell	Initialized for all VCs at chip initialization—nominally 0
CI	7	6	Congestion Indication	Initialized for all VCs at chip initialization—nominally 0
NI	7	6	No Increase	Initialized for all VCs at chip initialization—nominally 0

Table 3-1. Forward RM Cell Field Descriptions (cont)

FIELD	OCTET	BIT(s)	DESCRIPTION	VALUE
Reserved	7	4–1	Reserved	Initialized for all VCs at chip initialization—nominally 0
ER	8–9	All	Explicit Cell Rate	Set by (i)chipSAR+ to PCR
CCR	10–11	All	Current Cell Rate	Set by (i)chipSAR+ to ACR
MCR	12–13	All	Minimum Cell Rate	Set by (i)chipSAR+ to MCR
Reserved	14–21	All	Reserved	Set by (i)chipSAR+ to 0
Reserved	22–51	All	Reserved	Set by (i)chipSAR+ to 0x6a
Reserved	52	8–3	Reserved	Set by (i)chipSAR+ to 0
CRC-10	52	2–1	CRC-10	See ITU-T Recommendation I.610
	53	All		

RM Cell Protocol ID and Message Type on page 120 of *Chapter 6, Segmentation Engine Software Reference*, provides information about programming octets 6 and 7.

Backward RM Cell Structure

Table 3-2 lists the fields of backward RM cells that the (i)chipSAR+ turns around and specifies the information that goes into each field.

Table 3-2. Backward RM Cell Field Descriptions

FIELD	OCTET	BIT(s)	DESCRIPTION	VALUE
Header	1–5	All	ATM Header	Header fields initialized on a per VC basis at VC setup PTI = 110 CLP = 0 if in rate or CLP = 1 if out of rate
ID	6	All	Protocol Identifier	Copied from received forward RM Cell
DIR	7	8	Direction	Set by (i)chipSAR+ to 1
BN	7	7	BECN Cell	Copied from received forward RM Cell
CI	7	6	Congestion Indication	Set by (i)chipSAR+ to 1 if the last received data cell on this VC had the congestion codepoint set in the PTI field. Otherwise, copied from the received forward RM cell.

Table 3-2. Backward RM Cell Field Descriptions (cont)

FIELD	OCTET	BIT(s)	DESCRIPTION	VALUE
NI	7	6	No Increase	Copied from received forward RM Cell
Reserved	7	4–1	Reserved	Copied from received forward RM Cell
ER	8–9	All	Explicit Cell Rate	Copied from received forward RM Cell
CCR	10–11	All	Current Cell Rate	Copied from received forward RM Cell
MCR	12–13	All	Minimum Cell Rate	Copied from received forward RM Cell
Reserved	14–21	All	Reserved	Set by (i)chipSAR+ to 0 [1]
Reserved	22–51	All	Reserved	Set by (i)chipSAR+ to 0x6a [1]
Reserved	52	8–3	Reserved	Set by (i)chipSAR+ to 0
CRC–10	52	2–1	CRC-10	See ITU-T Recommendation I.610
	53	All		

[1] Any 8 contiguous bytes on a 4-byte offset from the start of the payload can also be copied from the received-forward RM cell.

In addition to handling the required fields for turned-around RM cells, the (i)chipSAR+ also allows eight additional bytes of the reserved fields to be turned around. These eight additional bytes must be contiguous, and must start on a 4-byte offset from the start of the payload. The 4-byte offset is programmable.

ABR VC Table on page 166 of *Chapter 7, Reassembly Engine Software Reference*, provides information about programming the eight additional bytes in the turned-around RM cells.

ABR in Hardware

The (i)chipSAR+ handles all of the required and recommended source and destination behaviors listed in ATM Forum Traffic Management 4.0 specification in hardware, with no intervention from the host. The host simply has to set up the ABR parameters at VC setup.

For each VC, the (i)chipSAR+ handles individual rates from link rate (OC-3) down to 0. The granularity of rates is based on the 9-bit mantissa of the 16-bit floating point representation defined in the ATM Forum Traffic Management 4.0 specification. The number of possible rates is not finite. Regardless of the number of VCs, each VC can have its own independent rate.

The (i)chipSAR+ also implements options for out-of-rate forward RM cells and out-of-rate backward RM cells. Specifically, if the allowed rate for a VC drops below 10 cells per second, the VC will begin to send out-of-rate forward RM cells at a rate of 10 cells per second.

Also, on a per VC basis, if the (i)chipSAR+ receives a forward RM cell before it has turned around the previously received forward RM cell, it replaces the previously received forward RM cell contents with the new cell contents. Then it sends the old cell as an out-of-rate backward cell. The new cell (with the same contents) remains scheduled.

The Segmentation Engine automatically generates forward RM Cells for each VC at the appropriate times according to the ATM Forum Traffic Management 4.0 specification. The Segmentation Engine works with the Reassembly Engine to turn around received-forward RM cells and to adjust rates based on received backward RM cells.

In addition to the required and recommended ABR behaviors of ATM Forum Traffic Management 4.0 specification, the (i)chipSAR+ also implements the optional use-it-or-lose-it behavior (if the host chooses to). (See Appendix I.8 of the ATM Forum Traffic Management 4.0 specification for a description of the use-it-or-lose-it behavior.) Specifically, the (i)chipSAR+ implements the Forward RM-Triggered method described in Appendix I.8 of the ATM Forum Traffic Management 4.0 specification.

The use-it-or-lose-it behavior is enabled on a per VC basis.

ABR Cell Scheduling

ABR VCs make use of an *appointment book* scheduler. This scheduler is transparent to system software. For scheduling, system software only needs to supply the (i)chipSAR+ with the applicable ABR parameters for each ABR VC. The (i)chipSAR+ can schedule individual VCs with the granularity defined by the 9-bit mantissa of the 16-bit floating point rate representation defined in the ATM Forum Traffic Management 4.0 specification.

The scheduler for ABR consists of two data structures:

- ABR Scheduling Table
- ABR Wait Queue

Each entry in each table represents a queue of VCs.

ABR Scheduling Table

The ABR Scheduling Table is the appointment book for future cell transmissions. This table is a queue of VC queues. Each entry in the ABR Scheduling Table represents a transmit cell time. Each table entry holds a queue of VCs, and contains a pointer for the VC queue it holds. In turn, each VC queue contains pointers for the VCs it holds.

Whenever a cell is segmented on a particular VC, the next appointment is computed, and the VC index is added to the ABR Scheduling Table at the appropriate entry location. If a VC is already at that entry location, the entry is treated as a queue, and the newly scheduled VC is added at the head of the queue.

Figure 3-7 illustrates what the ABR Scheduling Table might look like during normal scheduling.

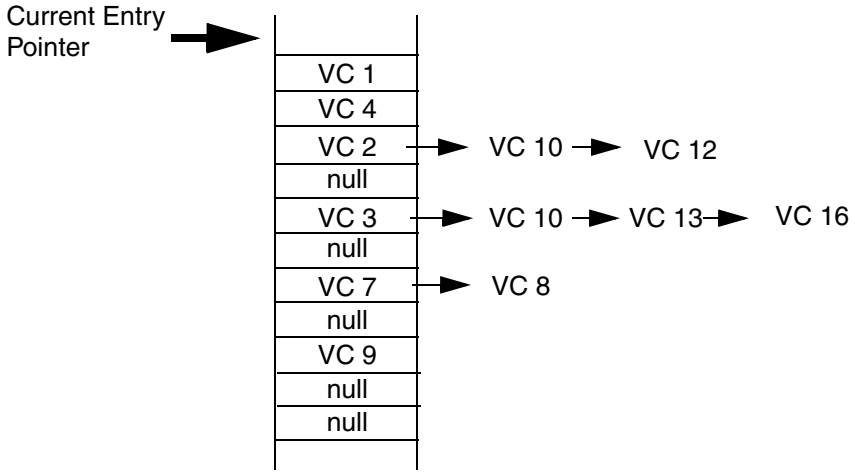

Figure 3-7. ABR Scheduling Table Example, Part 1

As part of this example, consider that VC 14 is currently segmenting and being rescheduled. If the reschedule computation says that VC 14 should be segmented again in 7 cell times, the resulting ABR Scheduling Table might look like Figure 3-8.

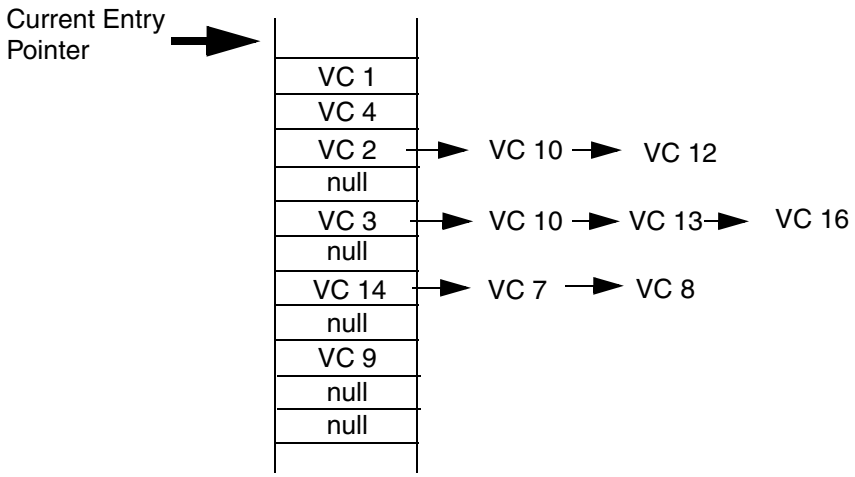

Figure 3-8. ABR Scheduling Table Example, Part 2

In this example the ABR Scheduling Table is actually overbooked. For the next 7 cell times after the current entry pointer, 12 VCs are actually scheduled for transmission. Also, notice that null entries are still within the scheduling table. Because the ABR service is overbooked at this time, the null entries must be removed so that a VC can be segmented every possible cell time.

ABR Wait Queue

The ABR Wait Queue performs the necessary function of removing null ABR Scheduling Table entries from the segmentation schedule. The ABR Wait Queue is a data structure similar to the ABR Scheduling Table. It is a queue of queues, with each entry being a pointer for the queue of VCs at a particular slot.

When the current entry pointer of the UBR Scheduling Table points to a non-null entry, the queue of that entry is moved to the next available entry in the ABR Wait Queue. When the current entry pointer encounters a null entry in the scheduling table, the null entry is not moved to the ABR Wait Queue.

Segmentation actually takes place outside of the ABR Wait Queue. Using the example shown in Figure 3-7 and Figure 3-8 on page 24 , and assuming an empty ABR Wait Queue at the start, segmentation would occur as follows:

1. At the next transmit cell time, VC 1 is moved to the ABR Wait Queue and then immediately segmented.
2. At the next transmit cell time, VC 4 is moved to the ABR Wait Queue and then immediately segmented.
3. At the next transmit cell time, the queue of VC 2, VC 10, and VC 12 is moved to the ABR Wait Queue. VC 2 is segmented.
4. At the next transmit cell time, the current entry pointer encounters a null in the ABR Scheduling Table, and nothing is moved to the ABR Wait Queue. VC 10 from the previous entry is segmented.
5. At the next transmit cell time, the queue of VC 3, VC 10, VC 13, and VC 16 is moved to the ABR Wait Queue. VC 12 from the previous wait queue entry is segmented. This completes the wait queue entry that originally consisted of VC 2, VC 10, and VC 12.
6. At the next transmit cell time, the current entry pointer encounters a null in the ABR Scheduling Table, and nothing is moved to the ABR Wait Queue. VC 3 is segmented.
7. The process continues in the same manner.

ABR Scheduling Table and ABR Wait Queue Maintenance

Note these points about ABR scheduling and the maintenance of the ABR Scheduling Table and ABR Wait Queue:

- The entry pointer for the ABR Scheduling Table is incremented and entries are placed in the ABR Wait Queue regardless of whether CBR transmission preempts ABR cell transmission. This action enables the ABR scheduler to maintain the ABR VCs' rates regardless of the amount of CBR traffic.
- The ABR Scheduling Table and ABR Wait Queue together hold each actively segmenting ABR VC exactly one time.
- To help preserve fairness of high-rate VCs in overbooked situations, VCs can also be rescheduled directly into the ABR Wait Queue.

- System software is responsible only for initializing these scheduling structures to zero (null VCs) during (i)chipSAR+ initialization.

Unspecified Bit Rate (UBR) Service Class

The UBR service class offers no rate guarantees nor cell spacing guarantees. Oversubscription is possible.

UBR VCs make use of an *appointment book* scheduler similar to the scheduler used in the ABR service. Note that there are physically separate scheduling structures for ABR and UBR services.

In the case of UBR services, only the Peak Cell Rate (PCR) needs to be specified. The (i)chipSAR+ always attempts to schedule UBR VCs smoothly at their Peak Cell Rate.

Note that, like with the ABR service, each VC's rate is independent of other VCs. In other words, the number of different rates is not finite. Each VC can have its own rate. The scheduler must assure fair and smooth scheduling in overbooked situations.

Priorities Among the Different Service Classes

The (i)chipSAR+ has different priorities for handling cells from CBR, ABR, and UBR service classes. CBR cells take first priority; however the priority assignment of ABR and UBR services is programmable.

CBR

CBR cells always have higher priority than either ABR or UBR cells. If a CBR cell is scheduled for a given cell slot, it will be sent. Any ABR or UBR cells scheduled for that same time slot will wait for the next available time slot.

Note, however, that CBR cells can be scheduled without CBR data being available. If that is the case, then either the ABR cell or UBR cell can be sent.

ABR and UBR

The (i)chipSAR+ can handle ABR and UBR VCs at the same time, as long as the priority assignment between the two services does not negate this ability.

Both ABR and UBR VCs can operate at the full link rate, if allowed by their Peak Cell Rates. Therefore, hardwiring either service to have priority over the other service can result in the lower priority service being starved if any VC of the higher priority service has something to send.

The (i)chipSAR+ solves this problem by allowing for a programmable priority between the two services. You can program the (i)chipSAR+ to give either service priority for a percentage of time. The rest of the time, the other service has priority.

You can program the service priority using a simple 8-bit register. During each cell time, the (i)chipSAR+ rotates a bit in the register. If both an ABR VC and a UBR VC need to be segmented during the same cell time:

- ABR has priority when the bit is set.
- UBR has priority when the bit is reset.

System software initializes the register at chip initialization. For example, if system software sets the register to alternating 1's and 0's (0x55), then ABR has priority over UBR half of the time and UBR has priority over ABR half of the time.

Note that the register has no meaning if only one of the two services needs to segment during a given cell time. That service is, of course, allowed to segment (providing there is not a CBR cell to be segmented).

ABR-UBR Programmable Priority Register on page 120 of *Chapter 6, Segmentation Engine Software Reference* provides information about programming ABR and UBR service priority.

VC Tables

Two separate tables contain VC specific information:

- 32-byte per VC Main VC Table
- 8-byte per VC Extended VC Table

These tables contain the headers, control information for the VC, and pointers for the queue entries used in scheduling. System software initializes the VC Tables upon completion of the signalling procedure for the VC. System software needs to perform no further intervention after table initialization.

Segmentation VC Table Entries on page 103 of *Chapter 6, Segmentation Engine Software Reference*, describes the fields in these tables in detail.

Data Transmission

Packet/Cell Buffer on page 7 of *Chapter 2, Packet/Cell Buffer and PCI Bus Interface Functional Description*, and *System Memory Structures* on page 43 of *Chapter 5, PCI Bus Interface Software Reference*, describe the mechanisms used to initiate data transmissions. This chapter explores only the Segmentation Memory data structures used by the Segmentation Engine.

Transmit Side Data Structures

Figure 3-9 shows the basic data structures for the Segmentation Engine of the (i)chipSAR+.

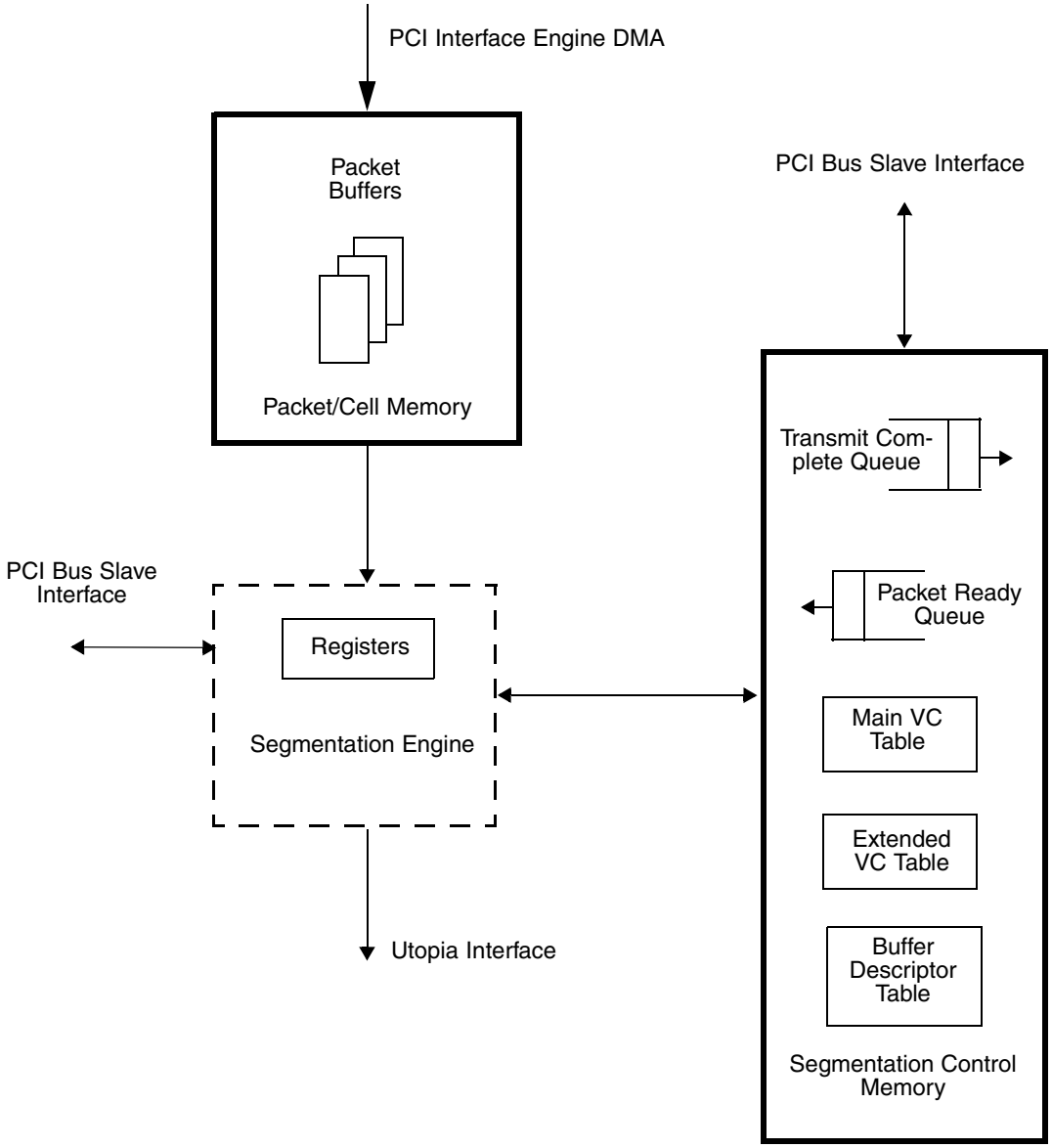

Figure 3-9. Segmentation Engine Data Structures

The data structures contained in the Packet/Cell memory are the packet buffers. When on-board segmentation is selected, the PCI Interface Engine performs the DMA operations to bring the system-resident data into these buffers. The Segmentation Engine pulls data out of these buffers 48 bytes at a time until packets are completely segmented.

The data structures in Segmentation Control memory are:

- Communication queues (Transmit Complete Queue and Packet Ready Queue)
- VC Table
- Buffer Descriptor Table

These data structures contain the overhead information and other segmentation variables. The system can access control memory through the PCI Bus slave interface.

The internal registers in the Segmentation Engine determine the device operation modes. Internal registers also contain the base addresses of the Buffer Descriptor Table, the VC Tables, and the various addresses and pointers associated with the communication queues. These registers are accessed using the PCI Bus slave interface. *Internal Registers* on page 116 of *Chapter 6, Segmentation Engine Software Reference* contains detailed information about these registers.

Each packet buffer contains either a packet ready to be segmented or a packet being segmented. Note that CBR and UBR traffic use the same packet buffer structure for data segmentation.

The Segmentation Buffer Descriptor Table contains packet-specific information for each packet to be segmented. Each table entry includes the following information:

- VC index (software-assigned identifier for the VC) associated with the packet
- Length of the packet to be segmented
- Location of the buffer (in the Packet/Cell Memory for on-board segmentation and in the Host Memory for off-board segmentation) and other temporary variables

The (i)chipSAR+ supports up to 4K descriptors.

The Main VC Table and Extended VC Table contain the ATM header information, along with various queue pointers and temporary information needed on a per VC basis. The (i)chipSAR+ supports between 512 and 4K virtual circuit entries in the VC table. The table entries are set up by software as virtual circuits are established.

The communication queues—the Packet Ready Queue and the Transmit Complete Queue—are used to pass buffer descriptor numbers between the (i)chipSAR+ and the system. The Packet Ready Queue contains buffer descriptor numbers of those buffer descriptors ready to be linked for segmentation by the (i)chipSAR+. The Transmit Complete Queue contains buffer descriptor numbers and completion codes for buffer descriptors that have completed segmentation.

Basic Segmentation Operation

This section provides an overview of the on-board and off-board system software process required to send a packet. The interfaces and programming details are described in greater detail in the following chapters.

On-Board Segmentation

To segment a packet using on-board segmentation, the system must determine a free Packet/Cell Memory buffer and initialize its Buffer Descriptor entry in the Buffer Descriptor Table. The free buffer can be determined by using the Transmit Complete Queue as a Free Buffer Queue.

After initializing the Buffer Descriptor entry, the system adds a DLE entry to the DLE queue for each system buffer to be chained for that packet. In the last DLE entry (the only one if not chaining), the system enters the buffer descriptor number of the Packet/Cell Memory Buffer to be used. The system starts the (i)chipSAR+ to service the DLE list. After the packet has been transferred to a packet buffer, the segmentation process begins automatically.

The system segmentation requires no further intervention.

Off-Board Segmentation

To segment a packet using off-board segmentation, the system must have the packet data available in a free host buffer. It must then initialize a Buffer Descriptor entry in the Buffer Descriptor Table with the host address and packet length. The system determines the free buffer by using the Transmit Complete Queue as a Free Buffer Queue.

After initializing the Buffer Descriptor entry, the system adds the Buffer Descriptor Index number to the Packet Ready Queue. Then, the system updates the Packet Ready Queue write pointer. At this point, the segmentation process begins automatically.

The system segmentation requires no further intervention.

Segmentation Control Memory Size Options

The (i)chipSAR+ has three options for Segmentation Control Memory sizes. These are 64 Kbytes, 256 Kbytes, and 0 bytes.

- The 64-Kbyte option is the 1K VC/1K Buffer Descriptor option.
- The 256-Kbyte option is the 4K VC/4K Buffer Descriptor option.
- The 0-byte option is intended for slower operation (25 Mbps). With this option, the 64-Kbyte reassembly control memory is also used for segmentation control. A maximum of 256 VCs is assumed in this case.

Note that the size of the segmentation and reassembly control memories affects the memory map of (i)chipSAR+ in PCI space.

- With 64 Kbytes in both memories, the PCI space needed for (i)chipSAR+ is 1 Mbyte.
- With 256 Kbytes in both memories, the PCI space needed for (i)chipSAR+ is 4 Mbytes.

Reassembly Engine Functional Description 4

Overview

This chapter describes the (i)chipSAR+ Reassembly Engine. It provides an overview of Reassembly Engine features. It describes the data and control structures used for moving data to the PCI Interface Engine and the mechanism used for decoding VPI/VCI combinations. It also describes methods for testing packet aging and identifies (i)chipSAR+ registers that can be accessed to determine reception statistics.

Reassembly Engine Features

The features of the (i)chipSAR+ Reassembly Engine are as follows:

- Full hardware support of ABR RM Cells and ABR rate calculations
- Support for B-ISDN standard ATM Adaptation Layer AAL5
- Support for a raw cell queue for OAM cells and null AAL cells
- Header checksum and CRC-32 checking
- Programmable support for up to 4K virtual circuits
- Flexible multiple path capability
- Extensive error checking and statistics gathering
- Partial packet discard in cases where the Free Buffer Queue goes from empty to non-empty
- Packet aging functionality

ABR Support

The (i)chipSAR+ Reassembly Engine recognizes and handles ABR RM cells without intervention from the host. In addition, the Reassembly Engine monitors for EFCI indications in data cells, and it sets the RM cell congestion indication accordingly.

EFCI Handling

On ABR VCs, the Reassembly Engine monitors the EFCI (Explicit Forward Congestion Indication) codepoint in all data cells received. It stores the state of the EFCI codepoint on a per VC basis. If the EFCI codepoint is set on a received data cell, the Reassembly Engine sets EFCI state bit. If the EFCI codepoint is not set on a received data cell, the Reassembly Engine resets the EFCI state bit.

RM Cell Filtering

The Reassembly Engine can filter RM cells based on their protocol ID. The host can write the valid protocol ID into a register on the (i)chipSAR+. The Reassembly Engine can then discard any received RM cell whose protocol ID does not match the protocol ID in this (i)chipSAR+ register. The host can disable this filter so that the Reassembly Engine accepts all protocol IDs.

In addition, the Reassembly Engine provides an option to transfer RM cells to the raw cell buffer for host inspection. This option is for diagnostic purposes only; the RM cells are still handled in hardware.

If a received RM cell has a header checksum error or a CRC-10 error, the (i)chipSAR+ discards the cell and increments an error count.

Turning Around Received Forward RM Cells

Upon receiving a forward RM cell on an ABR connection, the Reassembly Engine stores all required fields to be turned around in the VC's ABR VC Table entry. The required fields include:

- Protocol ID
- Message Type octet
- ER
- CCR
- MCR

In addition, you can program the (i)chipSAR+ to store eight contiguous octets on a programmable 4-byte boundary within the RM cell. These eight additional octets will be turned around with the required fields.

After storing the fields to be turned around, the Reassembly Engine notifies the Segmentation Engine of the arrival of the RM cell and then identifies the VC of the RM cell.

Later, when the Segmentation Engine determines that it is time to send the backward RM cell, it requests the stored fields from the Reassembly Engine. As the Reassembly Engine retrieves the stored fields, it sets the CI bit in the Message Type bit according to the saved EFCI state, and then it resets the saved EFCI state.

If a forward RM cell is received on a particular VC before the previously received forward RM cell is turned around, the stored contents are overwritten with the newly received forward RM cell. Then the Segmentation Engine is notified of the arrival of the new cell. At this notification, the Segmentation Engine understands that it needs to send an out-of-rate backward RM Cell.

Rate Calculations for Received Backward RM Cells

Upon receiving a backward RM cell, the Reassembly Engine again works with the Segmentation Engine by performing the calculations to determine the new rate.

The Reassembly Engine requests the present ACR from the Segmentation Engine. When it receives the present ACR, the Reassembly Engine computes the new rate, and then presents that rate back to the Segmentation Engine.

The Segmentation Engine checks the rate against the Peak Cell Rate and the Minimum Cell Rate, adjusts it accordingly, and then stores and begins to use the new rate.

Data Movement to the PCI Interface Engine

The Reassembly Engine can handle data received as AAL5 packets or as raw cells. *Figure 4-1* shows the data and control structures involved using on-board reassembly.

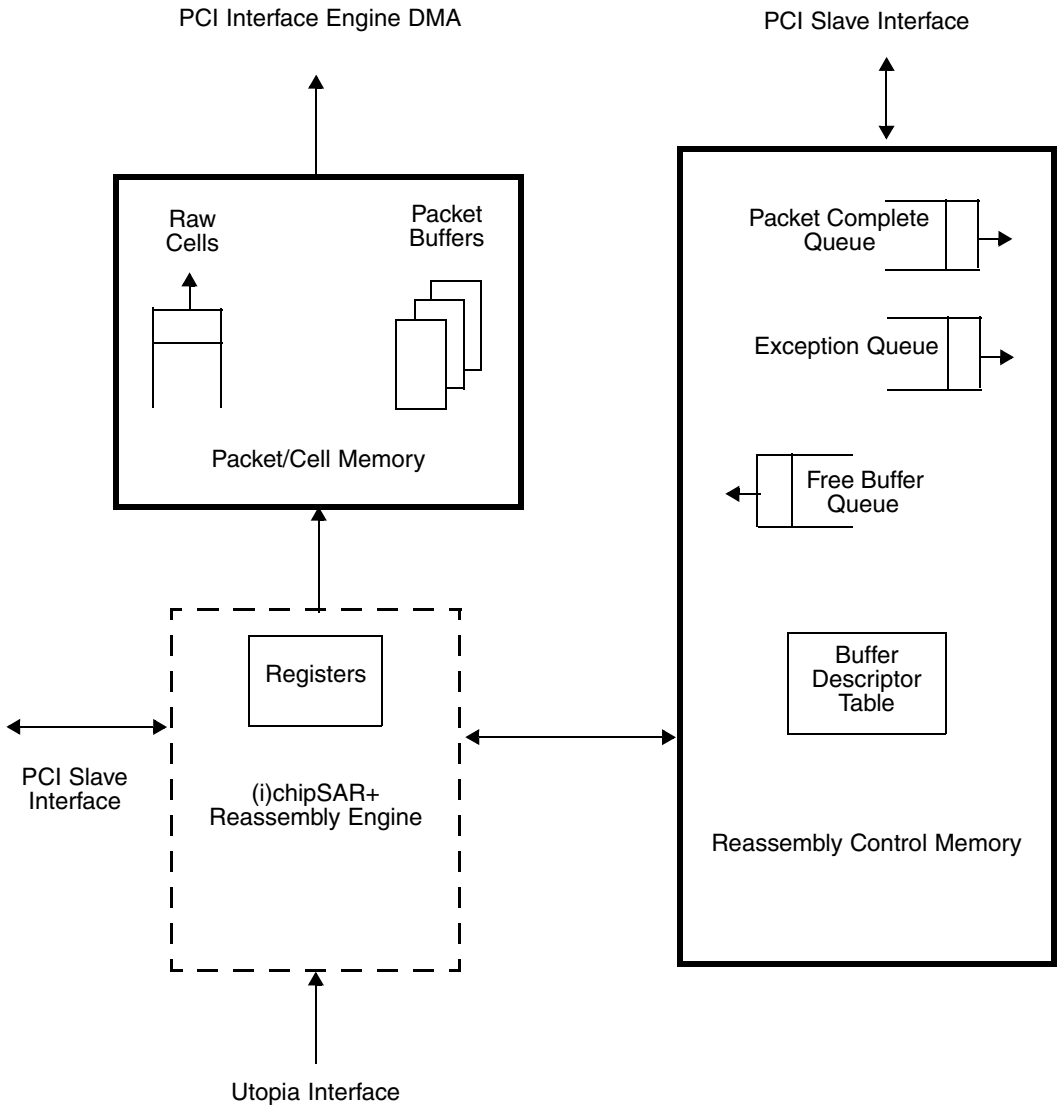

Figure 4-1. Reassembly Engine Data Movement Structures Using On-Board Reassembly

Figure 4-2 shows the data and control structures involved using Cell FIFO mode.

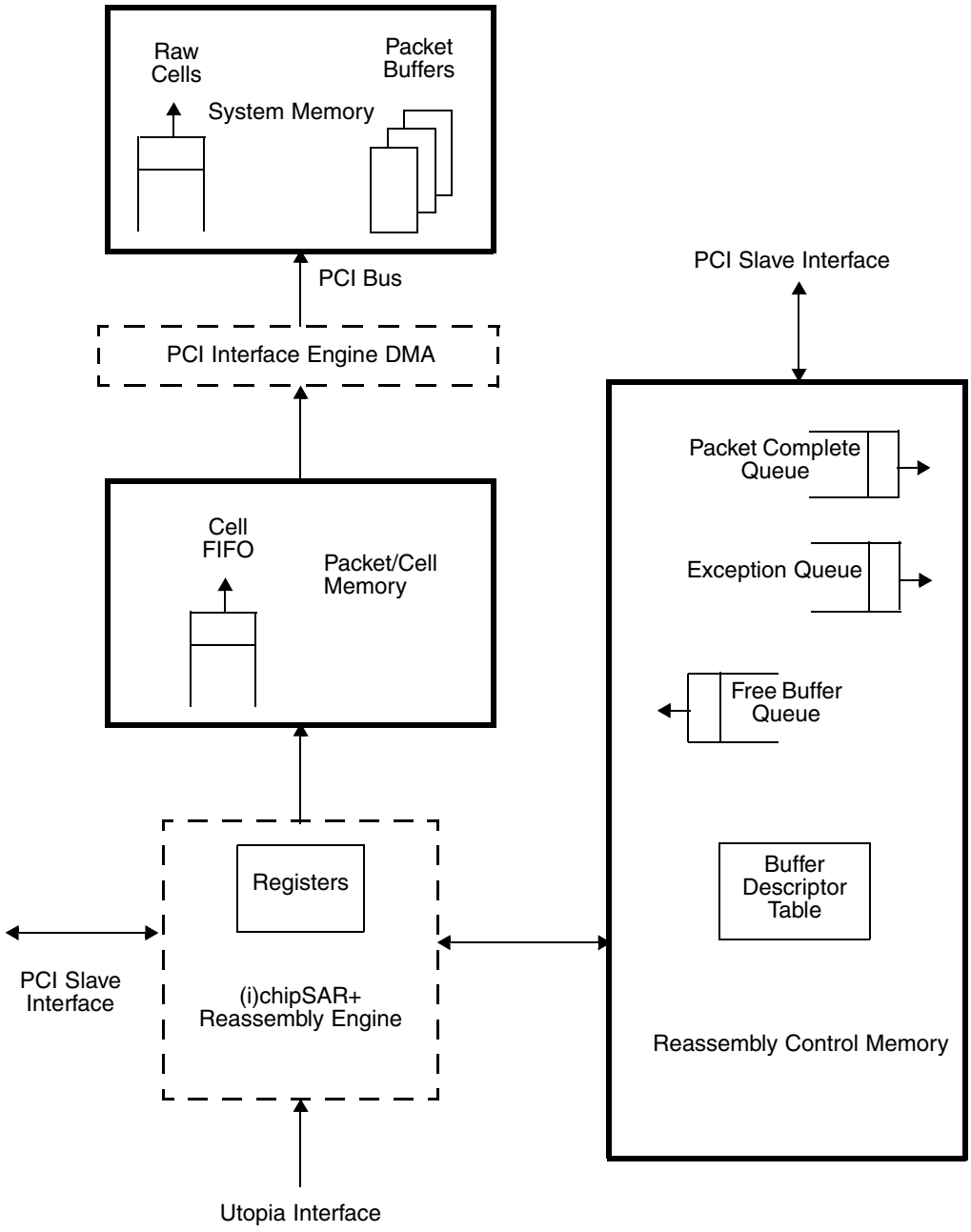

Figure 4-2. Reassembly Engine Data Movement Structures Using Cell FIFO Mode

ATM Adaptation Layer 5 (AAL5) Support

The (i)chipSAR+ passes the entire CPCS-PDU to the system. The (i)chipSAR+ Reassembly Engine strips off the ATM header of received cells and checks the header checksum. If the header checksum is in error, the (i)chipSAR+ drops the cell and increments a cell error counter. If the header checksum is not in error, the (i)chipSAR+ looks up the reassembly status of the VC of the received cell. The (i)chipSAR+ passes the 48 bytes of payload, along with the appropriate buffer address (either on-board or off-board) to the PCI Interface Engine, while keeping the results of a partial CRC-32 calculation.

The (i)chipSAR+ determines when the last cell of a packet is received by checking the PTI encoding of the header. At that time, the (i)chipSAR+ checks the CRC-32 and reports the error if the CRC-32 is incorrect. Regardless of whether the CRC-32 is incorrect, the (i)chipSAR+ then passes the last cell to the PCI Interface Engine and reports that the packet is complete.

As previously mentioned, the (i)chipSAR+ has two modes of reassembly:

- On-board reassembly
- Cell FIFO (off-board reassembly)

With on-board reassembly mode, the data buffers that the Reassembly Engine works with are the local buffers in the Packet/Cell Buffer. Local buffers cannot perform chaining, so these local buffers must all be the same size and must all be able to handle the largest possible packet to be received.

With Cell FIFO mode, the data buffers that the Reassembly Engine works with are the buffers in system memory. Again, since the Reassembly Engine cannot perform buffer chaining, the system buffers in Cell FIFO mode must also all be the same size and all be able to handle the largest possible packet to be received.

The Reassembly Engine has three structures in Reassembly Control Memory to facilitate the movement of packets into reassembly buffers:

- Buffer Descriptor Table
- Free Buffer Queue
- Packet Complete Queue

In addition, the Reassembly Engine has the Exception Queue for exception reporting.

Buffer Descriptor Table

Reassembly data buffers that the (i)chipSAR+ Reassembly Engine uses are all referenced by buffer descriptor numbers. The buffer descriptor numbers are used to index into the Buffer Descriptor Table. The Buffer Descriptor Table contains a 32-byte entry for each buffer with the relevant parameters of the buffer, including address and present byte count of the packet. The Reassembly Engine also uses Buffer Descriptor Table entries to hold intermediate parameters, such as partial CRC and the dynamic address for the next cell to be reassembled.

Up to 4K buffer descriptor entries can be supported in the (i)chipSAR+ Buffer Descriptor Table, depending on the size and organization of the Reassembly Control Memory.

Free Buffer Queue

The Free Buffer Queue holds the buffer descriptor numbers of buffers that are set up and ready to be used by the Reassembly Engine for reassembly of packets. System software puts buffer descriptor numbers in the queue, and the (i)chipSAR+ takes buffer descriptor numbers from the queue when it needs them.

The pointers of the Free Buffer Queue are contained in two (i)chipSAR+ registers:

- Free Buffer Queue Read Pointer
- Free Buffer Queue Write Pointer

The (i)chipSAR+ manipulates the Free Buffer Queue Read Pointer. System software manipulates the Free Buffer Queue Write Pointer.

Packet Complete Queue

The Packet Complete Queue holds the buffer descriptor numbers that the (i)chipSAR+ has reassembled. As the (i)chipSAR+ completes reassemblies, it places buffer descriptor numbers in the Packet Complete Queue and notifies the system that descriptors have been added to the queue. System software pulls buffer descriptor numbers out of the Packet Complete Queue; then, depending on the reassembly mode, the software either programs DLEs to move the on-board buffer to system memory or it uses the data that was already reassembled in system memory (Cell FIFO mode).

Two (i)chipSAR+ registers contain the pointers of the Packet Complete Queue:

- Packet Complete Queue Read Pointer
- Packet Complete Queue Write Pointer

The (i)chipSAR+ manipulates the Packet Complete Queue Write Pointer. System software manipulates the Packet Complete Queue Read Pointer.

The (i)chipSAR+ can be programmed to interrupt whenever the Packet Complete Queue goes from empty to non-empty.

If the Free Buffer Queue should ever empty and the (i)chipSAR+ begins receiving new AAL5 packets, the (i)chipSAR+ will drop the cells of the packet. The (i)chipSAR+ will also note on a per VC basis that it dropped the cells. Then as buffers are added to the empty Free Buffer Queue, the (i)chipSAR+ uses the buffers only for new reassemblies. AAL5 packets that had cells dropped at the start because of no buffer will continue have cells dropped until the PTI field specifies that the packet is complete.

Exception Queue

The Exception Queue can be used to report any of three possible exceptions:

- No buffer available
- Invalid VCI

- Invalid VPI

The invalid VCI and invalid VPI error conditions are discussed in *VPI/VCI Recognition on page 38* of this chapter.

The no buffer available exception occurs when the start of a new AAL5 packet is received but no buffers are available in the Free Buffer Queue. The (i)chipSAR+ drops the AAL5 packet and reports this exception.

The Exception Queue operates like the Packet Complete Queue in that there is a Read and Write Pointer in the Reassembly Engine registers, and the (i)chipSAR+ can be programmed to interrupt when the Exception Queue goes from empty to non-empty.

Raw Cell/OAM Cell Support

Each VC to be received can also be programmed (on a per VC basis) to treat incoming cells as simple raw cells, with no associated adaptation layer. In this case, cells (including the four header bytes) are written to a Raw Cell Queue. The Raw Cell Queue exists either in Packet/Cell Memory or in system memory, depending on the reassembly mode selected.

Two (i)chipSAR+ registers contain the pointers of the Raw Cell Queue:

- Raw Cell Queue Read Pointer
- Raw Cell Queue Write Pointer

The (i)chipSAR+ manipulates the Raw Cell Queue Write Pointer. System software manipulates the Raw Cell Queue Read Pointer.

Each cell entry of the Raw Cell Queue is 64 bytes long, consisting of 52 bytes of cell followed by 12 bytes of dummy data.

In addition to VCs that are specifically programmed for the Raw Cell Queue, the (i)chipSAR+ also handles OAM F5 cells. The (i)chipSAR+ checks the CRC-10 and, if the CRC-10 is okay, the (i)chipSAR+ sends the OAM F5 cell to the Raw Cell Queue. The (i)chipSAR+ sends OAM F5 cells to the Raw Cell Queue regardless of their VPI/VCI combination. The (i)chipSAR+ drops OAM F5 cells with CRC-10.

The (i)chipSAR+ can be programmed to interrupt when the Raw Cell Queue goes from empty to non-empty. The (i)chipSAR+ can also be programmed either to drop any raw cells that surpass its full capacity or to overwrite old cells in the Raw Cell Queue with new cells.

In addition, the (i)chipSAR+ can be programmed to place all received RM cells into the raw cell queue. Placing received RM cells into the raw cell queue does not affect the hardware operation of the RM cells. This option is for diagnostic purposes only.

VPI/VCI Recognition

The (i)chipSAR+ has a flexible mechanism for decoding VPI/VCI combinations and for supporting multiple VPIs. Multiple VPIs can be supported with programmable numbers of VCIs (programmable on a per VPI basis).

The block diagram in *Figure 4-3* shows the decoding of VPI/VCI information in a received cell.

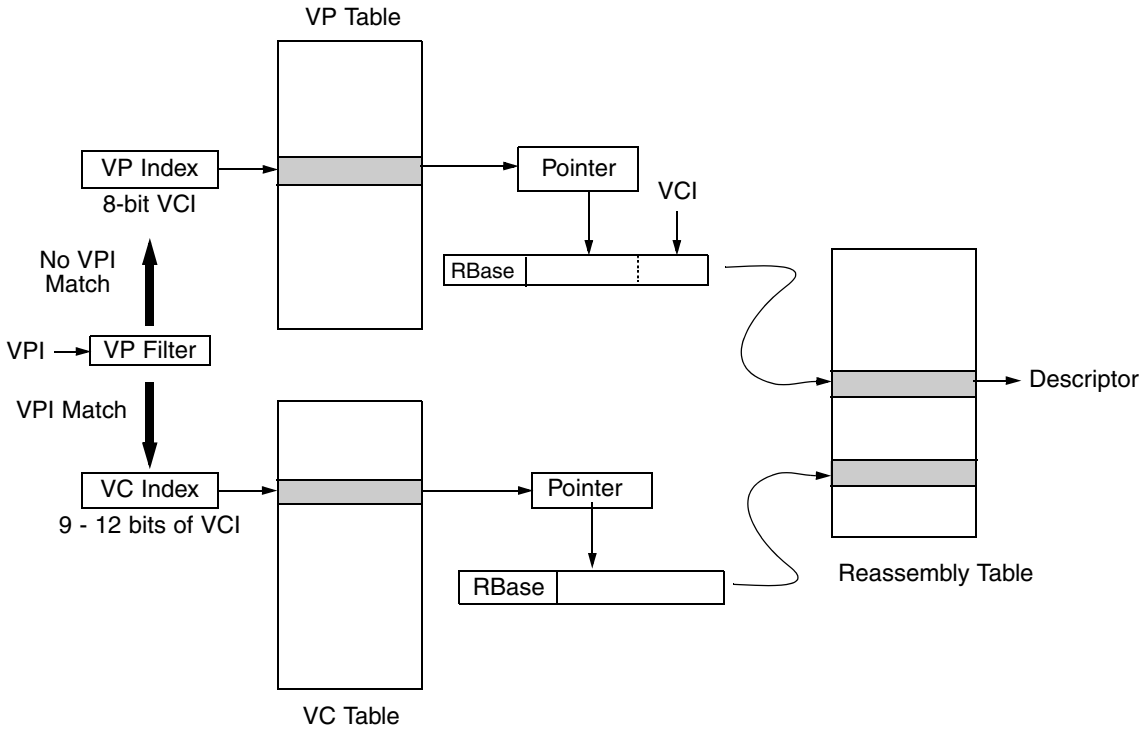

Figure 4-3. VPI/VCI Decoding

Control flow is as follows:

- When a cell arrives and the header checksum is validated, the 8-bit VPI is first checked against the 8-bit VPI Filter Register in the Reassembly Engine.
- If the VPI matches the VPI Filter Register, then the VCI of the cell is used to index into the VC Table.

  The VC Table contains as many entries as the maximum number of VCs that the (i)chipSAR+ is programmed to support. Each entry in the VC Table is 2 bytes long and contains a pointer into the reassembly table. Entries in the VC Table can also have a bit combination designating an invalid VCI.

  At (i)chipSAR+ initialization, system software sets all VC Table entries to indicate an invalid VCI, and then sets them to a correct value as each VC is activated.
- Decoding for all VPI combinations that do not match the VPI Filter Register goes through the VP Table.

  This table contains 256 2-byte entries. Each entry contains two fields. One of the fields is for a Reassembly Table pointer. The other field designates whether the VPI is valid. If the VPI is valid, the field also identifies how many VCI bits should be concatenated to the Reassembly Table pointer for the VPI.

At (i)chipSAR+ initialization, system software sets all VP table entries to indicate an invalid VPI, and then sets them to a correct value as each VP is activated.

The number of Reassembly Table entries equals the maximum number of VCs that the (i)chipSAR+ has been programmed to support. Each entry in the Reassembly Table is two bytes long and contains two fields. One field contains the buffer descriptor number of the buffer currently used for reassembly by this VC. The other field is an information field that designates the state of the reassembly and also designates whether AAL5 reassembly or the Raw Cell Queue is used.

This decoding capability creates a powerful mechanism for VPI/VCI recognition in cases where a primary VPI exists for most of the connections but where other VPI combinations are possible. VCI combinations for the primary VPI can be distributed throughout the range of VCIs, as defined by the maximum number of VCs. For example, if the (i)chipSAR+ is configured to support 4K VCs, and 2K VCs are in the primary VPI, those 2K VCs can be anywhere within the 4K range of VCI values.

Any number of other VPIs can also be used. Each VPI can support a programmable number of VCIs. A restriction of the secondary VPIs is that the VCI combinations must not exceed the maximum number of VCIs supported for that VPI.

If the VPI/VCI combination of a received cell maps to either an invalid VCI (from the Reassembly VC table) or an invalid VPI (from the Reassembly VP table), an exception is generated in the Exception Queue.

Packet Aging

The (i)chipSAR+ also has a flexible capability for determining whether a packet has timed out and whether its partially filled buffer should be freed.

A programmable contiguous range of VCs (represented in the Reassembly table) can be enabled for packet aging tests. The range can be from 32 VCs to the maximum number of VCs supported by the (i)chipSAR+.

Whenever a cell is reassembled into a VC's buffer, a timeout count is reset in the VC's current Buffer Descriptor table entry. At a programmable interval, a sequential VC within the enabled range is tested to see if it is in the middle of a reassembly, and if it is, the VC is further tested to determine its timeout count in its Buffer Descriptor table. If the timeout has not occurred, the timeout count is incremented and written back to the Buffer Descriptor table entry.

If a timeout occurs, the buffer is returned to the system through the Packet Complete Queue with a packet timeout indication set in the mode bits of the Buffer Descriptor entry.

Statistics Registers

The following (i)chipSAR+ registers can be accessed to determine statistics of cell reception:

- 32-Bit Cell Counter

- 16-Bit Dropped Packet Counter
- 16-Bit Error Counter

Each of these counters has a register address that gives the value and clears the counter, and also has a different address that gives the value without clearing the counter.

32-Bit Cell Counter

The 32-Bit Cell Counter counts all received cells except those with invalid header checksums and OAM F5 cells with invalid CRC-10 combinations.

Note that this counter does count cells that are dropped because a free buffer was lacking or because the Raw Cell Queue was full. It also counts cells with invalid VPI/VCI combinations.

16-Bit Dropped Packet Counter

As the name implies, the 16-Bit Dropped Packet Counter counts the number of AAL5 packets (not cells) that were dropped as a result of no free buffers.

16-Bit Error Counter

The 16-Bit Error Counter counts the number of cells that were received with either incorrect header checksums or incorrect CRC-10 values (only with OAM F5 cells or ABR RM cells).

PCI Bus Interface Software Reference 5

Overview

This chapter serves as a software reference for the PCI Bus interface of the (i)chipSAR+. It includes detailed descriptions of the (i)chipSAR+ host memory structures, PCI slave address maps, and PCI interface registers. It also describes the handling of PCI errors and interrupts, identifies the contents of the PCI EEPROM, and describes PMC BUSMODE signals.

System Memory Structures

The (i)chipSAR+ is a DMA Master in the movement of packet data into and out of system memory.

Transmit Data Buffers

System software is responsible for placing transmit data into a system memory buffer that is accessible by the (i)chipSAR+ DMA engine. Transmit buffers can have the following characteristics:

- Transmit buffers can be any size, up to a maximum of 64 Kbytes (maximum packet size for the (i)chipSAR+).
- Transmit buffers can start on any byte boundary.
- A single packet can be chained across multiple transmit buffers.

Receive Data Buffers

The characteristics of system memory receive data buffers differ depending on whether on-board reassembly or off-board reassembly (Cell FIFO) is used.

With on-board reassembly, receive buffers can have the following characteristics:

- Receive buffers can be any size, up to a maximum of 64 Kbytes (maximum packet size for the (i)chipSAR+).
- Receive buffers can start on any 32-bit word boundary.
- A single packet can be chained across multiple receive buffers.

With off-board reassembly, receive buffers can have the following characteristics:

- Because chaining of buffers is not allowed in off-board reassembly, receive buffers must be at least as large as the largest possible packet.
- Receive buffers can start on any 32-bit boundary.

Descriptor List Entries (DLEs)

DMA operations are described to the (i)chipSAR+ via two queues of control structures that reside in host memory. A set of pointer and counter registers inside the (i)chipSAR+ notify the (i)chipSAR+ of DMA requests from the host. The transfer direction of the DMA operation is determined by writing to one of two transaction counter registers in the (i)chipSAR+.

- The Transmit Transaction Counter is used for transfers when the source is system memory.
- Receive Transaction Counter is used for transfers when the destination is system memory.

The data structure the (i)chipSAR+ uses to move data is the Descriptor List Entry (DLE). The host builds the DLE and notifies the (i)chipSAR+ through the appropriate transaction counter to perform the DMA operation.

Table 5-1 shows the structure of a DLE:

Table 5-1. DLE Structure Definition

Address	Byte 3	Byte 2	Byte 1	Byte 0
0x00	System Address (32 bits)			
0x04	Reserved		Local Buffer Address (20 bits)	
0x08	Reserved	Reserved	Byte Count (64 Kbyte max)	
0x0C	Mode Word		PRQ_WR_PTR Data	

The 4-byte System Address is the starting address for the packet in host memory. This address can be on any byte boundary.

The 20-bit Local Buffer Address is the starting address at which this packet will be DMA'ed into the local SRAM buffer memory. This address can be on any byte boundary, but for full packets (not packet segments) or the first packet segment of a full packet, this address must be on a longword boundary.

The 16-bit Byte Count describes the size in bytes of the packet.

Table 5-2 describes the bits of a Transmit Mode word in the DLE structure.

Table 5-2. DLE Transmit Mode Bits

Bit	Definition
15:2	Reserved
01	DMA Interrupt Enable. 1 – Generate PCI Interrupt when the DMA is complete and all data has been transferred to Side RAM memory 0 – No Interrupt

Table 5-2. DLE Transmit Mode Bits (cont)

Bit	Definition
00	Packet Segment Indication (PSI). 0 – Last buffer of packet—AutoWrite to the (i)chipSAR+ Packet Ready Queue Write Pointer 1 – Not the last buffer of packet—No AutoWrite For transmit gathers, disable this for all fragments except last

Table 5-3 describes the bits of a Receive Mode word in the DLE structure.

Table 5-3. DLE Receive Mode Bits

Bit	Definition
15:2	Reserved
01	DMA Interrupt Enable. 1 – Generate PCI Interrupt when the DMA is complete and all data is written into system memory 0 – No Interrupt
00	Reserved.

In the case of transmit DLEs, system software must place the PRQ_WR_PTR for the packet into the PRQ_WR_PTR field in those DLEs that have the PSI bit set to 0.

The DLE structures reside in system memory in two circular queues:

- One queue is dedicated to transmits (data direction is from system memory to board buffer).
- The other queue is dedicated to receives (data direction is from board buffer to system memory).

Each queue is exactly 256 DLE entries long. Each queue must start on a 4-Kbyte boundary in system memory.

The beginning system address is programmed in the Transmit and Receive List Address registers during configuration.

PCI Slave Address Maps

This section includes address maps for the Configuration, Memory, and Expansion ROM address spaces. The (i)chipSAR+ does not respond to I/O cycles.

Configuration Address Map

PCI Configuration space is a 256-byte space used for device configuration. *Table 5-4* shows accessible addresses in configuration space.

Table 5-4. PCI Configuration Space Map

Function	Start Address	End Address
Configuration Registers	0x00 + IDSEL	0x3F + IDSEL
Reserved	0x40 + IDSEL	0xFF + IDSEL

Memory Address Map

Using memory accesses, everything on the board can be accessed except Expansion ROM, which can be accessed only in Expansion ROM space. Depending on the amount of SAR control memory installed, the (i)chipSAR+ occupies either 256 Kbytes, 512 Kbytes, or 1 Mbyte. The system writes the memory base address as part of initialization to set the base. Note that the configuration registers are also accessible in memory space.

The following tables show memory maps for each version of available memory space. The actual system address for each version is the address in the appropriate map plus the base address of the card.

Table 5-5. PCI Memory Space Map, 128K SAR Memory

Function	Start Address	End Address
PCI Configuration Registers	0x0000	
Bus Interface Control Registers	0x1000	
Fragmentation Control Registers	0x2000	
Reassembly Control Registers	0x3000	
Front End Registers/DMA Control	0x4000	
Fragmentation Control RAM	0x10000	0x1FFFF
Reassembly Control RAM	0x20000	0x2FFFF
Reserved (do not write)	0x30000	0x3FFFF

Table 5-6. PCI Memory Space Map, 512K SAR Memory

Function	Start Address	End Address
PCI Configuration Registers	0x0000	
Bus Interface Control Registers	0x1000	
Fragmentation Control Registers	0x2000	
Reassembly Control Registers	0x3000	

Table 5-6. PCI Memory Space Map, 512K SAR Memory

Function	Start Address	End Address
Front End Registers/DMA Control	0x4000	
Fragmentation Control RAM	0x40000	0x7FFFF
Reassembly Control RAM	0x80000	0xBFFFF
Reserved (do not write)	0xC0000	0xFFFFF

Table 5-7. PCI Memory Space Map, 1M SAR Memory

Function	Start Address	End Address
PCI Configuration Registers	0x0000	
Bus Interface Control Registers	0x1000	
Fragmentation Control Registers	0x2000	
Reassembly Control Registers	0x3000	
Front End Registers/DMA Control	0x4000	
Fragmentation Control RAM	0x80000	0xFFFFF
Reassembly Control RAM	0x100000	0x17FFFF
Reserved (do not write)	0x180000	0x1FFFFF

Expansion ROM Address Map

Using memory accesses, the Expansion ROM can be accessed at the address programmed into the Expansion ROM Base Address Configuration Register. In addition, the Expansion ROM space can be enabled and disabled via the Expansion ROM Decode Enable bit (bit 0 in the Expansion ROM Base Address Register). When this bit is set, the Expansion ROM is accessible. When this bit is cleared, the Expansion ROM is not accessible.

The Serial EEPROM also loads a bit into the (i)chipSAR+ that tells the (i)chipSAR+ whether the ROM is installed. If the ROM is not installed, then the Expansion ROM Base Register is not writable, which tells the system that ROM is not present. If the ROM is installed, the base register is writable, which tells the system that ROM is present.

The Expansion ROM can be read with 8-, 16- or 32-bit reads. Writes to the Expansion ROM must always be 32 bits. Writes are done only to update the Flash part. The Expansion ROM takes up 64K of space. *Table 5-8* shows the memory map for the Expansion ROM. The actual system address is the base register plus the address in the map.

Table 5-8. PCI Expansion ROM Space Map

Function	Start Address	End Address
Expansion ROM	0x00000	0x3FFFF

PCI Slave Addressing Specifics

This section provides details about addressing entities in the slave memory address map. It also provides details about Bus Interface registers and PCI Configuration registers. Details about Segmentation Control registers, Reassembly Control registers, and Front End registers are provided in other chapters of this guide.

PCI Configuration Registers

The (i)chipSAR+ contains internal registers used for PCI configuration that conform to PCI Specification Revision 2.1. These configuration registers are accessed by a C/$\overline{\text{BE}}$[3:0] command of 1010 (Configuration Read) or 1011 (configuration Write) plus the IDSEL signal active. These registers can also be accessed in memory space (see the appropriate memory space map on page).

The (i)chipSAR+ supports the 64-byte header portion of the configuration space. None of the device-specific registers in locations 65 through 255 are used. The configuration registers are accessible only by PCI configuration cycles, or by PCI memory cycles. Upon reset, the serial EEPROM contents are loaded into the configuration registers (if EE_EN pin is high). This load takes approximately 2 ms (milliseconds).

Until this load is complete, any configuration cycles to the (i)chipSAR+ will be retried. All write accesses to reserved locations will have no effect; reads from reserved locations will return a value of 0. Although these registers are depicted as 32-bit registers here, any size access is acceptable: byte, word, or Dword. Shaded bits are reserved.

Following is a map of the (i)chipSAR+ configuration registers:

Table 5-9. PCI Configuration Registers

Byte 3	Byte 2	Byte 1	Byte 0	Offset
Configuration ID Register				00h
Configuration Command/Status Register				04h
Configuration Class/Revision Register				08h
Configuration Latency/Type/Cache Register				0Ch
Configuration Memory Address Register				10h
Reserved				14h
Reserved				18h
Reserved				1Ch
Reserved				20h
Reserved				24h
Reserved				28h
Reserved				2Ch
Configuration Expansion ROM Base Address Register				30h

Table 5-9. PCI Configuration Registers (cont)

Byte 3	Byte 2	Byte 1	Byte 0	Offset
Reserved				34h
Reserved				38h
Configuration Interrupt Register				3Ch
Reserved				40h–FFh

Configuration ID Register

Register Name:Configuration ID Register
Function(s):Vendor and Device IDs
Address Offset:00h
Reset Init Value:0x0008107E
EEPROM Load Init Value:0x0008107E
Attribute: Read only

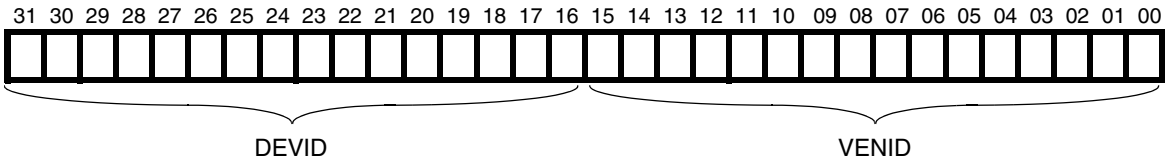

Figure 5-1. PCI Configuration ID Register

Table 5-10. PCI Configuration ID Register

Bit	Description
31:16 DEVID(15:00)	Device ID. Initialized to = 0x0008, indicating an (i)chipSAR+ based SAR.
15:00 VENID(15:00)	Vendor ID. Initialized to = 0x107E, which is the PCI SIG defined ID for Interphase Corp.

Configuration Command/Status Register

Register Name:Configuration Command/Status Register
Function(s):PCI Command and Status Registers
Address Offset:04h
Reset Init Value:0x02000000
EEPROM Load Init Value:0x02000000
Attribute: Status Register read/write with 1 to reset bits only
Command Register read/write

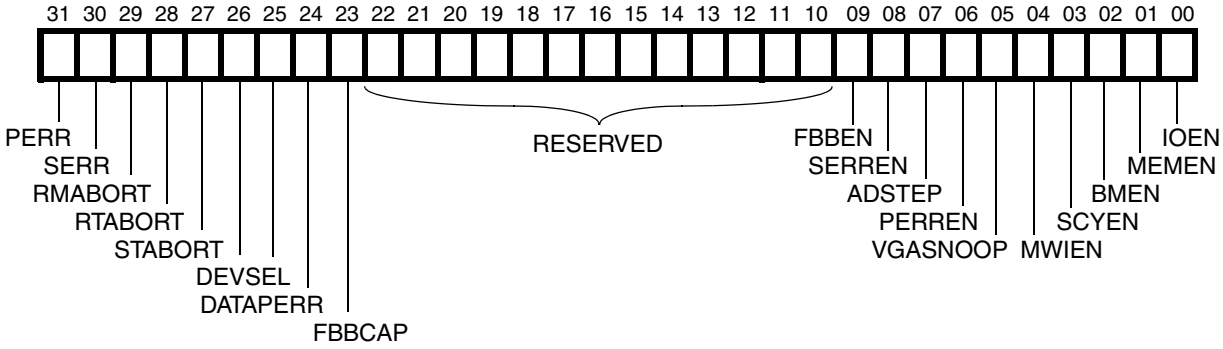

Figure 5-2. PCI Configuration Command/Status Register

Table 5-11. PCI Configuration Command/Status Register

Bit	Description
31 PERR	Status Register Parity Error status. This bit is set when the (i)chipSAR+ detects an address or data parity error as either target or master. Writing a 1 to this bit clears it.
30 SERR	Status Register System Error status. This bit is set to 1 when the (i)chipSAR+ asserts SERR#. The SERREN bit must be set before this bit can be set. Writing a 1 to this bit clears it.
29 RMABORT	Received Master Abort. As bus master, the (i)chipSAR+ sets this bit when its transaction is terminated with a master abort (meaning it did not receive DEVSEL). Writing a 1 to this bit clears it. The (i)chipSAR+ might also generate a SERR if it receives a master abort, if the SERREN bit is set in the Command Register. **NOTE:** If the RMABORT or RTABORT error occurs, the (i)chipSAR+ master machine will enter a freeze state until a reset (hard or soft) occurs. These errors should never occur in normal operation. The (i)chipSAR+ target interface will still function, however.

Table 5-11. PCI Configuration Command/Status Register (cont)

Bit	Description
28 RTABORT	Received Target Abort. As bus master, the (i)chipSAR+ sets this bit when its transaction is terminated with a target-abort. Writing a 1 to this bit clears it. The (i)chipSAR+ might also generate a SERR if it receives target abort, if the SERREN bit is set in the Command Register. **NOTE:** If the RMABORT or RTABORT error occurs, the (i)chipSAR+ master machine will enter a freeze state until a reset (hard or soft) occurs. These errors should never occur in normal operation. The (i)chipSAR+ target interface will still function, however.
27 STABORT	Signalled Target Abort. This bit is always 0. The (i)chipSAR+ will never generate target abort.
26:25 DEVSEL(01:00)	DEVSEL Timing. Initialized to 01h, meaning that the (i)chipSAR+ supports medium speed DEVSEL assertion timing. Read only.
24 DATAPERR	Data Parity Error detected. This bit is set when the (i)chipSAR+, as a master reading data, detects a data parity error, or when writing data, detects that the PERR signal is active. Writing a 1 to this bit clears it. The DATAPERR bit will be set only if the Parity Error Response Enable bit (bit 6, PERREN) is set.
23 FBBCAP	Fast Back to Back Capable. The (i)chipSAR+ is incapable of accepting fast back to back cycles, so this bit is = 0. Read only.
22:10 RESERVED	Reserved. Read as 0's.
09 FBBEN	Fast Back to Back Enable. Because the (i)chipSAR+ does not support fast back to back cycles, this bit is always read as 0. Not writable.
08 SERREN	SERR Enable. When this bit is 1, the SERR# line is enabled to be driven. When this bit is 0, the SERR# will never be driven. This bit must also be set before the SERR bit can be set. This bit and bit 6 (PERREN) must both be set to 1 to report address parity errors.
07 ADSTEP	Address Data Stepping. The (i)chipSAR+ does not use stepping, so this bit is always read as 0. Not writable.
06 PERREN	PERR Enable. When this bit is 0 and the (i)chipSAR+ detects a parity error, it sets only the detected parity error bit (PERR) in the Status Register. When this bit is 1, and a data parity error occurs during a master operation, the DATAPERR bit and the PERR bit will be set, the PERR pin will be driven active, and, if the mask bit is enabled, a PCI error interrupt will occur. If this bit is set and a parity error occurs during a target operation, then the PERR bit is set and the PERR pin will be driven active. This bit also enables reporting address parity errors through the SERR# pin and the SERR bit in the Control Register.
05 VGASNOOP	VGA Palette Snoop. Read as 0. Not writable.
04 MWIEN	Memory Write and Invalidate Cycle Enable. This bit, when set to 1, enables the (i)chipSAR+ master to use Memory Write and Invalidate commands if the burst size matches the Cache Line Size Register. Otherwise, Memory Write commands are used.
03 SCYEN	Special Cycle Enable. The (i)chipSAR+ does not generate this type of cycle, so this bit is read as 0. Not writable.

Table 5-11. PCI Configuration Command/Status Register (cont)

Bit	Description
02 BMEN	Bus Master Enable. This bit enables the (i)chipSAR+ to become bus master. Read/Writable.
01 MEMEN	Memory Space Access Enable. This bit enables the (i)chipSAR+ to respond to Memory space accesses. Read/Writable. **CAUTION**: When this bit is set, a memory access should not be attempted to the (i)chipSAR+ for at least 1 us. Failure to wait long enough could result in a Master Abort error.
00 IOEN	I/O Space Enable. The (i)chipSAR+ does not respond to I/O cycles. Read as 0. Not writable.

Configuration Class/Revision Register

Register Name:Configuration Class/Revision Register
Function(s):Board Class and Board Revision Information
Address Offset:08h
Reset Init Value:0x02030000
EEPROM Load Init Value:0x020300xx
Attribute: Read only

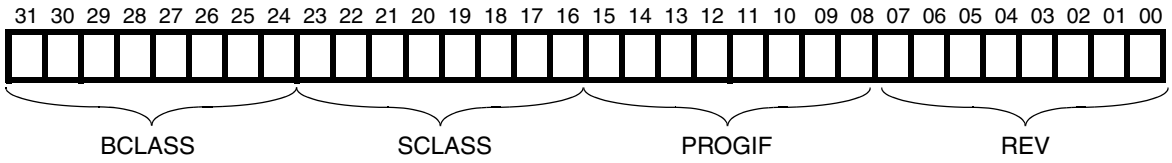

Figure 5-3. PCI Configuration Class/Revision Register

Table 5-12. PCI Configuration Class/Revision Register

Bit	Description
31:24 BCLASS(07:00)	Base Class. Initialized to 02h, indicating a network controller product.
23:16 SCLASS(07:00)	Sub Class. Initialized to 03h, indicating an ATM controller.
15:08 PROGIF(07:00)	Programming Interface. Initialized to 00h.
07:00 REV(07:00)	Revision. This indicates a board revision. It should be bumped each time the hardware level is changed, which implies that the firmware (or the data that gets loaded into EEPROM) must change each time the hardware level changes. The first hardware level will use 00h in this register, the second will use 01h, and so on.

Configuration Latency/Type/Cache Register

Register Name:Configuration Latency/Type/Cache Register
Function(s):Header Type and Latency Counter and Cache Line Size Register
Address Offset:0Ch
Reset Init Value:0x00000000
EEPROM Load Init Value:0x00000000
Attribute: Header Register is read only.
Latency Register is read/write capable.

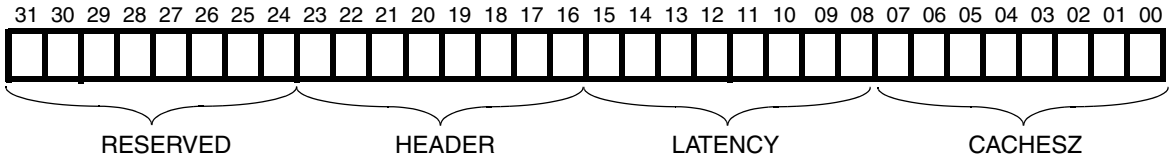

Figure 5-4. PCI Configuration Latency/Type Register

Table 5-13. PCI Configuration Latency/Type Register

Bit	Description
31:24 RESERVED	Reserved. Read as 0.
23:16 HEADER(07:00)	Header Type. Initialized to 00h, indicating header type 1 (single function, no registers beyond 3fh).
15:08 LATENCY(07:00)	Latency Timer/Counter. Specifies the maximum time the (i)chipSAR+ can continue with bus master transfers after the system arbiter has removed GNT#. The time is measured in CLK cycles. The working copy of the timer starts counting down when the (i)chipSAR+ asserts FRAME# for the first time during a bus master transfer. The counter stops at zero (0). When the counter is 0 and GNT is deasserted by the system arbiter, the (i)chipSAR+ finishes the current transfer and immediately releases the bus. This register is initialized to FFh. Loading a 00h into this register disables the latency timer. The Latency Timer Register is read/write capable.

Table 5-13. PCI Configuration Latency/Type Register (cont)

Bit	Description
07:00 CACHESZ(07:00)	Cache Line Size Register. This register specifies the system cache line size in units of 32-bit words. The (i)chipSAR+ uses this information to generate Memory Write and Invalidate, Memory Read Line, and Memory Read Multiple commands. Following are the rules that the (i)chipSAR+ uses to determine what type of command to use: READS: If this register = 00h—Memory Read Else If MEMLINE bit = 0—Memory Read Else If burst size = 1—Memory Read Else If 1 < burst size <= this register—Memory Read Line Else If burst size > this register—Memory Read Multiple WRITES: If this register = 00h—Memory Write Else If MWIEN bit = 0—Memory Write Else If burst size is even multiple of register—MWINV Else Memory Write **NOTE**: The (i)chipSAR+ supports only power-of-two cache-line sizes. Non-power-of-two sizes function as though 0 were written to this register. Also, it is possible to selectively turn off the (i)chipSAR+ use of Memory Write and Invalidate, Memory Read Line, and Memory Read Multiple commands by using the MWIEN bit in the Configuration Command Register, and the MEMLINE bit in the (i)chipSAR+ Control Register.

Configuration Memory Base Address Register 128K SAR Memory

Register Name:Configuration Memory Base Address Register
Function(s):Holds Memory base address
Address Offset:10h
Reset Init Value:0x00000000
EEPROM Load Init Value:0x00000000
Attribute: Bits 31:18 read/write; bits 17:00 read only

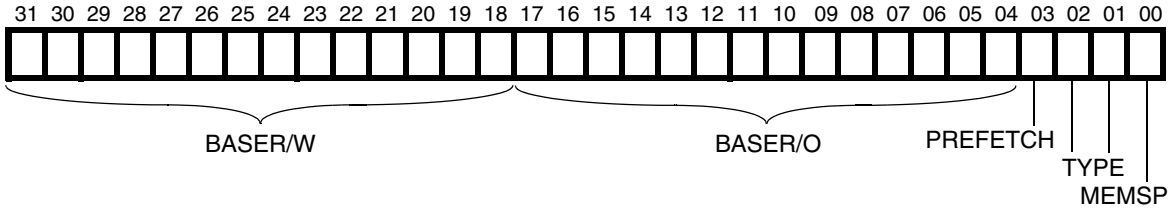

Figure 5-5. PCI Configuration Memory Base Address Register, 128K SAR Memory

Table 5-14. PCI Configuration Memory Base Address Register, 128K SAR Memory

Bit	Description
31:18 BASER/W(13:00)	Memory Base Address Read/Write bits. These bits control the base address. They are initialized to 000h, but the system should write the real address here.
17:04 BASER/O(13:00)	Memory Base Address Read Only. Initialized to 0. Read only.
03 PREFETCH	Prefetch. This bit indicates whether the memory space is prefetchable or not. Initialized to 0. Read only.
02:01 TYPE(01:00)	Memory Type. Initialized to 0, indicating that memory can be located anywhere in 32-bit space. Read only.
00 MEMSP	Memory Space Indicator. Initialized to 0. Read only.

Configuration Memory Base Address Register 512K SAR Memory

Register Name:Configuration Memory Base Address Register
Function(s):Holds Memory base address
Address Offset:10h
Reset Init Value:0x00000000
EEPROM Load Init Value:0x00000000
Attribute: Bits 31:20 read/write; bits 19:00 read only

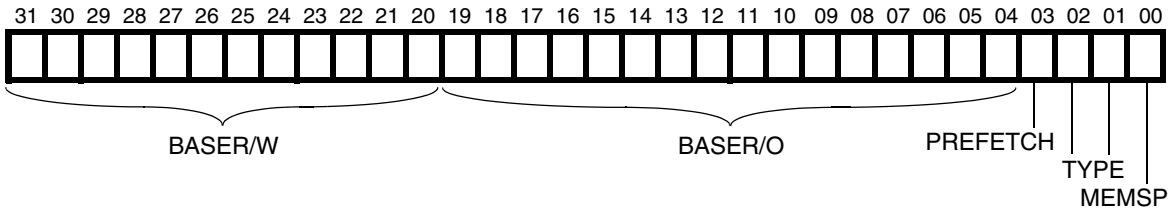

Figure 5-6. PCI Configuration Memory Base Address Register, 512 K SAR Memory

Table 5-15. PCI Configuration Memory Base Address Register, 512K SAR Memory

Bit	Description
31:20 BASER/W(11:00)	Memory Base Address Read/Write bits. These bits control the base address. They are initialized to 0, but the system should write the real address here.
19:04 BASER/O(15:00)	Memory Base Address Read Only. Initialized to 0. Read only.
03 PREFETCH	Prefetch. This bit indicates whether the memory space is prefetchable or not. Initialized to 0. Read only.
02:01 TYPE(01:00)	Memory Type. Initialized to 0, indicating that memory can be located anywhere in 32-bit space. Read only.
00 MEMSP	Memory Space Indicator. Initialized to 0. Read only.

Configuration Memory Base Address Register 1M SAR Memory

Register Name:Configuration Memory Base Address Register
Function(s):Holds Memory base address
Address Offset:10h
Reset Init Value:0x00000000
EEPROM Load Init Value:0x00000000
Attribute: Bits 31:21 read/write; bits 20:00 read only

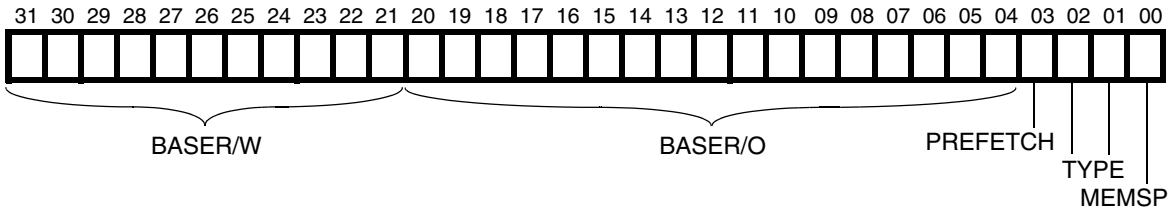

Figure 5-7. PCI Configuration Memory Base Address Register, 1M SAR Memory

Table 5-16. PCI Configuration Memory Base Address Register, 1M SAR Memory

Bit	Description
31:21 BASER/W(09:00)	Memory Base Address Read/Write bits. These bits control the base address. They are initialized to 0, but system should write the real address here.
20:04 BASER/O(16:00)	Memory Base Address Read Only. Initialized to 0. Read only.
03 PREFETCH	Prefetch. This bit indicates whether the memory space is prefetchable or not. Initialized to 0. Read only.
02:01 TYPE(01:00)	Memory Type. Initialized to 0, indicating that memory can be located anywhere in 32-bit space. Read only.
00 MEMSP	Memory Space Indicator. Initialized to 0. Read only.

Configuration Expansion ROM Base Address Register

Register Name:Configuration Expansion ROM Base Address Register
Function(s):Holds Expansion ROM base address
Address Offset:30h
Reset Init Value:0x00000001
EEPROM Load Init Value:0x00000000 if no Expansion ROM present
0x00000001 if Expansion ROM present
Attribute: Bits 31:18 read/write; bits 17:00 read only

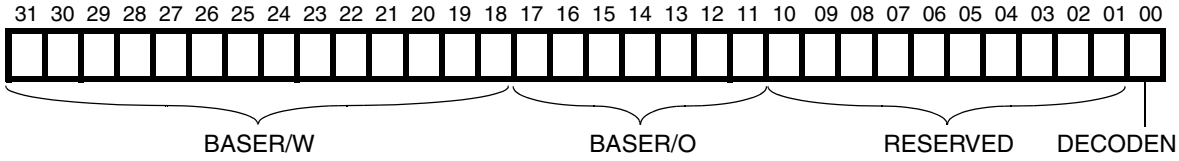

Figure 5-8. PCI Configuration Expansion ROM Base Address Register

Table 5-17. PCI Configuration Expansion ROM Base Address Register

Bit	Description
31:18 BASER/W(13:00)	Expansion ROM Base Address Read/Write bits. These bits control the base address. They are initialized to 0, but the system should write the real address here.
17:11 BASER/O(16:00)	Expansion ROM Base Address Read Only. Initialized to 0. Read only.
10:01 RESERVED	Reserved. Read as 0. Not writable.
00 DECODEN	Expansion ROM Decode Enable. This bit determines whether the (i)chipSAR+ can reply to a Expansion ROM access. The host can write this bit to control enable. 0 = disabled 1 = enabled.

NOTE

This register is programmable from the NOVRAM depending on whether Expansion ROM is present. If Expansion ROM is not present, this register is not writable and always reads as 0.

Configuration Interrupt Register

Register Name:Configuration Interrupt Register
Function(s):Two 8 bit registers: (1) Interrupt Line Register used by system
(2) Interrupt Pin Register used to indicate which interrupt is driven
Address Offset:3Ch
Reset Init Value:0x00000100
EEPROM Load Init Value:0x00000100
Attribute: Interrupt Pin Register read only
Interrupt Line Register read/write
Bits 31:08 read only; bits 07:00 read/write

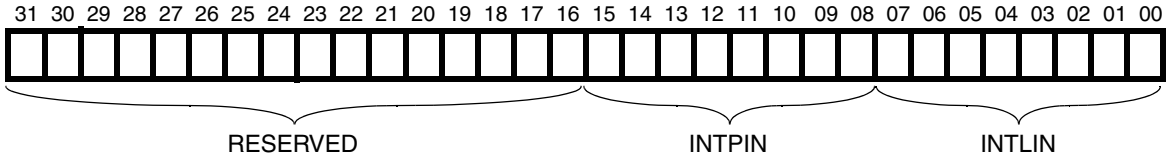

Figure 5-9. Configuration Interrupt Register

Table 5-18. Configuration Interrupt Register

Bit	Description
31:16 RESERVED	Reserved. Read as 0.
15:08 INTPIN(07:00)	Interrupt Pin Register. Tells which pin the (i)chipSAR+ will use for interrupts. INTA# is default. Read only.
07:00 INTLIN(07:00)	Interrupt Line Register. Programmed by POST code. Read/write.

Bus Interface Control Registers

Table 5-19 provides a map of the (i)chipSAR+ Bus Interface Control Registers that are accessible from PCI memory cycles.

Table 5-19. (i)chipSAR+ Bus Interface Control Registers

Byte 3	Byte 2	Byte 1	Byte 0	Offset
Bus Interface Control Register				00h
Bus Interface Status Register				04h
MAC Address 1				08h
(i)chipSAR+ Revision		MAC Address 2		0Ch
External Soft Reset Register				10h
Internal Soft Reset Register				14h
PCI Address Page Register				1Ch
EEPROM Access Register				28h
Cell FIFO Queue Size Register				2Ch
Cell FIFO Mark State Register				30h
Cell FIFO Read Pointer				34h
Cell FIFO Write Pointer				38h
Cell FIFO Cells Available				3Ch

Bus Interface Control Register

Register Name:Bus Interface Control Register
Function(s):Controls various functions of the bus interface
Address Offset:00h
Reset Init Value:0x00000000
EEPROM Load Init Value:Not initialized
Attribute: Read/write

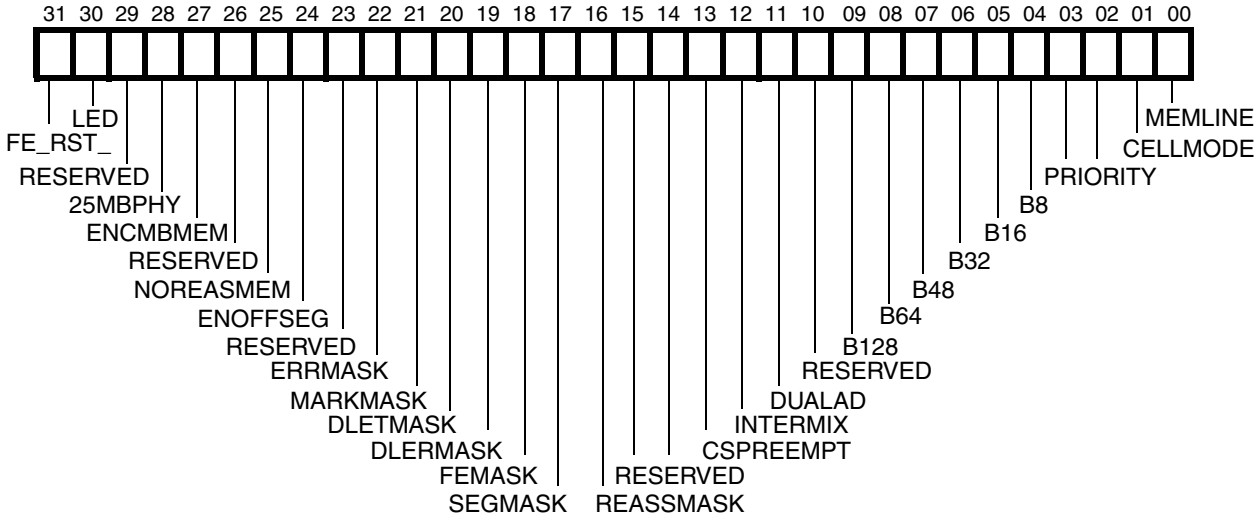

Figure 5-10. Bus Interface Control Register

Table 5-20. Bus Interface Control Register

Bit	Description
31 FE_RST_	Front End Reset. This bit directly sets the front end reset pin. Setting this bit to 1 sets the output pin to a logic 1. Setting this bit to 0 sets the output pin to a logic 0.
30 LED	LED. This bit directly controls the LED output. Setting this bit to 1 sets the output pin to a logic 1. Setting this bit to 0 sets the output pin to a logic 0.
29 RESERVED	Reserved.
28 25MBPHY	25 Mbps Physical Interface. This bit, when set to 1, allows access to PHY registers. This access also requires that bit 27 (Enable Combined Control Memory bit) be set to 1. This bit should be set to 1 only on hardware configured with the IDT 25 Mbit Phy.

Table 5-20. Bus Interface Control Register (cont)

Bit	Description
27 ENCMBMEM	Enable Combined Control Memory. This bit, when set to 1, enables the use of combined control memory mode. In this mode, both the Segmentation Engine and the Reassembly Engine access the same bank of control memory.
26 RESERVED	Reserved.
25 NOREASMEM	No Reassembly Memory. This bit, when set to 1, requires reassembly to be done in Cell FIFO (off-board) mode. When this bit is set to 0, reassembly can occur in on-board mode or in Cell FIFO mode using the internal packet memory. See *DMA of Receive Data on page 8* in *Chapter 5, PCI Bus Interface Software Reference* for general information about reassembly modes.
24 ENOFFSEG	Enable Off-Board Segmentation Mode. This bit, when set to 1, enables the Segmentation Engine to directly access host memory for packet segmentation. See *Basic Segmentation Operation on page 29* in *Chapter 3, Segmentation Engine Functional Description* for general information about off-board segmentation.
23 RESERVED	Reserved.
22 ERRMASK	Error Interrupt Mask. This bit, when set to 1, allows the PCI Error interrupt to cause a PCI INTA# interrupt. When this bit is set to 0, the PCI Error interrupt can still occur, and it will appear in the Status Register, but it will not cause a PCI INTA# interrupt.
21 MARKMASK	MARK Interrupt Mask. This bit, when set to 1, allows the Mark state interrupt to cause a PCI INTA# interrupt. When this bit is set to 0, the mark state interrupt can still occur, and it will appear in the Status Register, but it will not cause a PCI INTA# interrupt.
20 DLETMASK	DLE Transmit Interrupt Mask. This bit, when set to 1, allows a DLE transmit interrupt (if enabled in DLE) to cause a PCI INTA# interrupt. When this bit is set to 0, the DLE transmit interrupt can still occur, and it will appear in the Status Register, but it will not cause a PCI INTA# interrupt.
19 DLERMASK	DLE Receive Interrupt Mask. This bit, when set to 1, allows a DLE receive interrupt (if enabled in DLE) to cause a PCI INTA# interrupt. When this bit is set to 0, the DLE receive interrupt can still occur, and it will appear in the Status Register, but it will not cause a PCI INTA# interrupt.
18 FEMASK	Front End Interrupt Mask. This bit, when set to 1, allows the front end device interrupt to cause a PCI INTA# interrupt. When this bit is set to 0, the front end device interrupt can still occur, and it will show up in the Status Register, but it will not cause a PCI INTA# interrupt.
17 SEGMASK	Segmentation Interrupt Mask. This bit, when set to 1, allows the segmentation interrupt to cause a PCI INTA# interrupt. When 0, the segmentation interrupt can still occur, and it will appear in the Status Register, but it will not cause a PCI INTA# interrupt.
16 REASSMASK	Reassembly Interrupt Mask. This bit, when set to 1, allows the reassembly interrupt to cause a PCI INTA# interrupt. When this bit is set to 0, the reassembly interrupt can still occur, and it will appear in the Status Register, but it will not cause a PCI INTA# interrupt.
15:14 RESERVED	Reserved.

Table 5-20. Bus Interface Control Register (cont)

Bit	Description
13 CSPREEMPT	Control SRAM Preempt bit. When set to 1, PCI target accesses to Control SRAM are highest priority and preempt any internal activity that might be occurring at the time to control SRAM. This preempting is done to minimize the number of wait states for PCI accesses. When this bit is set to 0, preemption is turned off.
12 INTERMIX	Intermix. This bit is meaningful only when running in Cell FIFO Mode (CELLMODE bit = 1). This bit allows the Cell FIFO receives to intermix into the PCI DMA master stream with a transmit DLE that might be in progress. If this bit is 0, then Intermix is off, meaning that after a transmit DLE starts, it must finish before any receive cells can be DMA'ed. When this bit is 1, then intermixing of the transmits and receives is enabled. Intermix is normally desirable. **NOTE:** If transmits have been set to be highest priority, intermixing will not occur, and this bit is meaningless.
11 DUALAD	Dual Address. When this bit is set to 1, the (i)chipSAR+ generates 64-bit dual addresses using the PCI Address Page Register as the upper 32 bits. When this bit is set to 0, the (i)chipSAR+, when acting as PCI master, generates 32-bit addresses.
10 RESERVED	Reserved.
09 B128	Burst 128. This bit, when set to 1, enables the (i)chipSAR+ to use PCI burst sizes of 128 bytes (32 transfers). This bit is mutually exclusive with other burst bits. See the Burst 8 bit (bit 4) for an example.
08 B64	Burst 64. This bit, when set to 1, enables the (i)chipSAR+ to use PCI burst sizes of 64 bytes (16 transfers). This bit is mutually exclusive with other burst bits. See the Burst 8 bit (bit 4) for an example.
07 B48	Burst 48. This bit, when set to 1, enables the (i)chipSAR+ to use PCI burst sizes of 48 bytes (12 transfers). **NOTE:** 48-byte bursts are attempted ONLY on receives (PCI bus writes), never on transmits. 48-byte bursts are needed when receive Cell FIFO mode is enabled and when receive cell sizes are exactly 48 bytes long. If this bit is enabled, the (i)chipSAR+ writes an entire cell to the PCI bus in 1 burst. Use of this bit is not recommended if the PCI target cannot effectively handle non powers of 2 burst sizes. If this bit is disabled, then the 48-byte burst is done in the next most efficient manner, possibly as a 32- and a 16- byte burst if B32 and B16 are enabled. This bit is mutually exclusive with other burst bits. See the Burst 8 bit (bit 4) for an example.
06 B32	Burst 32. This bit, when set to 1, enables the (i)chipSAR+ to use PCI burst sizes of 32 bytes (8 transfers). This bit is mutually exclusive with other burst bits. See the Burst 8 bit (bit 4) for an example.
05 B16	Burst 16. This bit, when set to 1, enables the (i)chipSAR+ to use PCI burst sizes of 16 bytes (4 transfers). This bit is mutually exclusive with other burst bits. See the Burst 8 bit (bit 4) for an example.

Table 5-20. Bus Interface Control Register (cont)

Bit	Description
04 B8	Burst 8. This bit, when set to 1, enables the (i)chipSAR+ to use PCI burst sizes of 8 bytes (2 transfers). This bit is mutually exclusive with other burst bits. For example, it is possible to allow only 8-byte and 16-byte bursts by enabling B8 and B16. The (i)chipSAR+ will never do 32-, 48-, 64- or 128-byte bursts in this case.
03:02 PRIORITY(01:00	Priority. These bits control the priority of receive vs. transmit in the (i)chipSAR+. 00 = Receive highest 01 = Transmit highest 10 = Round Robin 11 = Undefined
01 CELLMODE	Cell FIFO Mode enable. When this bit is set to 1, the (i)chipSAR+ will reassemble off board (Cell FIFO mode). When the bit is 0, the (i)chipSAR+ will do reassembly on board (it will not use Cell FIFO mode).
00 MEMLINE	Memory Line Enable. When this bit is 0, the (i)chipSAR+ as bus master is free to use the Memory Read Line and Memory Read Multiple commands. If this bit is set to 1, or if the Cache Line Size Register equals 0, the (i)chipSAR+ will always use the Memory Read Command. See the *Configuration Latency/Type/Cache Register on page 55* for more details about these commands.

Bus Interface Status Register

Register Name:Bus Interface Status Register
Function(s):Bus interface real time status for interrupts
Address Offset:04h
Reset Init Value:0x00000000
EEPROM Load Init Value:Not initialized, except for cntl_mem_size and addr_par_chk
Attribute: Read/write to reset bits only

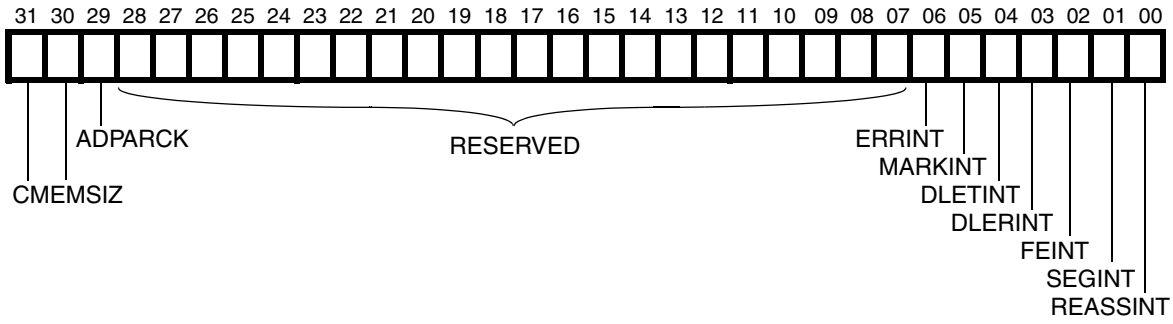

Figure 5-11. Bus Interface Status Register

Table 5-21. Bus Interface Status Register

Bit	Description
31:30 CMEMSIZ	SAR Control Memory Size. These two bits indicate the amount of SAR Control Memory installed. The value of these bits affects the memory map. Read only; settable only by the serial EEPROM. 00=128K 01=512K 10=1M
29 ADPARCK	Address Phase Parity Check. Setting this bit to 1 allows the use of parity checking during address phases. By default, addresses are matched even with parity errors. This matching does not affect the status reporting in any way. Read only; settable only by the Serial EEPROM.
28:07 RESERVED	Reserved.

Table 5-21. Bus Interface Status Register (cont)

Bit	Description
06 ERRINT	Error Interrupt. If set to 1, this bit indicates that a PCI error has occurred. This bit is the "or" of three bits in the Configuration Command/Status Register. These three bits are: Bit 29, RMABORT Bit 28, RTABORT Bit 24, DATAPERR If the corresponding enable bit in the Control Register is set, a PCI INTA# interrupt will occur. To clear this interrupt, clear the appropriate bit in the Configuration Command/Status Register.
05 MARKINT	Mark Interrupt. If set to 1, this bit indicates that the mark condition has completed. If the corresponding enable bit in the Control Register is set, a PCI INTA# interrupt will occur. To reset this interrupt, write a 1 to this bit.
04 DLETINT	DLE Transmit Interrupt. If set to 1, this bit indicates the completion of a transmit DLE with the Interrupt Enable bit on. If the corresponding enable bit in the Control Register is set, a PCI INTA# interrupt will occur. To reset this interrupt, write a 1 to this bit.
03 DLERINT	DLE Receive Interrupt. If set to 1, this bit indicates the completion of a receive DLE with the Interrupt Enable bit on. If the corresponding enable bit in the Control Register is set, a PCI INTA# interrupt will occur. To reset this interrupt, write a 1 to this bit.
02 FEINT	Front End Interrupt. If set to 1, this bit indicates that the front end device has asserted its interrupt line. If the corresponding enable bit in the Control Register is set, a PCI INTA# interrupt will occur. To clear this interrupt, clear the Front End interrupt at the front end device.
01 SEGINT	Segmentation Interrupt. If set to 1, this bit indicates that the segmentation circuit has asserted its interrupt line. If the corresponding enable bit in the Control Register is set, a PCI INTA# interrupt will occur. To clear this interrupt, clear the segmentation interrupt by accessing the Segmentation Status Register.
00 REASSINT	Reassembly Interrupt. If set to 1, this bit indicates that the reassembly circuit has asserted its interrupt line. If the corresponding enable bit in the Control Register is set, PCI INTA# interrupt will occur. To clear this interrupt, clear the Reassembly Interrupt by accessing the Reassembly Status Register.

MAC Address Register 1

Register Name:MAC Address Register 1
Function(s):Upper 32 bits of the 48-bit MAC address
Address Offset:08h
Reset Init Value:0x00000000
EEPROM Load Init Value:Upper 32 bits of Unique MAC Address
Attribute: Read/write

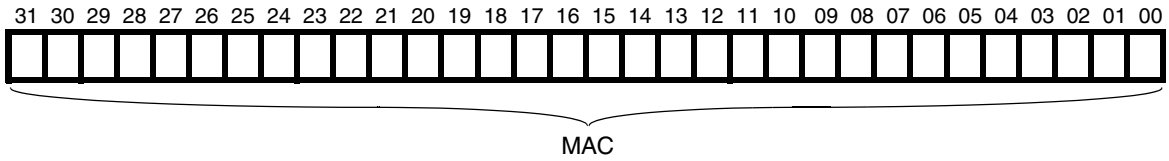

Figure 5-12. MAC Address Register 1

Table 5-22. MAC Address Register 1

Bit	Description
31:00 MAC(48:16)	MAC Address. Upper 32 bits of 48-bit MAC Address. Read/write.

MAC Address Register 2

Register Name:MAC Address Register 2 and (i)chipSAR+ Revision
Function(s):Lower 16 bits of the 48-bit MAC address and 16 bits of revision
Address Offset:0Ch
Reset Init Value:0x02000000
EEPROM Load Init Value:MAC2 initialized to lower 16 bits of 48 bit MAC Address
Revision/ID/Manuf not initialized
Attribute: MAC2 Read/write
Revision/ID/Manuf read only

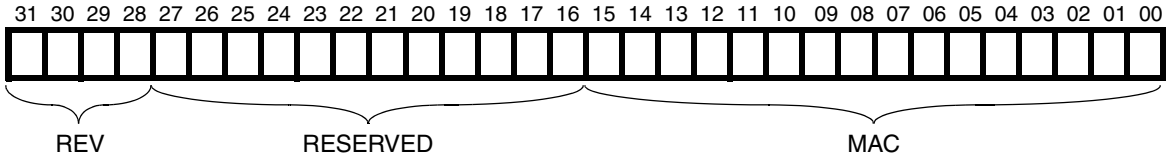

Figure 5-13. MAC Address Register 2

Table 5-23. MAC Address Register 2

Bit	Description
31:28 REV(03:00)	Revision. (i)chipSAR+ ASIC Revision Number. The first ASIC will be 0h, the second 1h, etc. Read only.
27:16 RESERVED	Read only and mask data.
15:00 MAC(15:00)	MAC Address. Lower 16 bits of 48-bit MAC Address. Read/write.

External Soft Reset Register

Register Name:(i)chipSAR+ External Soft Reset Register
Function(s):(i)chipSAR+ internal and external soft reset control with EEPROM load
Address Offset:10h
Reset Init Value:Not initialized
EEPROM Load Init Value:Not initialized
Attribute: Write only, will not store data

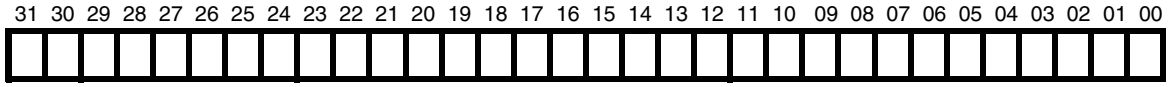

Figure 5-14. External Soft Reset Register

Table 5-24. External Soft Reset Register

Bit	Description
31:00	Soft Reset External. Writing anything to this register causes the (i)chipSAR+ to perform an internal and external (Will Drive LB_RST) reset and to load the serial EEPROM (if EE_EN pin high). This action is equivalent to a PCI RESET# hard reset. The (i)chipSAR+ will generate reset for approximately 1us (at 33 mhz PCI clock), during which time any PCI accesses to the (i)chipSAR+ will be retried. Also, until the serial EEPROM is loaded (approximately 2 ms) any PCI accesses will be retried. Software does not have to clear this bit, but it should wait for at least 5 us before accessing the board again in any way.

Internal Soft Reset Register

Register Name:(i)chipSAR+ Internal Soft Reset Register
Function(s):(i)chipSAR+ internal soft reset control without EEPROM load
Address Offset:14h
Reset Init Value:Not initialized
EEPROM Load Init Value:Not initialized
Attribute: Write only, will not store data

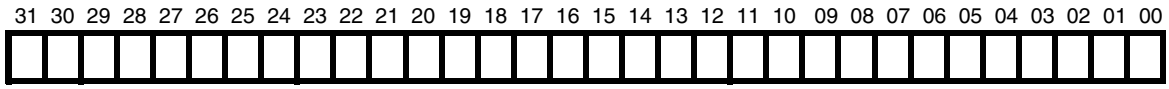

Figure 5-15. Internal Soft Reset Register

Table 5-25. Internal Soft Reset Register

Bit	Description
31:00	Soft Reset Internal. Writing anything to this register causes the (i)chipSAR+ to perform an internal reset only (the (i)chipSAR+ will not drive LB_RST), without reloading the serial EEPROM. The (i)chipSAR+ Internal Registers initialized by the serial EEPROM will not change. The registers initialized by reset will be reset. (To determine how a register will be initialized, look at *Reset Init Value* row in the top part of the register description in this chapter. If *Reset Init Value* says *Not initialized*, the register is not affected by a soft reset. If *Reset Init Value* contains a value, the register is reset to that value during a soft reset.) This type of reset is useful if the driver wants to reset all the state machines without having to configure and initialize all the registers. The (i)chipSAR+ will generate reset for approximately 1us (at 33 mhz PCI clock), during which time any PCI accesses to the (i)chipSAR+ will be retried. Software does not have to clear this bit, but it should wait for at least 5 us before accessing the board again in any way.

PCI Address Page Register

Register Name:(i)chipSAR+ PCI Address Page Register
Function(s):Sets the upper 32 bits in 64-bit dual address mode
Address Offset:1Ch
Reset Init Value:Not initialized
EEPROM Load Init Value:Not initialized
Attribute: Read/write

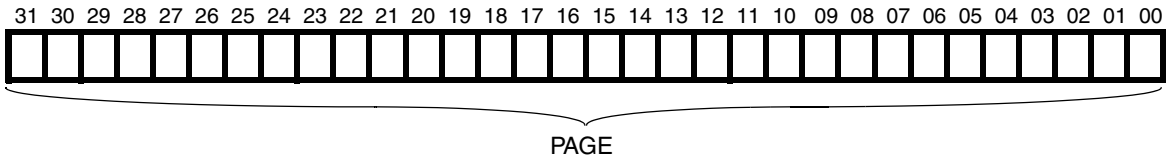

Figure 5-16. PCI Address Page Register

Table 5-26. PCI Address Page Register

Bit	Description
31:00 PAGE(31:00)	PCI Address Page Register. Upper 32 bits of 64-bit dual PCI address. The DUALAD bit in the Control Register must be 1; otherwise, this register is a don't care.

EEPROM Access Register

Register Name:(i)chipSAR+ EEPROM Access Register
Function(s):Allows accessing the EEPROM to store and read data
Address Offset:28h
Reset Init Value:0x00000000
EEPROM Load Init Value:Not initialized
Attribute: Bits 02:00 Read/write
Bit 03 Read only

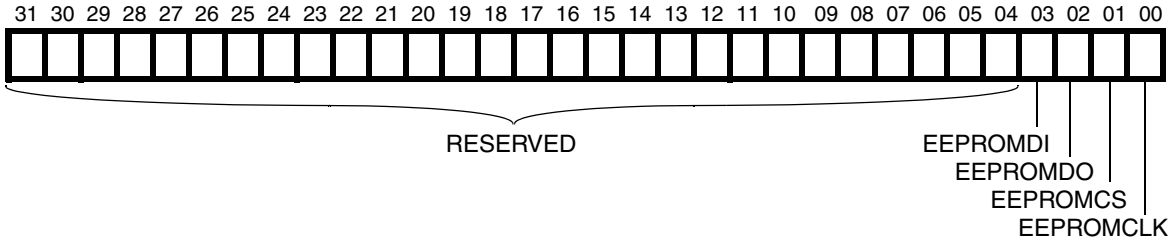

Figure 5-17. EEPROM Access Register

Table 5-27. EEPROM Access Register

Bit	Description
31:04 RESERVED	Reserved. Used for (i)chipSAR+ testing purposes only. Ignore these bits when accessing the EEPROM.
03 EEPROMDI	EEPROM Data In. Reads the real-time state of the EEPROM data output. Read only. See The 93C46 Specification for information on how to program the EEPROM.
02 EEPROMDO	EEPROM Data Out. Writes data to the EEPROM data-in bit. Read/write. See The 93C46 Specification for information on how to program the EEPROM.
01 EEPROMCS	EEPROM Chip Select. Controls the (i)chipSAR+ select line on the EEPROM. Read/write. See The 93C46 Specification for information on how to program the EEPROM.
00 EEPROMCLK	EEPROM Clock. Controls the clock line on the EEPROM. Read/write. See The 93C46 Specification for information on how to program the EEPROM.

Cell FIFO Queue Size Register

Register Name:Cell FIFO Queue Size Register
Function(s):Contains packet memory queue size in cells
Address Offset:2Ch
Reset Init Value:0x00000000
EEPROM Load Init Value:Not initialized
Attribute: Read/write

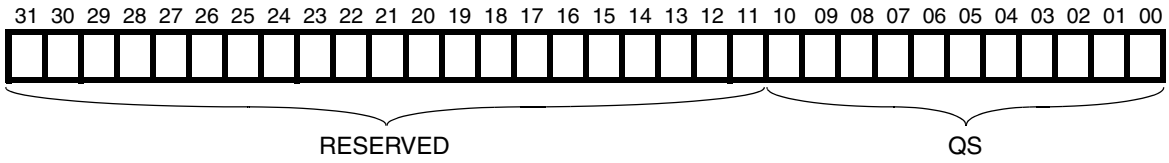

Figure 5-18. Cell FIFO Queue Size Register

Table 5-28. Cell FIFO Queue Size Register

Bit	Description
31:11 RESERVED	Reserved.
10:00 QS(10:00)	11-bit Queue Size Register. The packet memory state machine uses this register to determine how much packet memory to reserve for receives. The Host writes this register with the number of cells in the Cell FIFO as an initialization parameter. The same value should be written to the Cells Available Register. Initializing this register is required only for Cell FIFO usage.

Cell FIFO Mark State Register

Register Name:Cell FIFO Mark State Register
Function(s):Tells the (i)chipSAR+ to save state for valid data interrupt
Address Offset:30h
Reset Init Value:0x00000000
EEPROM Load Init Value:Not initialized
Attribute: Read/write

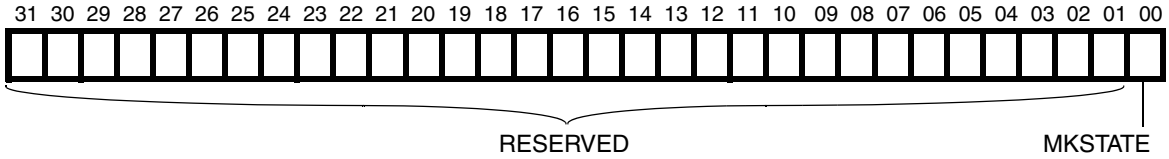

Figure 5-19. Cell FIFO Mark State Register

Table 5-29. Cell FIFO Mark State Register

Bit	Description
31:01 RESERVED	Reserved.
00 MKSTATE	This bit, when set by the host, tells the Mark state machine that a mark operation is requested. The Mark state machine will reset this bit at the same time it sets the mark complete status in the Bus Interface Status Register. The host should wait for the bit to be reset before setting it again. This bit is useful only in Cell FIFO mode.

Cell FIFO Read Pointer Register

Register Name:Cell FIFO Read Pointer Register
Function(s):Contains Cell FIFO pointer information
Address Offset:34h
Reset Init Value:0x00000000
EEPROM Load Init Value:Not initialized
Attribute: Read/write

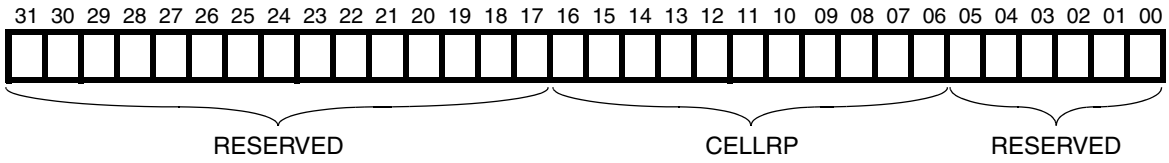

Figure 5-20. Cell FIFO Read Pointer Register

Table 5-30. Cell FIFO Read Pointer Register

Bit	Description
31:17 RESERVED	Reserved.
16:06 CELLRP(16:06)	17-bit Cell FIFO Read Pointer Register. This is the read pointer for the Cell FIFO. This register should be initialized to 0, and is otherwise not accessed. The lower 6 bits are always 0.
05:00 RESERVED	Reserved.

Cell FIFO Write Pointer Register

Register Name:Cell FIFO Write Pointer Register
Function(s):Contains Cell FIFO pointer information
Address Offset:38h
Reset Init Value:0x00000000
EEPROM Load Init Value:Not initialized
Attribute: Read/write

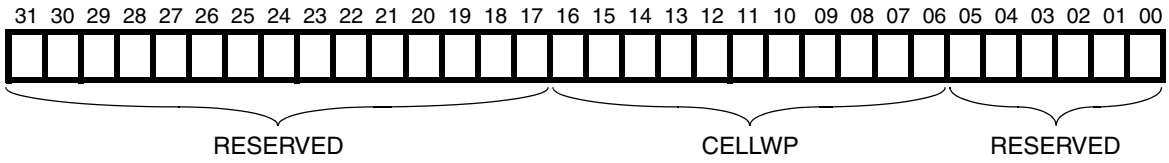

Figure 5-21. Cell FIFO Write Pointer Register

Table 5-31. Cell FIFO Write Pointer Register

Bit	Description
31:17 RESERVED	Reserved.
16:06 CELLWP	17-bit Cell FIFO Write Pointer Register. This is the write pointer for the Cell FIFO. This register should be initialized to 0, and is otherwise not accessed. The lower 6 bits are always 0.
05:00 RESERVED	Reserved.

Cell FIFO Cells Available Register

Register Name:Cell FIFO Cells available Register
Function(s):Contains the number of cell space available in receive side RAM FIFO
Address Offset:3Ch
Reset Init Value:0x00000000
EEPROM Load Init Value:Not initialized
Attribute: Read/write

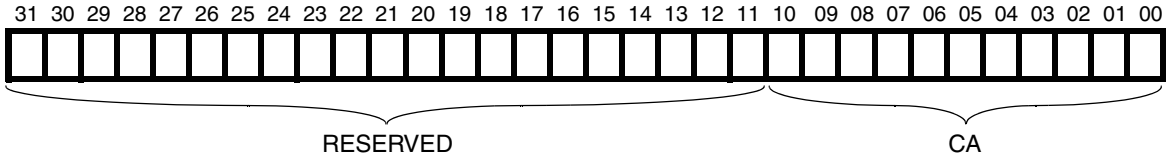

Figure 5-22. Cell FIFO Cells Available Register

Table 5-32. Cell FIFO Cells Available Register

Bit	Description
31:11 RESERVED	Reserved.
10:00 CA(10:00)	11-bit Cells Available Register. This register is used by the Cell FIFO to indicate the number of cell locations available in the Cell FIFO. This register should be initialized by the host to the same value as the Queue Size Register. Otherwise, it is not accessed. This register is functional only for Cell FIFO mode.

Front End Registers/DMA Control

Table 5-33 provides a map of the Front End registers/DMA Control registers that are accessible from PCI memory cycles:

Table 5-33. Front End Registers/DMA Control

Byte 3	Byte 2	Byte 1	Byte 0	Offset
			FE Register 0	000h
			FE Register 1	004h
			FE Register 2	008h
			FE Register 3	00ch
			etc. up to 512	etc.
Transmit Transaction Counter				800h
Receive Transaction Counter				A00h
Transmit List Address Register				C00h
Receive List Address Register				E00h

Front End Registers

Front End registers are specific to the front-end device being used. The table above shows the addressing if the registers are 8 bits wide. If the registers are 16 bits wide, bytes 1 and 0 are used.

Transmit Transaction Counter Register

Register Name:Transmit Transaction Counter Register
Function(s):DLE transmit transaction counter
Address Offset:4800h
Reset Init Value:0x00000000
EEPROM Load Init Value:Not initialized
Attribute: Read—Write to Add

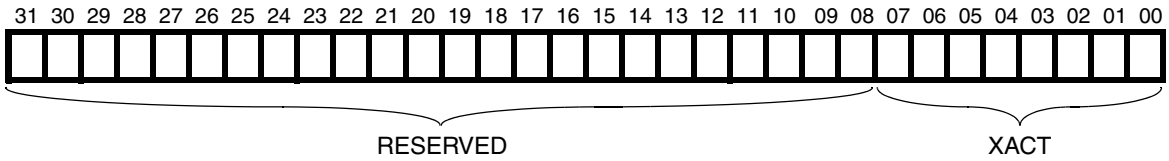

Figure 5-23. Transmit Transaction Counter Register

Table 5-34. Transmit Transaction Counter Register

Bit	Description
31:08 RESERVED	Reserved.
07:00 XACT(07:00)	Transmit Transaction Counter. The current value is added to new value when written to. Written by the host, decremented by the DLE State Machine.

Receive Transaction Counter Register

Register Name:Receive Transaction Counter Register
Function(s):DLE receive transaction counter
Address Offset:4800h
Reset Init Value:0x00000000
EEPROM Load Init Value:Not initialized
Attribute: Read—Write to add

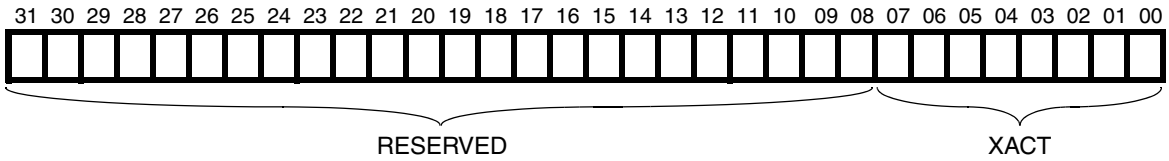

Figure 5-24. Receive Transaction Counter Register

Table 5-35. Receive Transaction Counter Register

Bit	Description
31:08 RESERVED	Reserved.
07:00 XACT(07:00)	Receive Transaction Counter. The current value is added to the new value when written to. Written by host, decremented by DLE state machine. This register is not used if the (i)chipSAR+ is in Cell FIFO Mode (CELLMODE bit in Control Register = 1).

Transmit List Address Register

Register Name:Transmit List Address Register
Function(s):Contains system address where transmit DLE structures are stored
Address Offset:0x4C00
Reset Init Value:Not initialized
EEPROM Load Init Value:Not initialized
Attribute: Read/write

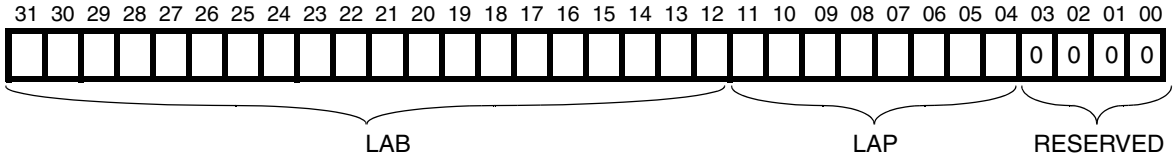

Figure 5-25. Transmit List Address Register

Table 5-36. Transmit List Address Register

Bit	Description
31:12 LAB(19:00)	List Address Base. Upper 20 bits of address where Host Transmit DLEs are stored.
11:04 LAP(07:00)	List Address Pointer. Lower 8 bits of Host Transmit DLE pointer. These bits will increment through the 256 DLE entries.
03:00 RESERVED	Reserved. These bits must = 0.

Receive List Address Register

Register Name:Receive List Address Register
Function(s):Contains system address where receive DLE structures are stored
Address Offset:0x4B00
Reset Init Value:Not initialized
EEPROM Load Init Value:Not initialized
Attribute: Read/write

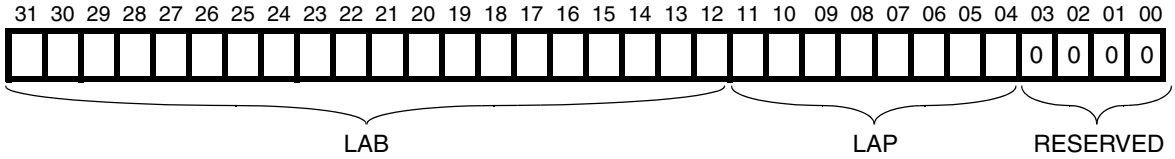

Figure 5-26. Receive List Address Register

Table 5-37. Receive List Address Register

Bit	Description
31:12 LAB(19:00)	List Address Base. Upper 20 bits of address where Host Receive DLEs are stored. This register is not used if the (i)chipSAR+ is in Cell FIFO Mode (CELLMODE bit in Control Register = 1).
11:04 LAP(07:00)	List Address Pointer. Lower 8 bits of Host Receive DLE pointer. These bits will increment through the 256 DLE entries. This register is not used if the (i)chipSAR+ is in Cell FIFO Mode (CELLMODE bit in Control Register = 1).
03:00 RESERVED	These bits must be = 0.

PCI Error Handling

The PCI bus contains two signals called PERR# and SERR#. These signals, in addition to error interrupts, are handled in the following ways.

PERR Handling

The (i)chipSAR+ generates and checks parity. It checks for different parity errors in the following ways:

- Target Data Parity Error

  If the (i)chipSAR+ is accessed with a target write, then the (i)chipSAR+ checks the write data and command/byte enable bits against the PAR bit on the PCI bus. If the (i)chipSAR+ detects a parity error, it sets the PERR bit (bit 31) in the Configuration Command/Status Register. If the PERREN bit (bit 6) is set in the Configuration Command/Status Register, then the PERR line is also activated.

- Master Read Data Parity Error

  If the master detects a parity error when it is reading data, it asserts the PERR bit (bit 31) in the Configuration Command/Status Register. If the PERREN bit (bit 6) is set in the Configuration Command/Status Register, then the PERR line is also activated, and the DATAPERR bit (bit 24) in the Configuration Command/Status Register is set. The (i)chipSAR+ will continue operations as if nothing happened in all other respects.

- Master Write Data Parity Error

  If the master detects that the PERR line is active when it is writing data, indicating that the target asserted a parity error, and if the PERREN bit (bit 6) is set in the Configuration Command/Status Register, the master will set the DATAPERR bit (bit 24) in the Configuration Command/Status Register. The (i)chipSAR+ will continue operations as if nothing happened in all other respects.

SERR Handling

The (i)chipSAR+ can generate SERR. SERR be will activated for the following reasons:

- The (i)chipSAR+ as a Master receives Target Abort.
- The (i)chipSAR+ as a Master receives Master Abort.
- The (i)chipSAR+ detects a Target Address Parity Error. In this case, the PERREN bit in the Configuration Command/Status Register must also be set.

In all of these cases, the SERREN bit (bit 8) in the Configuration Command/Status Register must be set to get a SERR. In addition to the SERR pin being driven, the SERR bit (bit 30) in the Configuration Command/Status Register will be set.

PCI Interrupt Handling

The (i)chipSAR+ always issues an interrupt on the PCI INTA# line. The host processor must determine who is interrupting (could be anyone on PCI bus) and then clear the interrupt. Some systems might have an interrupt controller to help with these tasks, but the (i)chipSAR+ only issues the interrupt and keeps it active until the host clears it.

The (i)chipSAR+ can generate a PCI interrupt from seven different sources. Each source can be individually enabled or disabled at the source. In addition, each source is associated with a mask bit in the Bus Interface Control Register. It is possible to use the mask bits to gate whether the interrupt will cause a PCI INTA# interrupt. Using the mask bits allows polling operations to occur (by reading the interrupt in the Bus Interface Status Register). Also, masking all the interrupt sources is an easy way to disable the (i)chipSAR+ from generating any PCI interrupts.

The (i)chipSAR+ will generate a PCI INTA# if one or more of the following sources is true and its mask bit is also set:

- The reassembly or segmentation side generates an interrupt.
- The front-end device generates an interrupt.
- A DLE receive interrupt occurs.
- A DLE transmit or transmit interrupt occurs.
- A PCI Errors interrupt occurs.

Reassembly and Segmentation Interrupts

The status of reassembly and segmentation interrupts can be read in either the Reassembly Interrupt Status Register or the Segmentation Interrupt Status Register, whichever is applicable. Accesses to the applicable reassembly or segmentation status register clear the interrupt. Individual sources of reassembly or segmentation interrupts can be masked through the applicable reassembly or segmentation interrupt mask register.

Front End Interrupts

The front end interrupt is passed through the (i)chipSAR+ from the front end device. To clear the interrupt, the host must access the front end device and turn it off there.

DLE Interrupts

Each DLE contains an interrupt bit that sends an interrupt at the completion of that DLE. If this bit is not set, then the interrupt is not activated. The Bus Interface Status Register contains two bits that indicate whether the source of the interrupt was the transmit DLE or receive DLE. Writing a 0 to the appropriate bit clears the interrupt. To mask the interrupt, clear the appropriate mask bit in the Bus Interface Control Register.

Mark Interrupts

The Mark Interrupt mechanism notifies the host when data related to Reassembly Packet Complete status is available in host memory for Cell FIFO mode. See *Mark Interrupt State Machine on page 90* for complete information.

PCI Errors Interrupt

The (i)chipSAR+ generates an error interrupt if one or more of the following error bits in the Configuration Command/Status Register is set:

- Bit 29, RMABORT
- Bit 28, RTABORT
- Bit 24, DATAPERR

If one or more of these three bits are set, then the ERRINT bit in the Bus Interface Status Register will be set. If the ERRMASK bit in the Bus Interface Control Register is set, then INTA# will be generated.

In addition, if an RTABORT or RMABORT error occurs, the SERR line, if enabled, will also become active. And if the PERR line is enabled, it will also become active if DATAPERR occurs.

To clear the error interrupt, clear the appropriate bit in the Configuration Command/Status Register.

CAUTION

If the RMABORT or RTABORT error occurs, the (i)chipSAR+ master machine will go into a freeze state until a reset (hard or soft) occurs. These errors should never occur in normal operation. The (i)chipSAR+ target interface will still function, however.

PCI EEPROM

The (i)chipSAR+ uses a serial NOVRAM (93C46) to load its internal configuration and other registers. These registers are first initialized by a reset to their reset init value and then written over with the EEPROM data when the load occurs. The EEPROM is loaded upon a PCI reset (RST#) or upon writing to the External Soft Reset Register. This load takes approximately 2 ms (milliseconds) with a 33 MHz PCI clock (4 ms @16 MHz). This load can be disabled by making the EE_EN pin low. The EEPROM contents can also be read or written from PCI, via the EEPROM Access Register. The EEPROM must be initialized with data at the factory before the board is shipped to a customer.

The EEPROM is organized as 64 16-bit locations. The most significant bit is read or written first. See the 93C46 data sheet for more information.

Bit 1 of the PCI Expansion ROM Base Register Low controls whether the Expansion ROM exists or not. If this bit is set to 0, Expansion ROM is not present, and the Expansion Base Register will not be writable from the PCI bus. If this bit is set to 1, Expansion ROM is present, and the base register is writable from the PCI configuration space.

Not all bits are loaded into the register specified. See the register descriptions for details of what is actually loaded.

The (i)chipSAR+ does not use address locations 24h through 7fh.

Table 5-38 provides a memory map of the (i)chipSAR+ EEPROM registers:

Table 5-38. (i)chipSAR+ EEPROM Memory Map

EEPROM WORD 15.....0	Address
PCI Device ID Register	00h
PCI Vendor ID Register	02h
PCI Status Register	04h
PCI Command Register	06h
PCI Class Register	08h
PCI Prog I/F and Revision Register	0ah
PCI Header Type Register	0ch
PCI Latency Register	0eh
PCI Memory Base Address Register (31–16)	10h
PCI Memory Base Address Register (15–0)	12h
PCI Expansion ROM Base Address Register (31–16)	18
PCI Expansion ROM Base Address Register (15–0)	1a
PCI Interrupt Register (31–16)	1c
PCI Interrupt Register (15–0)	1e
Bus Interface Status Register (31–16)	20
Bus Interface Status Register (15–0)	22
MAC Address Register 1 (31–16)	24
MAC Address Register 1 (15–0)	26
Reserved	28
MAC Address Register 2 (15–0)	2a
Reserved	27h - 7fh

PMC BUSMODE Signals

The (i)chipSAR+ can be used as the basis for a PMC card. The (i)chipSAR+ handles the four BUSMODE pins specified in the CMC/PMC specification. The (i)chipSAR+ monitors BUSMODE[4:2] and drives BUSMODE[1].

When BUSMODE[4:2] = 000, the system is requesting a *card present* test. The (i)chipSAR+ responds by driving BUSMODE[1] to 0 while tri-stating all of the PCI outputs.

When BUSMODE[4:2] = 001, the system is expecting the card to be PCI compatible. The (i)chipSAR+ responds by driving BUSMODE[1] to 0 and begins normal operation of its PCI signals.

When BUSMODE[4:2] are any other combination, the (i)chipSAR+ responds by driving BUSMODE[1] to 1 and tri-stating all PCI outputs.

Local Buffer

The Local Buffer is designed for a single bank of four 32Kx8 or one or two banks of 128Kx8 SRAMS. Using a 33-MHz clock, this buffer will have enough bandwidth to allow full speed transmits and receives at a 155-Mbit rate. The Local Buffer allows for either on-board packet reassembly or reassembly in host memory using the Cell FIFO. The Cell FIFO buffers cells in the Local Buffer until they can be transferred automatically by the PCI state machine.

Cell FIFO

The Cell FIFO allows the (i)chipSAR+ to be programmed as though packets are assembled in system memory, while providing the same the bus latency tolerance that is provided by using an intermediate buffer. The Cell FIFO is designed to require less buffer memory to achieve the same packet reassembly performance that is achieved using local memory. The Cell FIFO is a simple circular queue, with a mechanism to provide for data consistency in system memory.

The Cell FIFO is implemented as a fixed-sized circular queue. The Cell FIFO is empty when the head pointer equals the tail pointer. The Cell FIFO is considered full when less room is remaining than is required to transfer the largest burst the (i)chipSAR+ is capable of, plus 8 bytes for the system address and transfer count.

The following registers are required to provide this control:

- Cell Write Pointer
- Cell Read Pointer
- Cells Available Register

As data is stored in the queue, the system address and length of the data must also be stored. This information is stored at the beginning of the block.

Cell Write Pointer

The Cell Write Pointer points to the next location in the Cell FIFO for receive data.

The Cell Write Pointer is always either equal to or ahead of the Cell Read Pointer. This position does not determine available room in the Cell FIFO. Instead, a separate register called the Cells Available Register keeps track of the room available in the FIFO.

Cell Read Pointer

The Cell Read Pointer points to the oldest data in the Cell FIFO. The CELL FIFO has data available any time the tail pointer is not equal to the head pointer. The tail pointer always points to the next data location.

Cells Available Register

The Cells Available Register is initially loaded with the number of cells to allocate for the Cell FIFO. It is decremented when data is written to the Cell FIFO and incremented as data is read from the Cell FIFO. The FIFO is full anytime the Cells Available Register is equal to 0.

Cell FIFO Initialization

The Cell FIFO requires initialization before being enabled by software. The driver must set the Cells Available Register to the number of cells to be used by the Cell FIFO. After these values have been set, the Cell FIFO is enabled by setting CELLMODE in the (i)chipSAR+ Control Register to 1.

If the CELLMODE bit is not set, the (i)chipSAR+ will use receive addresses as pointers into the local memory, and the data will not be transferred into system memory.

Mark Interrupt State Machine

The (i)chipSAR+ will generate status indicating the reception of packets as data is transferred into the Cell FIFO. The system software can respond to this status before the data has been written to system memory from the Cell FIFO. This mechanism provides a way for the system software to know when the data has been transferred.

When the system responds to the (i)chipSAR+ data available status, the data can be in any of four places:

- System memory
- PCI Receive FIFO
- Side RAM
- (i)chipSAR+ Receive FIFO

There is no way of knowing where this data is, and it isn't practical to stop all transfers until the Cell FIFO is empty. The (i)chipSAR+ allows the software to mark the current state and enables notification of when the CELL FIFO transfers the existing data. The transferred data is guaranteed to be all of the data indicated by the interrupt status; it might be more data than was indicated from the status.

The Mark Interrupt state machine controls this function. Note that only one mark at a time is valid. A mark set while a earlier mark is in progress will be ignored and will probably cause data coherency problems. Therefore a mark should not be set while another mark is in progress.

Valid Data Interrupt State Machine

The Valid Data Interrupt state machine consists of four states:

- Mark state
- (i)chipSAR+ Receive FIFO state
- Side RAM state
- PCI Receive FIFO state

Figure 5-27 illustrates these states.

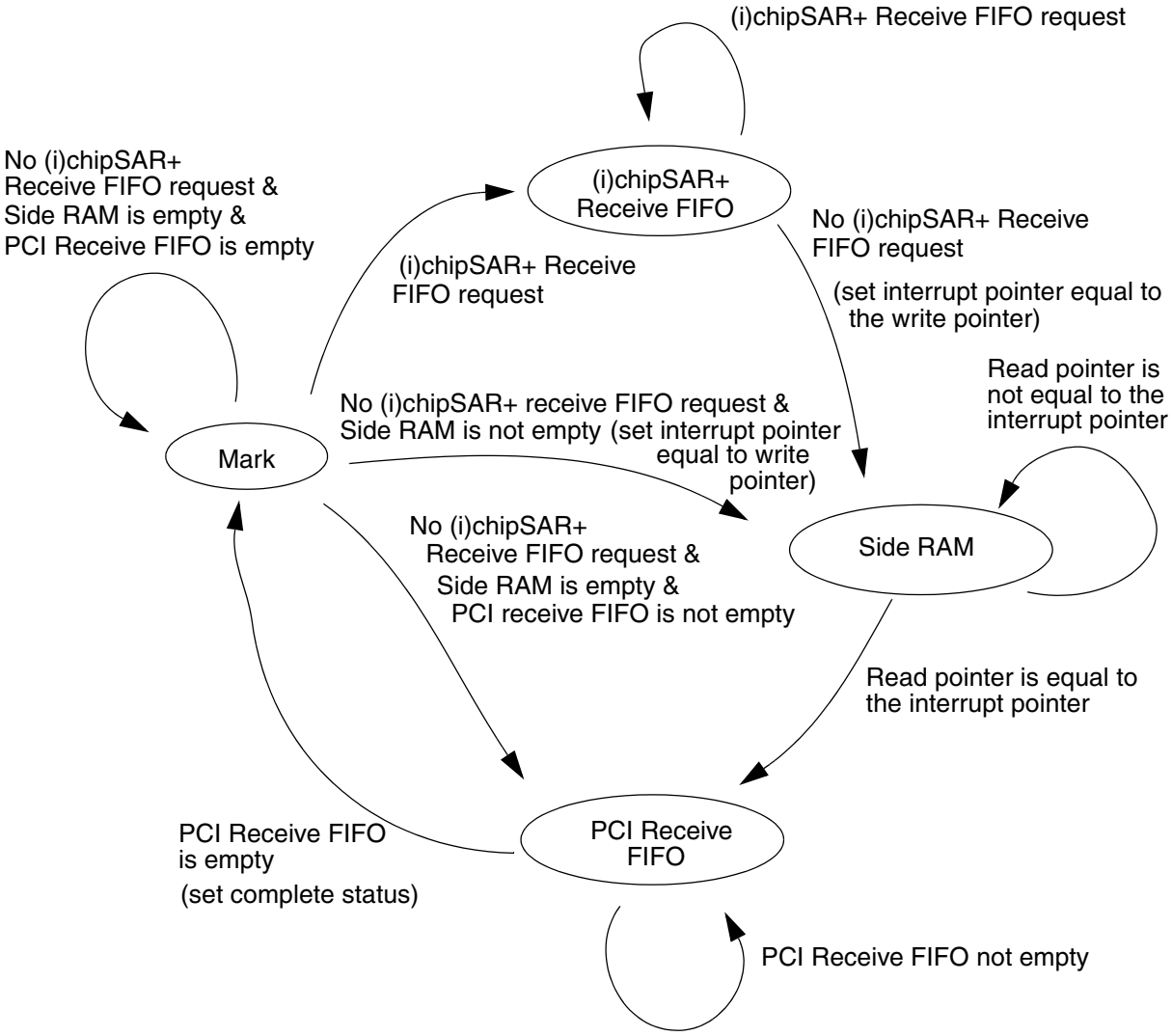

Figure 5-27. Valid Data Interrupt State Diagram

Mark State

The Mark state waits for an indication from the PCI Slave state machine that a mark operation is requested. It performs the process shown in the following decision tree when it receives that indication. Note that the system can perform the mark operation only while the state machine is in the mark state. Also, marking the FIFO while a mark is in progress can cause data coherency problems.

```
IF(   No ATLANTIC receive FIFO request &&
      The Side RAM is empty &&
      The PCI receive FIFO is empty )
{
      Set the Valid Data interrupt status;
      The next state is Mark;
}
ELSE  IF(   ATLANTIC Receive FIFO request )
      {
            The next state is ATLANTIC receive FIFO;
      }
      ELSE  IF(   Side RAM is not empty )
            {
                  Set the interrupt pointer equal to
                  the write pointer;
                  The next state is Local Buffer;
            }
            ELSE  IF(   PCI Receive FIFO is not empty )
                  {
                        The next state is PCI Receive FIFO;
                  }
```

The mark state might set the interrupt status directly if the Cell FIFO is empty, or it might branch to other states, depending on the state of the Cell FIFO.

(i)chipSAR+ Receive FIFO State

The (i)chipSAR+ Receive FIFO state indicates that a cell is still in the (i)chipSAR+ Receive FIFO before being written to the Side RAM. The Receive FIFO state waits for the Cell to be transferred to the Side RAM, and then sets the interrupt pointer to equal the Side RAM write pointer. The state machine then transfers to the Side RAM state.

Side RAM State

The Side RAM state waits for the Side RAM read pointer to equal the interrupt pointer. Note that further received cells might have incremented the write pointer before this happens. This is the reason the interrupt pointer is required. When the Side RAM read pointer is equal to the interrupt pointer, the state machine transfers to the PCI Receive FIFO state.

PCI Receive FIFO State

The PCI Receive FIFO state waits for the PCI Receive FIFO to become empty. After the PCI Receive FIFO becomes empty, the data since the last mark is guaranteed to be in system memory. The state machine sets the Valid Data Interrupt status and transfers to the Mark state.

Segmentation Engine Software Reference 6

Overview

This chapter provides information about the Segmentation Engine Control Memory structures and Segmentation Engine registers of the (i)chipSAR+.

Segmentation Engine Control Memory Structures

The Segmentation Engine Control Memory contains data structures that are used by both software and the (i)chipSAR+ to maintain dynamic information. Segmentation Engine Control Memory is also used for communication between the (i)chipSAR+ and the software entity. The chip's Segmentation and Reassembly Engines each have a separate control memory.

The Segmentation Engine Control Memory holds the following entities:

- Segmentation Buffer Descriptor table
- Segmentation Packet Ready queue
- Transmit Complete queue
- Segmentation VC table entries
- All of the scheduling control structures

Each of these entities is described in the following sections.

Segmentation Buffer Descriptor Table

The Segmentation Buffer Descriptor Table is a set of 32-byte entries. Each entry contains parameters for segmenting packets and identifies the location of the packet in the packet memory. When a packet is set up for segmentation, an available Segmentation Buffer Descriptor Table entry is initialized with the packet parameters, as shown in *Figure 6-1 on page 96*.

The most significant bit of the Segmentation Control Memory base address of the Segmentation Buffer Descriptor Table in memory is stored in the Descriptor Base Address Register (DESC_BASE). Each Segmentation Buffer Descriptor Table entry is referenced by its descriptor number, which can range from 1 to 8191. Descriptor number 0 is used internally by the (i)chipSAR+, and must not be used. The location of the entry is determined by concatenating the descriptor number with the most significant bit of the Segmentation Buffer Descriptor base address (as described in *Descriptor Table Base Address Register on page 132*).

Figure 6-1 shows the organization of each Segmentation Descriptor table entry, including the type of accesses by the software entity and the (i)chipSAR+ and the address of the individual entries in the table. Note that the addresses CH to 1E are used by the (i)chipSAR+ and must not be used for another purpose by software.

Address	15	0	SW	(i)chipSAR+
0H	Descriptor Mode Bits		W	R/W
2H	0 0 0	Segmentation VC Table Index	W	R
4H	Reserved			
6H	Packet Byte Count		W	R/W
8H	Packet Memory Start Address (High)		W	R
AH	Packet Memory Start Address (Low)		W	R
CH–1EH	Reserved		R	R/W

Figure 6-1. Segmentation Descriptor Table Entry Organization

The following sections describe Segmentation Descriptor Table entries.

Descriptor Mode Word Bits

Figure 6-2 shows the organization of the descriptor mode bit positions.

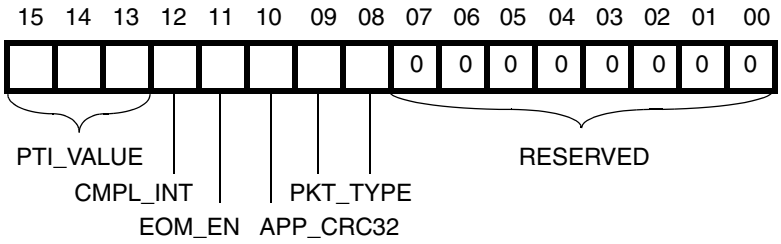

Figure 6-2. Descriptor Mode Bits

Table 6-1 describes the Descriptor Mode bits.

Table 6-1. Descriptor Mode Bits

Bit	Description
15:13 PTI_VALUE	PTI_VALUE. These bits are the substitute PTI_VALUE for the ATM cell header when the pkt_type field indicates that an OAM cell type is to be transmitted with the PTI values from the Descriptor Table (PKT_TYPE = 00 or 11).
12 CMPL_INT	Completion Interrupt. This bit, when set to 1, indicates that the (i)chipSAR+ must generate a maskable interrupt after the frame associated with this descriptor has been completely segmented and delinked.
11 EOM_EN	End of Message Enable. This bit, when set to 1, indicates that the (i)chipSAR+ must set the EOM bit (End of Message bit) in the ATM header by forcing the PTI field to xx1 for the last cell of a packet (required for AAL5 operation).

Table 6-1. Descriptor Mode Bits (cont)

Bit	Description
10 APP_CRC32	Append 32-bit CRC. This bit, when set to 1, indicates that the (i)chipSAR+ must generate and include a 32-bit frame check sequence (FCS) for the frame associated with this descriptor. Note that a bit can alternately be set in the Segmentation VC Table for similar effect. This bit provides FCS control on a per frame basis, whereas the bit in the Segmentation VC Table provides FCS control on a per connection basis. Note that this bit and the corresponding bit in the Segmentation VC Table must both be reset for this packet to not have the CRC32 appended.
09:08 PKT_TYPE	Packet Type. The settings of these two bits indicate the segmentation algorithm desired: 00 – No CRC-10 is appended to the cell. PTI values from the Descriptor Table are used. 01 – No CRC-10 is appended to the cell. PTI values from the Segmentation VC Table are used (normal AAL5 mode or null AAL mode). 10 – CRC-10 is appended to each cell. PTI values from the Segmentation VC Table are used. 11 – CRC-10 is appended to each cell. PTI values from the Descriptor Table are used. (OAM cell mode).
07:00 RESERVED SET TO 0	These bits MUST be initialized to 0 by software before passing descriptors to the (i)chipSAR+. The (i)chipSAR+ uses this area to store temporary variables; unpredictable results will occur if these are not initialized to 0.

Segmentation VC Table Index

The Segmentation VC Table Index field contains the identifier of the virtual circuit to which the packet associated with the descriptor belongs. This 13-bit field is used as a pointer to the Segmentation VC Table entry corresponding to the virtual circuit. The upper 3 bits of the 16-bit word must be set to 0.

Reserved (Descriptor Table Address 4H)

The (i)chipSAR+ ignores this 16-bit reserved field.

Packet Byte Count

The Packet Byte Count field is programmed with the length in bytes of the packet queued for segmentation. The packet byte count must be a multiple of four bytes (that is, the lower two bits must be 0). The (i)chipSAR+ decrements this field down to 0 during the segmentation process.

Packet Memory Start Address

The packet starting byte address, relative to the (i)chipSAR+, is generated by concatenating the high and low fields of the packet memory start address. The contents of the Packet Memory Start Address field must be initialized to the byte-address of the beginning of the packet buffer in the local Packet/Cell Memory. This address must also be 32-bit aligned.

Reserved (Descriptor Table Address CH–1EH)

The (i)chipSAR+ uses these reserved fields for internal purposes.

Segmentation Packet Ready Queue

The Segmentation Packet Ready Queue queues the descriptors of packets for segmentation. The Segmentation Packet Ready Queue is defined by the following four registers in the Segmentation Engine of the (i)chipSAR+:

- PRQ_ST_ADR (Packet Ready Queue Starting Address)
- PRQ_ED_ADR (Packet Ready Queue Ending Address)
- PRQ_RD_PTR (Packet Ready Queue Read Pointer)
- PRQ_WR_PTR (Packet Ready Queue Write Pointer)

Figure 6-3 shows the contents of each valid entry in the queue.

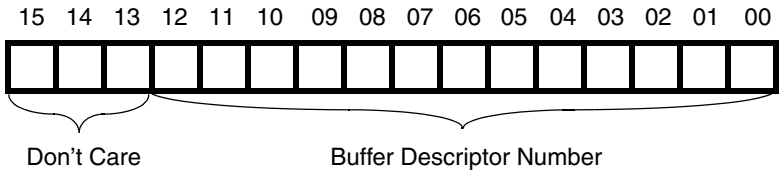

Figure 6-3. Segmentation Packet Ready Queue Entry

When this queue is non-empty, the Segmentation Engine reads the descriptor number from the queue and enters the associated connection appropriately into the scheduling tables.

Figure 6-4 shows the operational states of the Segmentation Packet Ready Queue.

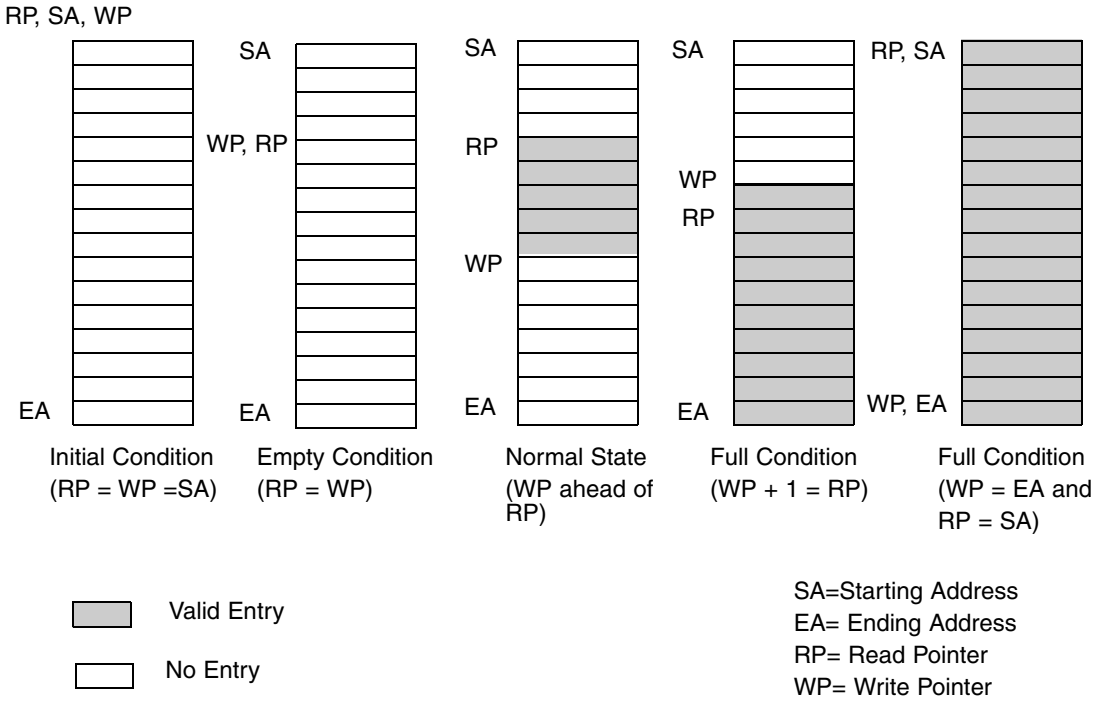

Figure 6-4. Communication Queue Operational States

The starting and ending addresses, when concatenated with the contents of the QUEUE_BASE Register, define the location of the queue in the control memory.

Figure 6-5 shows the address generation.

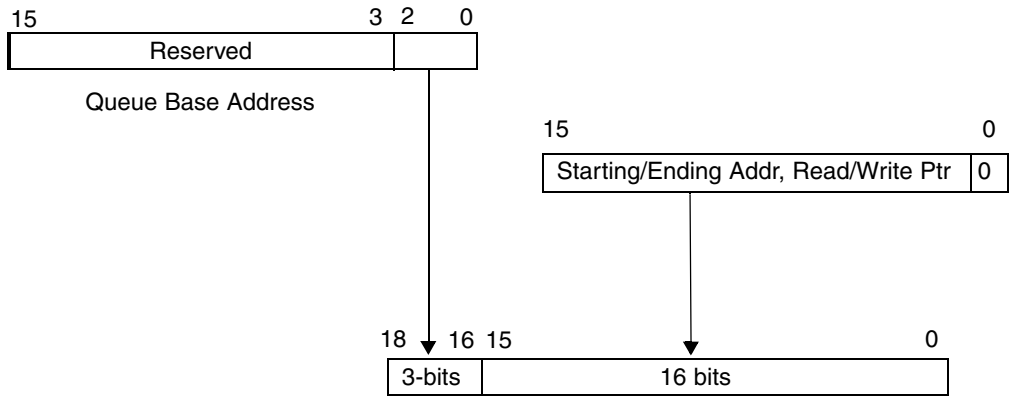

Figure 6-5. Address Generation of Queue Entry

The sequence of operations to load the Segmentation Packet Ready Queue is as follows:

1. Check that the Segmentation Packet Ready Queue is defined large enough to hold all of the possible descriptors + 1.
2. Read the Segmentation Packet Ready Queue write pointer from the (i)chipSAR+ (PRQ_WR_PTR).
3. Store the descriptor number of the buffer descriptor for segmentation into the queue location in the control memory pointed to by the write pointer.
4. Increment the write pointer to point to the next location in the queue, while taking care of the wrap-around condition. Multiple descriptors can be added to the Segmentation Packet Ready Queue by repeating Steps 3 and 4. After all the descriptors have been written into the Segmentation Packet Ready Queue, go to Step 5.
5. Store the incremented write pointer to the PRQ_WR_PTR to update the (i)chipSAR+ and trigger the segmentation process on these descriptors.

Transmit Complete Queue

The (i)chipSAR+ uses the Transmit Complete Queue to return the descriptor numbers of segmented packets. When this queue transitions from an empty to a non-empty state, the (i)chipSAR+ generates a maskable interrupt to the host. The descriptors are read from this queue to use for subsequent packet segmentation.

Like the Segmentation Packet Ready Queue, the Transmit Complete Queue is defined by four registers in the (i)chipSAR+:

- TCQ_ST_ADR (Transmit Complete Queue Starting Address)
- TCQ_ED_ADR (Transmit Complete Queue Ending Address)
- TCQ_RD_PTR (Transmit Complete Queue Read Pointer)
- TCQ_WR_PTR (Transmit Complete Queue Write Pointer)

Figure 6-6 shows the format of entries in this queue.

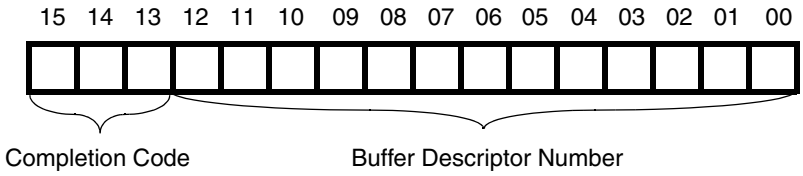

Figure 6-6. Transmit Complete Queue Entry

Figure 6-4 on page 99 shows the operational states of the Transmit Complete Queue, and *Figure 6-5 on page 100* shows the queue entry address generation.

Table 6-2 shows the completion codes for the Transmit Complete Queue.

Table 6-2. Completion Codes for Transmit Complete Queue

Completion Code	Description
000	Normal Completion
Others	Reserved

The following sequence of operations unloads the Transmit Complete Queue:

1. Check that the Transmit Complete Queue is not empty.
2. Read the Transmit Complete Queue read and write pointers from the (i)chipSAR+ (TCQ_RD_PTR and TCQ_WR_PTR respectively).
3. If the read pointer is not equal to the write pointer, read the descriptor number of the segmented packet from the queue location in the control memory pointed to by the read pointer. Otherwise skip to Step 5.
4. Increment the read pointer to point to the next location in the queue (take care of the wrap-around condition). Return to Step 3.

5. Store the incremented read pointer to the TCQ_RD_PTR Register in the (i)chipSAR+.
6. Read the Transmit Complete Queue write pointer from the (i)chipSAR+ (TCQ_WR_PTR). If the write pointer is not equal to the read pointer, that is, not empty, return to Step 3.

Segmentation VC Table Entries

The Segmentation VC Table contains entries for each active VC. (An active VC is defined as one that has completed the signalling process.) As part of the signalling process, the host chooses an unused VC Table Entry location (unused VC index), and performs slave writes to initialize the VC parameters in that VC Table Entry.

Because each service class of VC has different field definitions, there are three versions of VC tables.

- ABR VC Tables
- UBR VC Table
- CBR VC Table

For each service class, two tables hold Segmentation VC entries:

- The Main VC Table is composed of 32-byte VC Table entries.
- The Extended VC Table is composed of 8-byte Extended VC Table entries.

ABR VC Table Entries

Figure 6-7 represents the main VC Table entries for ABR.

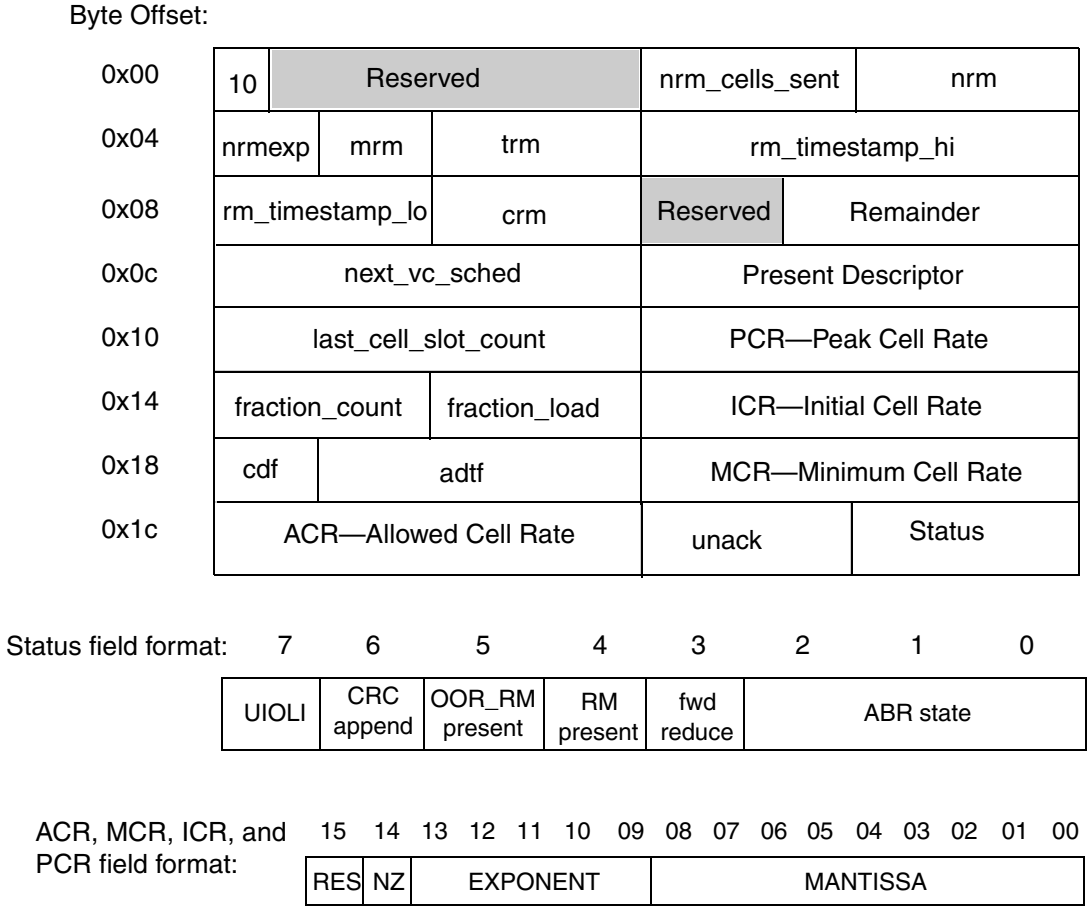

Figure 6-7. Main VC Table Entries For ABR

The ACR, MCR, ICR, and PCR fields are registers containing the 16-bit floating point format of the VC rates in the (i)chipSAR+. The register format is standardized according to the ATM Forum Traffic Management 4.0 Specification to represent the rates in terms of the exponent and mantissa together. The formula for each rate is:

$$1.\text{mantissa} \times 2^{\text{exponent}}$$

Figure 6-8 represents the extended VC Table entries for ABR.

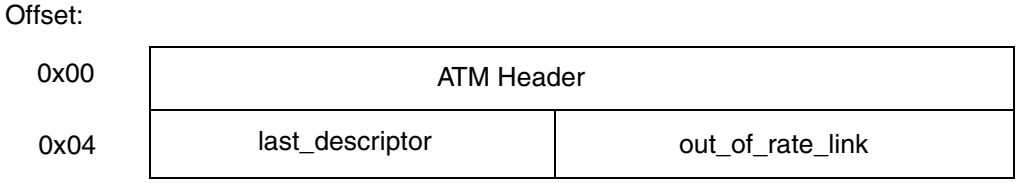

Figure 6-8. Extended VC Table Entry for ABR

Note that the host writes the VC Table entries only when the VC is set up. During normal VC operation, these entries do not need to be, and should not be, written by the host.

The following table describes the non-reserved VC Table entry fields for ABR.

Table 6-3. Non-reserved VC Table Entry Fields for ABR

Field	Description
nrm_cells_sent	Hardware count of number of in-rate cells since and including the last forward RM cell. Software initializes to nrm at VC setup.
nrm	Number of in-rate cells between and including in-rate forward RM cells. Written by software at VC setup. Range: 2–256 in powers of 2
nrmexp	Exponent of the power of 2 of nrm. Written by software at VC setup. Range: 3–7.
mrm	Minimum number of cells between, but not including, forward RM cells. Must be set to 3 (corresponds to an mrm of 2 in ATM Forum Traffic Management 4.0). Written by software at VC setup.
trm	Upper bound on the time between forward RM cells for an active source. Units of 1.024 milliseconds. Written by software at VC setup. Range: 1–255
rm _timestamp_hi/ rm_timestamp_lo	24-bit hardware timestamp for the last forward RM cell for this VC. Software must set this field to all 0's at VC setup.
crm	Maximum number of forward RM cells that can be sent without having received a backward RM cell. Written by software at VC setup. Range: 1–255
Remainder	The 10-bit Remainder field is maintained by hardware. Software should set this field to all 0's at VC setup.
next_vc_sched	The next_vc_sched field is maintained by hardware. It is used as a link for the Scheduling Table and waiting queue. Software should set this field to all 0's at VC setup.
Present Descriptor	The Present Descriptor field is maintained by hardware. It indicates the buffer descriptor number currently being segmented for this VC. Software should set this field to 0x0000 at VC setup.

Table 6-3. Non-reserved VC Table Entry Fields for ABR (cont)

Field	Description
last_cell_slot_count	The last_cell_slot_count field is maintained by hardware. It indicates the cell slot number of the last cell sent. It is used for schedule restart purposes. Software should set this field to 0x0000 at VC setup.
PCR	The Peak Cell Rate (PCR) is the maximum allowed rate for this VC. Software should set this number according to the floating point rate format at VC setup.
fraction_count	Used by hardware for extremely low rate cell scheduling. Software should set this field to all 0's at VC setup.
fraction_load	Used by hardware for extremely low rate cell scheduling. Software should set this field to all 0's at VC setup.
ICR	The Initial Cell Rate (ICR) is the initial rate for this VC. Software should set this number according to the floating point rate format at VC setup. Range: PCR–0.
cdf	Exponent for the power of 2 reduction of ACR associated with crm. Written by software at VC setup. Range: from 0–7 (1–1/128)
atdf	Time permitted between in-rate forward RM cells before rate is reduced to ICR. Units of 8.192 milliseconds. Range: 1–2047
MCR	The Minimum Cell Rate (MCR) is the minimum guaranteed rate for this VC. Software should set this number according to the floating point rate format at VC setup. Range: PCR–0
ACR	The Allowed Cell Rate (ACR) is the dynamic present allowed rate for this VC. Software should initialize this number to ICR according to the floating point rate format at VC setup. Range: PCR–0
unack	Updated by hardware to show the present number of unacknowledged in-rate forward RM cells. Software should initialize this field to 0.
Status	UIOLI—Enables the optional use-it-or-lose-it ABR behavior. 1 = enabled; 0 = disabled CRC_append—If set, CRC-32 is automatically calculated and appended to the end of the packet. Written by software at VC setup. OOR_Present—Hardware bit to signify that an OOR Backward RM cell is queued. Software should initialize to 0 at VC setup. RM_Present—Hardware bit to signify that an In-Rate Backward RM cell is queued. Software should initialize to 0 at VC setup. fwd_reduce—Under the optional use-it-or-lose-it behavior, hardware bit to signify that the last in-rate forward RM cell triggered a rate reduction. Software should initialize to 0 at VC setup. ABR State—3-bit field used by hardware to signify behavior state. Software must set this field to 010 at VC setup.

Table 6-3. Non-reserved VC Table Entry Fields for ABR (cont)

Field	Description
ATM Header	This 4-byte field is the ATM header for this VC. Software sets this field at VC setup.
Last Descriptor	The Last Descriptor field is maintained by hardware. This field indicates the last buffer descriptor number in a list of buffer descriptor numbers that are awaiting segmentation on this VC. Software should set this field to 0x0000 at VC setup.
out_of_rate_link	The out_of_rate_link field is used by hardware to link together multiple OOR Backward RM cells from multiple VCs. Software should set this field to 0's at VC setup.

UBR VC Table Entries

Figure 6-9 represents the main VC Table entries for UBR.

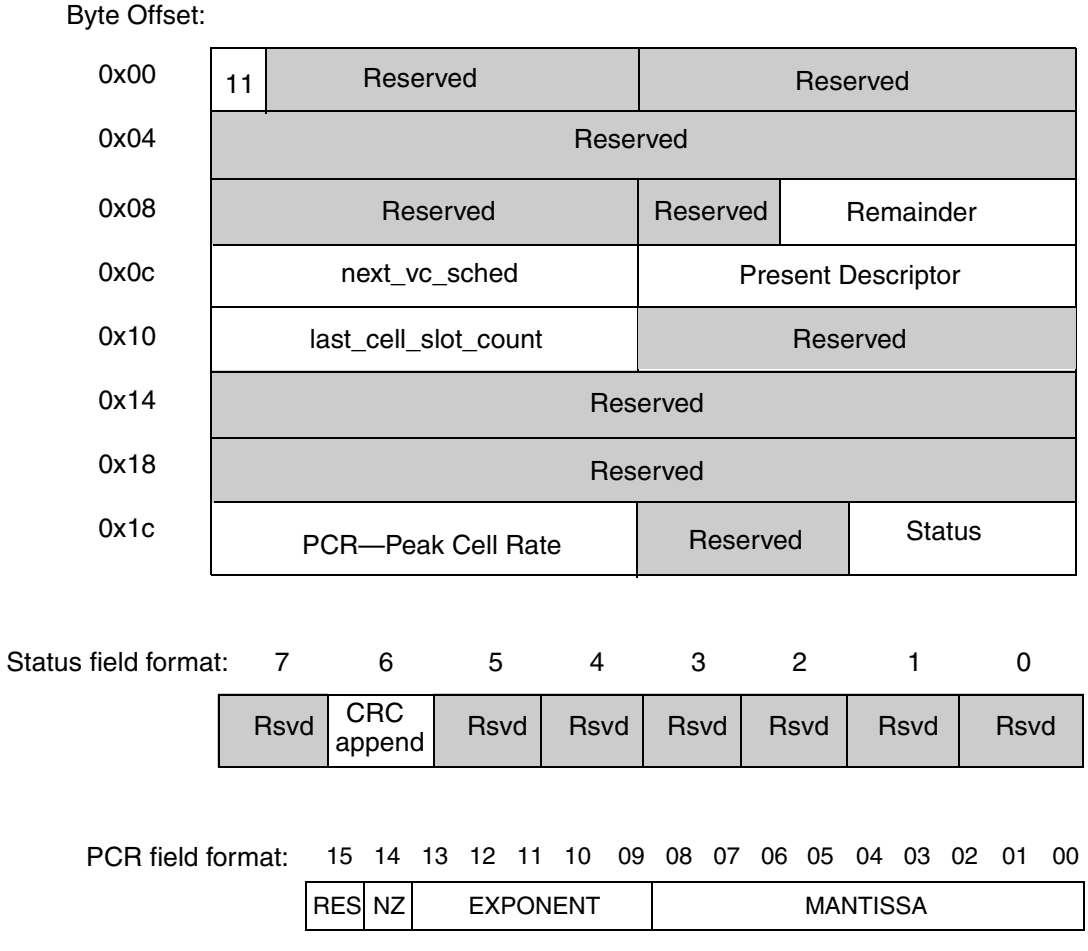

Figure 6-9. Main VC Table Entry For UBR

The PCR field is a register containing the 16-bit floating point format of the peak cell rate for a VC in the (i)chipSAR+. The register format is standardized according to the ATM Forum Traffic Management 4.0 Specification to represent the rate in terms of the exponent and mantissa together. The formula for the rate is:

$$1.\text{mantissa} \times 2^{\text{exponent}}$$

Figure 6-10 represents the extended VC Table entries for UBR.

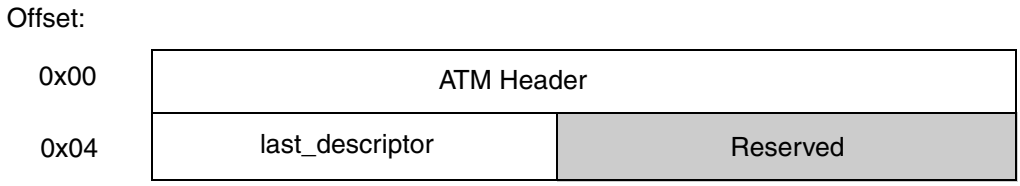

Figure 6-10. Extended VC Table Entry for UBR

Note that the host writes the VC Table entries only when the VC is set up. During normal VC operation, these entries do not need to be, and should not be, written by the host.

The following table describes the non-reserved VC Table entry fields for UBR.

Table 6-4. Non-reserved VC Table Entry Fields for UBR

Field	Description
Remainder	The 10-bit Remainder field is maintained by hardware. Software should set this field to all 0's at VC setup.
next_vc_sched	The next_vc_sched field is maintained by hardware. It is used as a link for the Scheduling Table and waiting queue. Software should set this field to all 0's at VC setup.
Present Descriptor	The Present Descriptor field is maintained by hardware. It indicates the buffer descriptor number currently being segmented for this VC. Software should set this field to 0x0000 at VC setup.
last_cell_slot_count	The last_cell_slot_count field is maintained by hardware. It indicates the cell slot number of the last cell sent. It is used for schedule restart purposes. Software should set this field to 0x0000 at VC setup.
PCR	The Peak Cell Rate (PCR) is the allowed rate for this VC. Software should set this number according to the floating point rate format at VC setup.
Status	Only 1 bit of the Status field is used: CRC_append—If set, CRC-32 is automatically calculated and appended to the end of the packet. This bit is written by software whenever the VC is set up.
ATM Header	This 4-byte field is the ATM header for this VC. Software sets this field at VC setup.
Last Descriptor	The Last Descriptor field is maintained by hardware. This field indicates the last buffer descriptor number in a list of buffer descriptor numbers that are awaiting segmentation on this VC. Software should set this field to 0x0000 at VC setup.

CBR VC Table Entries

Figure 6-11 represents the main VC Table entries for CBR.

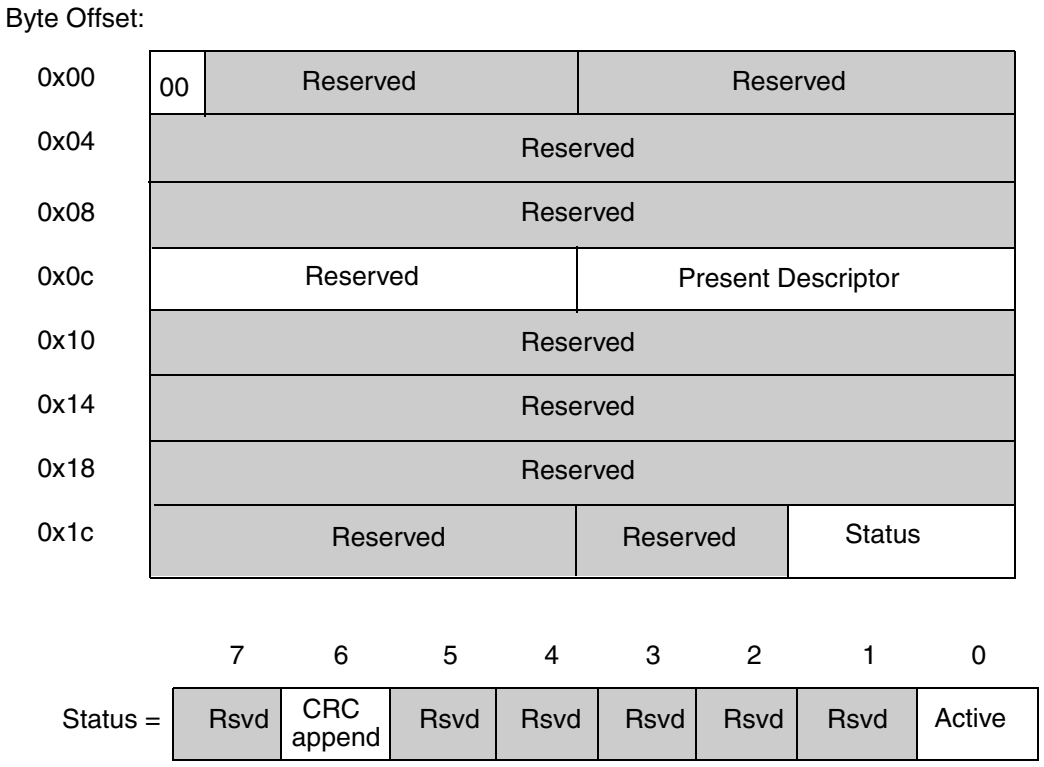

Figure 6-11. Main VC Table Entry For CBR

Figure 6-12 represents the extended VC Table entries for CBR.

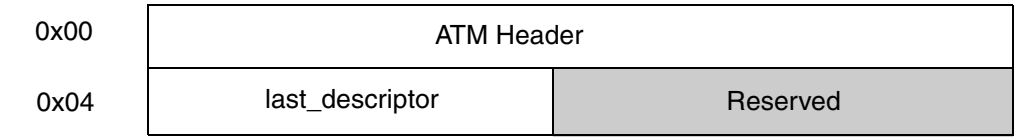

Figure 6-12. Extended VC Table Entry for CBR

Table 6-5 describes the non-reserved VC Table entry fields for CBR.

Table 6-5. Non-reserved VC Table Entry Fields for CBR

Field	Description
Type Field (Byte Offset 0x00)	Software must set the first two bits of the 0x00 offset word to 00 at VC setup. This indicates to hardware that this VC Table entry represents a CBR VC.

Table 6-5. Non-reserved VC Table Entry Fields for CBR (cont)

Field	Description
Present Descriptor	The Present Descriptor field is maintained by hardware. This field indicates the buffer descriptor number currently being segmented for this VC. Software should set this field to 0x0000 at VC setup.
Status	Two bits of the Status field are used: CRC_append—If set, CRC-32 is automatically calculated and appended to the end of the packet. This bit is written by software whenever the VC is setup. Active—This bit is set and reset by hardware. When set to 1, the bit indicates that the VC has a packet segmenting. When the VC is set up, software should set this bit to 0.
ATM Header	This 4-byte field is the ATM header for this VC. Software sets this field at VC setup.
Last Descriptor	The Last Descriptor field is maintained by hardware. It indicates the last buffer descriptor number in a list of buffer descriptor numbers that are awaiting segmentation on this VC. Software should set this field to 0x0000 at VC setup.

Scheduling Control Structures

The three traffic classes handled by the (i)chipSAR+, that is, ABR, UBR, and CBR, have separate scheduling control structures.

ABR Scheduling Structures

Two scheduling structures are associated with the ABR traffic class:

- ABR Scheduling Table
- ABR Wait Queue

ABR Scheduling Table

Hardware uses the ABR Scheduling Table as the appointment book for ABR VC scheduling. The size of the ABR Scheduling Table does not depend upon the lowest rate supported by the ABR service implementation. The (i)chipSAR+ ABR rates can always go down to 0.

However, the size of the ABR Scheduling Table does affect scheduling efficiency for large numbers of simultaneous VCs. *Table 6-6* shows the recommended sizes of the ABR Scheduling Table with different numbers of expected simultaneous VC segmentations.

Table 6-6. Recommended ABR Scheduling Table Size for Simultaneous VCs

Number of Expected Simultaneous VCs	ABR Scheduling Table Size
1K	1K entries (2K bytes)

Table 6-6. Recommended ABR Scheduling Table Size for Simultaneous VCs

Number of Expected Simultaneous VCs	ABR Scheduling Table Size
2K	2K entries (4K bytes)
4K	4K entries (8K bytes)

The Traffic Management Parameters (STPARMS) register settings control the Scheduling Table size for both the ABR Scheduling Table and the UBR Scheduling Table. Note that both scheduling tables will be the same size.

The ABR Scheduling Table Base Register (ABR_SBPTR_BASE) register settings control the starting location of the ABR Scheduling Table.

The host is responsible for initializing all bytes of the ABR Scheduling Table to 0 at (i)chipSAR+ initialization and before the (i)chipSAR+ is enabled. Hardware maintains the ABR Scheduling Table totally after initialization.

ABR Wait Queue

Hardware uses the ABR Wait Queue with the ABR Scheduling Table to perform cell scheduling. The ABR Wait Queue size relates to the maximum number of actively segmenting VCs. The ABR Wait Queue size, in bytes, must be 2 times the maximum number of actively segmenting VCs. For example, if the (i)chipSAR+ is configured to handle 1 K active VCs, the ABR Wait Queue must be 2 Kbytes long.

The least significant bits of the Main Segmentation VC Table Base Address (VCT_BASE) Register control the ABR Wait Queue size. The ABR Wait Queue Base Register (ABRWQ_BASE) Register settings control the starting location of the ABR Wait Queue.

The host is responsible for initializing all bytes of the ABR Wait Queue to 0 at (i)chipSAR+ initialization and before the (i)chipSAR+ is enabled. Hardware maintains the ABR Wait Queue totally after initialization.

UBR Scheduling Structures

Two scheduling structures are associated with UBR:

- UBR Scheduling Table
- UBR Wait Queue

UBR Scheduling Table

Hardware uses the UBR Scheduling Table as the appointment book for UBR VC scheduling. The maximum number of simultaneous VCs to be segmented determines the size of the UBR Scheduling Table. The size of the UBR scheduling table, in turn, determines the lowest rate supported with UBR service. *Table 6-7* shows this relationship:

Table 6-7. UBR Scheduling Table Size and Lowest Supported Rate

Number of Simultaneous VCs	UBR Scheduling Table Size	Lowest Supported Rate for 155 Mbps	Lowest Supported Rate for 25.6 Mbps
1K	1K entries (4K bytes)	345 cells/s (132 Kbps)	58 cells/s (22 Kbps)
2K	2K entries (8K bytes)	173 cells/s (66 Kbps)	29 cells/s (11 Kbps)
4K	4K entries (16K bytes)	86 cells/s (33 Kbps)	14 cells/s (5 Kbps)

Note that, due to implementation differences, the number of bytes for each entry of the UBR Scheduling Table is 4 while the number of bytes for each entry of the ABR Scheduling Table is 2.

As mentioned in *ABR Scheduling Table on page 110*, the Traffic Management Parameters Register (STPARMS) Register settings control the scheduling table size. The UBR Scheduling Table Base Register (UBR_SBPTR_BASE) Register controls the starting location of the UBR Scheduling Table.

The host is responsible for initializing all bytes of the UBR Scheduling Table to 0 at (i)chipSAR+ initialization and before the (i)chipSAR+ is enabled. Hardware maintains the UBR Scheduling Table totally after initialization.

UBR Wait Queue

Hardware uses the UBR Wait Queue with the UBR Scheduling Table to perform cell scheduling. The UBR Wait Queue size relates to the maximum number of actively segmenting VCs. The UBR Wait Queue size, in bytes, must be 4 times the maximum number of actively segmenting VCs. For example, if the (i)chipSAR+ is configured to handle 1 K active VCs, the UBR Wait Queue must be 4 Kbytes long. Note that this size requirement is different from the ABR Wait Queue, which must be 2 bytes per entry.

The least significant bits of the Main Segmentation VC Table Base Address (VCT_BASE) Register control the UBR Wait Queue size. The starting location of the UBR Wait Queue is programmed through the UBR Wait Queue Base Register (UBRWQ_BASE) Register.

The host is responsible for initializing all bytes of the UBR Wait Queue to 0 at (i)chipSAR+ initialization and before the (i)chipSAR+ is enabled. Hardware maintains the UBR Wait Queue totally after initialization.

CBR Scheduling Table

The (i)chipSAR+ features a separate CBR Scheduling Table in Segmentation Control Memory. This feature ensures that no other traffic class will interfere with CBR scheduling, regardless of the number of VCs and the amount of traffic with other traffic classes.

The CBR Scheduling Table for the (i)chipSAR+ is simple. It is a circular table with a programmable number of entries. Each entry contains either a CBR VC index or the null VC index. Sequential entries in the table are checked every transmit cell time. If a non-null VC index is in an entry when the entry is checked, that VC will segment a cell during that cell time. If a null VC index is in an entry when the entry is checked, other service classes can use that cell slot.

Figure 6-13 provides a high-level illustration of the CBR Scheduling Table.

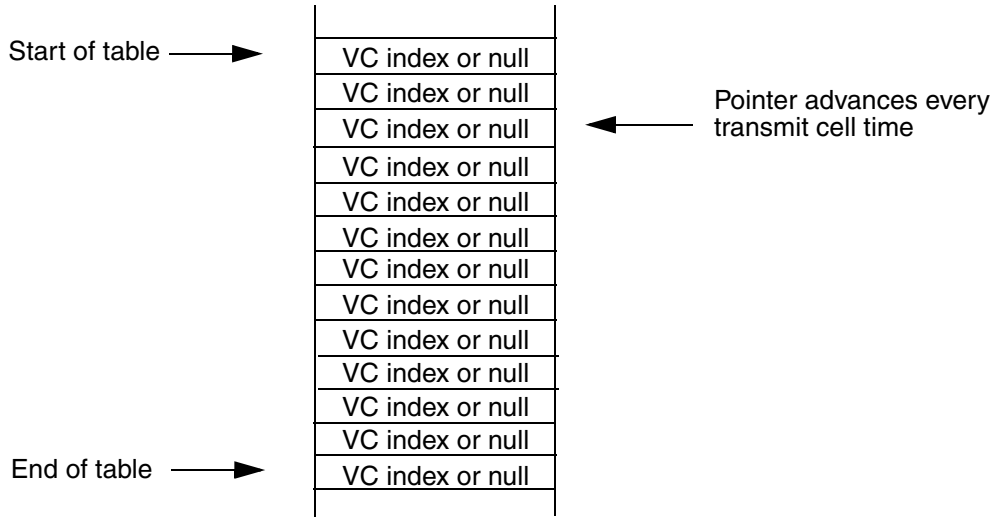

Figure 6-13. CBR Scheduling Table

System software loads the CBR Scheduling Table before the CBR VC is used. Since system software controls the size of the table and the placement of VC indices within the table, system software controls the exact rate and the exact cell spacing of the cells.

The CBR service class can be completely disabled if not needed. Disabling the CBR service class eliminates the requirement for a CBR Scheduling Table in memory.

Also, each VC within the CBR Scheduling Table has its own enable bit within its VC Table structure. This means that a VC can be resident in the table without actually having any data to segment. Keeping the VC resident in the table gives system software the ability to preload the CBR Scheduling Table at initialization, providing that system software knows the rates and cell spacing requirements of CBR VCs in advance.

Appendix B, Setup for CBR Scheduling, provides a detailed explanation, along with an example, of the algorithm used to load VC indices for particular rates and cell delay variations.

Segmentation Engine Registers

All (i)chipSAR+ registers are accessed as 16 bits, with bit 0 as the Least Significant Bit (LSB). All reserved bits in these registers must be set to 0 for proper operation unless otherwise mentioned. *Table 6-8* shows the (i)chipSAR+ Segmentation Engine internal registers used for (i)chipSAR+ setup. The Reset Value column specifies the default reset/power-up value of the registers.

Table 6-8. Segmentation Engine Internal Registers

Register Name	Addr. (Hex)	Description	Type[a]	Reset Value
IDLEHEADHI	00	Idle Cell Header (High) Register	R/W	0000H
IDLEHEADLO	01	Idle Cell Header (Low) Register	R/W	0000H
MAXRATE	02	Maximum Rate Register	R/W	0000H
STPARMS	03	Traffic Management Parameters Register	R/W	0000H
ABRUBR_ARB	04	ABR-UBR Programmable Priority Register	R/W	0000H
RM_TYPE	05	RM Cell Protocol ID and Message Type Register	R/W	0000H
COMMAND	17	Command Register	W	N/A
CBR_PTR_BASE	20	CBR Pointer Base Register	R/W	0000H
RESERVED	21	Reserved	R/W	N/A
ABR_SBPTR_BASE	22	ABR Scheduling Table Base Register	R/W	0000H
UBR_SBPTR_BASE	23	UBR Scheduling Table Base Register	R/W	0000H
RESERVED	25	Reserved	R/W	N/A
ABRWQ_BASE	26	ABR Wait Queue Base Register	R/W	0000H
UBRWQ_BASE	27	UBR Wait Queue Base Register	R/W	000H
VCT_BASE	28	Main Segmentation VC Table Base Address Register	R/W	0000H
VCTE_BASE	29	Extended Segmentation VC Table Base Address Register	R/W	0000H
CBR_TAB_BEG	2C	CBR Table Begin Register	R/W	0000H
CBR_TAB_END	2D	CBR Table End Register	R/W	0000H
PRQ_ST_ADR	30	Segmentation Packet Ready Queue Starting Address Register	R/W	0000H
PRQ_ED_ADR	31	Segmentation Packet Ready Queue Ending Address Register	R/W	0000H
PRQ_RD_PTR	32	Segmentation Packet Ready Queue Read Pointer Register	R/W	0000H
PRQ_WR_PTR	33	Segmentation Packet Ready Queue Write Pointer Register	R/W	0000H
TCQ_ST_ADR	34	Transmit Complete Queue Starting Address Register	R/W	0000H
TCQ_ED_ADR	35	Transmit Complete Queue Ending Address Register	R/W	0000H
TCQ_RD_PTR	36	Transmit Complete Queue Read Pointer Register	R/W	0000H
TCQ_WR_PTR	37	Transmit Complete Queue Write Pointer Register	R/W	0000H

Table 6-8. Segmentation Engine Internal Registers (cont)

Register Name	Addr. (Hex)	Description	Type[a]	Reset Value
QUEUE_BASE	40	Queue Base Register (base address for Packet Ready and Transmit Complete queues)	R/W	0000H
DESC_BASE	41	Descriptor Table Base Address Register	R/W	0000H
MODE_REG_0	45	Mode Register 0	R/W	0000H
MODE_REG_1	46	Mode Register 1	R/W	0000H
INTR_STATUS_REG	47	Interrupt Status Register	R	0000H
MASK_REG	48	Interrupt Mask Register	R/W	FFFFH
CELL_CTR_HIGH	49	Cell Counter High Register—Total cells transferred counter (high); auto clears on read	R	0000H
	C9	Cell Counter High Register—Total cells transferred counter (high); Does not auto clear on read.	R	0000H
CELL_CTR_LO	4A	Cell Counter Low Register—Total cells transferred counter (low); auto clears on read.	R	0000H
	CA	Cell Counter Low Register—Total cells transferred counter (low); Does not auto clear on read.	R	0000H

a. R/W—Read and Write
R—Read Only
W—Write Only

Table 6-9 shows the (i)chipSAR+ Segmentation Engine diagnostic registers. The diagnostic registers are used mainly in debugging environments, and do not require programming for normal operation. The Reset Value column specifies the default reset/power-up value of the registers.

Table 6-9. Diagnostic Registers

Register Name	Addr. (hex)	Description	R/W	Reset Value
NEXTDESC	59	Next Descriptor Register	R	0000H
NEXTVC	5A	Next VC Register	R	0000H
PSLOTCNT	5D	Present Slot Count Register	R/W	0000H
NEWDN	6A	New Descriptor Number Register	R	0000H
NEWVC	6B	New VC Register	R	0000H
SBPTR	6C	Schedule Table Pointer Register	R/W	0000H
RESERVED	6D	Reserved		N/A
RESERVED	6E	Reserved		N/A
ABRWQ_WRPTR	6F	ABR Wait Queue Write Pointer Register	R	0000H

Table 6-9. Diagnostic Registers (cont)

Register Name	Addr. (hex)	Description	R/W	Reset Value
ABRWQ_RDPTR	70	ABR Wait Queue Read Pointer Register	R	0000H
UBRWQ_WRPTR	71	UBR Wait Queue Write Pointer Register	R	0000H
UBRWQ_RDPTR	72	UBR Wait Queue Read Pointer Register	R	0000H
CBR_VC	73	CBR VC Register	R	0000H
RESERVED	74	Reserved		N/A
ABR_SBVC	75	ABR Should Be VC Register	R	0000H
UBR_SBVC	76	UBR Should Be VC Register	R	0000H
RESERVED	77	Reserved		N/A
ABRNEXTLINK	78	ABR Next Link Register	R	0000H
UBRNEXTLINK	79	UBR Next Link Register	R	0000H
RESERVED	7C	Reserved for future ABR service		N/A

Internal Registers

This section describes the ATM (i)chipSAR+ Segmentation Engine internal registers listed in *Table 6-8* on page *114*.

Idle Header Registers

There are two Idle Header Registers (IDLEHEADHI, IDLEHEADLO). These registers contain the header bytes to be appended to (i)chipSAR+ generated idle cells, as shown in *Figure 6-14* and *Figure 6-15*.

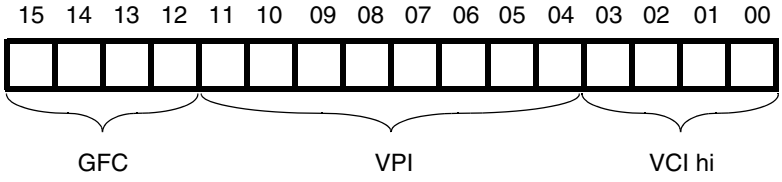

Figure 6-14. Idle Header Register High Register

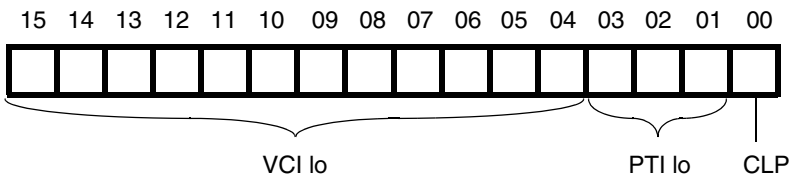

Figure 6-15. Idle Header Register Low Register

The reset value for these registers is 0, which corresponds to the standard Idle header.

Maximum Rate Register

The Maximum Rate Register (MAXRATE) contains the 16-bit floating point format of the maximum rate (in cells/second) that a VC can have in the (i)chipSAR+. *Figure 6-16* shows this register.

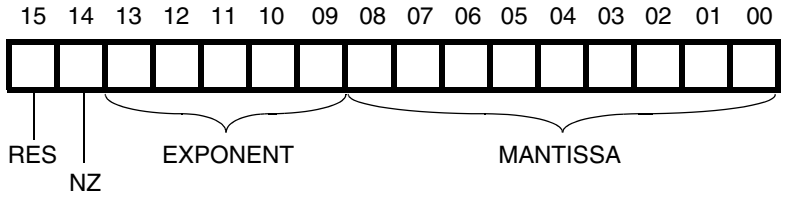

Figure 6-16. Maximum Rate Register

The register format is standardized according to the ATM Forum Traffic Management 4.0 Specification to represent the rate in terms of the exponent and mantissa together. The formula for the rate is:

$$1.\text{mantissa} \times 2^{\text{exponent}}$$

For operation at 155 Mbps (SONET OC-3), this register should be set to 64b1H. This setting corresponds to 352,768 cells/second.

For operation at 25.6 Mbps, this register should be set to 5f9dH. This setting corresponds to 59,200 cells/second.

Traffic Management Parameters Register

The Traffic Management Parameters Register (STPARMS) controls cell scheduling and the size of the ABR Scheduling Table and UBR Scheduling Table, as shown in *Figure 6-17*.

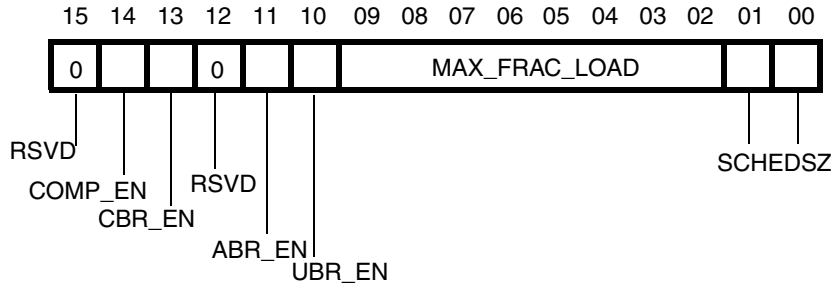

Figure 6-17. Traffic Management Parameters Register

Table 6-10 describes the individual Traffic Management Parameters data bits.

Table 6-10. Traffic Management Parameters Register

Bit	Description
15 RESERVED	Reserved. These bits must be set to 0.
14 COMP_EN	Cell Time Compensation. When this bit is set to 1, Cell Time Compensation is enabled. Cell Time Compensation is the (i)chipSAR+ ability to automatically compensate when the cell scheduler is behind because of excessive host activity on either the Segmentation Control Memory or the Packet/Cell Memory or both. The Cell Time Compensation circuit monitors the CELLCOMP input pin of the (i)chipSAR+. The CELLCOMP input pin is connected to a clock whose period corresponds to the exact rate at which cells are being transmitted over the physical medium. The CELLCOMP input signal can have any duty cycle, as long as both the high pulse and low pulse are at least 30 ns. wide. The CELLCOMP input signal can be asynchronous to the (i)chipSAR+ system clock. The Cell Time Compensation circuit ensures that situations do not arise that compromise the programmed Cell Delay Variation of CBR traffic. However, simulations have not produced any scenarios that caused the CBR scheduling to get behind, and this circuit should rarely, if ever, be needed.
13 CBR_EN	CBR Lookup Enable. Setting this bit to 1 enables the lookup operations on the CBR Scheduling Table. When this bit is set to 0, the CBR Scheduling Table is not needed in Segmentation Control Memory.
12 RESERVED	Reserved. This bit must be set to 0.

Table 6-10. Traffic Management Parameters Register (cont)

Bit	Description
11 ABR_EN	ABR Lookup Enable. Setting this bit to 1 enables the lookup operations on the ABR Scheduling Table and the ABR Wait Queue. When this bit is set to 0, the ABR Scheduling Table and ABR Wait Queue are not needed in Segmentation Control Memory.
10 UBR_EN	UBR Lookup Enable. Setting this bit to 1 enables the lookup operations on the UBR Scheduling Table and the UBR Wait Queue. When this bit is set to 0, the UBR Scheduling Table and UBR Wait Queue are not needed in Segmentation Control Memory.
09:02 MAX_FRAC_LOAD	Maximum Fraction Load bit. These 8 bits determine the minimum rate for an ABR VC before out-of-rate forward RM cells are sent. For a value of 10 cells per second (ABR spec), the following table shows the MAX_FRAC_LOAD values needed for each possibility of scheduling table size: SCHEDSIZE — MAX_FRAC_LOAD value 00 — 23H 01 — 12H 10 — 09H
01:00 SCHEDSIZE	Scheduling Table Size. These two bits specify the size of the ABR Scheduling Table and the UBR Scheduling Table, as shown below: SCHEDSIZE — Number of Entries 00 — 1K 01 — 2K 10 — 4K 11 — 8K

ABR-UBR Programmable Priority Register

The ABR-UBR Programmable Priority Register (ABRUBR_ARB) is an 8-bit variable that controls how long either ABR or UBR has priority over the other when both traffic types need to segment during the same cell time. *Figure 6-18* shows this register.

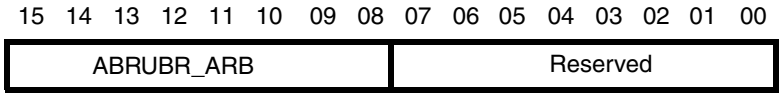

Figure 6-18. ABR-UBR Programmable Priority Register

Hardware rotates through the ABRUBR_ARB register on a cell-by-cell basis, with one bit rotated per cell time. Whenever both ABR and UBR services need to segment a cell during the same cell time:

- ABR has priority over UBR when the bit is set.
- UBR has priority over ABR when the bit is not set.

A typical setting of this 8-bit variable alternates 0 and 1, that is, AAH. If you want ABR to have priority over UBR more of the time, increase the density of 1's, for example, EEH. If you want UBR to have priority over ABR more of the time, decrease the density of 1's, for example, 22H.

RM Cell Protocol ID and Message Type

The RM Cell Protocol ID and Message Type Register (RM_TYPE) contains the 8-bit protocol ID and the 8-bit message type to be used for all ABR forward RM cells generated from this end station. *Figure 6-19* illustrates this register.

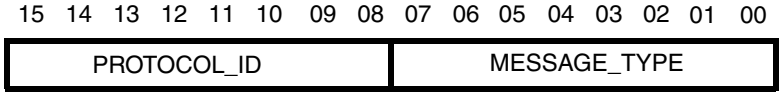

Figure 6-19. RM Cell Protocol ID and Message Type Register

For ATM Forum Traffic Management 4.0 support, set the PROTOCOL_ID field to 01H. Set the MESSAGE_TYPE field to 00H.

Command Register

The Command Register (COMMAND) is a write-only register used to issue software commands to the (i)chipSAR+ as follows:

- Reset Segmentation Engine state machine and registers when xx55 is written to this register.
- Reset Segmentation Engine state machine only when xxAA is written to this register.
- Reset Transmit Cell Counters only when xxCC is written to this register.

Values other than xxAA, xx55, or xxCC written to this register will be ignored.

CBR Pointer Base Register

The CBR Pointer Base Register (CBR_PTR_BASE) contains the base address of the CBR Scheduling Table. The two least significant bits of this register are concatenated with the 16-bit CBR Pointer Register (CBR_PTR) to form the Segmentation Control Memory address of the current entry in the CBR Scheduling Table. *Figure 6-20* shows this address generation.

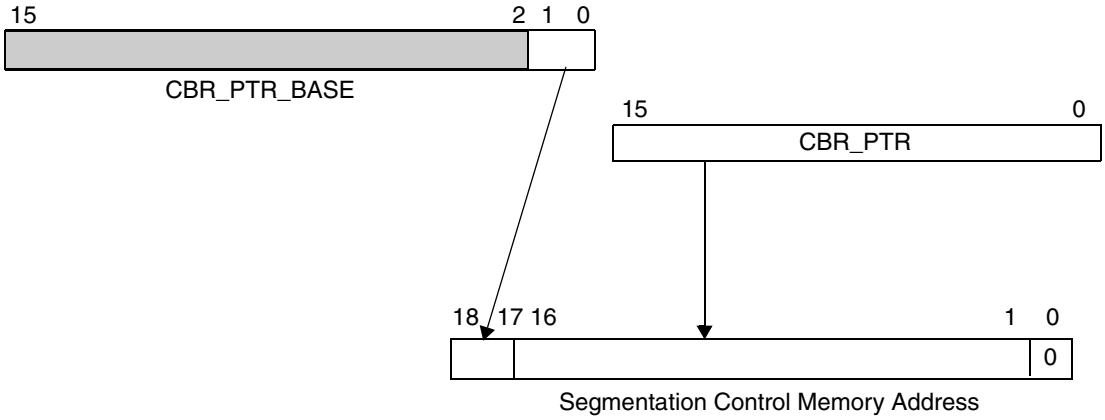

Figure 6-20. CBR Scheduling Table Address Generation

ABR Scheduling Table Base Register

The ABR Scheduling Table Base Register (ABR_SBPTR_BASE) contains the base address of the ABR Scheduling Table. A number of least significant bits of this register are concatenated with a number of least significant bits of the 13-bit Schedule Table Pointer Register (SBPTR) to form the Segmentation Control Memory address of the current entry in the UBR Scheduling Table. The SCHEDSIZE bit value in the Traffic Management Parameters Register (STPARMS) determines the number of bits from each entity. *Figure 6-21* shows this address generation.

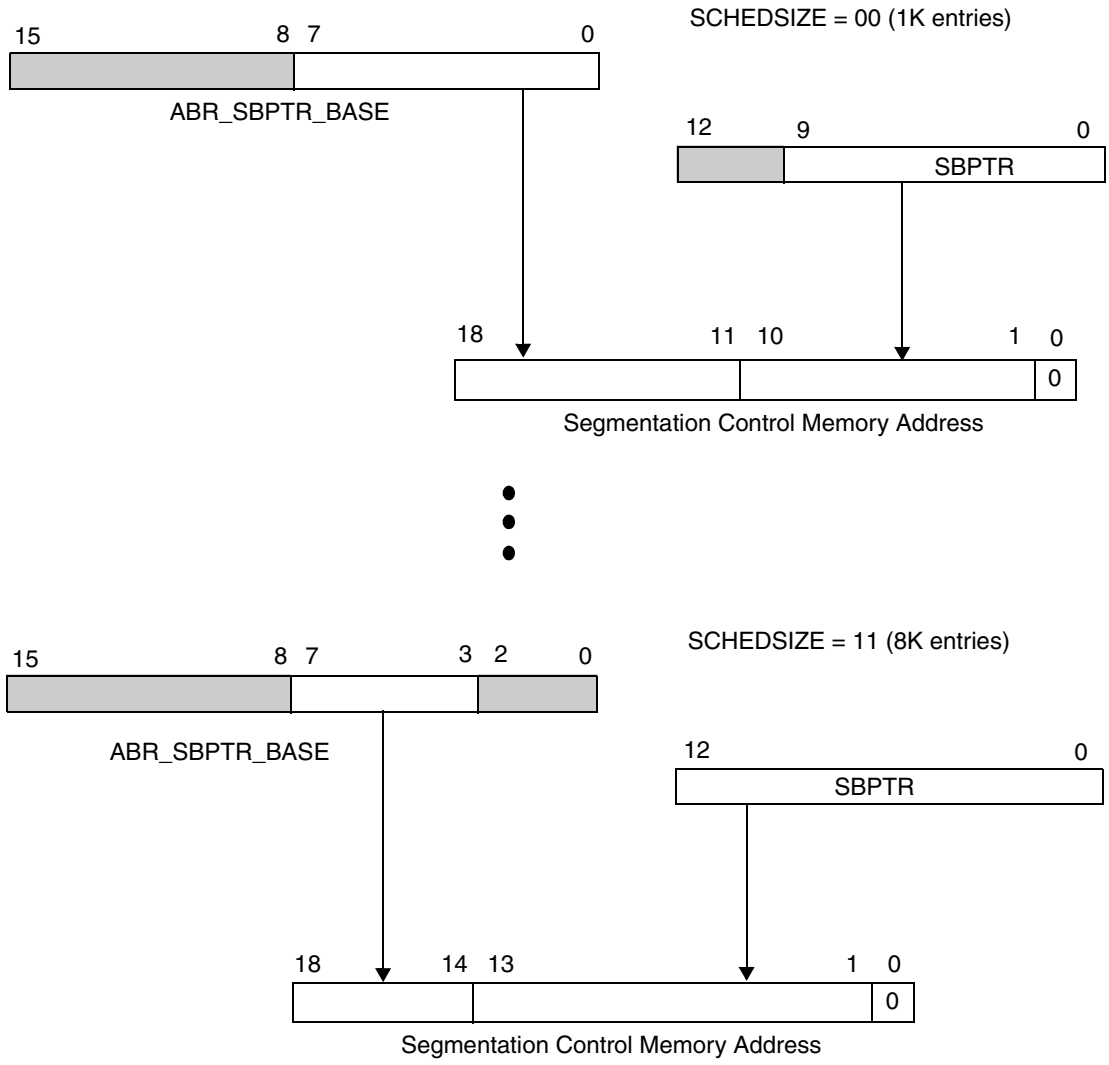

Figure 6-21. ABR Scheduling Table Address Generation

UBR Scheduling Table Base Register

The UBR Scheduling Table Base Register (UBR_SBPTR_BASE) contains the base address of the UBR Scheduling Table. A number of least significant bits of this register are concatenated with a number of least significant bits of the 13-bit Schedule Table Pointer Register (SBPTR) to form the Segmentation Control Memory address of the current entry in the UBR Scheduling Table. The SCHEDSIZE bit value in the Traffic Management Parameters Register (STPARMS) determines the number of bits from each entity.

The UBR Scheduling Table must start on a boundary that is the same size as the table. For example, a 1-Kbyte entry requires a 4-Kbyte table; therefore, the UBR Scheduling Table for a 1-Kbyte entry must start on a 4-Kbyte boundary.

Figure 6-22 shows this address generation.

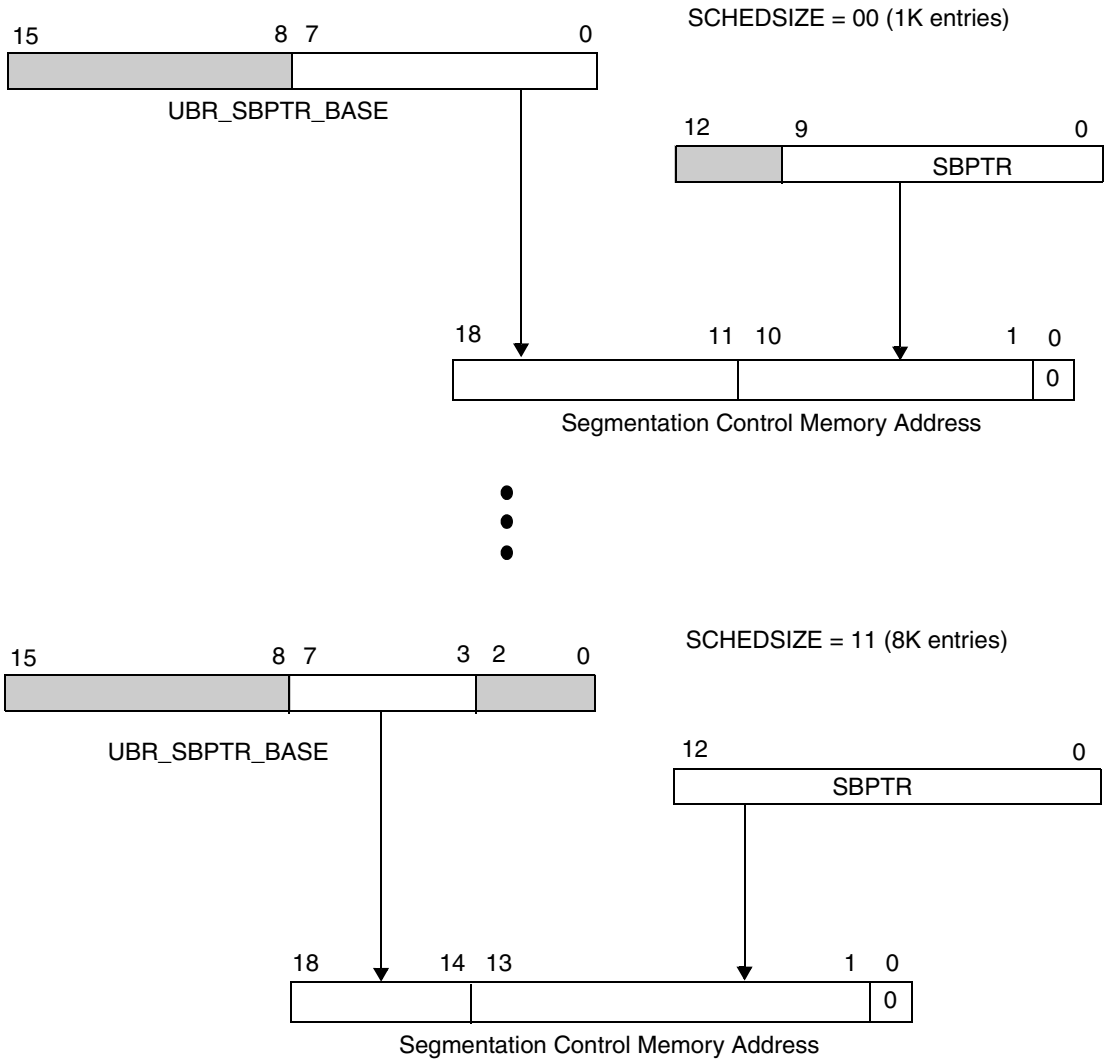

Figure 6-22. UBR Scheduling Table Address Generation

ABR Wait Queue Base Register

The ABR Wait Queue Base Register (ABRWQ_BASE) contains the base address of the ABR Wait Queue. A number of least significant bits of this register are concatenated with a number of least significant bits of the 13-bit wait queue pointer register to form the Segmentation Control Memory address of the current entry in the ABR Wait Queue. The wait queue pointer register might be either the ABR Wait Queue Read Pointer (ABRWQ_RDPTR) or the ABR Wait Queue Read Pointer (ABRWQ_WRPTR).

The three least significant bits of the Main Segmentation VC Table Base Address Register (VCT_BASE) determine the number of bits from each entity. *Figure 6-23* shows this address generation:

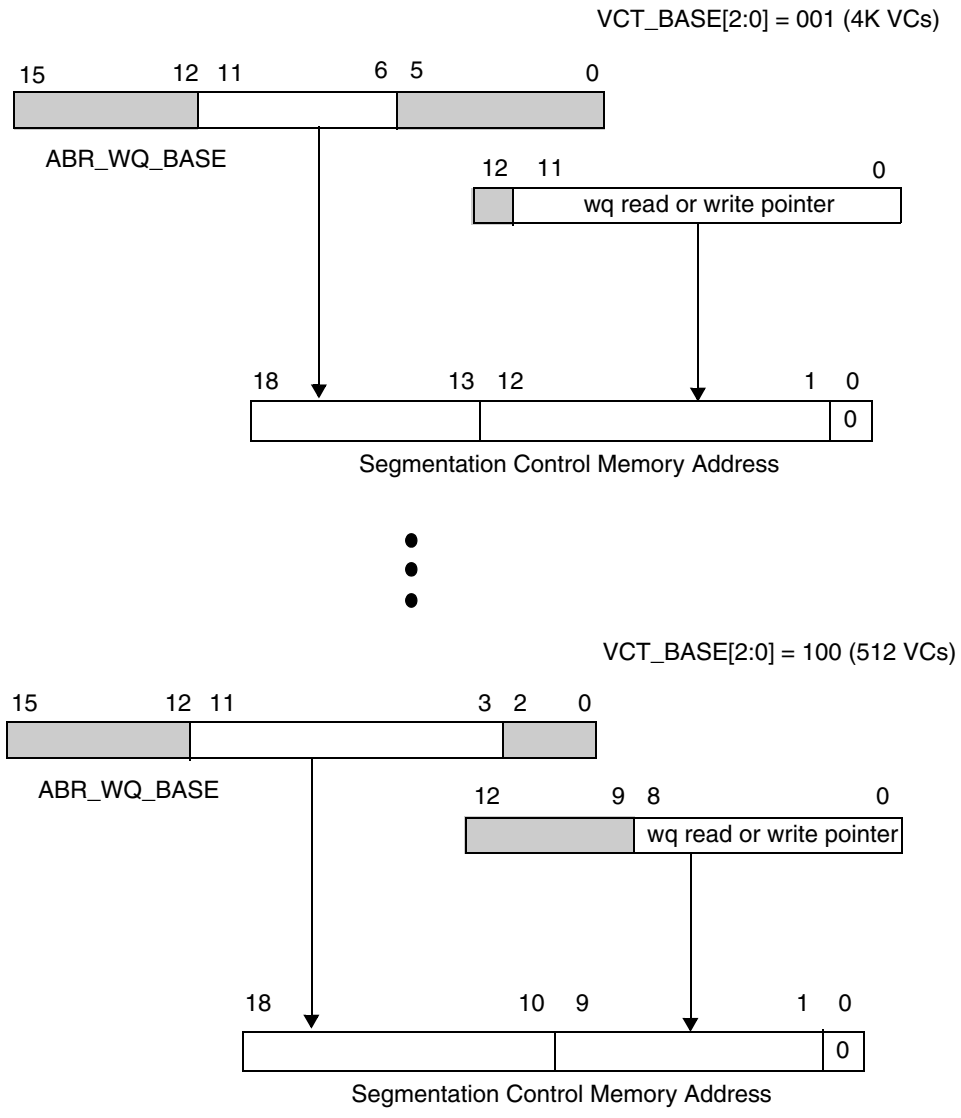

Figure 6-23. ABR Wait Queue Address Generation

UBR Wait Queue Base Register

The UBR Wait Queue Base Register (UBRWQ_BASE) contains the base address of the UBR Wait Queue. A number of least significant bits of this register are concatenated with a number of least significant bits of the 13-bit wait queue pointer register to form the Segmentation Control Memory address of the current entry in the UBR Wait Queue. The wait queue pointer register might be either the UBR Wait Queue Read Pointer (UBRWQ_RDPTR) or the UBR Wait Queue Read Pointer (UBRWQ_WRPTR).

The three least significant bits of the Main Segmentation VC Table Base Address Register (VCT_BASE) determine the number of bits from each entity. *Figure 6-24* shows this address generation.

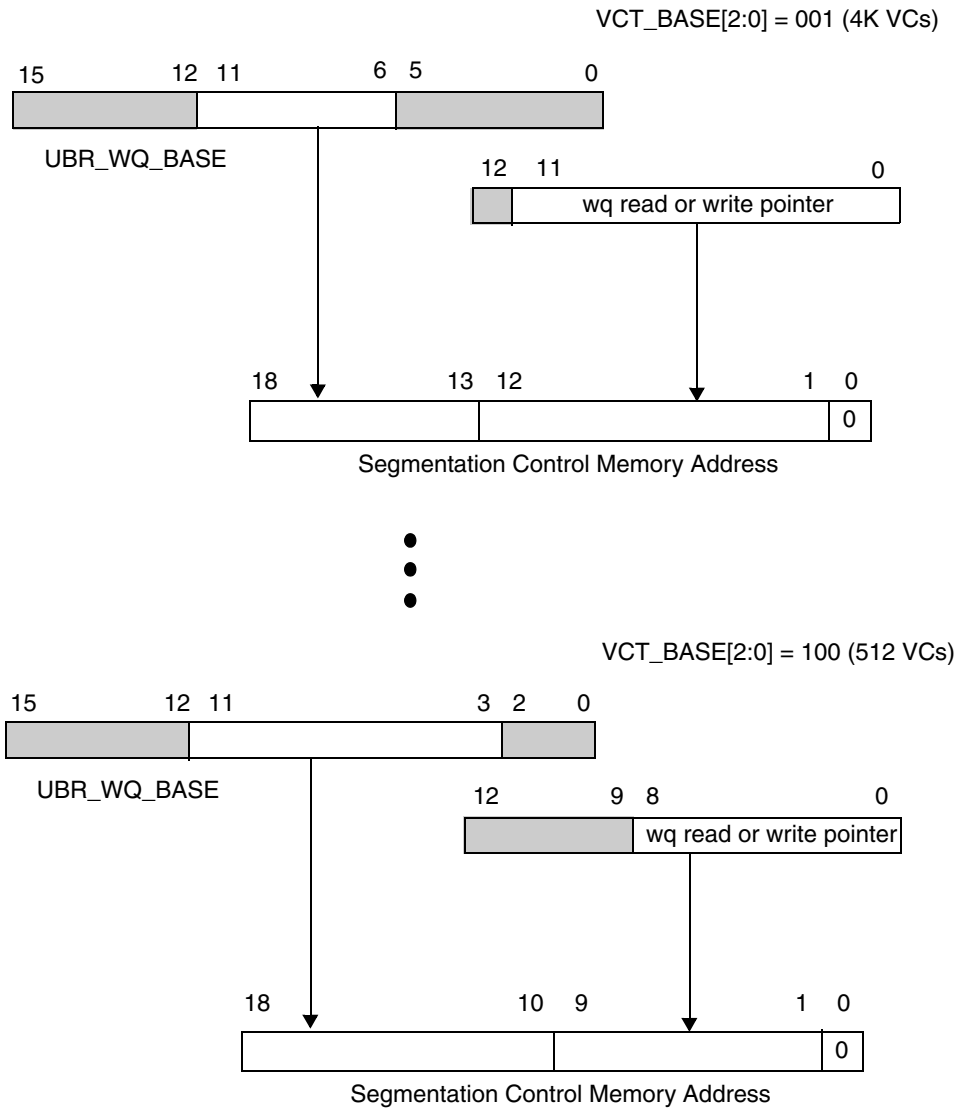

Figure 6-24. UBR Wait Queue Address Generation

Main Segmentation VC Table Base Address Register

The Main Segmentation VC Table Base Address Register (VCT_BASE) contains the base address of the Segmentation Main VC Table. The value in the VCT_BASE Register is concatenated with the lower bits of the VC index to generate the Segmentation Main VC Table entry address.

The least significant three bits of this base register determine how many bits of the base register are used in generating the address. This address generation is useful in configuring the control memory for various table sizes, and is illustrated in *Figure 6-25*.

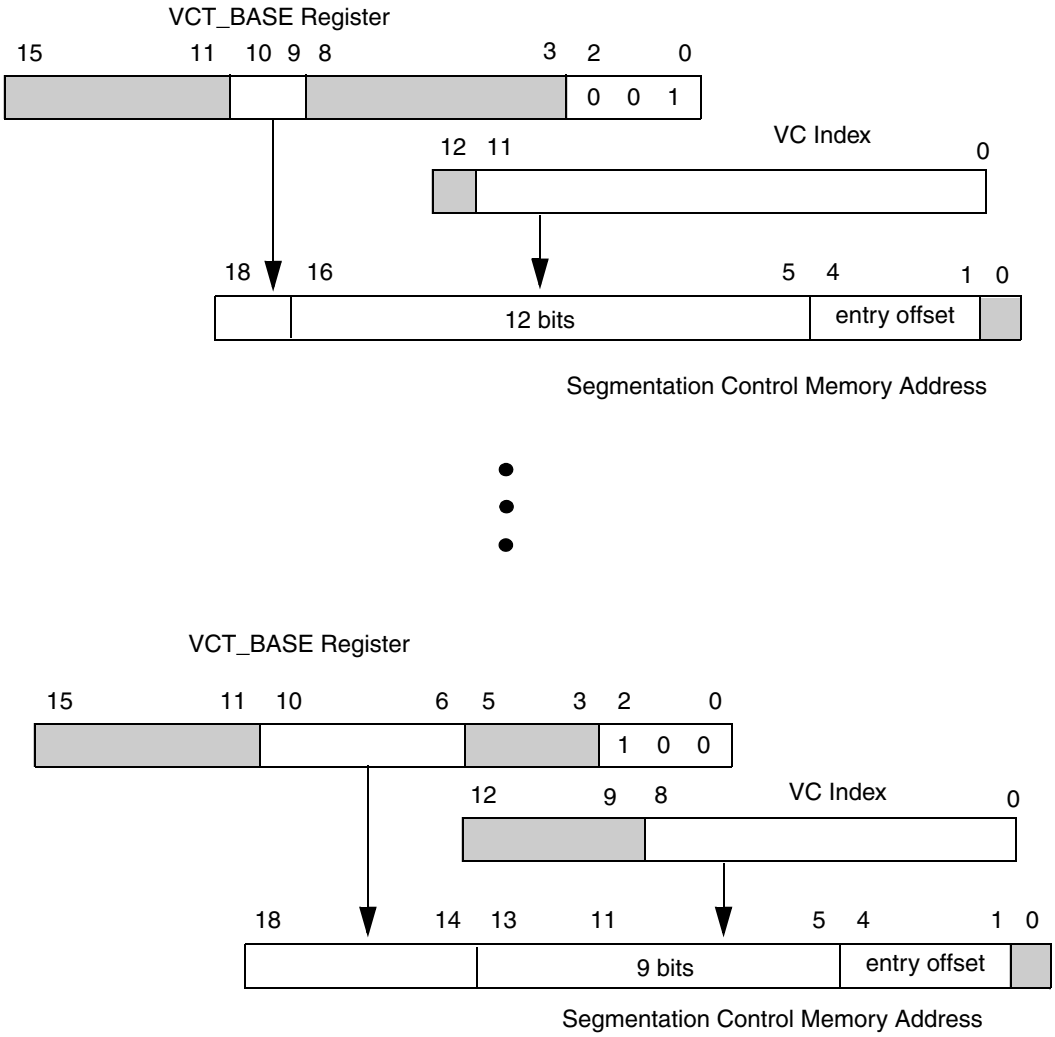

Figure 6-25. Segmentation Main VC Table Entry Address Generation

Table 6-11 on page 128 shows the number of maximum Segmentation VC Table entries possible for various values in the least significant three bits of the VCT_BASE Register. The number of bits in the VCT_BASE Register is concatenated with the number of bits in the VC index and with the offset into the 32-byte Main Segmentation VC Table entry.

Table 6-11. Number of Segmentation VC Table Entries for VCT_BASE Values

Least Significant Bits of VCT_BASE Register	Memory Address		Max Number of VCs Possible
	Number of bits used from VCT_BASE	Number of bits used from VC Index	
001	2	12	4096
010	3	11	2048
011	4	10	1024
100	5	9	512

Extended Segmentation VC Table Base Address Register

The Extended Segmentation VC Table Base Address Register (VCTE_BASE) contains the base address of the Segmentation Extended VC Table. The value in the VCTE_BASE Register is concatenated with the lower bits of the VC index to generate the Segmentation Extended VC Table entry address.

The least significant three bits of the VCT_BASE Register determine the number of bits of the base register used in generating the address. *Table 6-11 on page 128* shows the valid values of the lower 3 bits of the VCT_BASE Register.

Figure 6-26 shows the address generation of the 8-byte Extended VC Table entries.

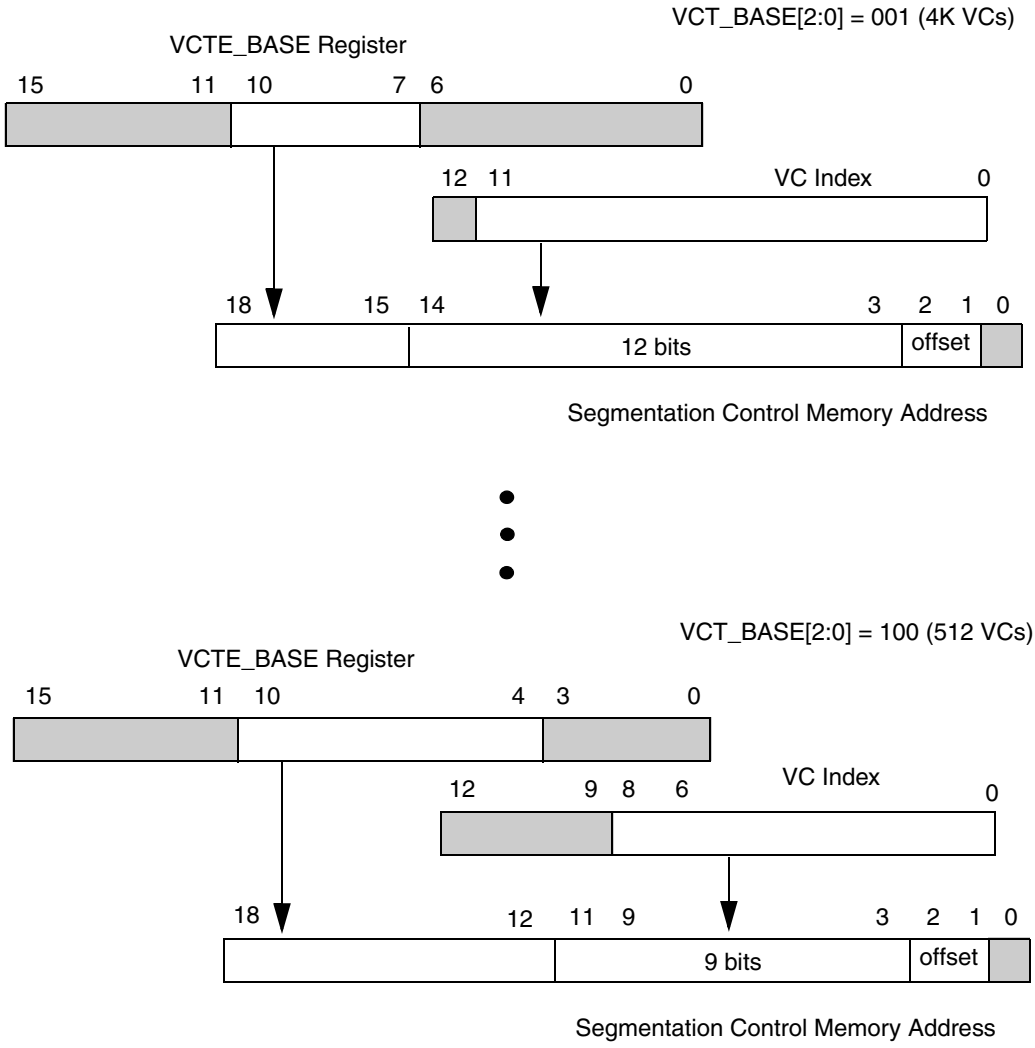

Figure 6-26. Segmentation Extended VC Table Entry Address Generation

CBR Table Begin and End Registers

The 16-bit CBR Table Begin and End Registers (CBR_TAB_BEG, CBR_TAB_END) define the starting and ending point of the CBR Scheduling Table. With these two registers, the CBR Scheduling Table can be placed anywhere within the 128K addressing range defined by the CBR_PTR_BASE Register.

At reset, the CBR_PTR Register is set to the CBR_TAB_BEG value. When the CBR_PTR Register has incremented past the CBR_TAB_END value, it will automatically be set to the CBR_TAB_BEG value again.

Figure 6-20 on page 122, illustrates address building specifics for the CBR Scheduling Table.

Segmentation Packet Ready Queue Pointers, Transmit Complete Queue Pointers, and Queue Base Register

The Segmentation Packet Ready Queue pointers are enumerated as:

- PRQ_ST_ADR (Segmentation Packet Ready Queue Starting Address)
- PRQ_ED_ADR (Segmentation Packet Ready Queue Ending Address)
- PRQ_RD_PTR (Segmentation Packet Ready Queue Read Pointer)
- PRQ_WR_PTR (Segmentation Packet Ready Queue Write Pointer)

The Transmit Complete Queue pointers are enumerated as:

- TCQ_ST_ADR (Transmit Complete Queue Starting Address)
- TCQ_ED_ADR (Transmit Complete Queue Ending Address)
- TCQ_RD_PTR (Transmit Complete Queue Read Pointer)
- TCQ_WR_PTR (Transmit Complete Queue Write Pointer)

The Queue Base Register is enumerated as QUEUE_BASE.

See *Segmentation Packet Ready Queue on page 99* for information about the use of these registers.

Descriptor Table Base Address Register

The Descriptor Table Base Address Register (DESC_BASE) contains the base address of the Segmentation Buffer Descriptor Table. The value in the descriptor base address, when concatenated with the descriptor number and the table offset, points to the buffer descriptor entry in control memory. Each Segmentation Buffer Descriptor Table entry is 32 bytes long. The table offset determines the specific 16-bit word within the table entry being accessed. *Figure 6-27* shows how the Descriptor Table byte address is formed.

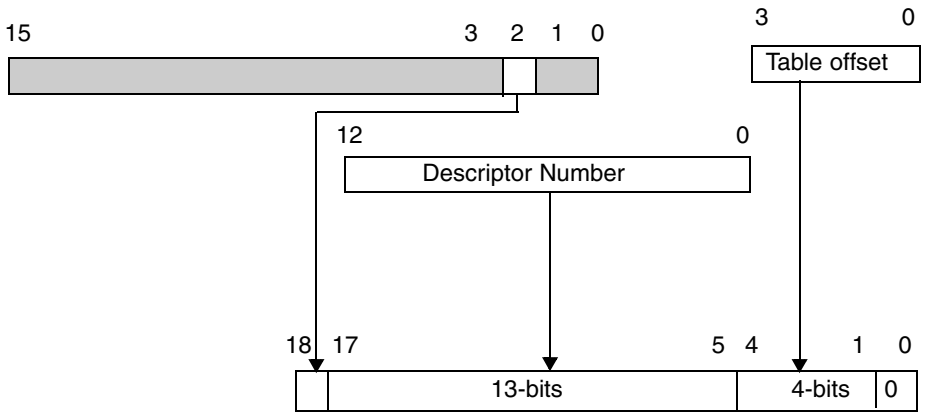

Figure 6-27. Descriptor Table Address Generation

Mode Register 0

Figure 6-28 shows the bits in the Mode Register 0 (MODE_REG_0):

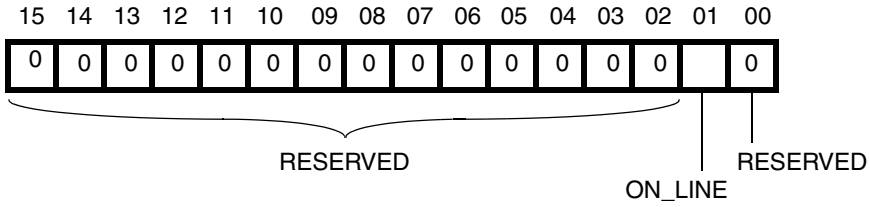

Figure 6-28. Mode Register 0 Data Bits

The following table describes the individual Mode Register 0 data bits:

Table 6-12. Mode Register 0 Register

Bit	Description
15:02 RESERVED	Reserved. These bits must be written as 0. Their read value is indeterminate, and they should be masked out.
01 ON_LINE	Online. This bit is used to control the segmentation process. Upon hard reset or soft reset, the (i)chipSAR+ defaults to the offline mode so that the host can set up the (i)chipSAR+ properly. 1 – (i)chipSAR+ is online. 0 – Schedule (i)chipSAR+ to enter offline gracefully. Programming this bit to 0 during the middle of a segmentation cycle forces the (i)chipSAR+ to gracefully enter the offline state after completing the current segmentation cycle(s), and sets the OFF_LINE bit in the State Register. Further segmentation and control memory accesses are inhibited during the offline state and will resume only after the (i)chipSAR+ is reprogrammed to be online.
00 RESERVED	Reserved. This bit must be written as 0. Its read value is indeterminate and it should be masked out.

Mode Register 1

Figure 6-29 shows the bits in the Segmentation Engine the (i)chipSAR+ Mode Register 1 (MODE_REG_1).

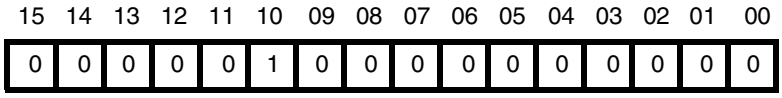

Figure 6-29. Mode Register 1

This register must be written as 0400H for proper device operation. When read, all bits except bit 10 should be masked out.

Interrupt Status Register

The Interrupt Status Register (INTR_STATUS_REG) contains *sticky* bits, which are set by certain events. Every bit in the Status Register is associated with a bit in the Interrupt Mask Register. If a particular mask bit is set to 0, then the corresponding status bit, when set to 1, generates an interrupt. If a particular mask bit is set to 1, the corresponding status bit does not generate an interrupt, but is set when the event occurs. Status bits are cleared when read by the processor.

Figure 6-30 shows the Segmentation Engine (i)chipSAR+ Interrupt Status Register bits.

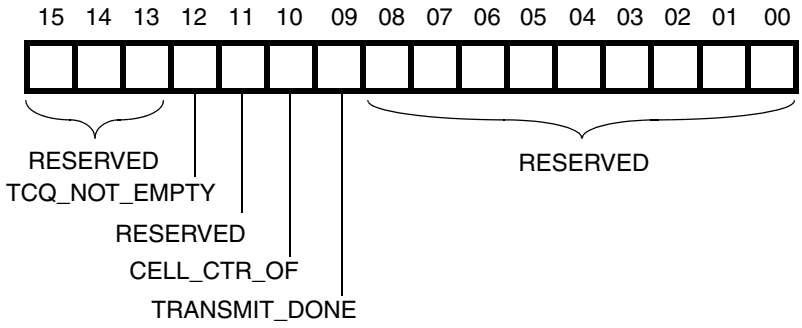

Figure 6-30. Interrupt Status Register

Table 6-13 describes the Interrupt Status Register bits.

Table 6-13. Interrupt Status Register

Bit	Description
15:13 RESERVED	Reserved. These bits must be masked out and ignored.
12 TCQ_NOT_EMPTY	Transmit Complete Queue Not Empty. After segmentation of any packet is completed, the buffer descriptor is delinked from the Segmentation Engine (i)chipSAR+ internal data structures and queued in the Transfer Complete Queue (TCQ). When the TCQ first goes non-empty, this bit is set to 1, indicating that descriptors are queued in the TCQ. The host can then choose to process the completed descriptor.
11 RESERVED	Reserved. This bit must be masked out and ignored.
10 CELL_CTR_OF	Cell Counter Overflow. When set, this bit indicates that the 32-bit Transmit Cell Counter has overflowed. The counter will then wrap around and continue counting from 0.
09 TRANSMIT_DONE	Transmit Done bit. When a descriptor with CMPL_INT bit set is delinked after segmentation, this status bit will be set by the Segmentation Engine of the (i)chipSAR+. (The CMPL_INT bit is in the Descriptor Mode field of the Segmentation Buffer Descriptor Table.)
08:00 RESERVED	Reserved. These bits must be masked out and ignored.

Interrupt Mask Register

Figure 6-31 shows the (i)chipSAR+ Interrupt Mask Register (MASK_REG) bits. Each bit in the Interrupt Mask Register corresponds to an associated bit in the Interrupt Status Register. If a particular mask bit is set to 0, then the corresponding status bit, when set to 1, will result in an interrupt generation. Setting a mask bit to 1 will suppress only the interrupt generation due to the associated status condition.

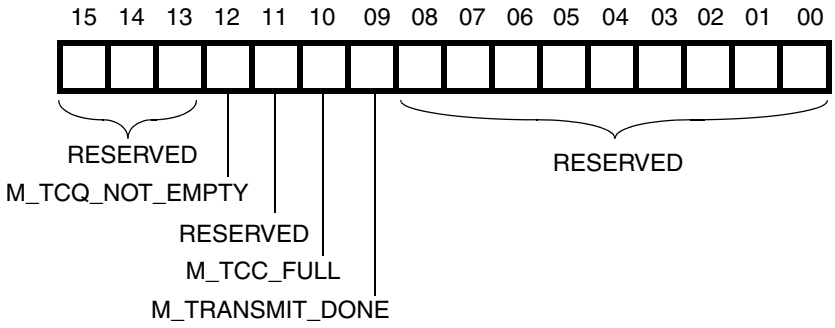

Figure 6-31. Interrupt Mask Register

Table 6-14 describes the Interrupt Mask Register bits.

Table 6-14. Interrupt Mask Register

Bit	Description
15:13 RESERVED	Reserved. These bits must be set to 1.
12 M_TCQ_NOT_EMPTY	Mask Transmit complete Queue Not Empty Interrupt. 1 – Mask interrupt generation because of the TCQ_NOT_EMPTY bit Interrupt Status Register. 0 – Unmask interrupt generation because of the TCQ_NOT_EMPTY TCQ_NOT_EMPTY bit.
11 RESERVED	Reserved. This bit must be set to 1.
10 M_TCC_OF	Mask Transmit Cell Counter Overflow Interrupt. 1 – Mask interrupt generation because of Transmit Cell Counter overflow. 0 – Unmask interrupt generation because of Transmit Cell Counter overflow.
09 M_TRANSMIT_DONE	Mask Transmit Done Interrupt bit. 1 – Mask interrupt generation because of a transmit done on specially marked descriptors. 0 – Unmask interrupt generation because of a transmit done on specially marked descriptors.
08:00 RESERVED	Reserved. These bits must be set to 1.

Cell Counter Registers

Two separate 16-bit, read-only Cell Counter registers (CELL_CTR_HIGH and CELL_CTR_LO) comprise a 32-bit cell counter.

The cell count indicates the total number of cells the (i)chipSAR+ has transmitted since being previously reset.

Each counter register can be accessed by reading two separate locations. Reading a counter from the first location automatically clears the respective counter, while reading from the alternate location preserves the count value without clearing the counter. In a system environment where different network entities can access the counters, we recommend that the counters do not automatically clear when read.

For an accurate count, we recommend that CELL_CTR_LO register be read before CELL_CTR_HIGH register is read. If the (i)chipSAR+ transmits cells back-to-back on a continuous basis at a 150 Mbps rate, it takes over 8 days for this counter to overflow. When the counter overflows, an interrupt status bit is set, generating a maskable interrupt.

Diagnostic Registers

This section provides descriptions of the diagnostic registers listed in *Table 6-9 on page 115*.

CBR Pointer Register

The 16-bit, writable CBR Pointer Register (CBR_PTR) holds the present pointer into the CBR Scheduling Table.

This register is available for diagnostic reasons; it must not be written during normal operation.

State Register

The bits of the (i)chipSAR+ Transmit SAR State Register (STATE_REG) are read-only status bits indicating the dynamic state of the chip. These bits do not generate an interrupt. Values written to these locations are ignored. All bits in this register are cleared automatically when the condition that them to be set is ended.

Figure 6-32 shows the (i)chipSAR+ Transmit SAR State Register bits.

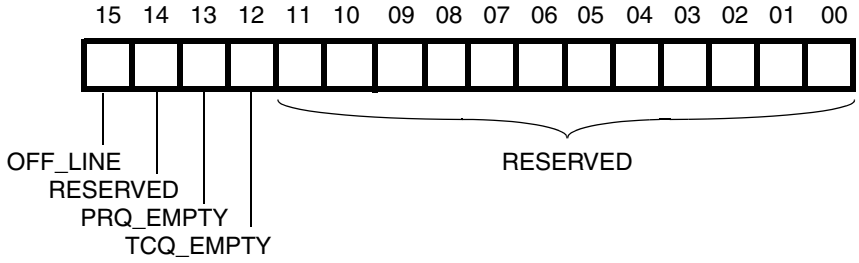

Figure 6-32. State Register

Table 6-15 describes the State Register bits.

Table 6-15. State Register

Bit	Description
15 OFF_LINE	Offline. This bit indicates the state of the chip. It is set to 1 immediately after a hard/soft reset ((i)chipSAR+ is in offline state). Packet segmentation and control memory accesses will not be performed until the (i)chipSAR+ is placed back online (ON_LINE of Mode Register 0). When the processor forces the (i)chipSAR+ to enter the offline state, this bit must be polled to see if the (i)chipSAR+ has entered offline state gracefully. If this bit still indicates online state, then the (i)chipSAR+ is in the middle of a segmentation cycle and has not yet entered the offline state. During online mode, this bit is in the 0 state.
14 RESERVED	Reserved. This bit must be masked out and ignored.
13 PRQ_EMPTY	Packet Ready Queue Empty. This bit indicates the Segmentation Packet Ready Queue empty state. When the queue is empty, this bit reads 1. This bit can be used to quickly see if the (i)chipSAR+ has linked descriptors queued for segmentation into its internal data structures.
12 TCQ_EMPTY	Transmit Complete Queue Empty. This bit indicates the Transmit Complete Queue empty state. When the queue is empty, this bit reads 1. This bit can be used to quickly see if the (i)chipSAR+ has delinked and freed up descriptors after packet segmentation. This bit is a 0 when the queue is not empty.
11:00 RESERVED	Reserved. These bits must be masked out and ignored.

Current Descriptor Number Register

The Current Descriptor Number Register (CURR_DESC_NUM) can be read to find the descriptor that the (i)chipSAR+ has most recently processed or is currently processing.

This register is made accessible for diagnostic reasons; it must not be programmed for normal operation.

Next Descriptor Register

The 16-bit, read-only Next Descriptor Register (NEXTDESC) contains the 13-bit NEXTDESC Register value. This register is passed from the Traffic Management Scheduling state machine to the Segmentation Engine state machine to identify which descriptor number is segmenting next.

Figure 6-33 shows the organization of the 16-bit read of this register.

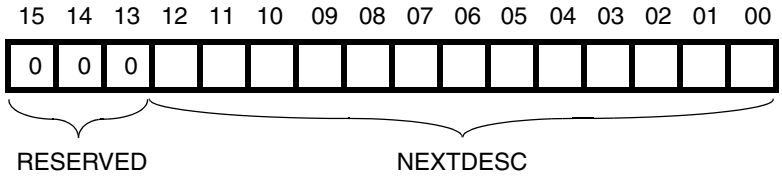

Figure 6-33. Next Descriptor Register

Next VC Register

The 16-bit, read-only Next VC Register (NEXTVC) contains the 13-bit NEXTVC Register value. This register is passed from the Traffic Management Scheduling state machine to the Segmentation Engine state machine to identify which VC is segmenting next.

Figure 6-34 shows the organization of the 16-bit read of this register.

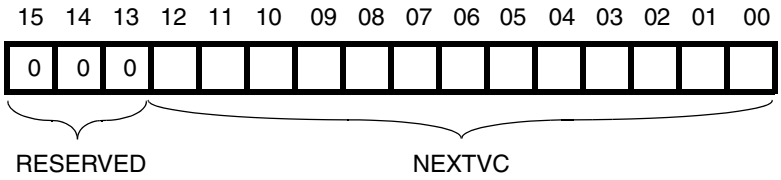

Figure 6-34. Next VC Register

Present Slot Count Register

The Present Slot Count Register (PSLOTCNT) is the working register for the Present Slot Count value in the Main VC Table entries. The Present Slot Count value keeps a timestamp of when the last cell for the VC was sent. The Traffic Management Scheduling state machine uses this value to determine whether a previously idle VC can begin transmitting cells immediately or whether it must wait for a time that would place it within its negotiated rate.

This register is available for diagnostic purposes; it must not be programmed for normal operation.

New Descriptor Number Register

The New Descriptor Number Register (NEWDN) passes the descriptor number of a new packet from the Segmentation Engine to the Traffic Management Scheduling state machine.

Figure 6-35 shows the organization of the 16-bit read of this register.

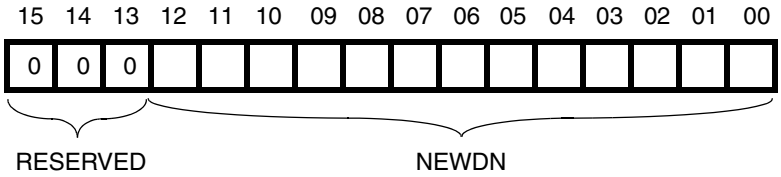

Figure 6-35. New Descriptor Number Register

New VC Number Register

The New VC Number Register (NEWVC) passes the VC index of a new packet from the Segmentation Engine to the Traffic Management Scheduling state machine.

Figure 6-36 shows the organization of the 16-bit read of this register.

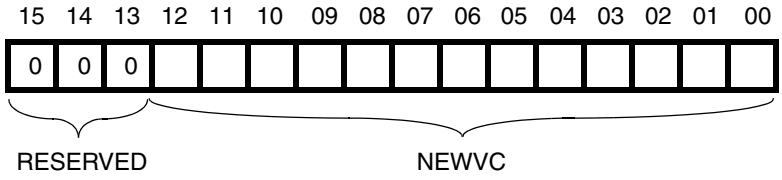

Figure 6-36. New VC Number Register

Schedule Table Pointer Register

The Schedule Table Pointer Register (SBPTR) holds the offset into the ABR and UBR Scheduling Tables.

This register is available for diagnostic purposes; it must not be programmed for normal operation.

ABR Wait Queue Write Pointer Register

The 13-bit ABR Wait Queue Write Pointer Register (ABRWQ_WRPTR) holds the write pointer of the ABR Wait Queue.

Figure 6-37 shows the organization of the 16-bit read of this register.

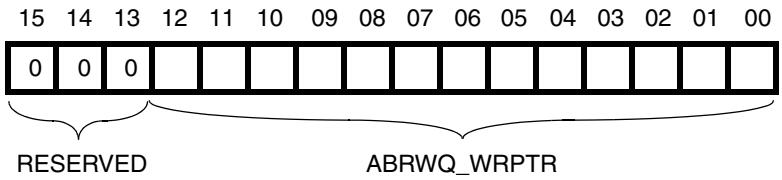

Figure 6-37. ABR Wait Queue Write Pointer Register

ABR Wait Queue Read Pointer Register

The 13-bit ABR Wait Queue Read Pointer Register (ABRWQ_RDPTR) holds the read pointer of the ABR Wait Queue.

Figure 6-38 shows the organization of the 16-bit read of this register.

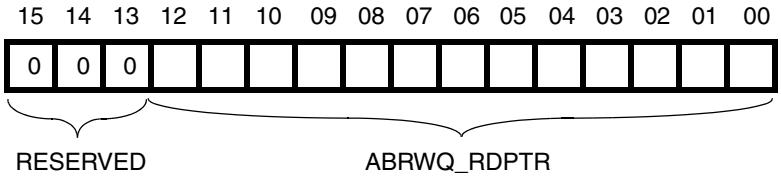

Figure 6-38. ABR Wait Queue Read Pointer Register

UBR Wait Queue Write Pointer Register

The 13-bit UBR Wait Queue Write Pointer Register (UBRWQ_WRPTR) holds the write pointer of the UBR Wait Queue.

Figure 6-39 shows the organization of the 16-bit read of this register.

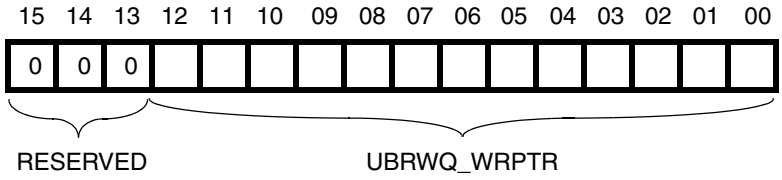

Figure 6-39. UBR Wait Queue Write Pointer Register

UBR Wait Queue Read Pointer Register

The 13-bit UBR Wait Queue Read Pointer Register (UBRWQ_RDPTR) holds the read pointer of the UBR Wait Queue.

Figure 6-40 shows the organization of the 16-bit read of this register.

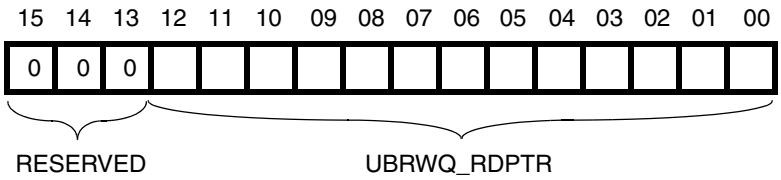

Figure 6-40. UBR Wait Queue Read Pointer Register

CBR VC Register

The CBR VC Register (CBR_VC) holds the latest 13-bit CBR VC index. It can be either null or non-null. If non-null, then this VC index will be sent to the Segmentation Engine for segmentation during the next cell slot.

Figure 6-41 shows the organization of the 16-bit read of this register.

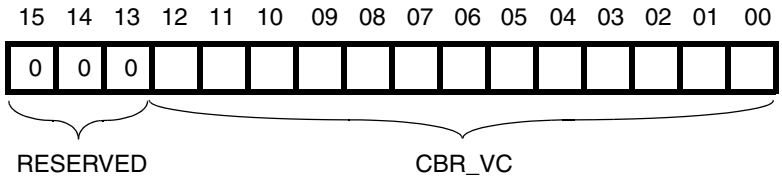

Figure 6-41. CBR VC Register

ABR Should Be VC Register

The ABR Should Be VC Register (ABR_SBVC) holds the 13-bit ABR VC index that was last read from the ABR Scheduling Table. It can be either null or non-null.

If non-null, then this VC index is sent to the Segmentation Engine for segmentation during the next cell slot, unless a CBR VC, a backup of ABR VCs, or a UBR VC is pending with higher priority. In any of these cases, this VC is placed on the ABR Wait Queue.

Figure 6-42 shows the organization of the 16-bit read of this register.

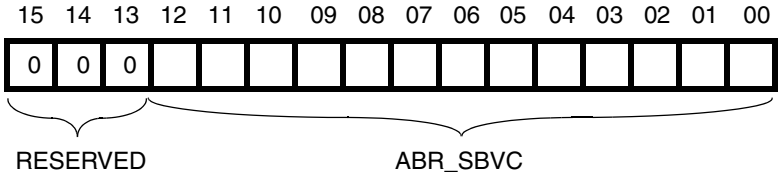

Figure 6-42. ABR Should Be VC Register

UBR Should Be VC Register

The UBR Should Be VC Register (UBR_SBVC) holds the 13-bit UBR VC index that was last read from the UBR Scheduling Table. It can be either null or non-null.

If non-null, then this VC index is sent to the Segmentation Engine for segmentation during the next cell slot, unless a CBR VC, a backup of ABR VCs, or a UBR VC is pending with higher priority. In any of these cases, this VC is placed on the UBR Wait Queue.

Figure 6-43 shows the organization of the 16-bit read of this register.

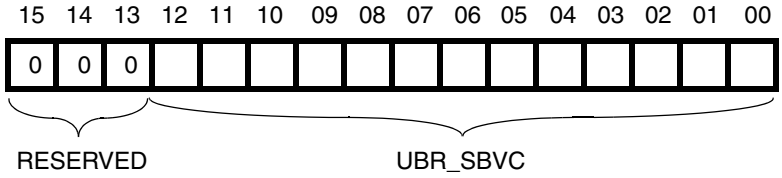

Figure 6-43. UBR Should Be VC Register

ABR Next Link Register

The ABR Next Link Register (ABRNEXTLINK) holds the 13-bit ABR Next Link VC index. This register represents the next link in a linked list of VCs that collided at a particular location in the ABR Scheduling Table.

Figure 6-44 shows the organization of the 16-bit read of this register.

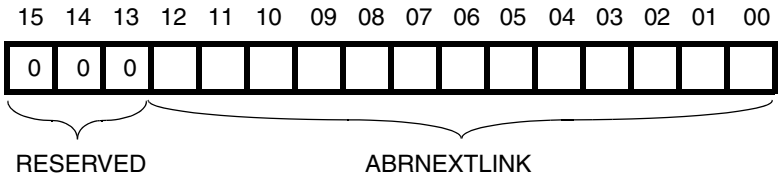

Figure 6-44. ABR Next Link Register

UBR Next Link Register

The UBR Next Link Register (UBRNEXTLINK) holds the 13-bit UBR Next Link VC index. This register represents the next link in a linked list of VCs that collided at a particular location in the UBR Scheduling Table.

Figure 6-45 shows the organization of the 16-bit read of this register.

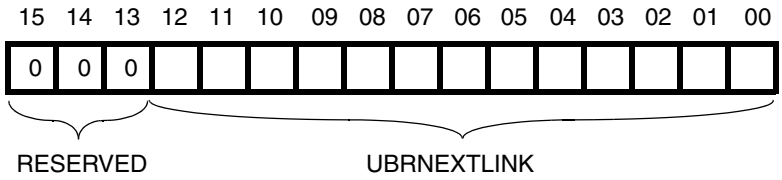

Figure 6-45. UBR Next Link Register

Reassembly Engine Software Reference 7

Overview

This chapter provides information about the Reassembly Control Memory Structures and the Reassembly Registers of the (i)chipSAR+.

Reassembly Control Memory Structures

Reassembly Control memory structures include the following tables and queues:

- Reassembly Buffer Descriptor Table
- VC Table
- VP Table
- Reassembly Table
- ABR VC Table
- Free Descriptor Queue
- Packet Complete Queue
- Exception Queue

Reassembly Buffer Descriptor Table

The Reassembly Buffer Descriptor Table is a set of 32-byte entries. Each entry contains information about a reassembled packet and the location of the packet buffer in the Packet/Cell Memory or system memory, depending on reassembly mode. When the (i)chipSAR+ reassembles a packet, it stores the packet parameters in the Reassembly Buffer Descriptor and stores the packet in the packet buffer specified in the descriptor.

Each Reassembly Buffer Descriptor Table entry is referenced by its buffer descriptor number, which can range from 1 to 4K. Buffer descriptor number 0 should not be used, to assure compatibility with the Segmentation Engine of the (i)chipSAR+. The location of the entry is determined by concatenating the buffer descriptor number with the descriptor base address.

Reassembly Buffer Descriptor Table Organization

Figure 7-1 shows the organization of each Reassembly Buffer Descriptor Table entry. It also shows the addresses of the individual entries in the table and the types of accesses available to the software entity and the (i)chipSAR+.

Address	15	0	SW	(i)chipSAR+
0H	Descriptor Mode/Status Bits		R/W	R/W
2H	VC Index		R	W
4H	Virtual-Path Identifier (VPI)		R	R/W
6H	Packet Byte Count		R	R/W
8H	Packet Memory Start Address (High)		W	R
AH	Packet Memory Start Address (Low)		W	R
CH	DMA Address—Upper		R	R/W
EH	DMA Address - Lower		R	R/W
10H	Residual CRC - Upper		R	R/W
12H	Residual CRC - Lower		R	R/W
14H	Reserved	Packet Time-out Count	R	R/W
16H - 1EH	Reserved			R/W

Figure 7-1. Reassembly Descriptor Table Entry Organization

The following sections describe Reassembly Descriptor Table entries.

Descriptor Mode/Status Bits

The Descriptor Mode bits are used to set up the descriptor entry. The Status bits are used by the (i)chipSAR+ to report the error status of the reassembled packet for normal or test operation. *Figure 7-2* shows the Descriptor Mode/Status bit positions.

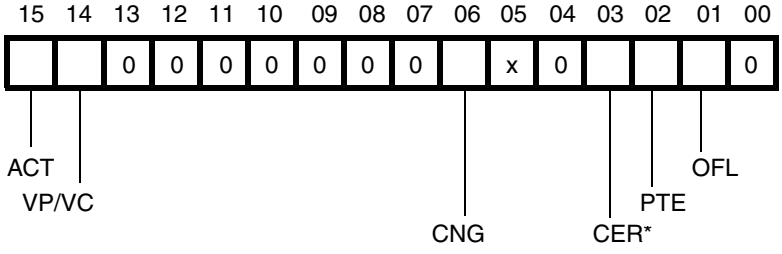

Figure 7-2. Descriptor Mode/Status Bits

Table 7-1 describes the descriptor mode and status bits.

Table 7-1. Descriptor Mode/Status Bits

Bit	Description
15 ACT	Active. The ACT bit is set high when the (i)chipSAR+ is reassembling the packet associated with this descriptor. The (i)chipSAR+ sets this bit to 0 when the reassembly process is completed, before the descriptor is passed to the software entity. **CAUTION: This bit must not be modified**.
14 VP/VC	VP/VC Reassembly. The VP/VC bit indicates whether the packet reassembly was performed on the VCI or VCI/VPI. 0 – VCI reassembly 1 – VPI/VCI reassembly and VP index is in entry address 4H
13:07 RESERVED	Reserved. Don't care on writes. Always read as 0.
06 CNG	Congestion. This bit is set when the (i)chipSAR+ receives a cell with the Congestion bit set in the cell header, as an indicator of congestion on the circuit. 0 – No congestion experienced by this packet in the network. 1 – Congestion was experienced by one or more cells of this packet during transit across the network.
05 RESERVED	Reserved. This bit should be masked out and ignored.
04 RESERVED	Reserved. Don't care on writes. Always read as 0.
03 CER	CRC Error. This bit is set to 1 when a packet CRC error occurs. CER should be ignored if packet CRC is not being used on this VC. 0 – No error 1 – Packet CRC error
02 PTE	Packet Timeout Error. This bit is set to 1 when a packet has not completed within the pre-programmed time (aged out) and is terminated. 0 – No error 1 – Packet timeout
01 OFL	Overflow. This bit is set to 1 when a packet overflows the packet buffer in the packet memory and is terminated. 0 – No error 1 – Buffer overflow error
00 RESERVED	Reserved. Don't care on writes. Always read as 0.

VC Index

The VC Index field determines the identifier of the virtual circuit that the packet was received on. This 13-bit field is used as a pointer to reference the Reassembly Table entry corresponding to the virtual circuit.

VP Identifier (VPI)

The VP Identifier field (VPI) indicates the VPI (in bits 7:0) on which the packet was received.

Packet Byte Count

The (i)chipSAR+ loads the Packet Byte Count field from the BUF_SIZE Register for AAL5 packets. The (i)chipSAR+ counts down the contents during the packet reassembly process. The purpose is to detect the end of packets and buffer overflows.

Packet Memory Start Address

The packet starting location in the packet memory address space, relative to the (i)chipSAR+, is generated by concatenating the high and low fields of the packet memory start address. The contents of the Packet Memory Start Address field must be initialized to the 32-bit word address of the beginning of the packet buffer in either the Packet/Cell Memory or system memory, depending on the mode of reassembly chosen.

DMA Address

The contents of the DMA Address field are initialized from the Packet Memory Start Address field, and contain a pointer to the location immediately following the end of the packet.

Residual CRC

The Residual CRC field is used to maintain the residual 32-bit CRC value of the packet in reassembly (between cells).

Packet Time-out Count

The 8-bit Packet Time-out Count field is initialized by the (i)chipSAR+ at the start of the packet reassembly, and is used to maintain the age of the packet. This field is incremented at periodic intervals while the packet is in the active reassembly process. If this field overflows, the packet is terminated and the PTE bit is set in the Descriptor Status field.

Reassembly Table Pointer/Descriptor Table Entry Generation

Figure 7-3 shows the process of generating the Reassembly Table pointer and the Reassembly Buffer Descriptor Table entry, depending on whether VPI/VCI or VCI-only reassembly is being done. The VP Filter Register determines whether the VP Table or VC Table will be used to generate the reassembly pointer.

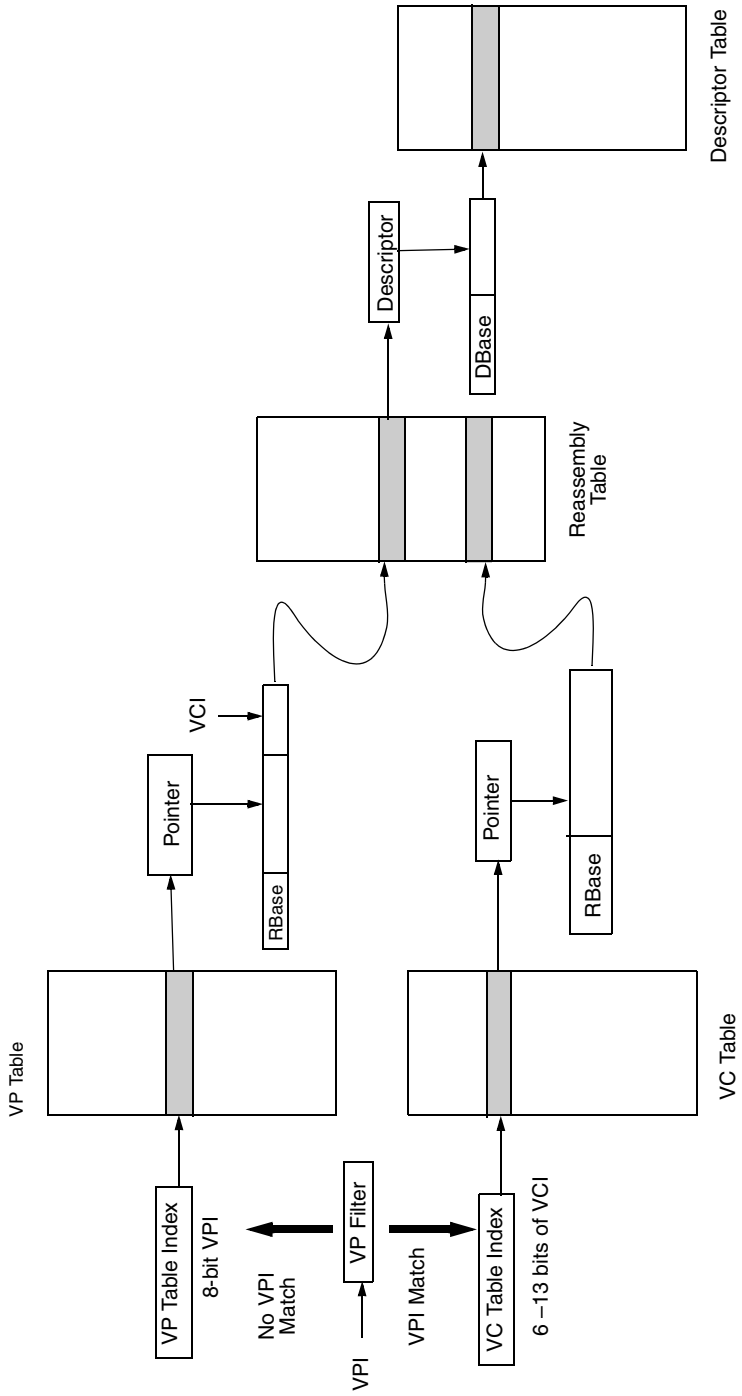

Figure 7-3. Reassembly Table Pointer/Reassembly Buffer Descriptor Table Entry Generation

This generation process allows for a flexible implementation using either a single virtual path or multiple virtual paths. If either a single virtual path or a primary virtual path is desired, then the path is entered into the VP Filter Register.

If the path of the received cell matches this VP, the Reassembly Engine does VCI-only reassembly using the VC Table. This capability provides the highest level of flexibility in the assignment of VCI bits.

If the path of the received cell does not match the VP Filter, the Reassembly Engine attempts reassembly using the VP Table.

Details of the addressing through the VC or VP Tables through the Reassembly Table are discussed in the following sections and also in the section titled *Reassembly Engine Registers on page 173* of this chapter.

VC Table

Figure 7-4 shows the organization of each VC Table entry.

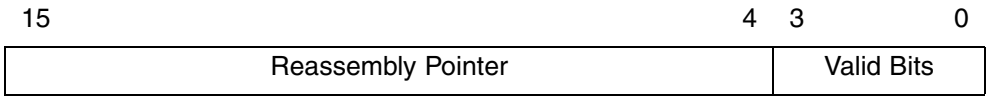

Figure 7-4. VC Table Entry

The Reassembly Pointer is concatenated with the Reassembly Table Base Address Register (REASS_BASE) to form the address for the Reassembly Table. The number of REASS_BASE bits and Reassembly Pointer bits is determined by the value of the 4 least significant bits (LSBs) of the VC Table Base Address Register (VC_LKUP_BASE), as specified in section *Reassembly Table Base Address Register on page 189* of this chapter.

The Valid Bits allow for the determination of valid versus invalid VCs.

- If the Valid Bits are set to 1111, then the (i)chipSAR+ knows that the received cell has an invalid VCI.
- If the Valid Bits are any other combination (e.g. 0000), then the entry is valid.

The VC Table is initialized by the software and read by the (i)chipSAR+ during the reassembly process. The (i)chipSAR+ never writes into the VC Table.

CAUTION

For ABR, it is critical that the reassembly pointer match the VC Index on the segmentation side. This is so the reassembly side can relate this reassembly side VC to a segmentation side VC for RM cell handling.

VP Table

Figure 7-5 shows the organization of each VP Table entry.

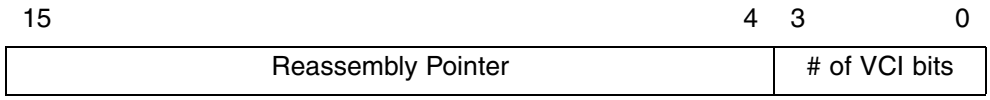

Figure 7-5. VP Table Entry

The Reassembly Pointer is concatenated with the Reassembly Table Base Address Register (REASS_BASE) and with a number of VCI bits to form the address for the Reassembly Table. The number of REASS_BASE bits, Reassembly Pointer bits, and VCI bits is determined by the value of the four least significant bits of the VC Table Base Address Register (VC_LKUP_BASE) and by the four least significant bits of the VP Table Entry.

See section *Reassembly Table Base Address Register on page 189* of this chapter for details about the meaning of the four least significant bits of the VP Table entry.

The VP Table is initialized by the software and read by the (i)chipSAR+ during the reassembly process. The (i)chipSAR+ never writes into the VP Table.

CAUTION

For ABR, it is critical that the reassembly pointer match the VC Index on the segmentation side. This is so the reassembly side can relate this reassembly side VC to a segmentation side VC for RM cell handling.

Reassembly Table

The Reassembly Table is comprised of 2-byte entries that contain information relating to each virtual connection. At startup, the contents of the Reassembly Table are initialized by the software. The (i)chipSAR+ maintains the state and the descriptor number of an active packet in the Reassembly Table.

Figure 7-6 shows the organization of Reassembly Table entries:

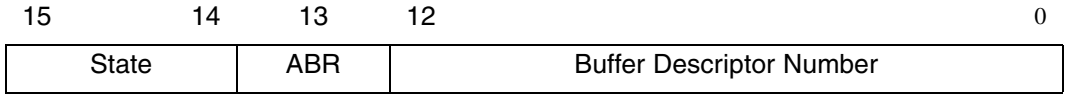

Figure 7-6. Reassembly Table Entry

Table 7-2 describes State bit settings, which indicate the state of the packet reassembly on the virtual circuit.

Table 7-2. Virtual Circuit Table Entry State Bit Settings

State Bit Setting[a]	Description
00	No AAL 5 packet in reassembly.
01	AAL 5 packet is being reassembled.
State Bit Setting[b]	**Description**
10	AAL 5 packet was terminated due to errors.
11	Virtual circuit belongs to Raw Cell traffic. All cells are loaded into the Raw Cell Queue in packet memory.

a. When the State bits are not set to “11”, the (i)chipSAR+ maintains the state of the packet reassembly.
b. When the State bits are not set to “11”, the (i)chipSAR+ maintains the state of the packet reassembly.

Bit 13, the ABR bit, specifies whether the reassembly is for an ABR VC. Setting the bit to 1 indicates ABR and setting the bit to 0 indicates non-ABR. The hardware looks at this bit to determine whether it needs to access the ABR VC Table for EFCI handling.

The buffer descriptor number is the index of the buffer descriptor associated with the packet in the reassembly process.

ABR VC Table

Figure 7-7 represents the ABR VC Table entries. The reassembly pointer is used as an index into the ABR VC Table for each ABR VC's 32-byte entry.

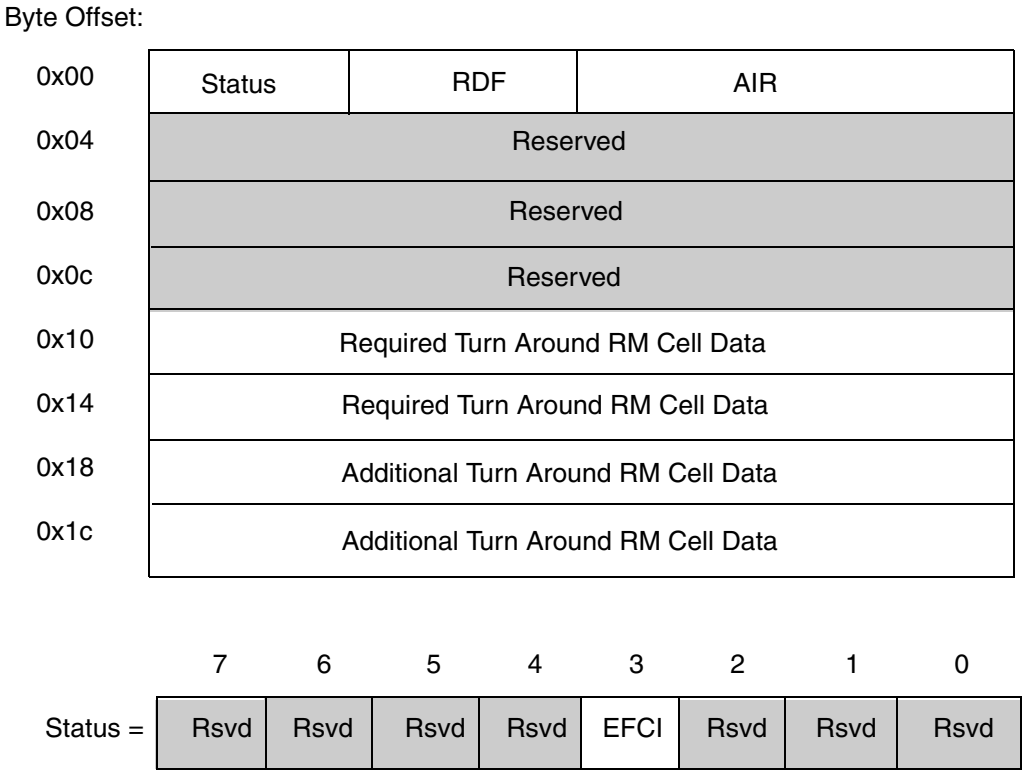

Figure 7-7. ABR VC Table Entry

Note that the host writes the VC Table entries only when the VC is set up. During normal VC operation, these entries do not need to be, and should not be, written by the host.

Note, also, that the ABR VC Table is in addition to the VC Table. ABR VCs require both entries.

Table 7-3 describes the non-reserved VC Table entry fields for ABR.

Table 7-3. Non-reserved ABR VC Table Entry Fields

Field	Description
Status	Status. Only one bit of the STATUS Field is used: EFCI—Hardware sets this bit whenever a data cell is received with the EFCI codepoint set in the header. Hardware resets the bit whenever a data cell is received without the EFCI codepoint set in the header. Software should initialize this bit to 0.
RDF	Rate Decrease Factor. RDF in this table is the exponent of the power of 2 for the RDF in the ATM Traffic Management 4.0 Specification. For example, if RDF in the specification is 1/16, RDF in this table is 4.

Table 7-3. Non-reserved ABR VC Table Entry Fields (cont)

Field	Description
AIR	Additive Increase Rate. AIR is the 16-bit floating point representation of the product of PCR and RIF i.e. AIR is added to ACR if the received backward RM Cell indicates a rate increase is required.
Required Turn Around RM CELL Data	Required Turn Around RM CELL. These eight octets are used for storage of received forward RM Cell data awaiting turn around. The data includes the protocol ID, message type, ER, CCR, and MCR fields of the received forward RM cell. Software should initialize these octets to 0.
Additional Turn Around RM Cell Data	Additional Turn Around RM Cell Data. These eight octets are used for storage of received forward RM Cell data awaiting turn around. The data is optional and includes the eight contiguous octets in the received forward RM Cell starting at the location specified by the XTRA_RM_OFFSET register. Software should initialize these octets to 0.

Free Descriptor Queue

The Free Descriptor Queue is used to queue the free descriptors for packet reassembly. When the (i)chipSAR+ receives a packet, it fetches a descriptor from this queue.

The Free Descriptor Queue is defined by the four registers in the Reassembly Engine of the (i)chipSAR+:

- FREEQ_ST_ADR (Free Descriptor Queue Starting Address)
- FREEQ_ED_ADR (Free Descriptor Queue Ending Address)
- FREEQ_RD_PTR (Free Descriptor Queue Read Pointer)
- FREEQ_WR_PTR (Free Descriptor Queue Write Pointer)

Figure 7-8 on page 168 shows the operational states of the Free Descriptor Queue. The addressing of the queues is explained in the section titled *Communication Queue Base Address Register on page 197* of this chapter.

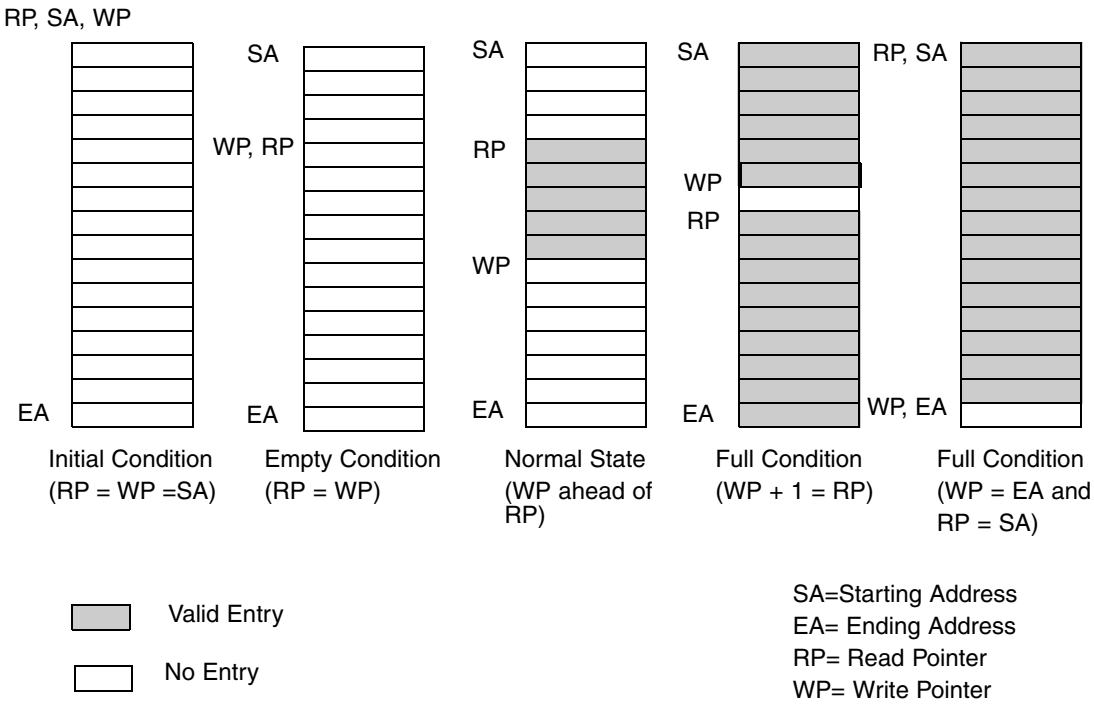

Figure 7-8. Queue Operational States

The sequence of operations to load the Free Descriptor Queue is as follows:

1. Check that the Free Descriptor Queue is large enough to hold all of the free buffer descriptor numbers.
2. Read the Free Descriptor Queue Write Pointer (FREEQ_WR_PTR) from the Reassembly Engine of the (i)chipSAR+.
3. Store the descriptor number of the free buffer descriptor for packet reassembly in the queue location in the control memory pointed to by the write pointer.

4. Increment the write pointer to point to the next location in the queue. (Take care of the wrap-around condition.) Multiple descriptors can be added to the Free Descriptor Queue by repeating Steps 3 and 4. After all the descriptors have been written into the Free Descriptor Queue, go to Step 5.

5. Store the incremented write pointer in the FREEQ_WR_PTR Register in the Reassembly Engine of the (i)chipSAR+.

Note that the size of the Free Descriptor Queue must be at least one greater than the maximum number of entries that can be written to the queue. The (i)chipSAR+ does not handle the error case where this queue could be overwritten.

Figure 7-9 shows the format of each Free Descriptor Queue entry.

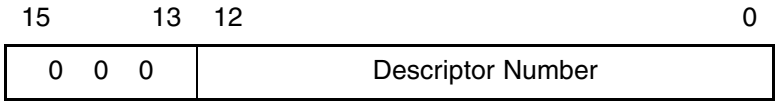

Figure 7-9. Free Descriptor Queue Entry

Packet Complete Queue

The Packet Complete Queue is used by the Reassembly Engine of the (i)chipSAR+ to queue the descriptors of reassembled packets. When this queue becomes non-empty, the (i)chipSAR+ generates a maskable interrupt to the host. The descriptors are read from this queue, and the received packet is processed by software. The following Reassembly Engine (i)chipSAR+ registers store Packet Complete Queue addresses and pointers:

- PCQ_ST_ADR (Packet Complete Queue Starting Address)
- PCQ_ED_ADR (Packet Complete Queue Ending Address)
- PCQ_RD_PTR (Packet Complete Queue Read Pointer)
- PCQ_WR_PTR (Packet Complete Queue Write Pointer)

Figure 7-8 on page 168 shows the operational states of the Packet Complete Queue. The queue locations are addressed as shown in the section titled *Communication Queue Base Address Register on page 197* of this chapter. *Figure 7-10* shows the format of each Packet Complete Queue entry.

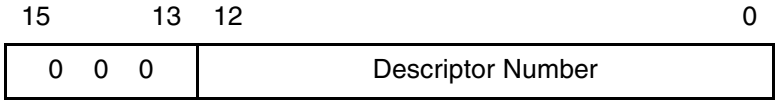

Figure 7-10. Packet Complete Queue Entry

The sequence of operations used to unload the Packet Complete Queue is similar to the sequence used to unload the Transmit Complete Queue.

1. Check that the Packet Complete Queue is not empty.
2. Read the Packet Complete Queue read and write pointers (PCQ_RD_PTR, PCQ_WR_PTR respectively) from the Reassembly Engine of the (i)chipSAR+.
3. If the read pointer is not equal to the write pointer, read the descriptor number of the reassembled packet from the queue location in the control memory pointed by the read pointer; otherwise skip to Step 5.
4. Increment the read pointer to point to the location of the next entry in the queue. (Take care of wrap-around condition.) Return to Step 3.
5. Store the incremented read pointer to the Packet Complete Queue Read Pointer (PCQ_RD_PTR) in the Reassembly Engine of the (i)chipSAR+.
6. Read the Packet Complete Queue Write Pointer (PCQ_WR_PTR) from the Reassembly Engine of the (i)chipSAR+. If the write pointer is not equal to the read pointer, return to Step 3.

Note that the size of the Packet Complete Queue must be at least one greater than the maximum number of entries that can be written to the queue. The (i)chipSAR+ does not handle the error case where this queue could be overwritten.

Exception Queue

The Exception Queue is used by the Reassembly Engine of the (i)chipSAR+ to transfer exception errors to the software entity. The Exception Queue is used to transfer error conditions that cannot be associated with a packet. Error conditions that can be associated with a descriptor of a packet are transferred through the Descriptor Status bits.

When this queue becomes non-empty, the (i)chipSAR+ generates a maskable interrupt to the host. The software reads the descriptors from this queue, and processes the received exception.

The following Reassembly Engine (i)chipSAR+ registers store Exception Queue addresses and pointers:

- EXCP_Q_ST_ADR (Exception Queue Starting Address)
- EXCP_Q_ED_ADR (Exception Queue Ending Address)
- EXCP_Q_RD_PTR (Exception Queue Read Pointer)
- EXCP_Q_WR_PTR (Exception Queue Write Pointer)

Figure 7-8 on page 168 shows the operational states of the Exception Queue. The queue locations are addressed as shown in the section titled *Communication Queue Base Address Register on page 197* of this chapter.

The following sequence of operations used to unload the Exception Queue is similar to the sequence of operations used to unload the Packet Complete Queue, except that each entry in the Exception Queue is four bytes long:

1. Check that the Exception Queue is not empty.
2. Read the Exception Queue read and write pointers (EXCP_Q_RD_PTR, EXCP_Q_WR_PTR respectively) from the Reassembly Engine of the (i)chipSAR+.
3. If the read pointer is not equal to the write pointer, read the descriptor number of the reassembled packet from the queue location in the control memory that the read pointer points to; otherwise skip to Step 5.
4. Increment the read pointer to point to the location of the next entry in the queue (note any wrap-around condition). Return to Step 3.
5. Store the incremented read pointer to the Packet Complete Queue Read Pointer (PCQ_RD_PTR) in the (i)chipSAR+.
6. Read the Exception Queue Write Pointer (PCQ_WR_PTR) from the (i)chipSAR+. If the write pointer does not equal the read pointer, return to Step 3.

Figure 7-11 shows the contents of the entries in the Exception Queue.

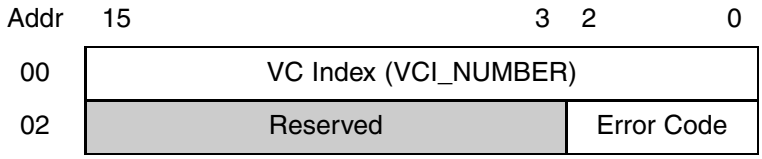

Figure 7-11. Exception Queue Entry

Note that the VC Index listed in the Exception Queue entry is the lowest 16 bits of the address of the Reassembly Table. Host software can mask out the upper bits that correspond to base address bits.

Table 7-4 describes the Exception Queue error code settings.

Table 7-4. Error Code Explanations for Exception Queue

Error Code Setting	Description
011	Reserved.
101	No buffers available (packet dropped).
110	Invalid VCI (cell received on invalid VC).
111	Invalid VPI (cell received on invalid VP).

Reassembly Engine Registers

All (i)chipSAR+ Reassembly Engine registers are accessed as 16 bits. Bit 0 is the least significant bit. All reserved and unused bits in these registers must be set to 0 for proper operation unless otherwise stated. *Table 7-5* shows the Reassembly Engine (i)chipSAR+ registers used for chip setup. The *Reset Value* column in the table specifies default reset/power-up values of the registers.

Table 7-5. Reassembly Engine (i)chipSAR+ Internal Registers

Register Name	Addr (Hex)	Description	Type[a]	Reset Value
MODE_REG	00	Mode Register	R/W	0000H
PROTOCOL_ID	01	ABR RM Cell Protocol ID Register	R/W	0001H
MASK_REG	02	Interrupt Mask Register	R/W	FFFFH
INTR_STATUS_REG	03	Interrupt Status Register	R/O	0000
DRP_PKT_CNTR	04	Dropped Packet Counter Register (cleared when read)	R/W	0000H
ERR_CNTR	05	Error Counter Register (cleared when read)	R/W	0000H
RAW_BASE_ADR	08	Raw Cell Queue Base Address Register	R/W	0000H
CELL_CTR0	0C	Received Cell Count 0 Register (cleared when read)	R/W	0000H
CELL_CTR1	0D	Received Cell Count 1 Register (cleared when read)	R/W	0000H
COMMAND_REG	0F	Command Register	W/O	N/A
DESC_BASE	10	Descriptor Table Base Address Register	R/W	0000H
VC_LKUP_BASE	11	VC Lookup Table Base Address Register	R/W	0000H
REASS_BASE	12	Reassembly Table Base Address Register	R/W	0000H
QUEUE_BASE	13	Communication Queue Base Address Register	R/W	0000H
PKT_TM_CNT	16	Packet Timeout Count Register	R/W	0000H
TMOUT_RANGE	17	Timeout Index Range Register (Range of reassembly IDs for timeout)	R/W	0000H
INTRVL_CNTR	18	Packet Aging Interval Counter	R/W	0000H
TMOUT_INDX	19	Timeout Index Register (Index of packet being tested for aging)	R/W	0000H
VP_LKUP_BASE	1C	VP Table Base Address Register	R/W	0000H
VP_FILTER	1D	VP Filter Register	R/W	0000H
ABR_LKUP_BASE	1E	ABR VC Table Base Address Register	R/W	0000H
FREEQ_ST_ADR	24	Free Descriptor Queue Starting Address Register	R/W	0000H

Table 7-5. Reassembly Engine (i)chipSAR+ Internal Registers (cont)

Register Name	Addr (Hex)	Description	Type[a]	Reset Value
FREEQ_ED_ADR	25	Free Descriptor Queue Ending Address Register	R/W	0000H
FREEQ_RD_PTR	26	Free Descriptor Queue Read Pointer Register	R/W	0000H
FREEQ_WR_PTR	27	Free Descriptor Queue Write Pointer Register	R/W	0000H
PCQ_ST_ADR	28	Packet Complete Queue Starting Address Register	R/W	0000H
PCQ_ED_ADR	29	Packet Complete Queue Ending Address Register	R/W	0000H
PCQ_RD_PTR	2A	Packet Complete Queue Read Pointer Register	R/W	0000H
PCQ_WR_PTR	2B	Packet Complete Queue Write Pointer Register	R/W	0000H
EXCP_Q_ST_ADR	2C	Exception Queue Starting Address Register	R/W	0000H
EXCP_Q_ED_ADR	2D	Exception Queue Ending Address Register	R/W	0000H
EXCP_Q_RD_PTR	2E	Exception Queue Read Pointer Register	R/W	0000H
EXCP_Q_WR_PTR	2F	Exception Queue Write Pointer Register	R/W	0000H
CC_FIFO_ST_ADR	34	Raw Cell Queue Starting Address Register	R/W	0000H
CC_FIFO_ED_ADR	35	Raw Cell Queue Ending Address Register	R/W	0000H
CC_FIFO_RD_PTR	36	Raw Cell Queue Read Pointer Register	R/W	0000H
CC_FIFO_WR_PTR	37	Raw Cell Queue Write Pointer Register	R/W	0000H
STATE_REG	38	State Register	R/O	8F7FH
BUF_SIZE	42	Buffer Size Register	R/W	0000H
XTRA_RM_OFFSET	44	Offset of the additional turned around RM cell bytes from the end of the required bytes	R/W	0000H
DRP_PKT_CNTR_NC	84	Dropped Packet Counter Register (Don't auto-clear on read)	R/W	0000H
ERR_CNTR_NC	85	Error Counter Register (Don't auto-clear on read)	R/W	0000H
CELL_CTR0_NC	8C	Cell Counter 0 Register (Don't auto-clear on read)	R/W	0000H
CELL_CTR1_NC	8D	Cell Counter 1 Register (Don't auto-clear on read)	R/W	0000H

a. R/W—Read and Write; R/O—Read Only; W/O—Write Only

Mode Register

Figure 7-12 shows the (i)chipSAR+ Reassembly Engine Mode Register (MODE_REG). This register is used to set up the operating modes of the chip's Reassembly Engine.

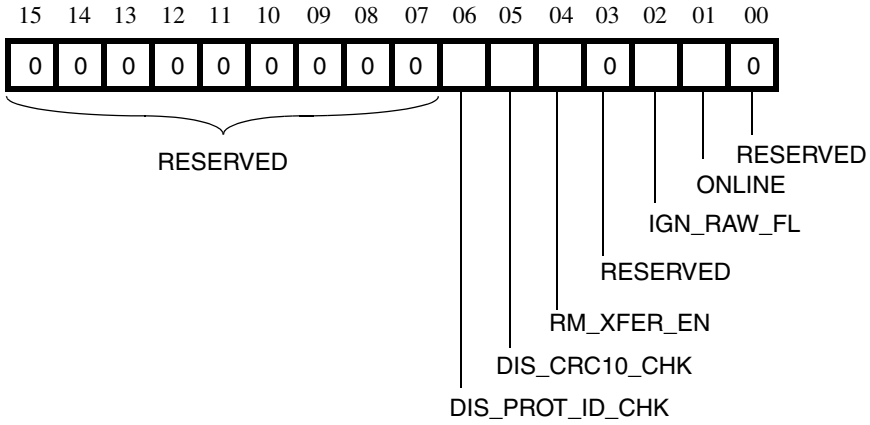

Figure 7-12. Mode Register Bit Positions

Table 7-6 describes the bits of the Mode Register:

Table 7-6. Mode Register

Bit	Description
15:07 RESERVED	Reserved. Don't care on writes. Always read as 0.
06 DIS_PROT_ID_CHK	Discard Protocol ID Check. 1 - Accept all received RM Cells. 0 - Accept received RM Cells only if the PROTOCOL_ID in cells matches the PROTOCOL_ID register.
05 DIS_CRC10_CHK	Discard CRC-10 Check. 1 - Accept all RM Cells and OAM F5 Cells regardless of CRC-10. 0 - Discard RM Cells and OAM F5 Cells if CRC-10 check fails.
04 RM_XFER_EN	RM Transfer Enable. 1 - Send all RM Cells to Raw Cell Queue (diagnostics only). 0 - Do not send RM Cells to Raw Cell Queue (normal mode).
03 RESERVED	Reserved. Don't care on writes. Always read as 0.
02 *IGN_RAW_FL	Ignore Raw Cell Full Flag. If this bit is set to 1, the Raw Cell/RM Cell queue is overwritten with new data even if the queue full condition is reached, ignoring the FULL flag. 1 – Ignore Raw Cell/RM Cell queue full flag. 0 – Do not Ignore Raw Cell/RM Cell queue full flag.

Table 7-6. Mode Register (cont)

Bit	Description
01 ONLINE	Online. This bit controls the reassembly process. Upon either hard or soft reset, the (i)chipSAR+ defaults to the offline mode so that the host can set up the (i)chipSAR+ properly. 1 – (i)chipSAR+ is on-line. 0 – Schedule (i)chipSAR+ to enter off-line gracefully. Programming this bit to 0 during the middle of a reassembly cycle forces the Reassembly Engine of the (i)chipSAR+ to gracefully enter the offline state after completing the current cycle(s) and sets the OFF_LINE bit in the State Register. Further reassembly and control memory accesses are inhibited during the offline state and resume only after being reprogrammed to be online.
00 RESERVED	Reserved. Don't care on writes. Always read as 0.

ABR RM Cell Protocol ID Register

Figure 7-13 shows the Reassembly Engine ABR RM Cell Protocol ID Register (PROTOCOL_ID). This register holds the protocol ID for the (i)chipSAR+ to check against the protocol ID in received RM Cells:

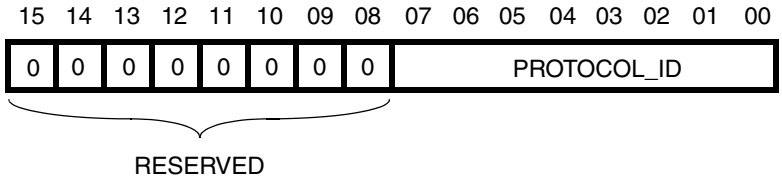

Figure 7-13. ABR RM Cell Protocol ID Register

Interrupt Mask Register

The (i)chipSAR+ Interrupt Mask Register (MASK_REG) is written by the host/processor to prevent the generation of interrupts caused by events recorded in the Status Register. Each Interrupt Mask Register bit, when set, masks out the interrupt generated by the corresponding bit in the Status Register. *Figure 7-14* shows the Interrupt Mask Register bit positions.

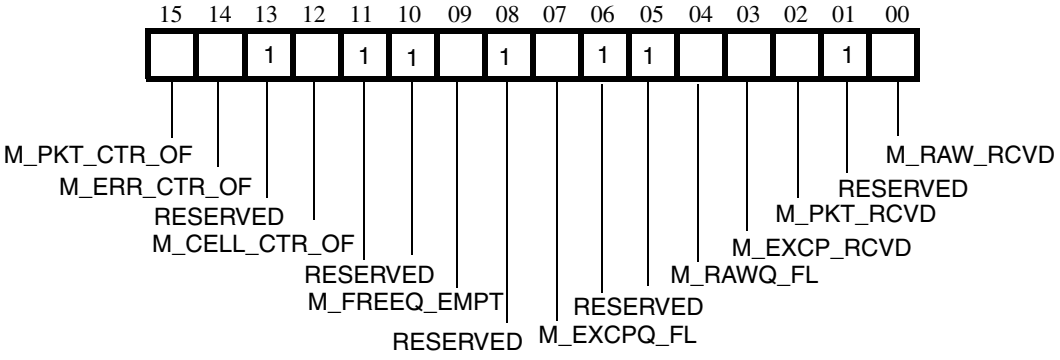

Figure 7-14. Interrupt Mask Register

Table 7-7 describes the Reassembly Engine Interrupt Mask Register bits.

Table 7-7. Interrupt Mask Register

Bit	Description
15 M_PKT_CTR_OF	Mask Packet Counter Overflow. 1 – Mask interrupt generation because of the PKT_CTR_OF bit in the Interrupt Status Register. 0 – Unmask interrupt generation because of the PKT_CTR_OF bit.
14 M_ERR_CTR_OF	Mask Error Counter Overflow. 1 – Mask interrupt generation because of the ERR_CTR_OF bit in the Interrupt Status Register. 0 – Unmask interrupt generation because of the ERR_CTR_OF bit.
13 RESERVED	Reserved. This bit must be set to 1.
12 M_CELL_CTR_OF	Mask Cell Counter Overflow. 1 – Mask interrupt generation for cell counter overflow. 0 – Unmask interrupt generation for cell counter overflow.
11:10 RESERVED	Reserved. These bits must be set to 1.
09 M_FREEQ_EMPT	Mask Free Descriptor Queue Empty. 1 – Mask interrupt generation because of the FREEQ_EMPT bit in the Interrupt Status Register. 0 – Unmask interrupt generation because of the FREEQ_EMPT bit.

Table 7-7. Interrupt Mask Register (cont)

Bit	Description
08 RESERVED	Reserved. This bit must be set to 1.
07 M_EXCPQ_FL	Mask Exception Queue Full. 1 – Mask interrupt generation because of the EXCPQ_FL_I bit in the Interrupt Status Register. 0 – Unmask interrupt generation because of the EXCPQ_FL_I bit.
06:05 RESERVED	Reserved. These bits must be set to 1.
04 M_RAWQ_FL	Mask Raw Cell Queue Full. 1 – Mask interrupt generation because of the RAWQ_FL_I bit in the Interrupt Status Register. 0 – Unmask interrupt generation because of the RAWQ_FL_I bit.
03 M_EXCP_RCVD	Mask Exception Received. 1 – Mask interrupt generation because of the EXCP_RCVD bit in the Interrupt Status Register. 0 – Unmask interrupt generation because of the EXCP_RCVD bit.
02 M_PKT_RCVD	Mask Packet Received. 1 – Mask interrupt generation because of the PKT_RCVD bit in the Interrupt Status Register. 0 – Unmask interrupt generation because of the PKT_RCVD bit.
01 RESERVED	Reserved. This bit must be set to 1.
00 M_RAW_RCVD	Mask Raw Cell Received. 1 – Mask interrupt generation because of the RAW_RCVD bit in the Interrupt Status Register. 0 – Unmask interrupt generation because of the RAW_RCVD bit.

Interrupt Status Register

The Interrupt Status Register (INTR_STATUS_REG) contains sticky bits, which are set by certain events. Every bit in the Status Register is associated with a bit in the Mask Register. If a particular mask bit is set to 0, then the corresponding status bit, when set, results in the generation of an interrupt. If a particular mask bit is set to 1, then the corresponding status bit does not result in the generation of an interrupt, but is set when the event occurs. The status bits are cleared when read by the host. *Figure 7-15* shows the (i)chipSAR+ Reassembly Engine Interrupt Status Register bits:

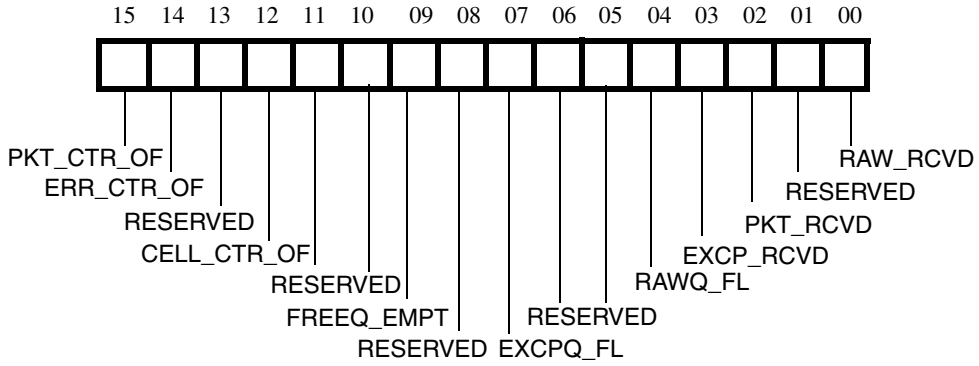

Figure 7-15. Interrupt Status Register Bit Positions

Table 7-8 describes the Reassembly Engine Interrupt Status Register bits.

Table 7-8. Interrupt Status Register

Bit	Description
15 PKT_CTR_OF	Packet Counter Overflow. This bit is set when the Dropped Packet Counter (DRP_PKT_CNTR) overflows.
14 ERR_CTR_OF	Error Counter Overflow. This bit is set when the error counter (ERR_CNTR) overflows.
13 RESERVED	Reserved. This bit must be masked out and ignored.
12 CELL_CTR_OF	Cell Counter Overflow. This bit is set when the 32-bit cell counter overflows.
11 RESERVED	Reserved. These bits must be masked out and ignored.
09 FREEQ_EMPT	Free Descriptor Queue Empty. This bit, when set, indicates that the Free Descriptor Queue was empty and that one or more packets were due to this condition. This condition occurs when an AAL5 packet is received and the queue is empty.
08 RESERVED	Reserved. This bit must be masked out and ignored.

Table 7-8. Interrupt Status Register (cont)

Bit	Description
07 EXCPQ_FL	Exception Queue Full. This bit, when set, indicates that the Exception Queue was full and that one or more errors were unreported as a result of this condition.
06:05 RESERVED	Reserved. These bits must be masked out and ignored.
04 RAWQ_FL	Raw Cell Queue Full. This bit, when set, indicates that the Raw Cell Queue was full and that one or more raw cells were dropped as a result of this condition. This bit should be masked out and ignored when the IGN_RAW_FL mode bit is set in the Interrupt Mask Register.
03 EXCP_RCVD	Exception Received. This bit, when set, indicates that an exception error has occurred, causing the Exception Queue to not be empty. The Exception Queue must become empty before this bit can be set again.
02 PKT_RCVD	Packet Received. This bit, when set, indicates that a descriptor is written into the Packet Complete Queue, causing the queue to not be empty. The Packet Complete Queue must become empty before this bit can be set again.
01 RESERVED	Reserved.
00 RAW_RCVD	Raw Cell Received. This bit, when set, indicates that a raw cell, OAM F5 cell, or RM Cell has been received by the (i)chipSAR+ and has been placed in the Raw Cell Queue in packet memory, causing the queue to not be empty. The Raw Cell Queue must become empty before this bit can be set again.

Dropped Packet Counter Register

The 16-bit Dropped Packet Counter Register (DRP_PKT_CNTR) indicates the number of packets dropped by the (i)chipSAR+ Reassembly Engine because of a shortage of available free buffers in packet memory. This counter is cleared when read by the host at address DRP_PKT_CNTR, and not cleared when read at address DRP_PKT_CNTR_NC. It can be written by the host in controlled write mode for diagnostic purposes. A bit is set in the Status Register when this counter overflows.

Error Counter Register

The 16-bit Error Counter Register (ERR_CNTR) indicates the number of cells received with either header checksum errors or CRC-10 (in the case of OAM cells). This counter is cleared when read by the processor at address ERR_CNTR and not cleared when read at address ERR_CNTR_NC. It can be written by the processor in controlled write mode for diagnostic purposes. A bit is set in the Status Register when this counter overflows.

Raw Cell Queue Base Address Register

The Raw Cell Queue can be located in either packet memory or host memory, depending on the reassembly mode being used. The contents of the Base Address Register (RAW_BASE_ADR) register depends on the location of the queue.

On-Board Reassembly Mode

In on-board reassembly mode, the Raw Cell Queue is located in packet memory. In this mode, the Raw Cell Queue Base Address Register contains the upper three bits of the Raw Cell Queue location in the packet memory. These bits, when concatenated with the 16-bit write pointer of the corresponding queue, point to the beginning of the cell entry. *Figure 7-16* illustrates the Raw Cell Queue address generation.

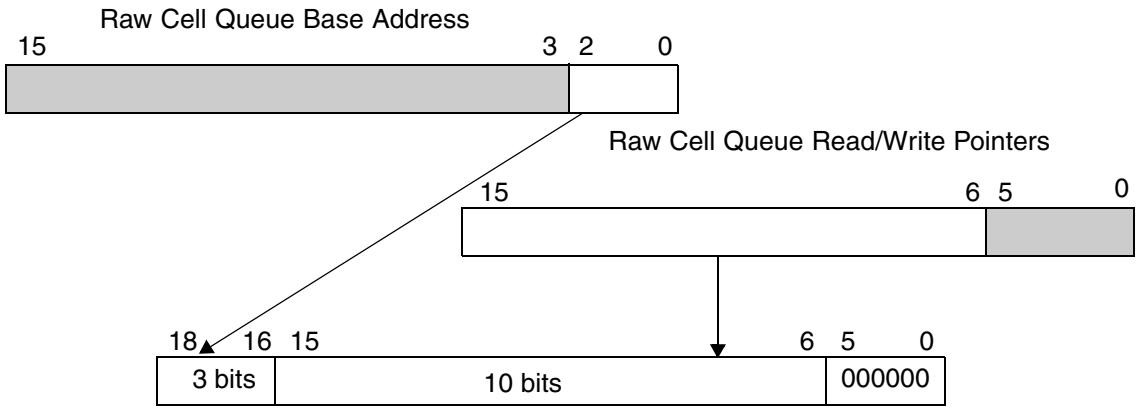

Figure 7-16. Raw Cell Queue Address Generation (On-Board Reassembly Mode)

Cell FIFO Mode

In Cell FIFO mode, the Raw Cell Queue is located in host memory. In this mode, the Raw Cell Queue Base Address Register contains the upper sixteen bits of the Raw Cell Queue location in host memory. These bits, when concatenated with the 16-bit write pointer of the corresponding queue, point to the beginning of the cell entry. *Figure 7-17* illustrates the Raw Cell Queue address generation.

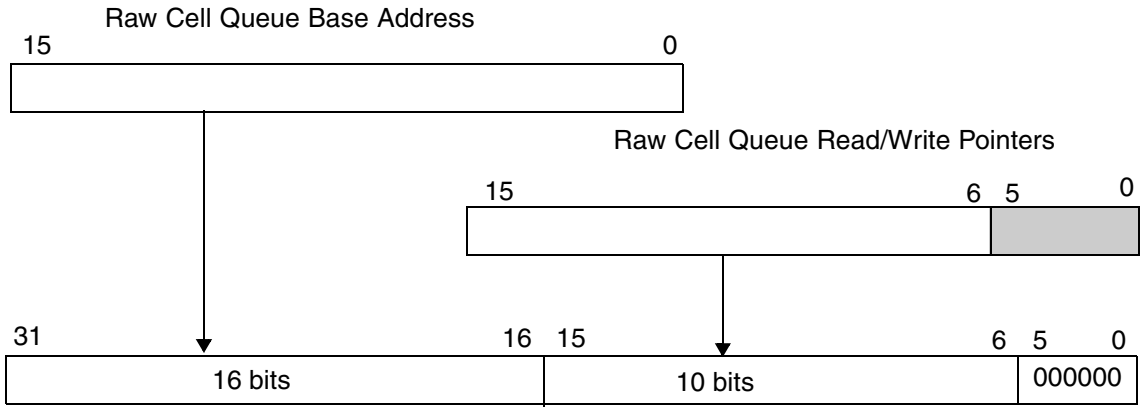

Figure 7-17. Raw Cell Queue Address Generation (Cell-FIFO Mode)

Received Cell Count Register

The 32-bit Received Cell Count Register is accessible by reading two 16-bit register locations (CELL_CTR0, CELL_CTR1). This register counts the total number of non-error and non-idle cells received. Non-error cells are defined as cells that have neither header checksum errors nor CRC-10 errors (in the case of OAM cells).

The counters are auto-cleared when they are read using addresses CELL_CTR0 and CELL_CTR1, and not cleared when read using addresses CELL_CTR0_NC and CELL_CTR1_NC.

The sum of the Error Counter Register and the Cell Counter Register equals the total number of non-idle cells received.

Command Register

The Command Register (COMMAND_REG) is a write-only register used to issue software commands to the Reassembly Engine of the (i)chipSAR+ as follows:

- Reset the (i)chipSAR+ Reassembly Engine when xx55 is written to this register.
- Reset the (i)chipSAR+ Reassembly Engine state machine when xxAA is written to this register.
- Reset the (i)chipSAR+ Reassembly Engine Dropped Packet Counter Register (DRP_PKT_CNTR) when xxF1 is written to this register.
- Reset the (i)chipSAR+ Reassembly Engine Error Counter Register (ERR_CNTR) when xxF2 is written to this register.
- Reset the (i)chipSAR+ Reassembly Engine Cell Counter 0/1 Register (CELL_CNTR 0/1) when xxF8 is written to this register.
- Reset all the above counters when xxFF is written to this register.

Values written to this register other than those listed are ignored.

Descriptor Table Base Address Register

The Descriptor Table Base Address Register (DESC_BASE) contains the base address of the Buffer Descriptor Table. The value in the descriptor base address, when concatenated with the descriptor number and the table offset, points to the descriptor entry in control memory. Each Buffer Descriptor Table entry is 32 bytes long. The offset determines the specific 16-bit word within the table entry being accessed. *Figure 7-18* shows how the Descriptor Table address is formed.

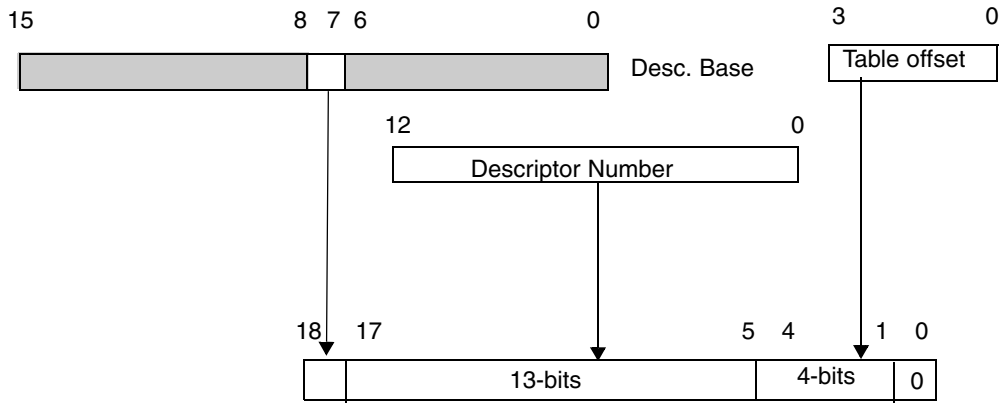

Figure 7-18. Reassembly Descriptor Table Address Generation

VP Lookup Table Base Address Register

The VP Lookup Table Base Address Register (VP_LKUP_BASE) contains the base address of the VP Table. This base address is concatenated with the 8-bit VPI field to index to the VP Table. This address generation is illustrated in *Figure 7-19*.

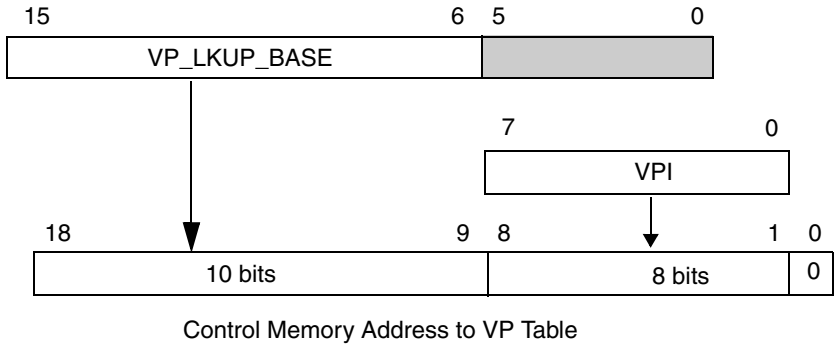

Figure 7-19. Reassembly Engine VP Lookup Table Base Address Generation

Reassembly Table Base Address Register

The method for creating the address in the Reassembly Table depends on whether VC Table reassembly or VP Table reassembly is used.

VC Table Reassembly

In the case of VC Table reassembly, the number of Reassembly Table Base Address Register (REASS_BASE) bits versus the number of Reassembly Pointer bits depends on the three least significant bits of the VC Table Base Address Register (VC_LKUP_BASE) Register. (The three least significant bits of the VC_LKUP_BASE Register define the maximum number of VCs allowed.) *Table 7-9* illustrates the Reassembly Table pointer construction for VC Table reassembly.

Table 7-9. Reassembly Table Pointer Construction for VC Table Reassembly

Bits 2, 1, and 0 of VC_LKUP_BASE Pointer	Number of REASS_BASE Register MSB Bits to be Concatenated	Number of MSBs of VC Table Reassembly Pointer	Maximum Number of VCs Possible
000	5	13	8,192
001	6	12	4,096
010	7	11	2,048
011	8	10	1,024
100	9	9	512

Figure 7-20 illustrates the Reassembly Table addressing process for VC Table reassembly.

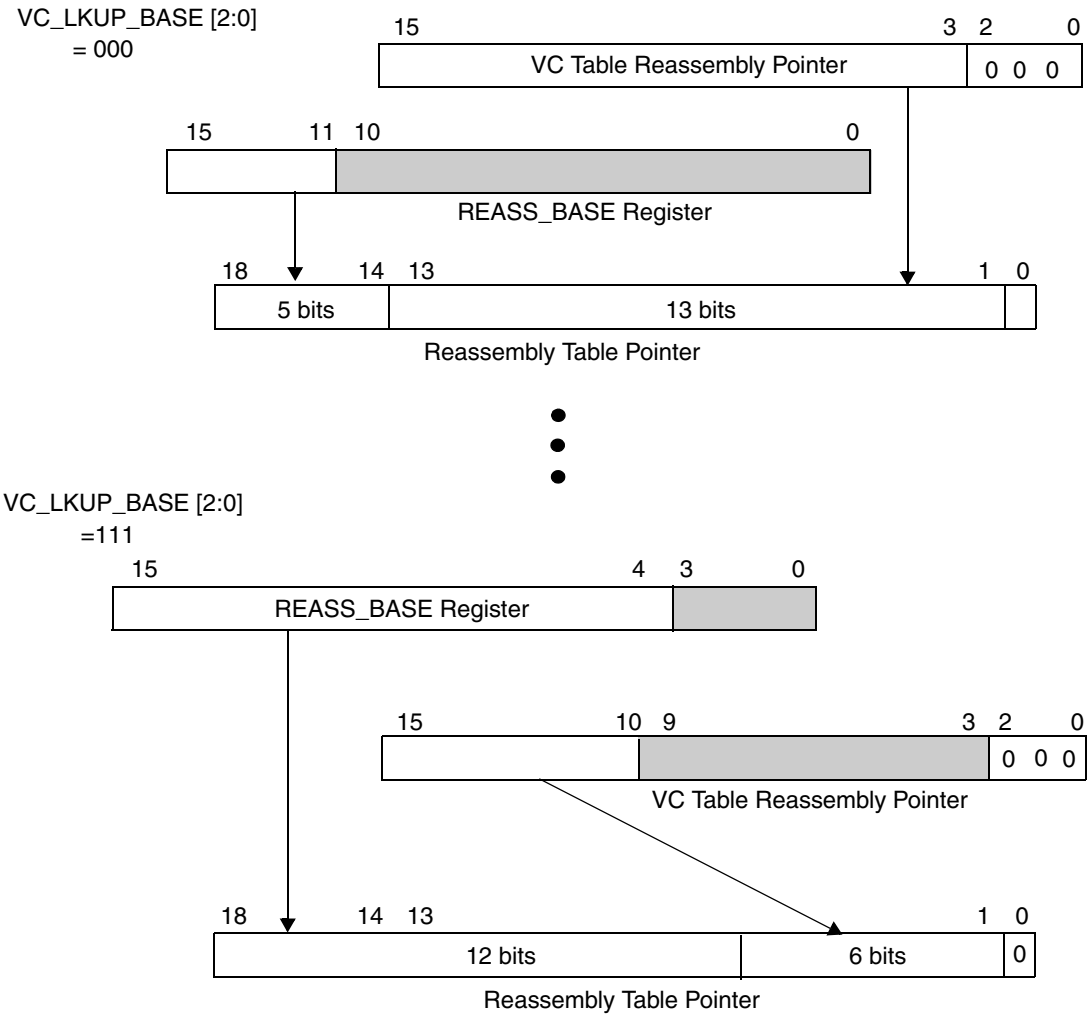

Figure 7-20. Reassembly Table Addressing For VC Table Reassembly

Note that the lower three bits of the VC Table Reassembly Pointer are always listed as 0. The hardware checks for the condition of 1111 in the lower 4 bits. If it detects 1111, an illegal VC error is noted and reassembly of the cell in question does not occur. It is assumed that the host initializes all unused VC Table Reassembly Pointers with 1111 in the lower 4 bits and that all used VC Table Reassembly Pointers have a different value (for example, x000).

CAUTION

It is critical that the reassembly pointer in the VC Table entry match the VC Index for the same VC on the segmentation side. This is so that the reassembly side can relate a received RM Cell to a segmentation VC.

VP Table Reassembly

In the case of VP Table reassembly, Reassembly Table address generation is more complex than for the VC Reassembly Table. The Reassembly Table address in this case has three components:

- Reassembly Base Register
- VP Table Reassembly Pointer
- VCI bits from the ATM header

The same mechanism used for choosing the number of Reassembly Base Register bits during VC Table reassembly is also used for VP Table reassembly. However, an additional parameter determines the number of Reassembly Pointer bits versus VCI bits.

The mechanism for determining the number of Reassembly Pointer bits versus VCI bits uses the 4 least significant bits (LSBs) of the Reassembly Pointer from the VP Table. This mechanism is illustrated in the following tables and figures:

Table 7-10. Number of REASS_BASE Register Bits for VP Table Reassembly

Bits 2, 1, and 0 of VC_LKUP_BASE Pointer	Number of REASS_BASE Bits
000	5
001	6
010	7
011	8
100	9

Number of REASS_ BASE Bits	Reassembly Pointer [3:0]	Number of RP Bits	Number of VCI Bits	Number of VCs
5	xxx0	13	0	1
5	0001	9	4	16
5	0011	8	5	32
5	0101	7	6	64
5	0111	6	7	128
5	1001	5	8	256
5	1011	4	9	512
5	1101	3	10	1024
5	1111	Invalid VP	Invalid VP	

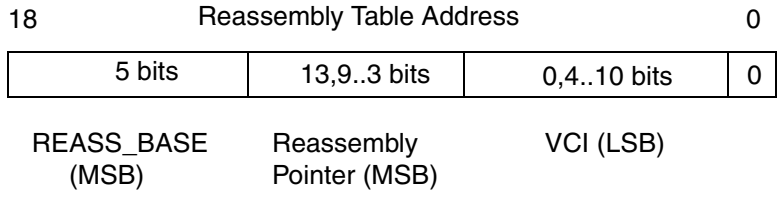

Figure 7-21. VP Reassembly Table Address Generation with Five REASS_BASE Bits

Number of REASS_ BASE Bits	Reassembly Pointer [3:0]	Number of RP Bits	Number of VCI Bits	Number of VCs
6	xxx0	12	0	1
6	0001	8	4	16
6	0011	7	5	32
6	0101	6	6	64
6	0111	5	7	128
6	1001	4	8	256
6	1011	3	9	512
6	1101	2	10	1024
6	1111	Invalid VP	Invalid VP	

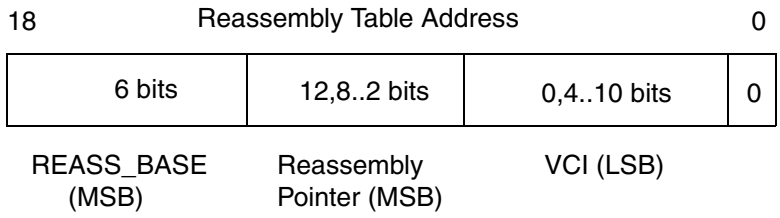

Figure 7-22. VP Reassembly Table Address Generation with Six REASS_BASE Bits

Number of REASS_ BASE Bits	Reassembly Pointer [3:0]	Number of RP Bits	Number of VCI Bits	Number of VCs
7	xxx0	11	0	1
7	0001	7	4	16
7	0011	6	5	32
7	0101	5	6	64
7	0111	4	7	128
7	1001	3	8	256
7	1011	2	9	512
7	1101	1	10	1024
7	1111	Invalid VP	Invalid VP	

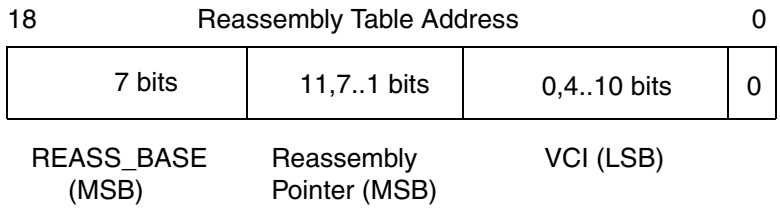

Figure 7-23. VP Reassembly Table Address Generation with Seven REASS_BASE Bits

Number of REASS_ BASE Bits	Reassembly Pointer [3:0]	Number of RP Bits	Number of VCI Bits	Number of VCs
8	xxx0	10	0	1
8	0001	6	4	16
8	0011	5	5	32
8	0101	4	6	64
8	0111	3	7	128
8	1001	2	8	256
8	1011	1	9	512
8	1101	0	10	1024
8	1111	Invalid VP	Invalid VP	

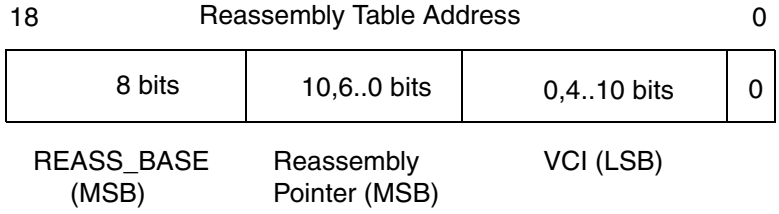

Figure 7-24. VP Reassembly Table Address Generation with Eight REASS_BASE Bits

Number of REASS_ BASE Bits	Reassembly Pointer [3:0]	Number of RP Bits	Number of VCI Bits	Number of VCs
9	xxx0	9	0	1
9	0001	6	3	8
9	0011	5	4	16
9	0101	4	5	32
9	0111	3	6	64
9	1001	2	7	128
9	1011	1	8	256
9	1101	0	9	512
9	1111	Invalid VP	Invalid VP	

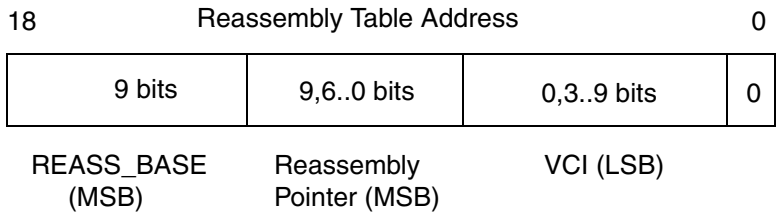

Figure 7-25. VP Reassembly Table Address Generation with Nine REASS_BASE Bits

CAUTION

The concatenation of reassembly pointer bits and VCI bits with VP reassembly must match the VC Index of the same VC on the segmentation side. This is so that the reassembly engine can relate a received RM Cell with the correct segmentation VC.

Communication Queue Base Address Register

The Communication Queue Base Address Register (QUEUE_BASE) contains the base address of the four communication queues in control memory. This base address, when concatenated with the queue read or write pointer, generates the control memory address to the appropriate queue entry. *Figure 7-26* shows the address generation.

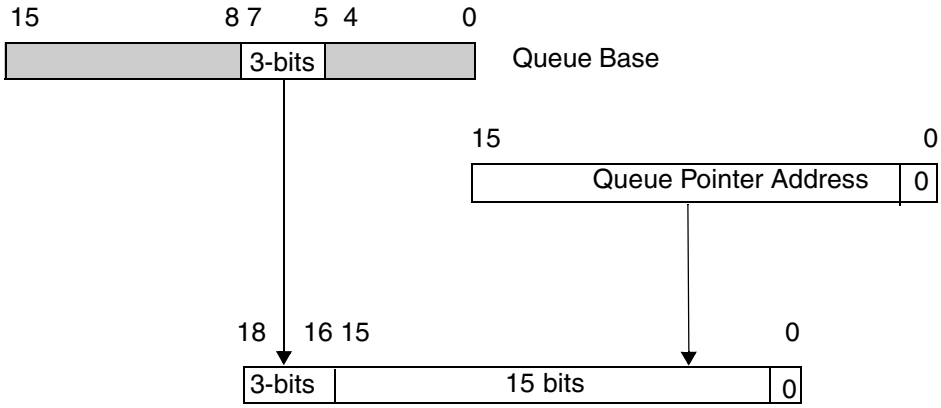

Figure 7-26. Reassembly Engine Queue Address Generation

Packet Timeout Count Register

The Packet Timeout Count Register (PKT_TM_CNT) is used for setting up the packet timeout interval. The upper and lower eight bits of this register serve different functions.

The upper eight bits initialize the Packet Timeout Counter field in the Descriptor Table when a buffer is first assigned to a packet.

The lower eight bits define the terminal count of the Packet Aging Interval Counter, which is used to determine how often a packet is aged.

Timeout Index Range Register

The Reassembly Table can be set up anywhere in memory by appropriately initializing the VP and VC tables. The Reassembly Engine needs the dimension of this table to be defined for packet aging.

The Timeout Index Range Register (TMOUT_RANGE) defines the range of the Reassembly Table that is scanned to age out packets in the middle of reassembly. The range is defined in increments of 32 locations of the Reassembly Table.

Figure 7-27 shows the Timeout Index Range Register:

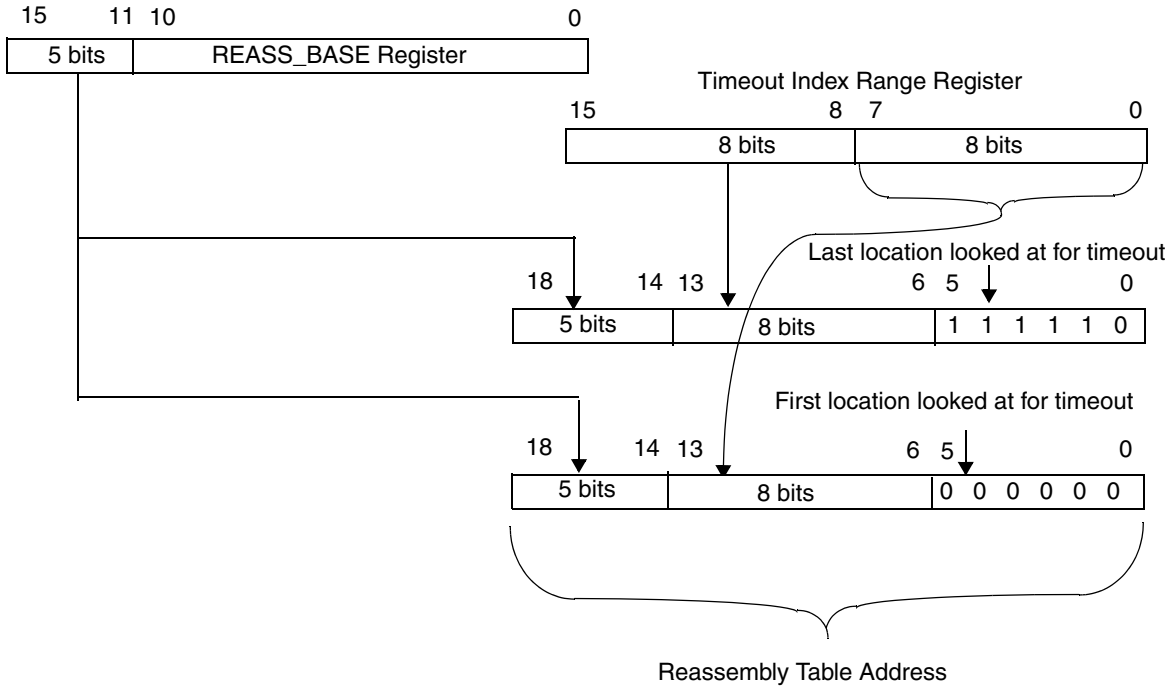

Figure 7-27. Timeout Index Range Register

Packet Aging Interval Counter Register

The Packet Aging Interval Counter Register (INTRVL_CNTR) is a free-running 12-bit counter used to age packets. The upper eight bits are refreshed from the lower eight bits of the Packet Timeout Count Register. The lower four bits are initialized to 0 when refreshed.

Whenever the Packet Aging Interval Counter overflows, the Timeout Index Register is incremented. When the Timeout Index Register is incremented, a new time stamp is applied to the active descriptor associated with the VC pointed to by the timeout index. Aging, if any, is done by incrementing the Packet Timeout Counter field in the Descriptor Table entry. An overflow of the packet timeout counter terminates the packet reassembly process of that descriptor. When the packet reassembly is terminated, the descriptor is returned to the processor by writing it into the Packet Complete Queue with an error status bit set in the Descriptor Table entry.

This counter is accessible for diagnostic purposes.

Timeout Index Register

The Timeout Index Register (TMOUT_INDX) contains the reassembly pointer whose packet is being aged. It is incremented each time the packet aging interval counter overflows. The aging process occurs only if a packet is active in the Reassembly Table.

The Timeout Index Register counts through its full 64K range. Only those values within the range of the Timeout Index Range Register are considered. This register is accessible for diagnostic purposes.

VC Table Base Address Register

The VC Table Base Address Register (VC_LKUP_BASE) contains the base address of the VC Table. The value in this VC_LKUP_BASE Register is concatenated with the lower bits of the reassembly pointer to generate the VC Table address.

The least significant three bits of this base register determines the number of bits of the base register used in generating the address. This address generation method is useful in configuring the control memory for various table sizes, and is illustrated in *Figure 7-28*.

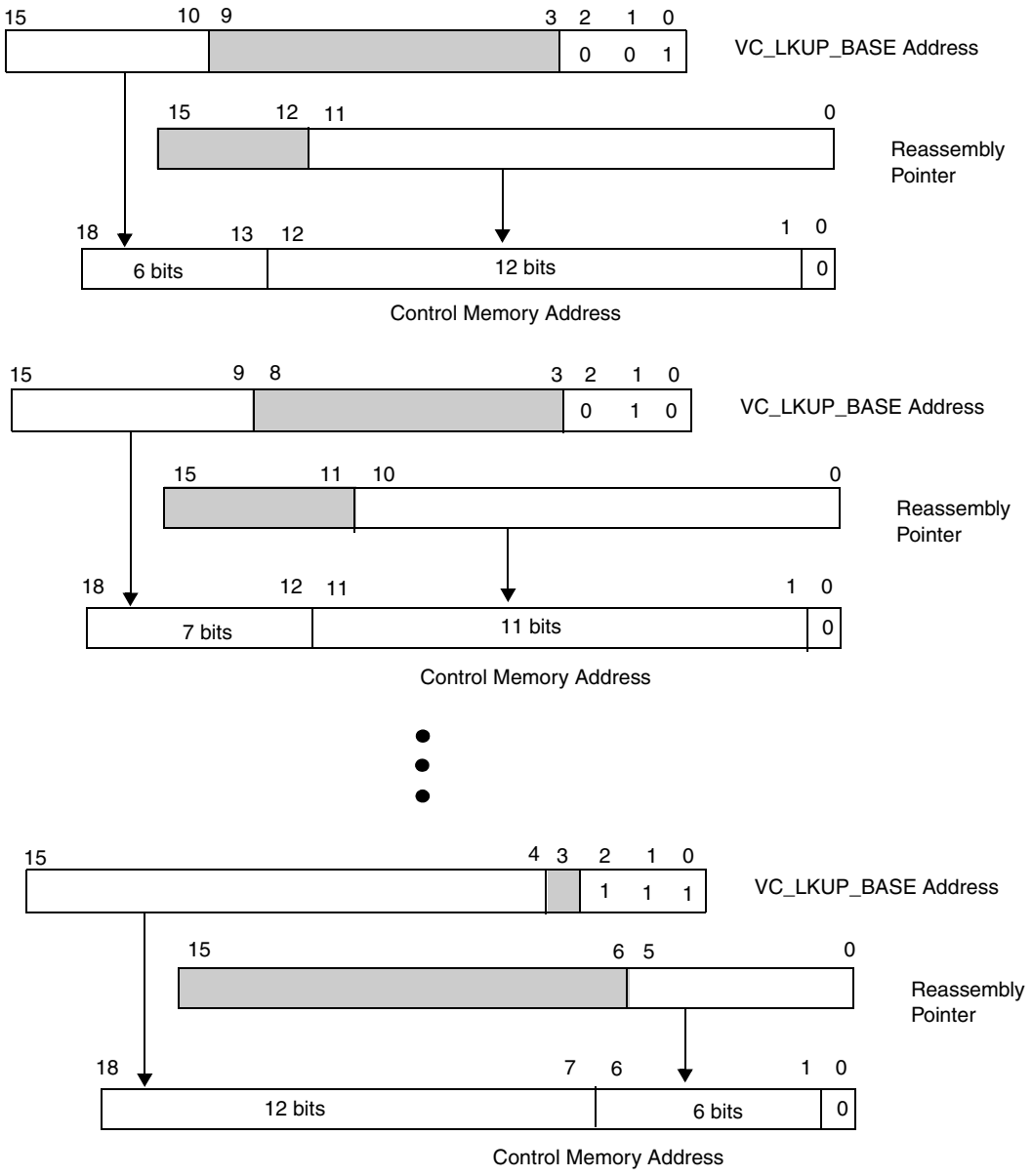

Figure 7-28. Reassembly Engine VC Table Address Generation

Table 7-11 shows the maximum number of VC Table entries for various values in the least significant three bits of the VC_LKUP_BASE Register.

Table 7-11. Number of VC Table Entries for the VC_LKUP_BASE Register

	Memory Address		
Least Significant Bits of VC_LKUP_BASE	Number of Bits Used From VC_LKUP_BASE	Number of Bits Used From VC_Index	Maximum Number of VCs Possible
001	6	12	4,096
010	7	11	2,048
011	8	10	1,024
100	9	9	512
101	10	8	256
110	11	7	128
111	12	6	64

ABR VC Table Base Address Register

The ABR VC Table Base Address Register (ABR_LKUP_BASE) contains the base address of the ABR VC Table. The value in the ABR_LKUP_BASE Register is concatenated with the lower bits of the Reassembly Pointer to generate the ABR VC Table address.

The least significant three bits of the reassembly VC Table Base Address Register (VC_LKUP_BASE) determines the number of bits of the ABR VC Table Base Register used in generating the address. This address generation method is useful in configuring the control memory for various table sizes, and is illustrated in *Figure 7-29*.

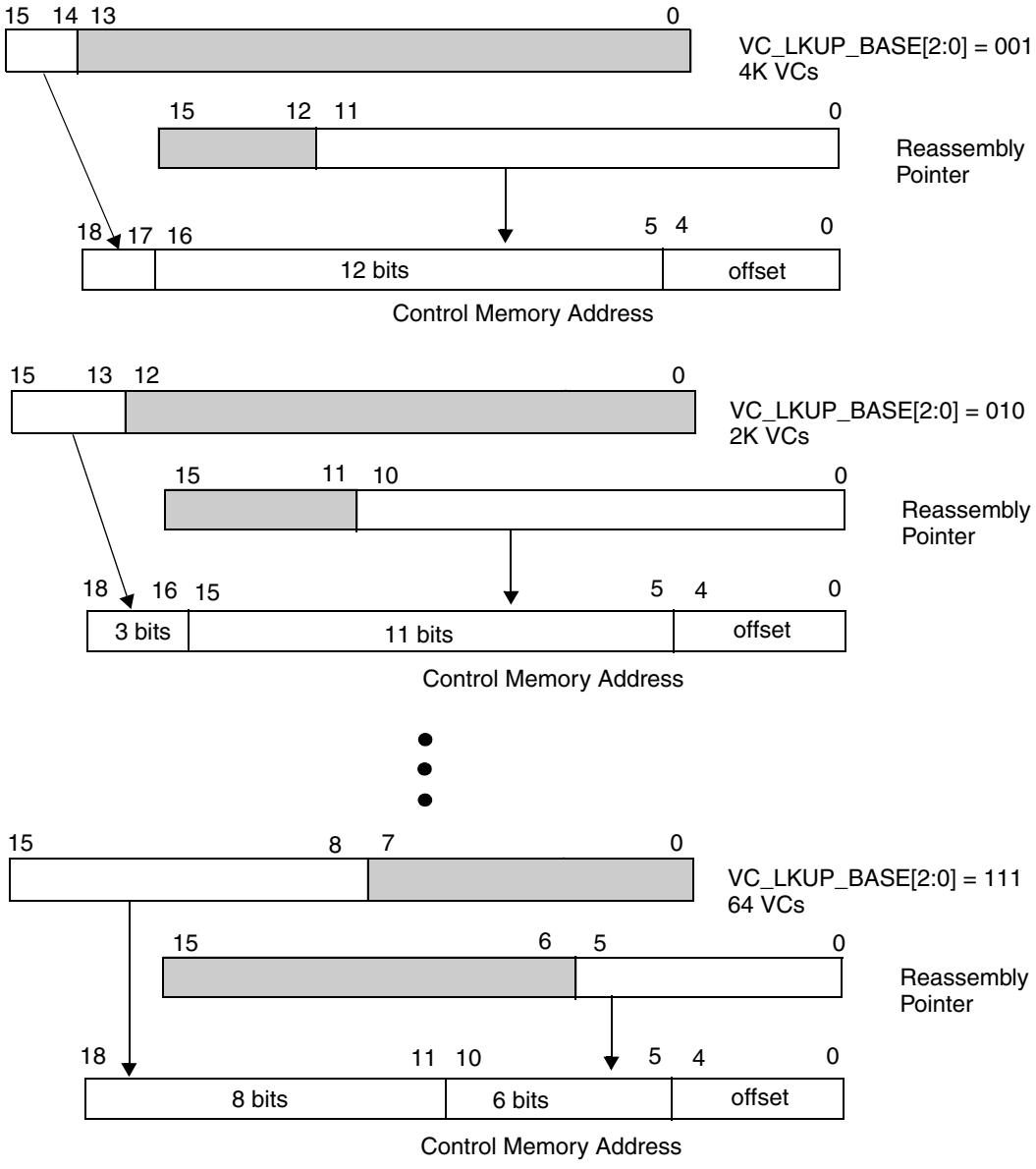

Figure 7-29. Reassembly Engine ABR VC Table Address Generation

Table 7-11 shows the maximum number of ABR VC Table entries for various values in the least significant three bits of the VC Table Base Address Register.

Table 7-12. Number of ABR VC Table Entries for the VC_LKUP_BASE Register

	Memory Address		
Least Significant Bits of VC_LKUP_BASE	Number of Bits Used From ABR_LKUP_BASE	Number of Bits Used From Reassembly Pointer	Maximum Number of VCs Possible
001	6	12	4,096
010	7	11	2,048
011	8	10	1,024
100	9	9	512
101	10	8	256
110	11	7	128
111	12	6	64

VP Filter Register

The VP Filter Register (VP_FILTER) is used to determine whether reassembly will be done on VPI/VCI or on VCI only. Depending on the result, either the VP or VC Table will be used.

The VP Filter Register uses the most significant eight bits (15:8) as mask bits, and the least significant eight bits (7:0) as compare bits. The compare bits are Exclusive NOR'ed with the VPI; this result in turn is OR'ed with the mask bits. This result is then compared with eight 1's.

If the result compares, the assembly is on VCI, if the results do not compare, the assembly is on VPI and VCI.

Figure 7-30 is a diagram of the VP Filter Register.

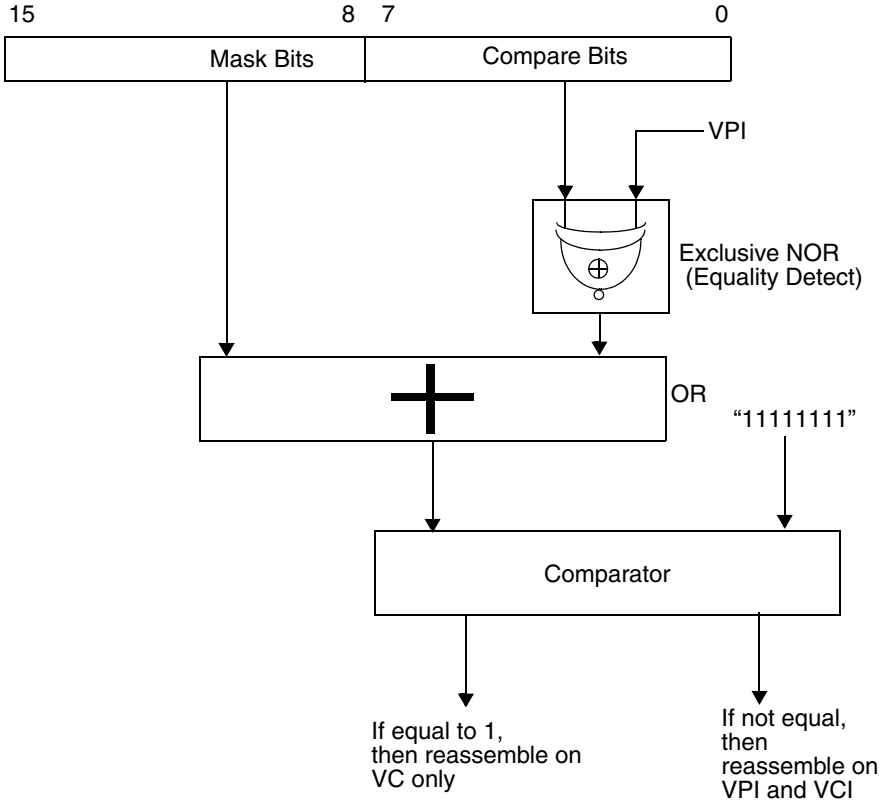

Figure 7-30. VP Filter Register

Communication Queues

The Reassembly Engine of the (i)chipSAR+ uses three queues to communicate with the software for passing descriptors:

- Free Descriptor Queue
- Packet Complete Queue
- Exception Queue

Table 7-13 identifies the address and pointer registers of these queues.

Table 7-13. Communication Queue Address and Pointer Registers

Queue	Starting Address	Ending Address	Read Pointer	Write Pointer
Free Descriptor	FREEQ_ST_ADR	FREEQ_ED_ADR	FREEQ_RD_PTR	FREEQ_WR_PTR
Packet Complete	PCQ_ST_ADR	PCQ_ED_ADR	PCQ_RD_PTR	PCQ_WR_PTR
Exception	EXCP_Q_ST_ADR	EXCP_Q_ED_ADR	EXCPQ__RD_PTR	EXCP_Q_WR_PTR

The queue length (the difference between the end address and the start address) should be programmed greater (by at least one entry) than the maximum number of descriptors used, to prevent any possible queue overflow.

Figure 7-31 shows the operational states of the queues during various conditions.

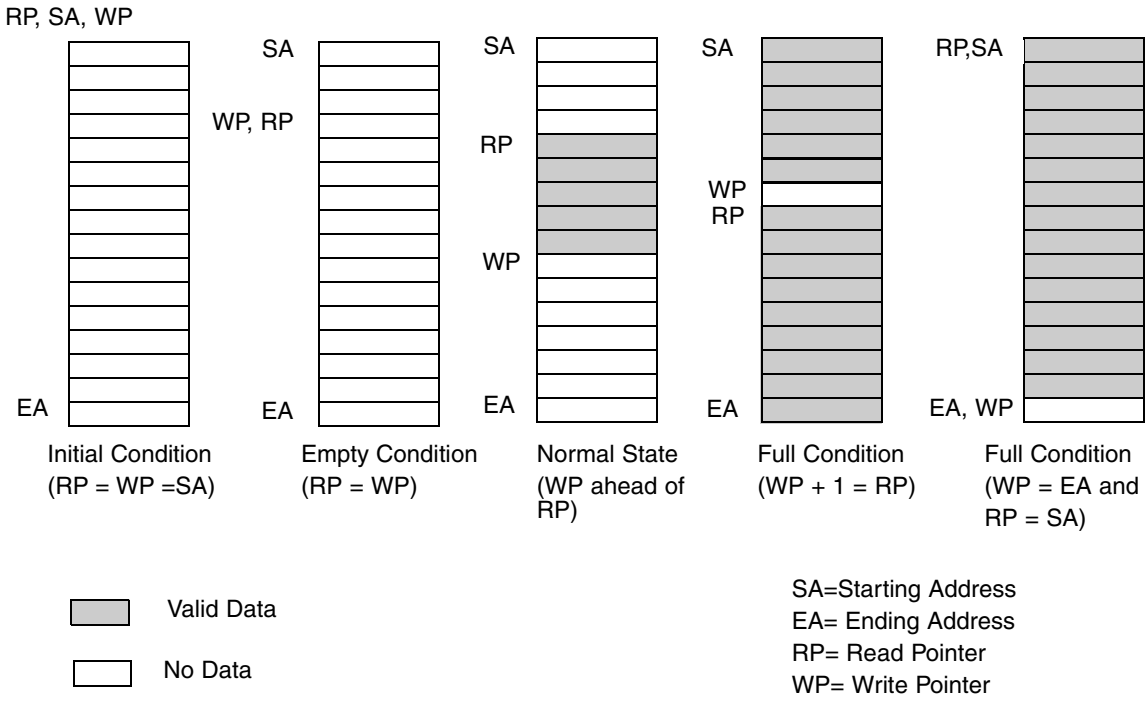

Figure 7-31. Reassembly Engine Communication Queue Operational States

The queues operate as follows:

- At initialization, the pointers should be programmed to be read pointer (RP) = write pointer (WP) = starting address (SA).
- Any time RP = WP, the queue is empty. Descriptors are added to the queue by incrementing the WP by one entry length.
- The WP wraps around when it exceeds the end address (EA). The (i)chipSAR+ automatically wraps the pointer (WP or RP) as it increments this pointer.
- Software is responsible for detecting the EA boundary and wrapping the pointer to SA.
- When the WP is one entry short of the RP, the queue full condition is reached.

Free Descriptor Queue

The Free Descriptor Queue is filled by the software with the descriptor numbers of the buffers that are available for packet reassembly.

The (i)chipSAR+ reads a descriptor from this queue whenever a packet is being reassembled.

Note that the size of this queue must be at least 1 greater than the maximum number of entries possible for the queue. The (i)chipSAR+ does not handle overflow error conditions for this queue.

Packet Complete Queue

The (i)chipSAR+ loads the Packet Complete Queue with the descriptor numbers of the buffers that contain completely reassembled packets.

Note that the size of this queue must be at least 1 greater than the maximum number of entries possible for the queue. The (i)chipSAR+ does not handle overflow error conditions for this queue.

Exception Queue

The (i)chipSAR+ loads the Exception Queue with the VCI of the cell that caused the exception.

Figure 7-32 shows an entry in the Exception Queue:

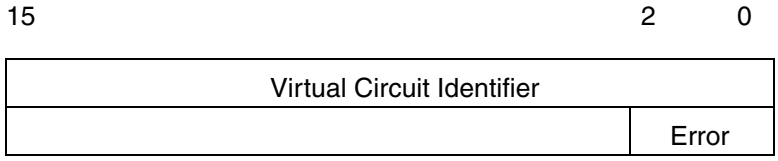

Figure 7-32. Entries in Exception Queue

The following error codes are defined in the Exception Queue:

- 000 – No error
- 001 – Not used
- 010 – Not used
- 100 – Not used
- 101 – No free descriptor available and packet has been dropped
- 110 – Cell received on invalid VC
- 111 – Cell received on invalid VPI/VCI

Raw Cell Queue Registers

The Raw Cell Queue contains the received raw cells or OAM F5 cells. The Raw Cell Queue may exist either in packet memory (in the case of on-board reassembly) or in host memory (in the case of Cell-FIFO reassembly). OAM F5 cells are indicated by the payload type field in the ATM header. This queue is similar in definition to the communication queues described on *page 207* in this chapter. This queue contains the following start and end address registers and read and write pointers:

- CC_FIFO_ST_ADR (Raw Cell Queue Starting Address)
- CC_FIFO_ED_ADR (Raw Cell Queue Ending Address)
- CC_FIFO_RD_PTR (Raw Cell Queue Read Pointer)
- CC_FIFO_WR_PTR (Raw Cell Queue Write Pointer)

Each entry is 64 bytes long, of which only the first 52 bytes are used to store the first four bytes of the header and the 48 bytes of payload associated with the raw cell.

State Register

The (i)chipSAR+ Reassembly Engine State Register (STATE_REG) bits do not generate an interrupt. These are read-only status bits indicating the dynamic states of all the queues in control memory and the cell queues in the packet memory. All bits in this register are cleared automatically when the condition that caused them to be set ends. *Figure 7-33* shows the (i)chipSAR+ Reassembly Engine State Register bits.

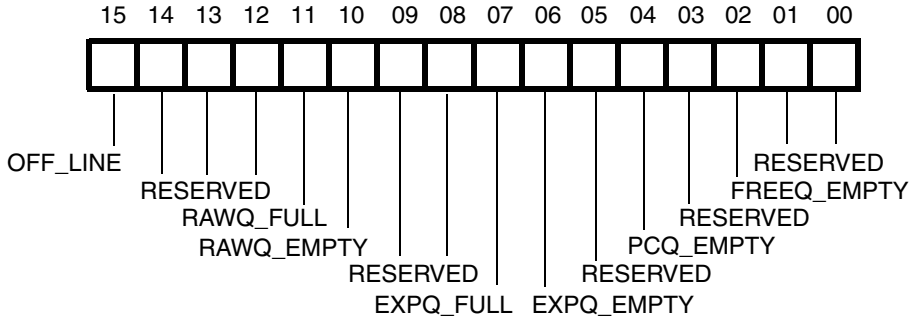

Figure 7-33. State Register

Table 7-14 describes the Reassembly Engine State Register bits.

Table 7-14. State Register

Bit	Description
15 OFF_LINE	Offline. This bit indicates the state of the (i)chipSAR+, and is set to 1 immediately after a hard or soft reset ((i)chipSAR+ is in off-line state). Packet reassembly and control memory accesses are not performed until the (i)chipSAR+ is placed back online (ON_LINE of Mode Register 0). When the processor forces the (i)chipSAR+ to enter the offline state, this bit must be polled to see if the (i)chipSAR+ has gracefully entered offline. If this bit still indicates online state, the (i)chipSAR+ is still in the middle of a reassembly cycle and has not yet gracefully entered the offline state. During online mode, this bit is set to 0.
14:12 RESERVED	Reserved. These bits must be masked out and ignored.
11 RAWQ_FULL	Raw Cell Queue Full. This bit indicates that the Raw Cell Queue full condition exists.
10 RAWQ_EMPTY	Raw Cell Queue Empty. This bit indicates that the Raw Cell Queue empty condition exists.
09:08 RESERVED	Reserved. These bits must be masked out and ignored.
07 EXCPQ_FULL	Exception Queue Full. This bit, when set, indicates that the Exception Queue full condition exists.
06 EXCPQ_EMPTY	Exception Queue Empty. This bit, when set, indicates that the Exception Queue empty condition exists.

Table 7-14. State Register (cont)

Bit	Description
05 RESERVED	Reserved. This bit must be masked out and ignored.
04 PCQ_EMPTY	Packet Complete Queue Empty. This bit, when set, indicates that the Packet Complete Queue empty condition exists.
03 RESERVED	Reserved. This bit must be masked out and ignored.
02 Q_EMPTY	Free Descriptor Queue Empty. This bit, when set, indicates that the descriptor queue empty condition exists.
01:00 RESERVED	Reserved. This bit must be masked out and ignored.

Buffer Size Register

The Buffer Size Register (BUF_SIZE) contains the size (in bytes) of the buffers in packet memory for AAL 5 circuits. It is programmed to the actual buffer size.

XTRA_RM_OFFSET Register

Figure 7-34 shows the (i)chipSAR+ Reassembly Engine Extra RM Cell Offset Register (XTRA_RM_OFFSET). This register defines the offset from the start of RM cell payload of 8 additional bytes to be turned around with forward RM cells. This functionality can be used for the turn-around of either ITU-T fields or any proprietary fields.

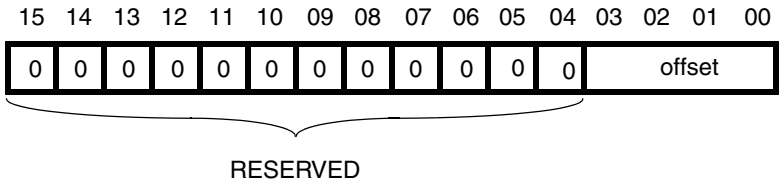

Figure 7-34. Extra RM Cell Offset Register

Table 7-6 describes the different possible combinations of the Offset field in the XTRA_RM_OFFSET Register.

Table 7-15. XTRA_RM_OFFSET Register Combinations

Offset	Description
0	No extra RM bytes (default)
2	Offset 8 bytes from start of payload
3	Offset 12 bytes from start of payload
4	Offset 16 bytes from start of payload
5	Offset 20 bytes from start of payload
6	Offset 24 bytes from start of payload
7	Offset 28 bytes from start of payload
8	Offset 32 bytes from start of payload
9	Offset 36 bytes from start of payload
all others	Illegal. May produce erroneous results.

Pinout and Pin Descriptions

8

Overview

This chapter details the pins of the (i)chipSAR+.

Pin Descriptions

Table 8-1. PCI Signal Pin Descriptions

Pin #	Pin Name	Description	I/O	Drive
13	PCLK	33 Mhz. PCI Clock	Input	
2	PINTAL	Low true PCI Interrupt A	Open Drain	PCI
102	PRSTL	Low true PCI Reset	Input	
278	PREQL	Low true PCI Request	Output[1]	PCI
3	PGNTL	Low true PCI Grant	Input	
202	PFRAMEL	Low true PCI Master Frame Signal	Bidir	PCI
203	PIRDYL	Low true PCI Initiator Ready Signal	Bidir	PCI
112	PTRDYL	Low true PCI Target Ready Signal	Bider	PCI
14	PDEVSELL	Low true PCI Device Select Signal	Bidir	PCI
107	PIDSEL	PCI Initialization Device Select	Input	
113	PSTOPL	Low true PCI Stop Signal	Bidir	PCI
15	PPERRL	Low true PCI Parity Error Signal	Bidir	PCI
114	PSERRL	Low true PCI System Error Signal	Bidir	PCI
205	PPAR	PCI Even Parity Bit	Bidir	PCI
198	PCBE3	PCI Bus Command/Byte Enable 3	Bidir	PCI
285	PCBE2	PCI Bus Command/Byte Enable 2	Bidir	PCI
288	PCBE1	PCI Bus Command/Byte Enable 1	Bidir	PCI
292	PCBE0	PCI Bus Command/Byte Enable 0	Bidir	PCI
103	PAD31	PCI Multiplexed Address/Data Bus	Bidir	PCI
195	PAD30	PCI Multiplexed Address/Data Bus	Bidir	PCI
5	PAD29	PCI Multiplexed Address/Data Bus	Bidir	PCI
196	PAD28	PCI Multiplexed Address/Data Bus	Bidir	PCI
105	PAD27	PCI Multiplexed Address/Data Bus	Bidir	PCI
6	PAD26	PCI Multiplexed Address/Data Bus	Bidir	PCI
106	PAD25	PCI Multiplexed Address/Data Bus	Bidir	PCI

Table 8-1. PCI Signal Pin Descriptions (cont)

Pin #	Pin Name	Description	I/O	Drive
281	PAD24	PCI Multiplexed Address/Data Bus	Bidir	PCI
282	PAD23	PCI Multiplexed Address/Data Bus	Bidir	PCI
199	PAD22	PCI Multiplexed Address/Data Bus	Bidir	PCI
108	PAD21	PCI Multiplexed Address/Data Bus	Bidir	PCI
9	PAD20	PCI Multiplexed Address/Data Bus	Bidir	PCI
109	PAD19	PCI Multiplexed Address/Data Bus	Bidir	PCI
284	PAD18	PCI Multiplexed Address/Data Bus	Bidir	PCI
201	PAD17	PCI Multiplexed Address/Data Bus	Bidir	PCI
110	PAD16	PCI Multiplexed Address/Data Bus	Bidir	PCI
206	PAD15	PCI Multiplexed Address/Data Bus	Bidir	PCI
289	PAD14	PCI Multiplexed Address/Data Bus	Bidir	PCI
17	PAD13	PCI Multiplexed Address/Data Bus	Bidir	PCI
116	PAD12	PCI Multiplexed Address/Data Bus	Bidir	PCI
18	PAD11	PCI Multiplexed Address/Data Bus	Bidir	PCI
117	PAD10	PCI Multiplexed Address/Data Bus	Bidir	PCI
208	PAD9	PCI Multiplexed Address/Data Bus	Bidir	PCI
209	PAD8	PCI Multiplexed Address/Data Bus	Bidir	PCI
20	PAD7	PCI Multiplexed Address/Data Bus	Bidir	PCI
119	PAD6	PCI Multiplexed Address/Data Bus	Bidir	PCI
21	PAD5	PCI Multiplexed Address/Data Bus	Bidir	PCI
120	PAD4	PCI Multiplexed Address/Data Bus	Bidir	PCI
211	PAD3	PCI Multiplexed Address/Data Bus	Bidir	PCI
121	PAD2	PCI Multiplexed Address/Data Bus	Bidir	PCI
212	PAD1	PCI Multiplexed Address/Data Bus	Bidir	PCI
23	PAD0	PCI Multiplexed Address/Data Bus	Bidir	PCI
213	BUSMODE1	PMC BUSMODE1 Signal	Output[1]	PCI
24	BUSMODE2	PMC BUSMODE2 Signal	Input	
123	BUSMODE3	PMC BUSMODE3 Signal	Input	
124	BUSMODE4	PMC BUSMODE4 Signal	Input	

1. During normal operation these pins are outputs; however, they are implemented as bidirectionals for NAND Tree testing.

Table 8-2. Packet Memory Signal Pin Descriptions

Pin #	Pin Name	Description	I/O	Drive
27	SRWE3L	Low true Packet Memory Byte 3 Write Enable	Output[1]	12 ma.
125	SRWE2L	Low true Packet Memory Byte 2 Write Enable	Output[1]	12 ma.
28	SRWE1L	Low true Packet Memory Byte 1 Write Enable	Output[1]	12 ma.
215	SRWE0L	Low true Packet Memory Byte 0 Write Enable	Output[1]	12 ma.
29	SROEL	Low true Packet Memory Output Enable	Output[1]	12 ma.
30	SRDATA31	Packet Memory Data Bus	Bidir	3 ma.
128	SRDATA30	Packet Memory Data Bus	Bidir	3 ma.
31	SRDATA29	Packet Memory Data Bus	Bidir	3 ma.
217	SRDATA28	Packet Memory Data Bus	Bidir	3 ma.
129	SRDATA27	Packet Memory Data Bus	Bidir	3 ma.
32	SRDATA26	Packet Memory Data Bus	Bidir	3 ma.
300	SRDATA25	Packet Memory Data Bus	Bidir	3 ma.
219	SRDATA24	Packet Memory Data Bus	Bidir	3 ma.
301	SRDATA23	Packet Memory Data Bus	Bidir	3 ma.
220	SRDATA22	Packet Memory Data Bus	Bidir	3 ma.
131	SRDATA21	Packet Memory Data Bus	Bidir	3 ma.
34	SRDATA20	Packet Memory Data Bus	Bidir	3 ma.
132	SRDATA19	Packet Memory Data Bus	Bidir	3 ma.
35	SRDATA18	Packet Memory Data Bus	Bidir	3 ma.
303	SRDATA17	Packet Memory Data Bus	Bidir	3 ma.
222	SRDATA16	Packet Memory Data Bus	Bidir	3 ma.
36	SRDATA15	Packet Memory Data Bus	Bidir	3 ma.
304	SRDATA14	Packet Memory Data Bus	Bidir	3 ma.
223	SRDATA13	Packet Memory Data Bus	Bidir	3 ma.
134	SRDATA12	Packet Memory Data Bus	Bidir	3 ma.
305	SRDATA11	Packet Memory Data Bus	Bidir	3 ma.
224	SRDATA10	Packet Memory Data Bus	Bidir	3 ma.
135	SRDATA9	Packet Memory Data Bus	Bidir	3 ma.
39	SRDATA8	Packet Memory Data Bus	Bidir	3 ma.
40	SRDATA7	Packet Memory Data Bus	Bidir	3 ma.

1. During normal operation these pins are outputs; however, they are implemented as bidirectionals for NAND Tree testing.

Table 8-2. Packet Memory Signal Pin Descriptions (cont)

Pin #	Pin Name	Description	I/O	Drive
137	SRDATA6	Packet Memory Data Bus	Bidir	3 ma.
226	SRDATA5	Packet Memory Data Bus	Bidir	3 ma.
307	SRDATA4	Packet Memory Data Bus	Bidir	3 ma.
227	SRDATA3	Packet Memory Data Bus	Bidir	3 ma.
308	SRDATA2	Packet Memory Data Bus	Bidir	3 ma.
42	SRDATA1	Packet Memory Data Bus	Bidir	3 ma.
139	SRDATA0	Packet Memory Data Bus	Bidir	3 ma.
43	SRADDR17	Packet Memory Longword Address Bus	Output[1]	12 ma.
140	SRADDR16	Packet Memory Longword Address Bus	Output[1]	12 ma.
229	SRADDR15	Packet Memory Longword Address Bus	Output[1]	12 ma.
310	SRADDR14	Packet Memory Longword Address Bus	Output[1]	12 ma.
230	SRADDR13	Packet Memory Longword Address Bus	Output[1]	12 ma.
311	SRADDR12	Packet Memory Longword Address Bus	Output[1]	12 ma.
45	SRADDR11	Packet Memory Longword Address Bus	Output[1]	12 ma.
142	SRADDR10	Packet Memory Longword Address Bus	Output[1]	12 ma.
46	SRADDR9	Packet Memory Longword Address Bus	Output[1]	12 ma.
143	SRADDR8	Packet Memory Longword Address Bus	Output[1]	12 ma.
232	SRADDR7	Packet Memory Longword Address Bus	Output[1]	12 ma.
47	SRADDR6	Packet Memory Longword Address Bus	Output[1]	12 ma.
233	SRADDR5	Packet Memory Longword Address Bus	Output[1]	12 ma.
48	SRADDR4	Packet Memory Longword Address Bus	Output[1]	12 ma.
145	SRADDR3	Packet Memory Longword Address Bus	Output[1]	12 ma.
49	SRADDR2	Packet Memory Longword Address Bus	Output[1]	12 ma.
146	SRADDR1	Packet Memory Longword Address Bus	Output[1]	12 ma.
50	SRADDR0	Packet Memory Longword Address Bus	Output[1]	12 ma.

1. During normal operation these pins are outputs; however, they are implemented as bidirectionals for NAND Tree testing.

Table 8-3. Transmit Control Memory Signal Pin Descriptions

Pin #	Pin Name	Description	I/O	Drive
51	FCA18	Segmentation Control Memory Byte Address	Output[1]	9 ma.
52	FCA17	Segmentation Control Memory Byte Address	Output[1]	9 ma.
148	FCA16	Segmentation Control Memory Byte Address	Output[1]	9 ma.
53	FCA15	Segmentation Control Memory Byte Address	Output[1]	9 ma.
149	FCA14	Segmentation Control Memory Byte Address	Output[1]	9 ma.
54	FCA13	Segmentation Control Memory Byte Address	Output[1]	9 ma.
316	FCA12	Segmentation Control Memory Byte Address	Output[1]	9 ma.
237	FCA11	Segmentation Control Memory Byte Address	Output[1]	9 ma.
55	FCA10	Segmentation Control Memory Byte Address	Output[1]	9 ma.
238	FCA9	Segmentation Control Memory Byte Address	Output[1]	9 ma.
151	FCA8	Segmentation Control Memory Byte Address	Output[1]	9 ma.
56	FCA7	Segmentation Control Memory Byte Address	Output[1]	9 ma.
152	FCA6	Segmentation Control Memory Byte Address	Output[1]	9 ma.
57	FCA5	Segmentation Control Memory Byte Address	Output[1]	9 ma.
319	FCA4	Segmentation Control Memory Byte Address	Output[1]	9 ma.
240	FCA3	Segmentation Control Memory Byte Address	Output[1]	9 ma.
320	FCA2	Segmentation Control Memory Byte Address	Output[1]	9 ma.
241	FCA1	Segmentation Control Memory Byte Address	Output[1]	9 ma.
154	FCD15	Segmentation Control Memory Data Bus	Bidir	3 ma.
59	FCD14	Segmentation Control Memory Data Bus	Bidir	3 ma.
155	FCD13	Segmentation Control Memory Data Bus	Bidir	3 ma.
60	FCD12	Segmentation Control Memory Data Bus	Bidir	3 ma.
322	FCD11	Segmentation Control Memory Data Bus	Bidir	3 ma.
243	FCD10	Segmentation Control Memory Data Bus	Bidir	3 ma.
61	FCD9	Segmentation Control Memory Data Bus	Bidir	3 ma.
323	FCD8	Segmentation Control Memory Data Bus	Bidir	3 ma.
244	FCD7	Segmentation Control Memory Data Bus	Bidir	3 ma.
62	FCD6	Segmentation Control Memory Data Bus	Bidir	3 ma.
63	FCD5	Segmentation Control Memory Data Bus	Bidir	3 ma.
245	FCD4	Segmentation Control Memory Data Bus	Bidir	3 ma.

1. During normal operation these pins are outputs; however, they are implemented as bidirectionals for NAND Tree testing.

Table 8-3. Transmit Control Memory Signal Pin Descriptions (cont)

Pin #	Pin Name	Description	I/O	Drive
158	FCD3	Segmentation Control Memory Data Bus	Bidir	3 ma.
64	FCD2	Segmentation Control Memory Data Bus	Bidir	3 ma.
159	FCD1	Segmentation Control Memory Data Bus	Bidir	3 ma.
65	FCD0	Segmentation Control Memory Data Bus	Bidir	3 ma.
247	FOEL	Segmentation Control Memory Data Enable	Output[1]	9 ma.
66	FWEL	Segmentation Control Memory Write Enable	Output[1]	9 ma.

1. During normal operation these pins are outputs; however, they are implemented as bidirectionals for NAND Tree testing.

Table 8-4. Receive Control Memory Signal Pin Descriptions

Pin #	Pin Name	Description	I/O	Drive
327	RCA18	Reassembly Control Memory Byte Address	Output[1]	9 ma.
67	RCA17	Reassembly Control Memory Byte Address	Output[1]	9 ma.
162	RCA16	Reassembly Control Memory Byte Address	Output[1]	9 ma.
68	RCA15	Reassembly Control Memory Byte Address	Output[1]	9 ma.
163	RCA14	Reassembly Control Memory Byte Address	Output[1]	9 ma.
250	RCA13	Reassembly Control Memory Byte Address	Output[1]	9 ma.
329	RCA12	Reassembly Control Memory Byte Address	Output[1]	9 ma.
251	RCA11	Reassembly Control Memory Byte Address	Output[1]	9 ma.
330	RCA10	Reassembly Control Memory Byte Address	Output[1]	9 ma.
70	RCA9	Reassembly Control Memory Byte Address	Output[1]	9 ma.
165	RCA8	Reassembly Control Memory Byte Address	Output[1]	9 ma.
71	RCA7	Reassembly Control Memory Byte Address	Output[1]	9 ma.
166	RCA6	Reassembly Control Memory Byte Address	Output[1]	9 ma.
253	RCA5	Reassembly Control Memory Byte Address	Output[1]	9 ma.
72	RCA4	Reassembly Control Memory Byte Address	Output[1]	9 ma.
254	RCA3	Reassembly Control Memory Byte Address	Output[1]	9 ma.
73	RCA2	Reassembly Control Memory Byte Address	Output[1]	9 ma.
168	RCA1	Reassembly Control Memory Byte Address	Output[1]	9 ma.
333	RCD15	Reassembly Control Memory Data Bus	Bidir	3 ma.
169	RCD14	Reassembly Control Memory Data Bus	Bidir	3 ma.

Table 8-4. Receive Control Memory Signal Pin Descriptions (cont)

Pin #	Pin Name	Description	I/O	Drive
75	RCD13	Reassembly Control Memory Data Bus	Bidir	3 ma.
170	RCD12	Reassembly Control Memory Data Bus	Bidir	3 ma.
76	RCD11	Reassembly Control Memory Data Bus	Bidir	3 ma.
77	RCD10	Reassembly Control Memory Data Bus	Bidir	3 ma.
171	RCD9	Reassembly Control Memory Data Bus	Bidir	3 ma.
78	RCD8	Reassembly Control Memory Data Bus	Bidir	3 ma.
172	RCD7	Reassembly Control Memory Data Bus	Bidir	3 ma.
79	RCD6	Reassembly Control Memory Data Bus	Bidir	3 ma.
335	RCD5	Reassembly Control Memory Data Bus	Bidir	3 ma.
258	RCD4	Reassembly Control Memory Data Bus	Bidir	3 ma.
173	RCD3	Reassembly Control Memory Data Bus	Bidir	3 ma.
80	RCD2	Reassembly Control Memory Data Bus	Bidir	3 ma.
259	RCD1	Reassembly Control Memory Data Bus	Bidir	3 ma.
174	RCD0	Reassembly Control Memory Data Bus	Bidir	3 ma.
81	ROEL	Reassembly Control Memory Output Enable	Output[1]	9 ma.
175	RWEL	Reassembly Control Memory Write Enable	Output[1]	9 ma.

1. During normal operation these pins are outputs; however, they are implemented as bidirectionals for NAND Tree testing.

Table 8-5. Physical Interface Signal Pin Descriptions

Pin #	Pin Name	Description	I/O	Drive
176	PHYRDL	Phy Device Slave Read Strobe	Output[1]	3 ma.
261	PHYWRL	Phy Device Slave Write Strobe	Output[1]	3 ma.
338	PHYINTL	Phy Device Interrupt Input	Input	
339	TXDATA7	Utopia Transmit Data Bus	Output[1]	3 ma.
262	TXDATA6	Utopia Transmit Data Bus	Output[1]	3 ma.
177	TXDATA5	Utopia Transmit Data Bus	Output[1]	3 ma.
84	TXDATA4	Utopia Transmit Data Bus	Output[1]	3 ma.
178	TXDATA3	Utopia Transmit Data Bus	Output[1]	3 ma.
85	TXDATA2	Utopia Transmit Data Bus	Output[1]	3 ma.
341	TXDATA1	Utopia Transmit Data Bus	Output[1]	3 ma.
264	TXDATA0	Utopia Transmit Data Bus	Output[1]	3 ma.

Table 8-5. Physical Interface Signal Pin Descriptions (cont)

Pin #	Pin Name	Description	I/O	Drive
180	TXSOC	Utopia Transmit Start of Cell Indication	Output[1]	3 ma.
265	TXENBL	Utopia Transmit Valid Data Indication	Output[1]	3 ma.
179	TXCLAV	Utopia Transmit Cell Available Indication	Input	
266	RXDATA7	Utopia Receive Data Bus	Input	
181	RXDATA6	Utopia Receive Data Bus	Input	
89	RXDATA5	Utopia Receive Data Bus	Input	
182	RXDATA4	Utopia Receive Data Bus	Input	
90	RXDATA3	Utopia Receive Data Bus	Input	
183	RXDATA2	Utopia Receive Data Bus	Input	
268	RXDATA1	Utopia Receive Data Bus	Input	
345	RXDATA0	Utopia Receive Data Bus	Input	
346	RXSOC	Utopia Receive Start of Cell Indication	Input	
184	RXENBL	Utopia Receive Valid Data Indication	Output[1]	3 ma.
92	RXCLAV	Utopia Receive Cell Available Indication	Input	
185	CELLCOMP	Actual Cell Transmission Time Input	Input	

1. During normal operation these pins are outputs; however, they are implemented as bidirectionals for NAND Tree testing.

Table 8-6. Expansion ROM Interface Signal Pin Descriptions

Pin #	Pin Name	Description	I/O	Drive
93	EPCSL	Low true PCI Expansion ROM Chip Select	Output[1]	3 ma.
186	EPOEL	Low true PCI Expansion ROM Output Enable	Output[1]	3 ma.
271	EPWEL	Low true PCI Expansion ROM Write Enable	Output[1]	3 ma.

1. During normal operation these pins are outputs; however, they are implemented as bidirectionals for NAND Tree testing.

Table 8-7. Serial EEPROM Interface Signal Pin Descriptions

Pin #	Pin Name	Description	I/O	Drive
94	EEEN	PCI EEPROM Enable Bit	Input	
187	EEDI	PCI EEPROM Data In	Output[1]	3 ma.

1. During normal operation these pins are outputs; however, they are implemented as bidirectionals for NAND Tree testing.

Table 8-7. Serial EEPROM Interface Signal Pin Descriptions (cont)

Pin #	Pin Name	Description	I/O	Drive
272	EECLK	PCI EEPROM Clock	Output[1]	3 ma.
95	EECS	PCI EEPROM Chip Select	Output[1]	3 ma.
188	EEDO	PCI EEPROM Data Out	Input	

1. During normal operation these pins are outputs; however, they are implemented as bidirectionals for NAND Tree testing.

Table 8-8. Miscellaneous Control Signal Pin Descriptions

Pin #	Pin Name	Description	I/O	Drive
98	SRCLK	33 Mhz. Chip Clock	Input	
97	LED	Software programmable LED output	Output[1]	3 ma.
275	RESETL	Output reset signal for PHY device	Output[1]	3 ma.

1. During normal operation these pins are outputs; however, they are implemented as bidirectionals for NAND Tree testing.

Table 8-9. Factory Test Signal Pin Descriptions

Pin #	Pin Name	Description	I/O	Drive
191	BISTEN	Memory Module Built In Self Test Enable	Input	
190	TESTSEI	ATPG Parallel Load Enable	Input	
192	POUT	NAND Tree Output	Output[1]	9 ma.
276	MOUTEN	Output enable—disabled for NAND Tree Testing	Input	
99	TESTEN	ATPG Scan Enable	Input	
86	SCANIN1	ATPG Scan Inputs	Input	
87	SCANIN2	ATPG Scan Inputs	Input	
88	SCANIN3	ATPG Scan Inputs	Input	
269	SCANIN4	ATPG Scan Inputs	Input	
96	SCANIN5	ATPG Scan Inputs	Input	
189	SCANIN6	ATPG Scan Inputs	Input	
351	SCANIN7	ATPG Scan Inputs	Input	

1. During normal operation these pins are outputs; however, they are implemented as bidirectionals for NAND Tree testing.

Table 8-10. Die Fixed VCC Pin Descriptions

Pin #	Pin Name	Description	I/O	Drive
193	VCC	+3.3V		
197	VCC	+3.3V		
200	VCC	+3.3V		
204	VCC	+3.3V		
207	VCC	+3.3V		
210	VCC	+3.3V		
214	VCC	+3.3V		
218	VCC	+3.3V		
221	VCC	+3.3V		
225	VCC	+3.3V		
228	VCC	+3.3V		
231	VCC	+3.3V		
235	VCC	+3.3V		
239	VCC	+3.3V		
242	VCC	+3.3V		
246	VCC	+3.3V		
249	VCC	+3.3V		
252	VCC	+3.3V		
256	VCC	+3.3V		
260	VCC	+3.3V		
263	VCC	+3.3V		
267	VCC	+3.3V		
270	VCC	+3.3V		
273	VCC	+3.3V		

Table 8-11. Additional VCC Pins for Simultaneous Switching

Pin #	Pin Name	Description	I/O	Drive
104	VCC	+3.3V		
111	VCC	+3.3V		
291	VCC	+3.3V		
126	VCC	+3.3V		
33	VCC	+3.3V		

Table 8-11. Additional VCC Pins for Simultaneous Switching (cont)

Pin #	Pin Name	Description	I/O	Drive
38	VCC	+3.3V		
41	VCC	+3.3V		
44	VCC	+3.3V		
144	VCC	+3.3V		
234	VCC	+3.3V		
317	VCC	+3.3V		
58	VCC	+3.3V		
157	VCC	+3.3V		
69	VCC	+3.3V		
74	VCC	+3.3V		
167	VCC	+3.3V		
352	VCC	+3.3V		

Table 8-12. Die Fixed Ground Pins

Pin #	Pin Name	Description	I/O	Drive
280	GND	Ground		
283	GND	Ground		
287	GND	Ground		
290	GND	Ground		
293	GND	Ground		
296	GND	Ground		
299	GND	Ground		
302	GND	Ground		
306	GND	Ground		
309	GND	Ground		
312	GND	Ground		
315	GND	Ground		
318	GND	Ground		
321	GND	Ground		
326	GND	Ground		
328	GND	Ground		
331	GND	Ground		

Table 8-12. Die Fixed Ground Pins (cont)

Pin #	Pin Name	Description	I/O	Drive
334	GND	Ground		
337	GND	Ground		
340	GND	Ground		
344	GND	Ground		
347	GND	Ground		
350	GND	Ground		
277	GND	Ground		

Table 8-13. Additional Ground Pins for Simultaneous Switching

Pin #	Pin Name	Description	I/O	Drive
194	GND	Ground		
279	GND	Ground		
8	GND	Ground		
11	GND	Ground		
286	GND	Ground		
115	GND	Ground		
118	GND	Ground		
294	GND	Ground		
122	GND	Ground		
216	GND	Ground		
298	GND	Ground		
130	GND	Ground		
133	GND	Ground		
37	GND	Ground		
138	GND	Ground		
141	GND	Ground		
313	GND	Ground		
314	GND	Ground		
236	GND	Ground		
150	GND	Ground		
153	GND	Ground		
156	GND	Ground		

Table 8-13. Additional Ground Pins for Simultaneous Switching (cont)

Pin #	Pin Name	Description	I/O	Drive
324	GND	Ground		
326	GND	Ground		
248	GND	Ground		
164	GND	Ground		
332	GND	Ground		
255	GND	Ground		
257	GND	Ground		
336	GND	Ground		
83	GND	Ground		
342	GND	Ground		
343	GND	Ground		
91	GND	Ground		
348	GND	Ground		
349	GND	Ground		
274	GND	Ground		

Table 8-14. PCI Diode Protection Bias Pins

Pin #	Pin Name	Description	I/O	Drive
1	VIO	PCI Diode Protection Bias		
4	VIO	PCI Diode Protection Bias		
7	VIO	PCI Diode Protection Bias		
10	VIO	PCI Diode Protection Bias		
12	VIO	PCI Diode Protection Bias		
16	VIO	PCI Diode Protection Bias		
19	VIO	PCI Diode Protection Bias		
22	VIO	PCI Diode Protection Bias		
25	VIO	PCI Diode Protection Bias		

Table 8-15. Reserved Pins

Pin #	Pin Name	Description	I/O	Drive
26	RSVD	Reserved For Future Use		
82	RSVD	Reserved For Future Use		
100	RSVD	Reserved For Future Use		
101	RSVD	Reserved For Future Use		
127	RSVD	Reserved For Future Use		
136	RSVD	Reserved For Future Use		
147	RSVD	Reserved For Future Use		
160	RSVD	Reserved For Future Use		
161	RSVD	Reserved For Future Use		
295	RSVD	Reserved For Future Use		
297	RSVD	Reserved For Future Use		

Electrical Specifications 9

Overview

This chapter provides electrical specifications for the (i)chipSAR+ ABR. PCI interface signals comply with the PCI Local Bus Specification, Revision 2.1. All other signals are 5V TTL compatible I/O as specified in the following tables.

Absolute Maximum Ratings

Table 9-1. Absolute Maximum Ratings

Item	Symbol	Condition	Rated Value	Unit
Power Supply Voltage	V_{DD}		-0.5 to +4.6	V
Input Buffer Voltage[1]	V_I	$V_I<V_{DD}+0.5V$	-0.5 to +6.6	V
Output Buffer Voltage[1]	V_O	$V_O<V_{DD}+0.5V$	-0.5 to +6.6	V
Operating Temperature	T_A		0 to +70	^oC
			32 to 158	^oF
Storage Temperature	T_{stg}		-65 to +150	^oC
			-85 to +302	^oF

[1] These specifications apply to all signals except the PCI interface signals.

Recommended Operating Conditions

Table 9-2. Recommended Operating Conditions

VDD=3.3V +/- 0.3V; T_a=0 to 70^o (T_j=0 to 100^{o}C)						
Item	**Symbol**	**Condition**	**Min**	**Typ**	**Max**	**Unit**
Power Supply Voltage	V_{DD}		3.0	3.3	3.6	V
High-level input voltage[1]	V_{IH}		2.0		5.5	V
Low-level input voltage[1]	V_{IL}		0		0.8	V
Input Rise Time[1]	t_{ri}		0		200	ns
Input Fall Time[1]	t_{fl}		0		200	ns

[1] These specifications apply to all signals except the PCI interface signals.

DC Characteristics

Table 9-3. DC Characteristics

VDD=3.3V +/- 0.3V; T_a=0 to 70^O (T_j=0 to 100^{O}C)						
Item	**Symbol**	**Condition**	**Min**	**Typ**	**Max**	**Unit**
OFF-state output current[1]	I_{OZ}	V_O=V_{DD} or GND	2.0		5.5	V
Output short-circuit current[1,2]	I_{OS}	V_O=GND		+/-10^{-4}	+/-10	ua
Low-level output current[1,3]	I_{OL}	V_{OL}=0.4V				ma
3ma type			3.0			
9ma type			9.0			
12ma typ			12.0			
High-level output current[1,3]	I_{OH}	V_{OH}=2.4V				ma
3ma type			-3.0			
9ma type			-3.0			
12ma type			-3.0			

1 These specifications apply to all signals except the PCI interface signals.

2 The output short-circuit time is less than one second and for only one LSI pin.

3 The output type for each pin is specified in the pin description section.

AC Characteristics

PCI Interface Timing

The PCI interface is designed to operate at a maximum frequency of 33MHz. PCI timing is as specified in the PCI Local Bus Specification, Revision 2.1.

SRCLK Specification

SRCLK requires a 33MHz clock input with the following characteristics:

Table 9-4. SRCLK Requirements

Item	Value
Frequency	33MHz
Duty Cycle	45% to 55%
Frequency Stability	100ppm

Control Memory Interface

Each of the control memory interfaces (transmit and receive) is designed to operate using a single bank of 15ns asynchronous SRAMs with no external decoding logic. Output enable access must be a maximum of 8 ns. Writes must occur on the positive edge of a negative-going write pulse. Functionally, the bank may be 32K by 16 or 128K by 16 and may be implemented using a combination of 8- or16-bit wide SRAMS.

Examples of SRAMS meeting the timing requirements for the control memory interfaces include the following:

- ISSI (32K by 8) IS61C256AH-15T
- ISSI (128K by 8) IS61C1024-15T
- Motorola (32K by 8) MCM6206DJ15
- Motorola (128K by 8) MCM6226BWJ15

Packet Memory Interface

The packet memory interface is designed to operate using 15ns asynchronous SRAMS with no external decoding logic. Output enable access must be a maximum of 8 ns. Writes must occur on the positive edge of a negative going write pulse. Legal memory configurations are shown below:

Table 9-5. Packet Memory Configurations

Total Size	Bank	Bank Size
128K bytes	1	32K by 32 bits
512K bytes	1	128K by 32 bits
1M byte	1	128K by 32 bits
	2	128K by 32 bits

The above configurations can be designed using a combination of 8-, 16-, or 32-bit wide SRAMS. Examples of SRAMS meeting the timing requirements for the packet memory interface include the following:

- ISSI (32K by 8) IS61C256AH-15T
- ISSI (128K by 8) IS61C1024-15T
- Motorola (32K by 8) MCM6206DJ15
- Motorola (128K by 8) MCM6226BWJ15

Utopia Interface

The (i)chipSAR+ implements a Utopia Level 1-like interface that runs at 33MHz based on the SRCLK. Timing of this interface is designed to operate with the PMC-Sierra, Inc. PM5346 S/UNI-155-LITE 155 Sonet interface and compatible devices. It also interfaces with the IDT77105 25.6 Mbps ATM physical interface and compatible devices.

Control Memory Map Examples

A

Overview

This appendix provides information about and examples of control memory maps for the following types of operation:

- 1023 VC operation with separate segmentation and reassembly control memories (small memory configuration). The example is for CBR, ABR, and UBR.
- 127 VC operation with combined segmentation and control memories (25 Mbps operation). The example is for CBR, ABR, and UBR.
- 4095 VC operation with separate segmentation and reassembly control memories (large memory configuration). The example is for CBR and ABR.

1023 VC Operation

This section provides information about 1023 VC operation with separate segmentation and reassembly control memories (small memory configuration). The example is for CBR, ABR, and UBR.

Transmit Side Memory Map

Table A-1 shows the data structures required on the transmit side, along with the required sizes for an example 1023 VC max operation.

Table A-1. Transmit Side Control Memory Data Structures

Data Structure	Size per Entry	Number of Entries	Number of Bytes
Buffer Descriptor	32 bytes	128	4K
VC Table	32 bytes	1K	32K
Extended VC Table	8 bytes	1K	8K
Packet Ready Queue	2 bytes	128	256
Transmit Complete Queue	2 bytes	128	256
CBR Scheduling Table	2 bytes	as needed	as needed
UBR Scheduling Table	4 bytes	1K	4K
UBR Wait Queue	4 bytes	1K	4K
ABR Scheduling Table	2 bytes	1K	2K
ABR Wait Queue	2 bytes	1K	2K

Figure A-1 illustrates a segmentation control memory map for an example 1023 VC max.

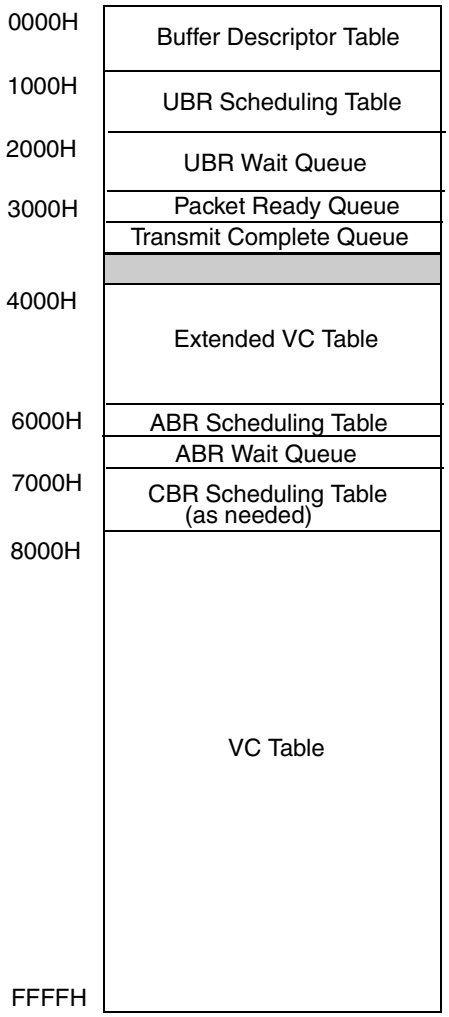

Figure A-1. Segmentation Control Memory Map Example

Buffer Descriptor Table

The Buffer Descriptor Table occupies locations from 0000H to 0FFFH. To achieve this placement, the Descriptor Table Base Address Register (DESC_BASE) is programmed as shown in *Table A-2*.

Table A-2. Buffer Descriptor Table

Register Name	Binary Representation	Hexadecimal Representation
DESC_BASE	000000000000000 0	0000

UBR Scheduling Table

The UBR Scheduling Table occupies the locations from 1000H to 1FFFH. To achieve this placement, the UBR Scheduling Table Base Address (UBR_SBPTR_BASE) is programmed as shown in *Table A-3*.

Table A-3. UBR_SBPTR_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
UBR_SBPTR_BASE	00000000 000 0001 0	0002

UBR Wait Queue

The UBR Wait Queue occupies the locations from 2000H to 2FFFH. To achieve this placement, the UBR Wait Queue Base Address Register (UBRWQ_BASE) is programmed as shown in *Table A-4*.

Table A-4. UBRWQ_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
UBRWQ_BASE	0000 000 0010 0 0000	0040

Communication Queues

The Packet Ready Queue and the Transmit Complete Queue occupy the block of memory from 3000H to 3FFFH. To achieve this placement, the Queue Base Address Register (QUEUE_BASE) is programmed to a 0 base and the actual locations of the individual queues are determined by the starting and ending pointers of the individual queues, as shown in *Table A-5*.

Table A-5. QUEUE_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
QUEUE_BASE	0000000000000 000	0000

Packet Ready Queue

The Packet Ready Queue occupies memory locations from 3000H to 30FFH. To achieve this placement, the Packet Ready address and pointer registers are programmed as shown in *Table A-6*.

Table A-6. Packet Ready Queue Registers

Register Name	Binary Representation	Hexadecimal Representation
PRQ_ST_ADR	0011 0000 0000 0000	3000
PRQ_ED_ADR	0011 0000 1111 1110	30FE
PRQ_RD_PTR	0011 0000 0000 0000	3000

Table A-6. Packet Ready Queue Registers (cont)

Register Name	Binary Representation	Hexadecimal Representation
PRQ_WR_PTR	0011 0000 0000 0000	3000

Transmit Complete Queue

The Transmit Complete Queue occupies memory locations from 3100H to 31FFH. To achieve this placement, the Transmit Complete address and pointer registers are programmed as shown in *Table A-7*.

Table A-7. Transmit Complete Queue Registers

Register Name	Binary Representation	Hexadecimal Representation
TCQ_ST_ADR	0011 0001 0000 0000	3100
TCQ_ED_ADR	0011 0001 1111 1110	31FE
TCQ_RD_PTR	0011 0001 0000 0000	3100
TCQ_WR_PTR	0011 0001 0000 0000	3100

Extended VC Table

The Extended VC Table occupies the locations from 4000H to 5FFFH. To achieve this placement, the Extended VC Table Base Address Register (VCTE_BASE) is programmed as shown in *Table A-8*.

Table A-8. VCTE_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
VCTE_BASE	00000 000 010 00000	0040

ABR Scheduling Table

The ABR Scheduling Table occupies the locations from 6000H to 67FFH. To achieve this placement, the ABR Scheduling Table Base Address Register (ABR_SBPTR_BASE) is programmed as shown in *Table A-9*.

Table A-9. ABR_SBPTR_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
ABR_SBPTR_BASE	00000000 000 0110 0	000C

ABR Wait Queue

The ABR Wait Queue occupies the locations from 6800H to 6FFFH. To achieve this placement, the ABR Wait Queue Base Register (ABRWQ_BASE) is programmed as shown in *Table A-10*.

Table A-10. ABRWQ_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
ABRWQ_BASE	0000 000 0110 1 0000	00D0

VC Table

The VC Table occupies the locations from 8000H to FFFFH. To achieve this placement, the Main Segmentation VC Table Base Address Register (VCT_BASE) is programmed as shown in *Table A-11*.

Table A-11. VCT_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
VCT_BASE	00000 000 1 0000 011	0083

Receive Side Memory Map

Table A-12 shows the data structures required on the receive side control memory, along with required sizes for an example 1023 VC–736 simultaneous reassemblies max.

Table A-12. Receive Side Control Memory Data Structures

Data Structure	Size per Entry	Number of Entries	Number of Bytes
Buffer Descriptor	32 bytes	736	23K
Free Buffer Queue	2 bytes	1K	2K
Packet Complete Queue	2 bytes	1K	2K
Exception Queue	4 bytes	128	512
Reassembly Table	2 bytes	1K	2K
VC Table	2 bytes	1K	2K
VP Table	2 bytes	256	512
ABR VC Table	32 bytes	1K	32K

Figure A-2 illustrates a reassembly control memory map for an example 1023 VC max.

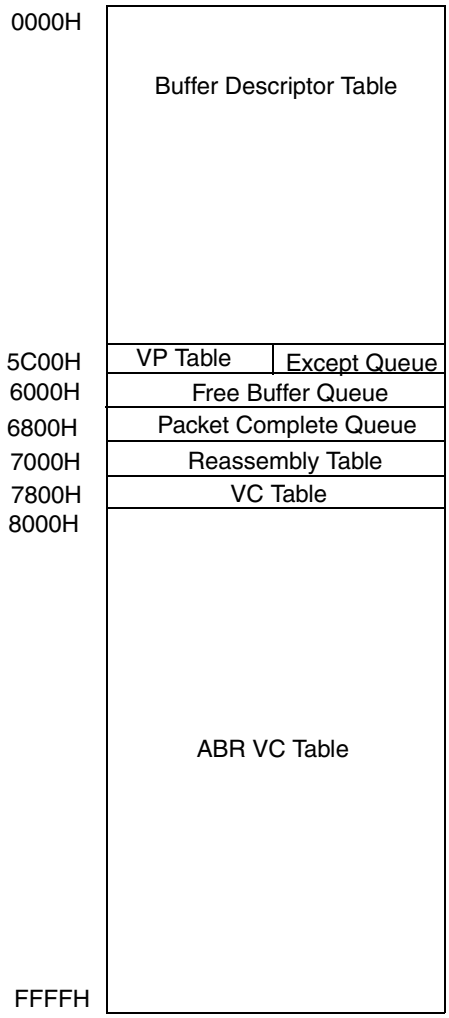

Figure A-2. Reassembly Control Memory Map Example

Buffer Descriptor Table

The Buffer Descriptor Table occupies the locations from 0000H to 5BFFH. To achieve this placement, the DESC_BASE Register is programmed as shown in *Table A-13*.

Table A-13. DESC_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
DESC_BASE	00000000 0 0000000	0000

Communications Queues

The QUEUE_BASE Register for the communications queues specifies in which 64K quadrant the queues reside. In this example, since only 64K reassembly control memory is used in this example, the QUEUE_BASE Register can be set to 0. The 16-bit starting and ending address for each queue can fully define the address space. The QUEUE_BASE Register is programmed as shown in *Table A-14*.

Table A-14. QUEUE_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
QUEUE_BASE	00000000 000 00000	0000

The following subsections describe the set up of individual starting and ending addresses.

Free Buffer Queue

The Free Buffer Queue occupies locations from 6000H to 67FFH. To achieve this placement, the Free Buffer Queue address and pointer registers are programmed as shown in *Table A-15*.

Table A-15. Free Buffer Queue Registers

Register Name	Binary Representation	Hexadecimal Representation
FREEQ_ST_ADR	0110 0000 0000 0000	6000
FREEQ_ED_ADR	0110 0111 1111 1110	67FE
FREEQ_RD_PTR	0110 0000 0000 0000	6000
FREEQ_WR_PTR	0110 0000 0000 0000	6000

Packet Complete Queue

The Packet Complete Queue occupies locations from 6800H to 6FFFH. To achieve this placement, the Packet Complete address and pointer registers are programmed as shown in *Table A-16*.

Table A-16. Packet Complete Queue Registers

Register Name	Binary Representation	Hexadecimal Representation
PCQ_ST_ADR	0110 1000 0000 0000	6800
PCQ_ED_ADR	0110 1111 1111 1110	6FFE
PCQ_RD_PTR	0110 1000 0000 0000	6800
PCQ_WR_PTR	0110 1000 0000 0000	6800

Exception Queue

The Exception Queue occupies locations from 5E00H to5FFFH. To achieve this placement, the Exception Queue address and pointer registers are programmed as shown in *Table A-17*.

Table A-17. Exception Queue Registers

Register Name	Binary Representation	Hexadecimal Representation
EXCP_Q_ST_ADR	0101 1110 0000 0000	5E00
EXCP_Q_ED_ADR	0101 1111 1111 1110	5FFE
EXCP_Q_RD_PTR	0101 1110 0000 0000	5E00
EXCP_Q_WR_PTR	0101 1110 0000 0000	5E00

VP Table

The VP Table occupies locations from 5C00H to 5DFFH. To achieve this placement, the VP Table Base Address Register (VP_LKUP_BASE) is programmed as shown in *Table A-18*.

Table A-18. VP_LKUP_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
VP_LKUP_BASE	0000101110 000000	0B80

VC Table

The VC Table occupies locations from 7800H to 7FFFH. To achieve this placement, the VC Table Base Address Register (VC_LKUP_BASE) is programmed as shown in *Table A-19*.

Table A-19. VC_LKUP_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
VC_LKUP_BASE	0000111 1 00000 011	0F03

NOTE

The lower three bits of the VC_LKUP_BASE Register define this implementation as being a 1023 VC max implementation. Changing this to a different number may require changes to all of the other base registers that are dependent on the number of VCs.

Reassembly Table

The Reassembly Table occupies locations from 7000H to 77FFH. To achieve this placement, the Reassembly Table Base Address Register (REASS_BASE) is programmed as shown in *Table A-20*.

Table A-20. REASS_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
REASS_BASE	000011100 0000000	0E00

ABR VC Table

The ABR VC Table occupies locations from 8000H to FFFFH. To achieve this placement, the ABR VC Table Base Address (ABR_LKUP_BASE) is programmed as shown in *Table A-21*.

Table A-21. ABR_LKUP_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
ABR_LKUP_BASE	000 1000 0 00000000	1000

127 VC Operation (25 Mbps Operation)

This section provides information about 127 VC operation with combined segmentation and reassembly control memories (small memory configuration). This section describes how to program the combined control memory for 127 VC max operation using ABR, CBR, and UBR.

Transmit Side Memory Map

Table A-22 shows the data structures that are required on the transmit side along with the required sizes for the 127 VC max case.

Table A-22. Transmit-Side Control Memory Data Structures

Data Structure	Size per Entry	Number of Entries	Number of Bytes
Buffer Descriptor	32 bytes	128	4K
VC Table	32 bytes	128	4K
Extended VC Table	8 bytes	128	1K
Packet Ready Queue	2 bytes	128	256
Transmit Complete Queue	2 bytes	128	256
CBR Scheduling Table	2 bytes	as needed	as needed
UBR Scheduling Table	4 bytes	1K	4K

Table A-22. Transmit-Side Control Memory Data Structures (cont)

Data Structure	Size per Entry	Number of Entries	Number of Bytes
UBR Wait Queue	4 bytes	1K	4K
ABR Scheduling Table	2 bytes	1K	2K
ABR Wait Queue	2 bytes	1K	2K

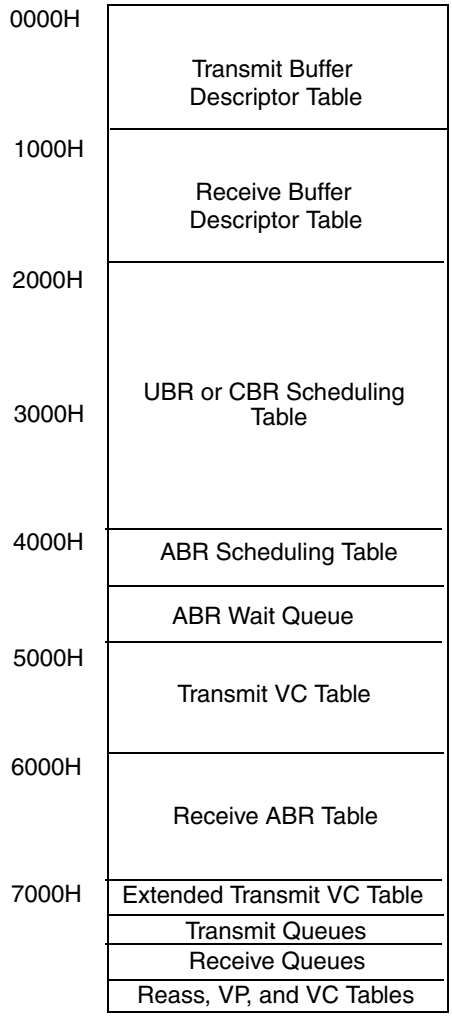

Figure A-3. Control Memory Map Example

Buffer Descriptor Table

The Buffer Descriptor Table occupies locations from 0000H to 0FFFH. To achieve this placement, the DESC_BASE Register is programmed as shown in *Table A-23*.

Table A-23. DESC_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
DESC_BASE	000000000000000 0	0000

In addition, the buffer numbers must range from 1 to 127.

ABR Scheduling Table

The ABR Scheduling Table occupies locations from 4000H to 47FFH. To achieve this placement, the ABR_SBPTR_BASE Register is programmed as shown *Table A-24*.

Table A-24. ABR_SBPTR_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
ABR_SBPTR_BASE	00000000 000 0100 0	0008

ABR Wait Queue

The ABR Wait Queue occupies locations from 4800H to 48FFH. To achieve this placement, the ABRWQ_BASE Register is programmed as shown in *Table A-25*.

Table A-25. ABRWQ_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
ABRWQ_BASE	0000 000 0100 0000 0	0080

Communication Queues

The Packet Ready Queue and the Transmit Complete Queue occupy the block of memory from 7400H to 77FFH. To achieve this placement, the QUEUE_BASE Register is programmed to a 0 base. The actual locations of the individual queues are determined by the starting and ending pointers of the individual queues, as shown in *Table A-26*.

Table A-26. QUEUE_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
QUEUE_BASE	0000000000000 000	0000

Packet Ready Queue

The Packet Ready Queue occupies memory locations from 7400H to 74FFH. To achieve this placement, the Packet Ready Queue address and pointer registers are programmed as shown in *Table A-27*.

Table A-27. Packet Ready Queue Registers

Register Name	Binary Representation	Hexadecimal Representation
PRQ_ST_ADR	0111 0100 0000 0000	7400
PRQ_ED_ADR	0111 0100 1111 1110	74FE
PRQ_RD_PTR	0111 0100 0000 0000	7400
PRQ_WR_PTR	0111 0100 0000 0000	7400

Transmit Complete Queue

The Transmit Complete Queue occupies memory locations from 7500H to 75FFH. To achieve this placement, the Transmit Complete Queue address and pointer registers are programmed as shown in *Table A-28*.

Table A-28. Transmit Complete Queue Registers

Register Name	Binary Representation	Hexadecimal Representation
TCQ_ST_ADR	0111 0101 0000 0000	7500
TCQ_ED_ADR	0111 0101 1111 1110	75FE
TCQ_RD_PTR	0111 0101 0000 0000	7500
TCQ_WR_PTR	0111 0101 0000 0000	7500

Extended VC Table

The Extended VC Table occupies locations from 7000H to 73FFH. To achieve this placement, the Extended Segmentation VC Table Base Address Register (VCTE_BASE) is programmed as shown in *Table A-29*.

Table A-29. VCTE_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
VCTE_BASE	00000 000 0111 00 00	0070

VC Table

The VC Table occupies locations from 5000H to 5FFFH. To achieve this placement, the Main Segmentation VC Table Register (VCT_BASE) is programmed as shown in *Table A-30*.

Table A-30. VCT_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
VCT_BASE	00000 000 0101 0 110	0056

In addition, bits 2 through 0 of the VCT_BASE Register should be set to 110 to represent 128 VCs.

Receive Side Memory Map

Table A-31 shows the data structures that are required on the receive side control memory along with required sizes for the 127 VC simultaneous reassemblies max case.

Table A-31. Receive Side Control Memory Data Structures

Data Structure	Size per Entry	Number of Entries	Number of Bytes
Buffer Descriptor	32 bytes	128	4K
Free Buffer Queue	2 bytes	128	256
Packet Complete Queue	2 bytes	128	256
Exception Queue	4 bytes	64	256
Reassembly Table	2 bytes	128	256
VC Table	2 bytes	128	256
VP Table	2 bytes	256	512
ABR Table	32 bytes	128	4K

Buffer Descriptor Table

The Buffer Descriptor Table occupies locations from 1000H to 1FFFH. To achieve this placement, the DESC_BASE Register is programmed as shown in *Table A-32*.

Table A-32. DESC_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
DESC_BASE	00000000 0 0000000	0000

To make the offset of 1000H, the buffer descriptor indices must be in the range of 129–255.

Communications Queues

The Communications Queues occupy locations from 7800H to 7BFFH. To achieve this placement, the QUEUE_BASE Register is programmed as shown in *Table A-23*.

Table A-33. QUEUE_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
QUEUE_BASE	00000000 000 00000	0000

NOTE

Since only a 32-Kbyte control memory is used, the 16-bit starting and ending address for each queue can fully define the address space, allowing this base register to be set to 0.

Individual starting and ending addresses for the communication queues are set up as described in the following subsections.

Free Buffer Queue

The Free Buffer Queue occupies locations from 7800H to78FFH. To achieve this placement, the Free Buffer Queue address and pointer registers are programmed as shown in *Table A-34*.

Table A-34. Free Buffer Queue Registers

Register Name	Binary Representation	Hexadecimal Representation
FREEQ_ST_ADR	0111 1000 0000 0000	7800
FREEQ_ED_ADR	0111 1000 1111 1111	78FF
FREEQ_RD_PTR	0111 1000 0000 0000	7800
FREEQ_WR_PTR	0111 1000 0000 0000	7800

Packet Complete Queue

The Packet Complete Queue occupies locations from 7900H to 79FFH. To achieve this placement, the Packet Complete Queue address and pointer registers are programmed as shown in *Table A-35*.

Table A-35. Packet Complete Queue Registers

Register Name	Binary Representation	Hexadecimal Representation
PCQ_ST_ADR	0111 1001 0000 0000	7900
PCQ_ED_ADR	0111 1001 1111 1111	79FF

Table A-35. Packet Complete Queue Registers

Register Name	Binary Representation	Hexadecimal Representation
PCQ_RD_PTR	0111 1001 0000 0000	7900
PCQ_WR_PTR	0111 1001 0000 0000	7900

Exception Queue

The Exception Queue occupies locations from 7A00H to7BFFH. To achieve this placement, the Exception Queue address and pointer registers are programmed as shown in *Table A-36*.

Table A-36. Exception Queue Registers

Register Name	Binary Representation	Hexadecimal Representation
EXCP_Q_ST_ADR	0111 1010 0000 0000	7A00
EXCP_Q_ED_ADR	0111 1011 1111 1111	7BFF
EXCP_Q_RD_PTR	0111 1010 0000 0000	7A00
EXCP_Q_WR_PTR	0111 1010 0000 0000	7A00

VP Table

The VP Table occupies locations from 7C00H to 7DFFH. To achieve this placement, the VP_LKUP_BASE Register is programmed as shown in *Table A-37*.

Table A-37. VP_LKUP_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
VP_LKUP_BASE	000 0111 110 000000	0F80

VC Table

The VC Table occupies locations from 7E00H to 7EFFH. To achieve this placement, the VC_LKUP_BASE Register is programmed as shown in *Table A-38*.

Table A-38. VC_LKUP_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
VC_LKUP_BASE	000 0111 1110 00 110	0FC6

NOTE

The lower three bits of this register define this implementation as being a 127 VC max implementation. Changing this to a different number may require changes to all of the other base registers that are dependent on the number of VCs.

Reassembly Table

The Reassembly Table occupies locations from 7F00H to 7FFFH. To achieve this placement, the REASS_BASE Register is programmed as shown in *Table A-39*.

Table A-39. REASS_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
REASS_BASE	000 0111 1111 00000	0FE0

ABR VC Table

The ABR VC Table occupies locations from 6000H to 6FFFH. To achieve this placement, the ABR_LKUP_BASE Register is programmed as shown in *Table A-40*.

Table A-40. ABR_LKUP_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
ABR_LKUP_BASE	000 0110 0 00000000	0C00

4095 VC Operation (Large Control Memories)

This section provides information about 4095 VC operation with separate segmentation and reassembly control memories (large memory configuration). It provides examples of CBR and ABR operation.

Transmit Side Memory Map

Table A-41 shows the data structures required on the transmit side, along with the required sizes for an example 4095 VC max configuration.

Table A-41. Transmit Side Control Memory Data Structures

Data Structure	Size per Entry	Number of Entries	Number of Bytes
Buffer Descriptor	32 bytes	2K	64K
VC Table	32 bytes	4K	128K
Extended VC Table	8 bytes	4K	32K

Table A-41. Transmit Side Control Memory Data Structures (cont)

Data Structure	Size per Entry	Number of Entries	Number of Bytes
Packet Ready Queue	2 bytes	2K	4K
Transmit Complete Queue	2 bytes	2K	4K
CBR Scheduling Table	2 bytes	as needed	as needed
ABR Scheduling Table	2 bytes	1K	2K
ABR Wait Queue	2 bytes	4K	8K

Figure A-4 illustrates a segmentation control memory map for an example 1023 VC max.

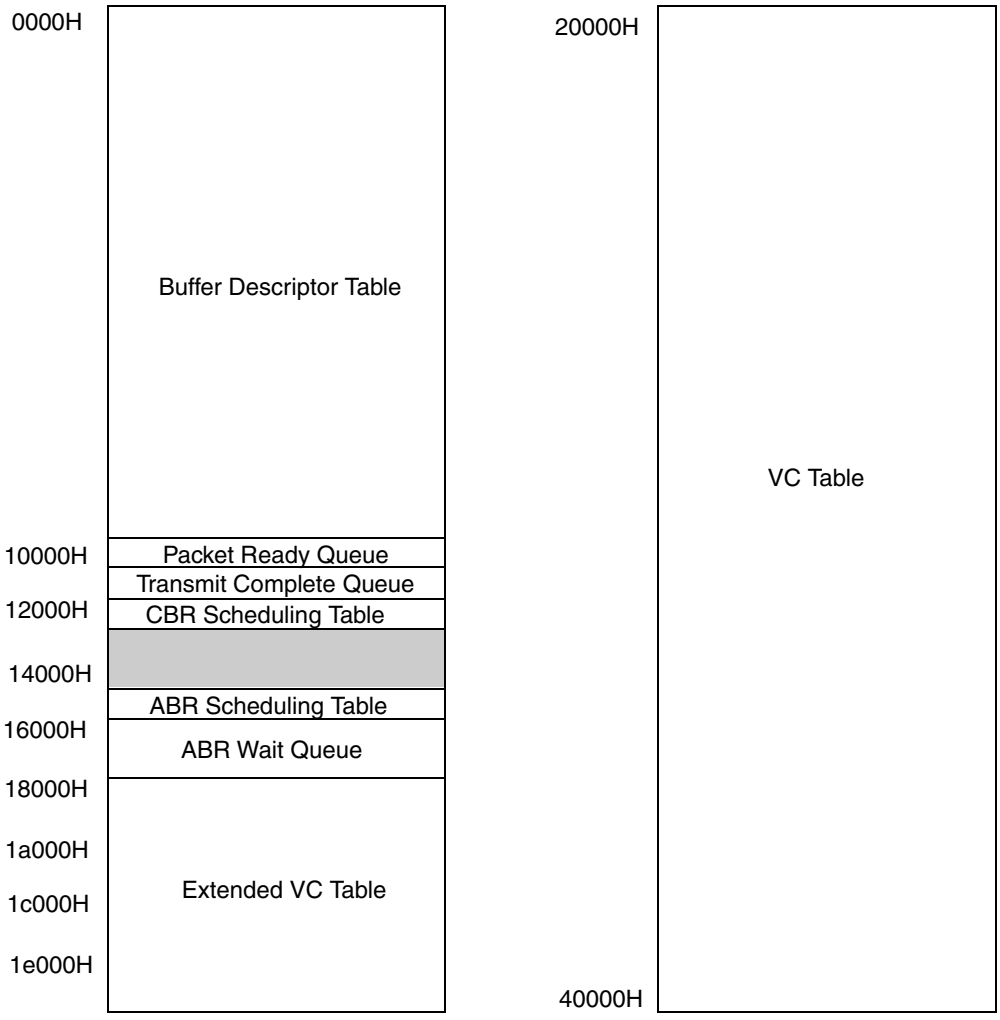

Figure A-4. Segmentation Control Memory Map Example

Buffer Descriptor Table

The Buffer Descriptor Table occupies locations from 00000H to 0FFFFH. To achieve this placement, the DESC_BASE Register is programmed as shown in *Table A-42*.

Table A-42. Buffer Descriptor Table

Register Name	Binary Representation	Hexadecimal Representation
DESC_BASE	000000000000000 0	0000

Communication Queues

The Packet Ready Queue and the Transmit Complete Queue occupy the block of memory from 10000H to 11FFFH. To achieve this placement, the QUEUE_BASE Register is programmed to a 10000H base and the actual locations of the individual queues are determined by the relative starting and ending pointers of the individual queues, as shown in *Table A-43*.

Table A-43. QUEUE_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
QUEUE_BASE	0000000000000 001	0001

Packet Ready Queue

The Packet Ready Queue occupies memory locations from 10000H to 10FFFH. This is an offset of 0000H off of the queue base. To achieve this placement, the Packet Ready Queue address and pointer registers are programmed as shown in *Table A-44*.

Table A-44. Packet Ready Queue Registers

Register Name	Binary Representation	Hexadecimal Representation
PRQ_ST_ADR	0000 0000 0000 0000	0000
PRQ_ED_ADR	0000 1111 1111 1110	0FFE
PRQ_RD_PTR	0000 0000 0000 0000	0000
PRQ_WR_PTR	0000 0000 0000 0000	0000

Transmit Complete Queue

The Transmit Complete Queue occupies memory locations from 11000H to 11FFFH. This is an offset of 1000H off of the queue base. To achieve this placement, the Transmit Complete Queue address and pointer registers are programmed as shown in *Table A-45*.

Table A-45. Transmit Complete Queue Registers

Register Name	Binary Representation	Hexadecimal Representation
TCQ_ST_ADR	0001 0000 0000 0000	1000
TCQ_ED_ADR	0001 1111 1111 1110	1FFE
TCQ_RD_PTR	0001 0000 0000 0000	1000
TCQ_WR_PTR	0001 0000 0000 0000	1000

ABR Scheduling Table

The ABR Scheduling Table occupies locations from 15000H to 157FFH. To achieve this placement, the ABR_SBPTR_BASE Register is programmed as shown in *Table A-46*.

Table A-46. ABR_SBPTR_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
ABR_SBPTR_BASE	00000000 001 0101 0	002a

ABR Wait Queue

The ABR Wait Queue occupies locations from 16000H to 17FFFH. To achieve this placement, the ABRWQ_BASE Register is programmed as shown in *Table A-47*.

Table A-47. ABRWQ_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
UBRWQ_BASE	0000 001 0110 0 0000	02c0

Extended VC Table

The Extended VC Table occupies locations from 18000H to 1FFFFH. To achieve this placement, the VCTE_BASE Register is programmed as shown in *Table A-48*.

Table A-48. VCTE_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
VCTE_BASE	00000 001 100 00000	0180

VC Table

The VC Table occupies locations from 20000H to 3FFFFH. To achieve this placement, the VCT_BASE Register is programmed as shown in *Table A-49*.

Table A-49. VCT_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
VCT_BASE	00000 010 0 0000 001	0201

Receive Side Memory Map

Table A-50 shows the data structures required on the receive side control memory, along with required sizes for an example 4095 VC–3008 simultaneous reassemblies max.

Table A-50. Receive Side Control Memory Data Structures

Data Structure	Size per Entry	Number of Entries	Number of Bytes
Buffer Descriptor	32 bytes	3008	96K
Free Buffer Queue	2 bytes	4K	8K
Packet Complete Queue	2 bytes	4K	8K
Exception Queue	4 bytes	256	1K
Reassembly Table	2 bytes	4K	8K
VC Table	2 bytes	4K	8K
VP Table	2 bytes	256	512
ABR VC Table	32 bytes	4K	128K

Figure A-5 illustrates a reassembly control memory map for an example 4095 VC max.

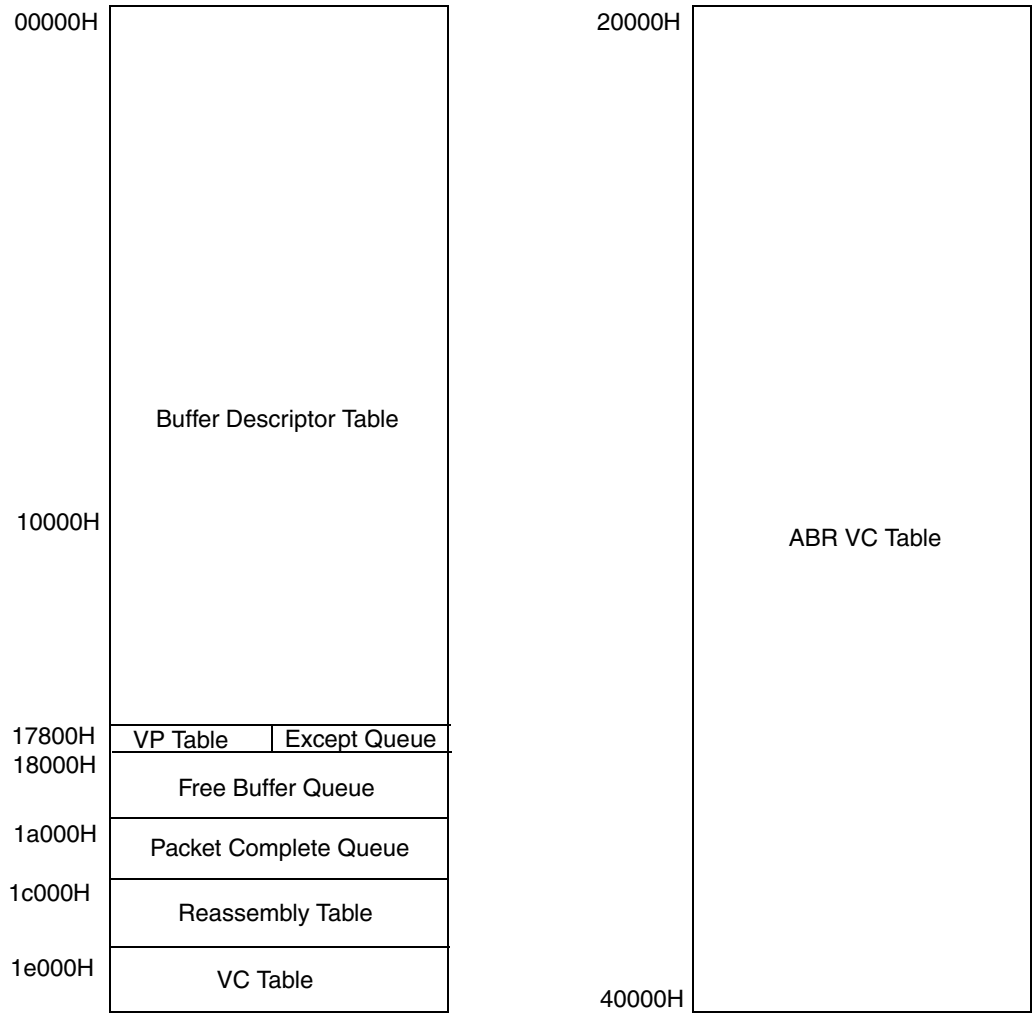

Figure A-5. Reassembly Control Memory Map Example

Buffer Descriptor Table

The Buffer Descriptor Table occupies locations from 00000H to 16FFFH. To achieve this placement, the DESC_BASE Register is programmed as shown in *Table A-51*.

Table A-51. DESC_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
DESC_BASE	00000000 0 0000000	0000

Communications Queues

The Communications Queues occupy the second 64K quadrant in the receive memory map. To achieve this placement, the QUEUE_BASE Register is programmed as shown in *Table A-52*.

Table A-52. QUEUE_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
QUEUE_BASE	00000000 001 00000	0020

NOTE

Within this quadrant, the 16-bit starting and ending address for each queue can fully define the address space, so this base register can be set to 10000H.

The following subsections describe the set up of individual starting and ending addresses.

Free Buffer Queue

The Free Buffer Queue occupies locations from 18000H to19FFFH. This is an offset of 8000H from the queue base. To achieve this placement, the Free Buffer Queue address and pointer registers are programmed as shown in *Table A-53*.

Table A-53. Free Buffer Queue Registers

Register Name	Binary Representation	Hexadecimal Representation
FREEQ_ST_ADR	1000 0000 0000 0000	8000
FREEQ_ED_ADR	1001 1111 1111 1110	9FFE
FREEQ_RD_PTR	1000 0000 0000 0000	8000
FREEQ_WR_PTR	1000 0000 0000 0000	8000

Packet Complete Queue

The Packet Complete Queue occupies locations from 1A000H to1BFFFH. This is an offset of A000H from the queue base. To achieve this placement, the Packet Complete Queue address and pointer registers are programmed as shown in *Table A-54*.

Table A-54. Packet Complete Queue Registers

Register Name	Binary Representation	Hexadecimal Representation
PCQ_ST_ADR	1010 0000 0000 0000	A000
PCQ_ED_ADR	1011 1111 1111 1110	BFFE

Table A-54. Packet Complete Queue Registers (cont)

Register Name	Binary Representation	Hexadecimal Representation
PCQ_RD_PTR	1010 0000 0000 0000	A000
PCQ_WR_PTR	1010 0000 0000 0000	A000

Exception Queue

The Exception Queue occupies locations from 17C00H to17FFFH. This is an offset of 7C00H from the queue base. To achieve this placement, the Exception Queue address and pointer registers are programmed as shown in *Table A-55*.

Table A-55. Exception Queue Registers

Register Name	Binary Representation	Hexadecimal Representation
EXCP_Q_ST_ADR	0111 1100 0000 0000	7C00
EXCP_Q_ED_ADR	0111 1111 1111 1110	7FFE
EXCP_Q_RD_PTR	0111 1100 0000 0000	7C00
EXCP_Q_WR_PTR	0111 1100 0000 0000	7C00

VP Table

The VP Table occupies locations from 17800H to 178FFH. To achieve this placement, the VP_LKUP_BASE Register is programmed as shown in *Table A-56*.

Table A-56. VP_LKUP_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
VP_LKUP_BASE	0010111100 000000	2F00

VC Table

The VC Table occupies locations from 1E000H to 1FFFFH. To achieve this placement, the VC_LKUP_BASE Register is programmed as shown in *Table A-57*.

Table A-57. VC_LKUP_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
VC_LKUP_BASE	00111100 00000 001	3C01

NOTE

The lower three bits of the VC_LKUP_BASE Register define this implementation as being a 4095 VC max implementation. Changing this to a different number may require changes to all of the other base registers that are dependent on the number of VCs.

Reassembly Table

The Reassembly Table occupies locations from 1C000H to 1DFFFH. To achieve this placement, the REASS_BASE Register is programmed as shown in *Table A-20*.

Table A-58. REASS_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
REASS_BASE	001110000 0000000	3800

ABR VC Table

The ABR VC Table occupies locations from 20000H to 3FFFFH. To achieve this placement, the ABR_LKUP_BASE Register is programmed as shown in *Table A-59*.

Table A-59. ABR_LKUP_BASE Register

Register Name	Binary Representation	Hexadecimal Representation
ABR_LKUP_BASE	010 0000 0 00000000	4000

Setup for CBR Scheduling

B

Overview

This appendix provides information about cell slot programming for the CBR Scheduling Table. It presents an algorithm in the form of an example C program for the case of multiple VCs having the same rate requirements (as in video servers or possibly desktop video windows). Some examples are run through the program to illustrate the precision of the CBR mechanism and the number of slave accesses needed per VC (at either chip initialization or VC setup) to program the CBR Scheduling Table.

An Example CBR Slot Assignment Algorithm

The following C program can be used to compute the programming that is needed to set up the CBR Scheduling Table for a given rate and number of VCs. Note that this example program addresses only the situation where one or more VCs are set up with the same rate. However, the basic algorithm can still be used with a modified program to solve other CBR applications, for example, random CBR setups at random rates.

The following algorithm produces a cell stream with an extremely accurate maximum source-generated jitter of one cell time (2.83 microseconds) per VC:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <stdio.h>

// This program assumes that there are a number of possible VCs to be setup
// as CBR VCs.  Each of these VCs will have the same rate (e.g. video server
// or possibly desktop video windows).  With this program, the CBR Scheduling
// Table for all of the VCs can be computed.  Software can program the CBR
// Scheduling Table at initialization time and then simply enable the VCs one
// at a time as they are signalled.

long bps, space, intslots, bytesincell, numvcs, ratetype, bestfitnum,
     bestfitstep, numentries, testloops, i, j, actualrate, schedsize,
     numempty, numinrow, inrowcount, startaddress, address;
double tolerance, exactslots, fraction, fractol, bestfit, fracstep,
       actualtol;

main()
{
   // Input the appropriate rates and constraints

   printf ("Enter the payload rate in bits / second   ");
   scanf ("%ld", &bps);
   printf ("Enter the number of payload bytes in a cell  ");
   scanf ("%ld", &bytesincell);
   printf ("Enter the number of VCs to be scheduled  ");
   scanf ("%ld", &numvcs);
   printf ("Enter the maximum space available for structure (in bytes)  ");
```

```
scanf ("%ld", &space);
printf ("Enter 1 if closest rate is desired\n");
printf ("      2 if final rate must be no slower than desired rate\n");
printf ("      3 if final rate must be no faster than desired rate     ");
scanf ("%ld", &ratetype);
printf ("Enter the frequency tolerance (in percent)   ");
scanf ("%8lf", &tolerance);

// That is all of the information that is needed.  First, compute
// the nominal number of slots that is needed for the rate. With this,
// we can check the validity of the input parameters.

exactslots = (1.0 / (((double)bps / (double)bytesincell) / 8.0))
               / 2.8312e-6;
if (((double)(space / 2)) < exactslots)
  // This is an error.
  printf ("INVALID INPUTS.  RATE TOO SMALL FOR STRUCTURE SPACE.");
else if (((double)numvcs) > exactslots)
  // This is an error
  printf ("INVALID INPUTS.  OVERBOOKED LINK.");
else
{
   // These are valid inputs. Make the calculations for different numbers
   // of entries up to the maximum memory size.

   // The algorithm for this is conceptually quite simple.  The CBR
   // Scheduling Table is assumed to have some number of entries per VC.
   // For example, if it is assumed that the CBR Scheduling Table has 4
   // entries per VC, then the CBR Scheduling Table has a length of roughly
   // 4 time the number of slots in the exactslots variable.  Since we
   // cannot have fractional slots, each number of entries trial will have
   // some number of entries at one integer number of slots and the rest of
   // the entries at the next integer number of slots.  In this way, the
   // average will be a fractional number of slots.  The trick is to
   // choose the minimum number of entries that will allow for a rate that
   // is within the specified tolerance.

   // The algorithm below will start at one entry per VC and continue to add
   // entries and increase the size of the CBR Scheduling Table until either
   // the tolerance is reached or until the specified maximum table size is
   // reached.

   // Once the optimum number of entries and spacing is reached, this
   // program will output which VC indices need to go into which locations
   // in the CBR Scheduling Table.

   intslots = (long)exactslots;

   fraction = exactslots - (double)intslots;

   // Convert the tolerance into a fraction tolerance

   fractol = (exactslots * (1.0 + (tolerance / 100.0))) - exactslots;

   // Perform the calculation for the case of one entry per VC.

   if ((fraction == 0.0) || (ratetype != 3))
```

```
{
   bestfit = fraction;
   bestfitnum = 1;  // So far, one entry per VC is the best fit
   bestfitstep = 0;
}
if (ratetype != 2)
   if ((1.0 - fraction) < bestfit)
   {
      bestfit = 1.0 - fraction;
      bestfitnum = 1;
      bestfitstep = 1;
   }

numentries = 1;
while (fractol < bestfit) // If we go into this while loop, we will stay
                          // there until a break statement is executed.
{
   // First check to see if this combination will fit into the
   // CBR scheduling table.  If not, we are through.

   if ((numentries * (intslots + 1) * 2) > space)
      break;

   fracstep = 0.0;
   for (testloops = 1; testloops < numentries; testloops++)
   {
      fracstep = fracstep + (1.0 / (double)numentries);
      if ((fracstep < fraction) && (ratetype != 3))
         // Perform a test to see if this is the best fit so far
         if ((fraction - fracstep) < bestfit)
         {
            bestfit = fraction - fracstep;
            bestfitnum = numentries;
            bestfitstep = testloops;
         }
      else if ((fracstep > fraction) && (ratetype != 2))
         // Perform a test to see if this is the best fit so far
         if ((fracstep - fraction) < bestfit)
         {
            bestfit = fracstep - fraction;
            bestfitnum = numentries;
            bestfitstep = testloops;
         }
      else if ((fracstep > fraction) && (ratetype == 2))
         break;  // Through with the for loop
   }
   numentries++;
}

// Now, output the results.

actualtol = (bestfit / exactslots) * 100.0;
actualrate = (long)(((double)bytesincell * 8.0)
             / (((double)intslots + ((double)bestfitstep
                    / (double)bestfitnum)) * 2.8312e-6));

printf ("\nActual rate = %ld.  Error = %f %%\n\n", actualrate,
```

```
                actualtol);

      schedsize = ((bestfitstep * (intslots + 1))
                   + ((bestfitnum - bestfitstep) * intslots)) * 2;
      printf ("Size of CBR Scheduling Table = %ld bytes\n\n", schedsize);

      // Determine the most uniform spacing of CBR VCs.  This is done to
      // ensure the most efficient functioning of cell time compensation.

      numempty = (intslots / numvcs) - 1;
      numinrow = (long)(1.0 / (((double)intslots / (double)numvcs) - 1.0)) - 1;
      inrowcount = 1;
      startaddress = 0;

      printf ("VC Index          Entry Numbers in CBR Scheduling Table\n\n");

      for (i=1; i<=numvcs; i++)
      {
         printf ("%6ld       ", i);
         address = startaddress;
         for (j=0; j<bestfitnum; j++)
         {
            printf ("%6ld   ", address);
            if (j < bestfitstep)
               address = address + intslots + 1;
            else
               address = address + intslots;
         }
         printf ("\n");
         if (numempty >= 1)
            startaddress = startaddress + numempty + 1;
         else
            if (inrowcount < numinrow)
            {
               startaddress++;
               inrowcount++;
            }
            else
            {
               startaddress = startaddress + 2;
               inrowcount = 1;
            }
      }
   }
}
```

The following examples were run to illustrate the power of the CBR functionality within the (i)chipSAR+. Only the rate results are provided. The program also computes exactly what to load into the CBR Scheduling Table for the number of VCs requested. The loading of the CBR Scheduling Table is the most efficient for smooth operation of other traffic types and also for the most efficient operation of the Cell Time Compensation circuit (if needed).

Each case assumes that the actual rate should be as close as possible to, but no less than, the requested rate.

64 Kbps Rate Example

Input parameters:

- 64,000 bps
- 47 bytes of payload per cell

Program results:

- Actual rate of 64,002 bps
- Error of 0.004426% from the requested rate
- Total CBR Scheduling Table size of 4150 bytes
- Up to 2076 VCs at this rate can be supported with the same size CBR Scheduling Table
- One slave access per VC for setup
- Zero cell-to-cell jitter per VC at OC-3 link rate
- One cell every 2076 cell times at OC-3 link rate

2.5 Mbps Rate Examples

Input parameters:

- 2,500,000 bps
- 47 bytes of payload per cell

Program results in the first case:

- Actual rate of 2,500,529 bps
- Error of 0.021158% from the requested rate
- Total CBR Scheduling Table size of 956 bytes
- Up to 53 VCs at this rate can be supported with the same size CBR Scheduling Table
- Nine slave accesses per VC for setup
- One cell slot time (2.83 µs) of cell-to-cell jitter per VC at OC-3 link rate
- One cell every 53–54 cell times at OC-3 link rate

Program results when the same test was run with a looser tolerance:

- Actual rate of 2,505,771 bps
- Error of 0.230319% from the requested rate
- Total CBR Scheduling Table size of 106 bytes
- Up to 53 VCs at this rate can be supported with the same size CBR Scheduling Table
- One slave access per VC for setup
- Zero cell slot time (2.83 µs) of cell-to-cell jitter per VC at OC-3 link rate
- One cell every 53 cell times at OC-3 link rate

4.5 Mbps Rate Examples

Input parameters:

- 4,500,000 bps
- 47 bytes of payload per cell

Program results in the first case:

- Actual rate of 4,500,033 bps
- Error of 0.000752% from the requested rate
- Total CBR Scheduling Table size of 2420 bytes
- Up to 29 VCs at this rate can be supported with the same size CBR Scheduling Table
- 41 slave accesses per VC for setup
- One cell slot time (2.83 µs) of cell-to-cell jitter per VC at OC-3 link rate
- One cell every 29–30 cell times at OC-3 link rate

Program results when the same test was run with a looser tolerance:

- Actual rate of 4,501,894 bps
- Error of 0.042074% from the requested rate
- Total CBR Scheduling Table size of 118 bytes
- Up to 30 VCs at this rate can be supported with the same size CBR Scheduling Table
- One slave access per VC for setup
- Zero cell slot time (2.83 µs) of cell-to-cell jitter per VC at OC-3 link rate
- One cell every 30 cell times at OC-3 link rate

Example Conclusions

These examples show the power and flexibility of the CBR function in the (i)chipSAR+.

The actual number of slave accesses needed per VC (at either chip initialization or VC setup) and the size of the CBR Scheduling Table are a function of the rate itself and also depend on the tolerance of the rate. Depending on the tolerance, very small numbers of slave accesses and very small CBR Scheduling Table sizes can be used.

This example code, or similar code, can be run in the driver and therefore relieves you from the task of determining the best way to set up the CBR Scheduling Table.

ABR Theory of Operation C

Overview

This appendix summarizes the ABR functionality defined in the ATM Forum Traffic Management 4.0 specification.

Overview of the Available Bit Rate Specification

The Available Bit Rate (ABR) specification defines a closed feedback dynamic rate control mechanism for ATM connections. The rate control mechanism is used to control congestion within network nodes traversed by the connection.

Figure C-1 illustrates how congestion occurs within a network.

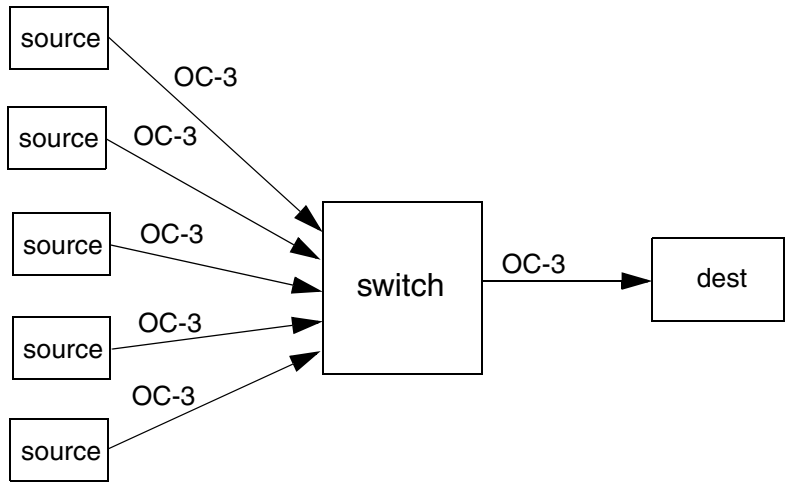

Figure C-1. Congestion Within an ATM Network

As *Figure C-1* shows, it is possible for switches to take in more cells through multiple inputs than they can push out through a single output. The result is that internal switch buffers fill to capacity and cells are dropped. Since cells might be small components of larger packets, the loss of a cell in a packet results in the loss of the packet. Significant loss of cells across multiple packets is disastrous to the performance of an ATM network.

ABR controls this cell loss through a feedback loop between the source and the destination of a connection. Based on the level of their internal congestion, network nodes along the connection can adjust parameters within control structures of the feedback loop. When these control structures return to the source of the data, the source can adjust its rate for that connection accordingly.

Resource Management Cells

The mechanism used for the closed loop feedback control is a special cell called a Resource Management cell (RM cell). RM cells are sent periodically on a per connection basis along with the data cells for the connection.

Figure C-2 provides a simplified view of RM cell flow.

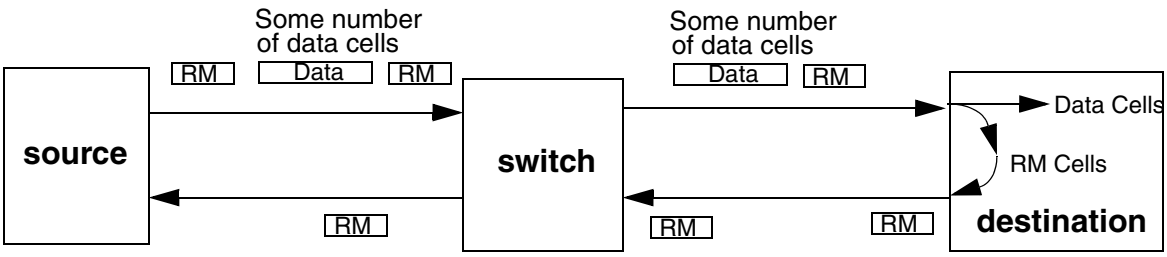

Figure C-2. RM Cell Flow

As this illustration shows, RM cells originate at the source and flow along the connection. At the destination, the data is consumed, and the RM cells are turned around to flow backward. Each network node can alter certain fields in the RM cell in response to the level of congestion in the node. *Table C-1* shows the structure of an RM cell as defined in the ATM Forum Traffic Management 4.0 specification.

Table C-1. RM Cell Fields

FIELD	OCTET	BIT(s)	DESCRIPTION
Header	1 - 5	all	Standard ATM Header with PTI = 110
ID	6	all	Protocol Identifier
DIR	7	8	Direction (0 = forward 1 = backward)
BN	7	7	BECN Cell
CI	7	6	Binary Congestion Indication
NI	7	5	No Increase
Reserved	7	4 - 1	Set to 0
ER	8 - 9	all	Explicit Rate
CCR	10 - 11	all	Current Cell Rate
MCR	12 - 13	all	Minimum Cell Rate
Reserved	14 - 21	all	Set to 0
Reserved	22 - 51	all	Set to 6A (hex) for each octet
Reserved	52	8 - 3	Set to 0
CRC-10	52	2 - 1	CRC-10 for the cell
	53	all	

Table C-2 describes the RM cell fields.

Table C-2. RM Cell Field Descriptions

RM Cell Field	Description
ID	Protocol Identifier. This octet is used to differentiate different versions of ABR RM cells. For ATM Forum Traffic Management 4.0, this octet is 1. The (i)chipSAR+ allows for this octet to be set under software control. Also, the (i)chipSAR+ can be set to either filter incoming RM cells using this protocol identifier or ignore the ID field on incoming RM cells.
DIR	Direction. This bit indicates the direction of this RM cell. If 0, then the RM cell is moving toward the source (forward RM cell). If 1, then the RM cell is moving toward the destination (backward RM cell). In turning around RM cells, the destination changes this bit from 0 to 1. Switches that generate backward RM cells (BECN (Backward Explicit Congestion Notification) cells) also set this bit to 1.
BN	BECN cell indication. This bit is set by switches when they generate optional BECN cells toward the source. The source uses this bit in its accounting procedures for RM cells.
CI	Binary Congestion Indication. The CI bit is initially set to 0 by the source as it initiates a forward RM cell. Any node along the connection (in either the forward or backward direction) can set this bit to 1 as a result of the node's present state of congestion. The destination shall also set the CI bit to 1 if the Explicit Forward Connection Indication (EFCI) on the last data cell received for this VC is set to 1 in the ATM header. After the CI bit is set to 1, no node in the connection is allowed to reset the bit back to 0. If the CI bit is set to 1 on a backward RM cell that has returned to the source, then the source reduces the VC's allowed rate. If the bit is not set to 1, the source can increase the VC's rate (depending on the state of the NI bit and the ER field).
NI	No Increase. The NI bit specifies that the rate for a node in the connection is not to be increased, even if the CI bit is not set to 1. The NI bit is initially set to 0 by the source while it initiates a forward RM cell. Any node along the connection (in either the forward or backward direction) can set this bit to 1 as a result of the node's present state of congestion.
ER	Explicit Rate. A node can use the Explicit Rate field to signal an exact rate for the VC to be sent. This field uses the 16-bit floating point rate format defined in ATM Forum Traffic Management 4.0. The ER field is set to the Peak Cell Rate by the source. It can be modified downward by any node in the connection. No node is allowed to modify the Explicit Rate field upward. When the backward RM cell returns to the source, the source sets the VC's rate to the minimum of the Explicit Rate and the result of the binary congestion calculation.

Table C-2. RM Cell Field Descriptions (cont)

RM Cell Field	Description
CCR	Current Cell Rate. The source places the current Allowed Cell Rate for the VC into this field when the forward RM cell is initiated. This field is used for informational purposes only to assist nodes in the connection with their optional Explicit Rate calculations. No node is allowed to modify this field.
MCR	Minimum Cell Rate. The source places the Minimum Cell Rate for the VC into this field when the forward RM cell is initiated. This field is used for informational purposes only to assist nodes in the connection with their optional Explicit Rate calculations. No node is allowed to modify this field.
CRC-10	CRC-10. This is the same CRC that is used for all OAM cells. It is described in the ATM Forum Traffic Management 4.0 specification.

Congestion Control Methods

Two methods are defined by the ABR specification for congestion control:

- Binary mode
- Explicit rate mode

ABR networks can use either of these modes or a combination of the two.

Binary Mode Congestion Control

The binary mode of congestion control, as the name implies, feeds back a binary congestion indication to the host. If congestion is present, there is no indication of how bad the congestion is.

Either of the following settings indicate binary congestion:

- The Congestion Indication Bit in the RM cell (bit 6 of octet 7) may be set to 1 (in either forward or backward RM cells).
- The EFCI (Explicit Forward Congestion Indication) codepoint in the ATM header of data cells may be set to 1 (forward direction only).

When setting the EFCI codepoint of data cells, the destination end station is required to monitor data cells for the EFCI codepoint and maintain the state, on a per-VC basis, of the last received data cell's EFCI codepoint. When it turns around a forward RM cell and sends it back to the source as a backward RM cell, the destination end station sets the Congestion Indication Bit in the RM cell if the last received data cell's EFCI codepoint was set.

The method used to indicate binary congestion is implementation dependent. The simplest implementation of ABR in switches is for the switch to set the EFCI codepoint of data cells whenever a threshold is reached in the switch internal buffers. This method is backward compatible with the congestion indication mechanism of ATMF UNI 3.x. With this method, switches do not have to understand RM cells; they pass them along with the data cells.

More advanced switches can set the Binary Congestion Indication (CI) bit of the RM cells directly rather than through the EFCI codepoint of data cells. When setting the CI bit directly, the switch should set the CI bit of backward RM cells rather than forward RM cells because the congestion information reaches the source faster if the RM cell is already heading back to the source. In addition, because the advanced switches must recognize RM cells anyway, they can treat the RM cells with greater priority, thus further decreasing the time required to send congestion information to the host.

Based on the information returned to them in backward RM cells, sources will either decrease the rate, increase the rate, or keep the rate the same.

If the CI bit in the RM cell is set to 1, the source decreases the allowed rate. The amount of decrease is determined by multiplying the present allowed rate by a fractional factor (which is determined by signalling). This produces an exponential rate of decrease. The larger the allowed rate, the more it is decreased when each backward RM cell indicates congestion.

If the CI bit in the RM cell is set to 0, and the No Increase (NI) bit also is set to 0, the source increases the allowed rate. The amount of increase is an additive amount (which is determined by signalling). This produces a linear rate of increase.

If the NI bit in the RM cell is set to 1, the source does not increase the rate, even if the CI bit is set to 0.

The source will not set the allowed rate to less than the minimum rate (signalled) nor greater than the peak rate (signalled).

Binary mode ABR is an effective tool for congestion control with high output rates in local area networks and other networks where physical layer propagation delays are low, such as in the case of software handling of RM cells or of wide area networks. Generally, the binary mode of operation does not produce optimal results when the round-trip time of RM cells is high. This is one reason why it is critical for RM cells and rate settings to be handled in hardware. If hardware handles RM cells and rate settings, RM cell round-trip times may be orders of magnitude less than if software handles RM cells and rate settings. Also, local area networks have inherently lower propagation delays than wide area networks.

Explicit Rate Congestion Control

ABR networks allow for a growth path where switches can become sophisticated in their approach to congestion control. Through the Explicit Rate field, switches can indicate exact rates that they can handle for VCs. This allows the networks to adjust differently for mild congestion than for severe congestion. Explicit rate congestion control allows switches to perform sophisticated congestion detection mechanisms internally, such as monitoring the rate of change of buffer levels rather than just the buffer level itself. (Refer to ATM Forum Traffic Management 4.0 Appendix I.5.2.)

With explicit-rate switches, congestion can be more tightly controlled than with binary mode congestion control. This allows ABR to work well even with higher RM cell round trip times, and promotes ABR for use in wide area networks.

Supporting Congestion Control Mechanisms

Note that end-station NICs are required to support both congestion control mechanisms simultaneously. NICs are required to perform both types of rate calculations and use the lessor of the two rates as the ACR for the VC. An end-station NIC used in a binary congestion control network can be used without modifications in an explicit rate congestion control network. However, the switches in the two types of networks implement a different algorithm.

In explicit rate networks, to give switches the flexibility to adjust the rate up or down, the additive increase amount for the binary calculation must be set to the Peak Cell Rate (PCR). If the switches never set the EFCI codepoints in data cells, and never set the CI or NI bits in RM cells, then the rate calculation that the NIC performs for binary congestion control will always attempt to set the rate at PCR. This allows the Explicit Rate field to totally control (either up or down) the final rate of the VC.

Virtual Sources and Virtual Destinations

The ABR closed feedback loop for RM cells is primarily designed as an end-to-end loop. This means that the RM cell is propagated all of the way from the source, through all switches in the connection, through the destination, back through all switches in the connection, and back to the source before the rate of the entire connection is determined.

The ATM Forum Traffic Management 4.0 specification allows for this loop to be broken into a series of smaller segments. The endpoints of these smaller segments are called *virtual* sources and destinations. *Figure C-3* illustrates a loop with a virtual source and destination (at the switch).

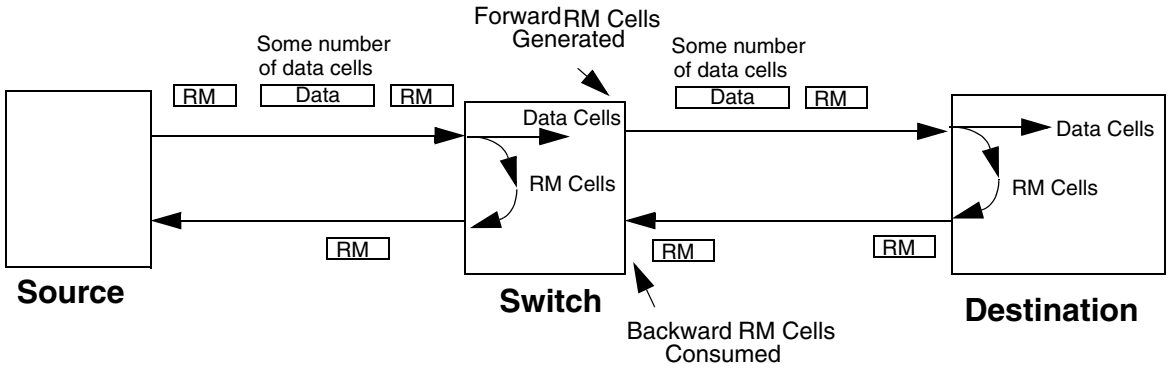

Figure C-3. RM Cell Flow With Virtual Source and Virtual Destination

A switch (or other network element) can take on the role of a virtual source and virtual destination in situations where segments of a single ABR connection can have different round trip time characteristics (e.g. a connection goes from a source in a LAN to a destination in a WAN).

Virtual sources and virtual destinations are required to adhere to all of the required behaviors of sources and destinations. The methods that the virtual source and virtual destination use to couple between different rates of the different segments are implementation specific.

Regulatory Statements D

FCC

4531 Communications Controller

FCC Part 15 Regulatory Compliance

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

This equipment complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions:

(1) This equipment may not cause harmful interference, and

(2) This equipment must accept any interference received, including interference that may cause undesired operation.

Canadian

Tested to Comply with Canadian Standards

This Class A digital apparatus complies with Canadian ICES-003.

Cet appareil numérique de la classe A est conforme à la norme NMB-003 du Canada.

European

Regulatory Information for Europe

This equipment displays the CE mark to show that it has been tested and found to be in full compliance with the requirements of the EMC and Low Voltage Directives (89/336/EEC and 72/23/EEC, as amended by Directive 93/68/EEC).

WARNING

This is a Class A product. In a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures.

Safety

EN60950–IEC950 Safety Standard

This equipment complies with the EN60950–IEC950 safety standard, with the following restrictions:

These cards are for use only with UL Listed personal computers that have instructions detailing installation of accessories by the user.

Glossary

802.2 IEEE ◆ Standards that govern the *LLC (Logical Link Control)* within the Data Link layer of the OSI model. LLC frames carry user information between the nodes on a network and define the transmission of a frame between two stations. These standards are common across the various lower level standards within the Data Link and the Physical layers.

802.3 IEEE ◆ Standards that govern the use of the CSMA/CD (Carrier Sense Multiple Access/Collision Detection) network access method used by Ethernet networks.

802.5 IEEE ◆ Standards that govern the use of the token ring indicator and frame priority.

AAL (ATM Adaptation Layer) ◆ Converts packets of data to 53-byte cells for transmission on the network. Several AALs are defined to provide different types of service for ATM connections, and to provide a method of mapping data from a particular class of service into ATM cells in such a way that the data can be remapped into its original format at the other end of an ATM network.

ABR (Available Bit Rate) ◆ One of the two non-guaranteed service types (the other is UBR). With ABR, the network makes no absolute guarantee of cell delivery, but does guarantee a minimum bit rate for user transmission. ABR also makes an effort to keep cell loss as low as possible through a closed feedback loop conveying congestion information back to the source so the source can adjust its rates.

Adapter ◆ A device, usually a user interface card, that physically connects an end station to the network medium (for example, twisted pair, coaxial, fiber).

Address Registration ◆ A subset of ILMI that enables the switch and end-station to dynamically construct an end-station address.

AESA (ATM End Station Address) ◆ The 20-byte address defined by the ATM Forum for identifying end stations on a private ATM network.

ANSI (American National Standards Institute) ◆ An organization that coordinates, develops, and publishes standards used in the United States.

ARP (Address Resolution Protocol) ◆ The Internet protocol used to dynamically translate the Internet address of a network host to its LAN hardware address. This action is limited to LANs that support hardware broadcasts.

ASIC (Application Specific Integrated Chip)

ATM (Asynchronous Transfer Mode) ◆ A switched, connection-oriented technology for LANs and WANs. ATM accommodates a mix of data types, such as audio, video, and data, on a single network. The multiplexed information is organized into cells, and can be transported between network nodes at many different access and transmission speeds. Synonyms: asynchronous transmission, cell relay.

ATM Forum ◆ A communications industry organization made up of hundreds of vendors and users that defines ATM networking protocols.

ATM LAN ◆ Topology that consists of ATM switches and computer interfaces that provide high data rate connectivity for voice, video, and data (IP and multimedia). In addition, ATM interfaces are being integrated into existing LAN hubs and bridge/router platforms. Besides supporting high bandwidth, ATM LANs map to the WAN via central office ATM switches and services being deployed in the telecommunications world.

Attenuation ◆ Signal power lost in a transmission medium as the signal travels from sender to receiver.

Backbone ◆ A network configuration that connects LANs to form an integrated network.

Bandwidth ◆ Capacity for transmitting data through a given circuit. Generally, the greater the bandwidth, the more information can be sent through a circuit during a given amount of time.

Best Effort ◆ A QoS class where no specific traffic parameters and no attempts are made to guarantee no cell loss or delay variation.

B-ISDN (Broadband ISDN)

Bridge ◆ An internetworking device used to connect two or more computer networks at the MAC level, and to forward MAC packets among those networks.

BUS (Broadcast and Unknown Server) ◆ A LAN emulation server for ATM networks that has the ability to receive broadcast requests and forward them to all the attached LECs, thus emulating the broadcast feature of Ethernet and Token Ring LANs.

CAC (Connection Admission Control) ◆ An ATM function that determines whether a VC connection request should be accepted or rejected.

Call Control ◆ A process that uses signalling procedures to establish VCs. These connections are much like telephone calls. Synonym: call setup.

CAT-3 (Category 3 UTP) ◆ A type of *UTP (Unshielded Twisted Pair)* commonly used with ATM interfaces for cell transmission at low speeds (25 Mbps) and at distances up to 100 meters.

CAT-5 (Category 5 UTP) ◆ A type of UTP commonly used with ATM interfaces for higher-speed cell transmission (155 Mbps).

CBR (Constant Bit Rate) ◆ The continuous transmission of data at a fixed and guaranteed rate over a network with a guaranteed cell delay variation.

CCITT (International Telephone and Telegraph Consultative Committee) ◆ See *ITU (International Telecommunication Union) on page 279*.

CDV (Cell Delay Variation) ◆ A QoS parameter that measures the difference between a single cell's transfer delay (CTD) and the expected transfer delay. It gives a measure of how closely cells are spaced in a VC. CDV can be introduced by ATM multiplexers (MUXs) or switches.

Cell ◆ Basic ATM transmission unit. A 53-byte packet comprised of a 5-byte header and a 48-byte payload. User traffic is segmented into cells at the source, transmitted through the ATM switched network, and reassembled at the destination.

Cell FIFO ◆ Optimized DMA of ATM cells from reassembly engine to system memory requiring minimal CPU intervention.

Cell Switching ◆ ATM technology that combines the best features of circuit-and packet-switching technology and supports several classifications of AAL service (voice/video, packet/video, data).

Circuit Switching ◆ A connection-oriented service that uses switching techniques such as time division. It is an ideal mode for continuous, constant bandwidth applications, such as voice and video.

CLIP (Classical IP) ◆ A set of IETF-defined protocols for developing IP over ATM networks. The main issues in the transport of IP over ATM that CLIP resolves are packet encapsulation and address resolution.

CLP (Cell Loss Priority) ◆ A 1-bit field in the ATM cell header that corresponds to the loss priority of a cell. Lower-priority (CLP=1) cells can be discarded in congestion situations.

Compression ◆ Method of reducing the quantity of data that must be transmitted across a network, primarily to enable the transmission of voice and video. The major compression standards are JPEG for still images, MPEG/MPEG-2 for full-screen motion images, and Px64 and H.261 for video conferencing.

Configuration Cycle ◆ A type of I/O cycle provided on the PCI bus to facilitate system configuration.

CPC (Cell Personality Card)

CPCS (Common Part Convergence Sublayer) ◆ Part of the AAL convergence sublayer (CS), it must be present in the AAL implementation. Its task is to pass primitives to the other AAL sublayers (SAR, SSCS). It supports the functions of the standardized Common Part AALs: AAL1, AAL3/4, and AAL5.

CRC (Cyclic Redundancy Check) ◆ A bit errors detection technique that employs an algorithm to calculate a value for the information bits in a packet. The receiver, using the same algorithm, recalculates that value and compares it to the value received. If the two values do not agree, the transmitted packet is considered to be in error.

DLE (DMA/Descriptor List Element) ◆ The structure that describes to the PCI bus interface how to move data between the packet buffer and system memory.

DMA (Direct Memory Access) ◆ A fast method of moving data between two subsystems without processor intervention.

DS3 (Digital Standard 3) ◆ This ANSI standard defines the format of asynchronous data sent at the rate of 44.736 Mbps.

E.164 ◆ An ITU-defined 8-byte address format. In ATM it is typically used in public networks and is provided by the telecommunication carriers, while 20-byte NSAP-format addresses are used in private networks.

EFCI (Explicit Forward Congestion Indication) ◆ A 1-bit field in the cell header that contains information about whether congestion at an intermediate node has been experienced. The EFCI bit is set when, for example, a buffer threshold has been exceeded. When recognized by end stations, causes a reduction in the data rate.

ELAN (Emulated LAN) ◆ See *LANE (LAN Emulation) on page 280*.

End Station ◆ A machine, intended for running user application programs, that is connected to a network. In an ATM network, the end station is where an ATM connection is terminated or initiated.

Fiber-Optic Cable ◆ A transmission medium designed to transmit digital signals in the form of pulses of light.

FIFO (First In First Out) memory ◆ A type of dual-ported memory where the data is read out in the same order in which it was written in.

Fragmentation ◆ A process where large frames from one network are broken up into smaller frames that are compatible with the frame size requirements of the network to which they will be forwarded.

Frame ◆ Data in a highly-structured format for the purpose of transmission. *Frame, packet,* and *PDU* are equivalent in most contexts.

FTP (File Transfer Protocol)

GFC (Generic Flow Control) ◆ A field in the ATM cell header that supports multiplexing functions. The GFC mechanism is intended to support simple flow control in ATM connections.

HDLC (High-level Data Link Control) ◆ A framing protocol specified by the ISO that can provide error-free Data Link layer services.

Header ◆ Control information attached to the front of a frame or packet. In ATM cells, the header is the bits in a cell allocated for functions required to transfer the cell payload within the network.

HEC (Header Error Control) ◆ A cell header CRC field that guarantees the integrity of the cell header information.

Host ◆ Generally, any computer on a network.

Host Name ◆ A unique name that identifies each host machine on a network.

ICMP (Internet Control Message Protocol) ◆ A Network layer Internet protocol which enables network IP packets to report errors and other relevant information for packet-processing purposes.

IEEE (Institute of Electrical and Electronic Engineers) ◆ An information exchange organization. Among other functions, it coordinates, develops, and publishes network standards for use in the United States, following ANSI rules.

IETF (Internet Engineering Task Force) ◆ Organization responsible for all Internet protocols (for example, IP, TCP, FTP).

ILMI (Integrated Local Management Interface) ◆ The ATM Forum standard, ILMI is used to manage the physical and logical interface between two ATM devices.

IME (Interface Management Entity) ◆ The logical management layer implementing ILMI.

IP (Internet Protocol) ◆ A networking protocol for providing a connectionless (datagram) service to the higher transport protocol. It is responsible for discovering and maintaining topology information and for routing packets across homogeneous or heterogeneous networks. Combined with TCP, it is commonly known as the TCP/IP platform.

IPX (Internetwork Packet Exchange Protocol) ◆ A connectionless Network layer protocol similar to IP.

ISDN (Integrated Services Digital Network) ◆ An early, CCITT-adopted protocol reference model intended to provide a ubiquitous, end-to-end, interactive digital service for data, audio, and video. Synonym: narrowband ISDN.

ISO (International Standards Organization) ◆ An international body that creates networking standards, including the OSI model.

ITU (International Telecommunication Union) ◆ The international standards organization for telecommunications, previously known as the CCITT (International Telephone and Telegraph Consultative Committee). For more information, see `http://www.itu.ch`

Isochronous ◆ A time-slot protocol that allows delivery of real-time data by dividing a time slot into equal-size mini slots allocated to different channels for synchronous transmission of information.

Jitter ◆ See *CDV (Cell Delay Variation) on page 278*.

KB (Kilobytes) ◆ One kilobyte is equivalent to 1024 bytes when referring to memory size, and 1000 bytes when referring to speed.

Kb (Kilobits) ◆ One kilobits is equivalent to 1024 bits when referring to memory size, and 1000 bits when referring to speed.

Kbps (Kilobits per second)

KBps (Kilobytes per second)

LAN (Local Area Network) ◆ A data communications system designed to operate over a limited geographic distance, such as a single building.

LANE (LAN Emulation) ◆ An ATM Forum service specification that allows a connection-oriented ATM network to emulate legacy LAN (for example, Ethernet or Token Ring) services, such as broadcast.

LAP-D (Link Access Procedure-D) ◆ A Data Link layer procedure using D channel communications, typical of ISDN.

LE-ARP (LAN Emulation ARP) ◆ The ARP used in LAN emulation for binding a requested ATM address to the MAC address.

LEC (LAN Emulation Client) ◆ Typically located in an ATM end system (for example, an ATM host), its task is to maintain address resolution tables and forward data traffic. It is uniquely associated with an ATM address.

LES (LAN Emulation Server) ◆ A server which provides support for the LAN emulation address resolution protocol (LE-ARP). The LECs register their own ATM and MAC addresses with the LES. An LES is uniquely identified by an ATM address.

LECS (LAN Emulation Configuration Server) ◆ A server whose main function is to provide configuration information to an LEC (such as the ELAN it belongs to or its LES).

LLC (Logical Link Control) ◆ The upper half of the Data Link layer in LANs. Performs error control, broadcasting, multiplexing, and flow control functions. See also *MAC (Medium Access Control) on page 280*.

LLC/SNAP (Logical Link Control/SubNetwork Attachment Point)

Local ◆ Describes files and devices, such as disk drives, that are attached to, or on, your machine.

MAC (Medium Access Control) ◆ A set of protocols that are the lower part of the Data Link layer and comprise the basis of the IEEE LAN specifications. In general, MAC determines the way devices can transmit in a broadcast network. See also *LLC (Logical Link Control) on page 280*.

Mbps (Megabits per second) ◆ Transmission speed or rate of one million bits per second.

MBps (Megabytes per second) ◆ Transmission speed or rate of one million bytes per second or 8 Mbps.

MIB (Management Information Base) ◆ The specification that defines objects for referencing variables such as integers and strings. In general, it contains information about the network's management and performance (for example, traffic parameters). See also *ILMI (Integrated Local Management Interface) on page 279*.

MTU (Maximum Transmission Unit) ◆ The largest packet that can be sent over a given medium.

Multicast ◆ A technique that allows copies of a single packet or cell to be passed to a set of destinations.

Multimode Fiber ◆ A large-core (62.5 micron) optical fiber through which multiple signals can propagate. Length constraint is 2 kilometers (1.2 miles).

Network ◆ An interconnection of multiple stations or systems that are able to send messages to and receive messages from one another.

NIC (Network Interface Card) ◆ A component that connects a station to a network (for example, a LAN). Synonym: adapter.

NMS (Network Management Station) ◆ The system responsible for managing a network or a portion of a network. The NMS communicates to network management agents (an agent resides in each managed node) using a network management protocol.

NNI (Network-to-Network Interface) ◆ The interface between two pieces of equipment on a public network.

Node ◆ A device, such as a station or concentrator, connected to the network media, usually with an adapter.

NRZI (Non-Return to Zero Inverted) ◆ A data transmission technique where a polarity transition from low to high, or high to low, represents a logical 1. The absence of a polarity transition represents a 0.

NSAP (Network Services Access Point) ◆ In the OSI environment it is the *SAP (Service Access Point)* between the network and the transport layers. It identifies a Data Terminal Equipment by a unique address.

OAM (Operations and Maintenance) ◆ Set of administrative and supervisory actions regarding network performance monitoring, failure detection, and system protection.

OAM Cell ◆ A cell that contains ATM layer management information. It does not form a part of the upper layer information transfer. Generated by hardware or network administrators. Cell header bits distinguish OAM cells from normal data cells.

OC-3c (Optical Carrier level 3, concatenated) ◆ An optical signal defined with a base rate of 155.52 Mbps. The concatenated signal is indivisible, that is, it cannot be multiplexed and demultiplexed.

OSI (Open Systems Interconnection) Model ◆ The 7-layer protocol model defined by the ISO for data communications.

Packet ◆ Data in a highly-structured format for the purpose of transmission. *Frame, packet,* and *PDU* are equivalent in most contexts.

Packet Buffer ◆ Memory used on the Interphase NIC to store data for fragmentation and reassembly. Synonym: Side RAM.

Packet Switching ◆ Statistical, connectionless switching based on information contained in variable-length packets.

Payload ◆ Part of the ATM cell, it contains the actual information to be carried, and may also contain overhead. It occupies 48 bytes.

PCI (Peripheral Component Interconnect) Bus ◆ A high-performance multiplexed address and data bus. Supporting 32-bit with optional 64-bit data transfers, the PCI bus is intended to be an interconnect between peripheral controllers, peripheral add-in boards, and processor/memory systems. The PCI bus operates at up to 33 MHz, providing burst transfer rates up to 132 MBps 32 bits wide, or up to 264 MBps 64 bits wide.

PDU (Protocol Data Unit) ◆ Data in a highly-structured format for the purpose of transmission. *Frame, packet,* and *PDU* are equivalent in most contexts.

PHY (Physical Layer) ◆ Layer 1 of the OSI model. Defines and handles the electrical and physical connections between systems. The Physical layer can also encode data in a form that is compatible with the medium (coaxial, twisted pair, fiber, and so on).

PING (Packet Internet Groper) ◆ An Internet protocol facility used to test the reachability of destinations by sending an ICMP echo request, and waiting for a reply.

PMC (PCI Mezzanine Card) ◆ A daughtercard form factor implementation of the PCI bus specification.

PMD (PHY Medium Dependent) ◆ A standard that defines the medium and protocols to transfer symbols between PHYs.

Point-to-Multipoint Connection ◆ A unidirectional, one-to-many VC that allows one station to simultaneously send data to the connected end stations.

Point-to-Point Connection ◆ A bidirectional VC between two end points.

Primitive ◆ Data and events passed between a layer service user and a layer service provider.

Protocol ◆ A set of rules and conventions that govern the exchange of information between communicating parties.

PVC (Permanent/Provisioned Virtual Connection) ◆ A VC provisioned for indefinite use in an ATM network, established by the network management system (NMS). See also *SVC (Switched Virtual Circuit) on page 283*.

Q.93B ◆ Early draft of the signalling specification now known as Q.2931, on which UNI 3.0 was based.

Q.2110 ◆ ITU Recommendation for specifying the UNI SSCOP.

Q.2931 ◆ ITU Recommendation derived from both Q.931 and Q.933 to provide SVC specifications and standards.

QoS (Quality of Service) ◆ The set of ATM performance parameters that characterize the traffic over a given VC.

QoS Classes ◆ Five service classes defined by the ATM Forum in terms of the QoS parameters.

Reassembly ◆ The portion of the ATM SAR that reassembles an incoming multiplexed stream of ATM cells into packets.

RFC (Request for Comment) ◆ IETF documents that contain proposed standards and specifications. RFCs can be either approved, or simply archived as historical recommendations.

RFC-1577 ◆ IETF standard for running Layer 3 IP traffic directly over ATM. See also *CLIP (Classical IP) on page 278*.

RJ-45 Connector ◆ Standard 8-wire connector for IEEE 802.3 networks and some telephone applications.

SAAL (Signalling AAL) ◆ Service-specific parts of the AAL protocol responsible for signalling. Its specifications, being developed by the ITU, were adopted from N-ISDN.

SAP (Service Access Point) ◆ Functional interface between the layers in the OSI model through which lower layers provide services to the higher layers along with PDUs.

SAR (Segmentation and Reassembly) ◆ The lower half of the AAL. The segmentation portion inserts data from packets into cells, adds any necessary header or trailer bits to the data, and passes the 48-octet payload to the ATM layer. Each AAL type has its own SAR format. At the destination, the reassembly portion extracts the cell payload and rebuilds the packet.

SAR-PDU ◆ The 48-octet PDU that the SAR sublayer exchanges with the ATM layer. It is comprised of the SAR-PDU payload and any control information that the SAR sublayer adds.

SC (Subscriber Connector) ◆ A connector where the transmit and receive fibers are one keyed module plug that latches.

SDH (Synchronous Digital Hierarchy) ◆ A hierarchy that designates signal interfaces for very high-speed digital transmission over optical fiber links. See also *SONET (Synchronous Optical Network) on page 282*.

SDU (Service Data Unit) ◆ User data passed through a SAP between the layers of the OSI or a similar model.

Signalling ◆ An ATM connection procedure that dynamically implements explicit routes through switches to establish a communication link with another station on the network.

Single-Mode Fiber ◆ A small-core (9 micron) optical fiber through which only one signal can propagate. Length constraint for the device used in Interphase products is 20 kilometers.

SLIP (Serial Line Internet Protocol) ◆ A protocol for transmitting and receiving IP datagrams via a serial interface.

SNMP (Simple Network Management Protocol) ◆ A high-level, standards-based protocol for network management, usually used in TCP/IP networks. An SNMP manager controls and measures the activities of SNMP agents that are embedded in nodes and network devices on the network. SNMP relies on MIBs embedded in the network resources to monitor and control the network.

SONET (Synchronous Optical Network) ◆ An ANSI-defined standard for high-speed and high-quality digital optical transmission. It has been recognized as the North American standard for SDH.

SSCF (Service Specific Coordination Function) ◆ Part of the SSCS portion of the SAAL. Among other functions, it provides a clear interface for relaying user data and providing independence from the underlying sublayers. See also *SSCOP (Service Specific Connection-Oriented Protocol) on page 282*.

SSCOP (Service Specific Connection-Oriented Protocol) ◆ Part of the SSCS portion of the SAAL. SSCOP is an end-to-end protocol that provides error detection and correction by retransmission and status reporting between the sender and the receiver. It also guarantees delivery integrity. See also *SSCF (Service Specific Coordination Function) on page 282*.

SSCS (Service Specific Convergence Sublayer) ◆ One of the two components of the Convergence Sublayer (CS) of the AAL. It supports the specific requirements of upper-layer protocols.

ST (Straight Through) connector ◆ A connector where the transmit and receive fibers have separate twist-on connections.

Station ◆ An addressable node on the network capable of transmitting and receiving data.

STM (Synchronous Transfer Mode) ◆ A packet-switching approach where time is divided into time slots assigned to single channels during which users can transmit periodically. Basically, time slots denote allocated (fixed) parts of the total available bandwidth.

STM-1 (Synchronous Transport Module-1) ◆ An ITU-defined SDH physical interface for ATM digital transmission at the rate of 155.52 Mbps.

STM-n (Synchronous Transport Module-n) ◆ An ITU-defined SDH physical interface for ATM digital transmission at *n* times the basic STM-1 rate. STM-n and SONET STS-3n transmission rates are equivalent.

STS (Synchronous Transport Signal)

Subnet Address ◆ An extension of the Internet addressing scheme. Using this method, a site can use a single Internet address for multiple physical networks.

SVC (Switched Virtual Circuit) ◆ A software-created dynamic connection between two network nodes. SVCs are created "on demand" and torn down upon completion of the data transfer.

Symbol ◆ The smallest signalling element used by the MAC sublayer. The symbol set consists of sixteen data symbols and sixteen non-data symbols. Each symbol corresponds to a specific sequence of code bits (code group) to be transmitted by the PHY.

Synchronous Transmission ◆ A data transmission scheme where the interval between transmitted characters is fixed so that start and stop bits are not required. As opposed to asynchronous transmissions, synchronous transmissions are guaranteed a specific percentage of bandwidth on the network medium.

TCP/IP (Transmission Control Protocol/Internet Protocol) ◆ A set of communications protocols that define how different types of computers talk to each other. It is the standard architecture for internetworking multiple organizations, and the common link that ties the huge Internet together.

TELNET ◆ A TCP/IP protocol that supports remote terminal operations via a network.

Token Ring ◆ A 4 Mbps or 16 Mbps network that uses a ring topology and a token-passing access method.

UBR (Unspecified Bit Rate) ◆ With UBR, the source specifies no traffic parameters, and therefore the network does not guarantee transmission quality.

UNI (User-Network Interface) ◆ Definition of the interface between an end system and an ATM switch. Defines a set of specifications for signalling. Produced by the ATM Forum.

UNI 3.0 ◆ ATM Forum UNI specification for the physical (PHY) and ATM layers, the ILMI, OAM (traffic control), and PVC support.

UNI 3.1 ◆ A corrected version of UNI 3.0, this specification also includes SSCOP standards.

UNI 4.0 ◆ This UNI specification covers signalling issues in ABR and VP, as well as QoS negotiation.

UTP (Unshielded Twisted Pair) ◆ Type 3 cable with one or more twisted pairs where the wiring is not protected from electromagnetic and radio frequency interferences. There are two main categories of UTP used in ATM: category 3 for 25 Mbps, and category 5 for 155 Mbps. See also *CAT-3 (Category 3 UTP) on page 278* and *CAT-5 (Category 5 UTP) on page 278*.

VBR (Variable Bit Rate) ◆ The bit rate available to a user for the transfer of user information that requires a guaranteed service for a bounded variable transmission rate.

VBR-RT (Variable Bit Rate—Real Time) ◆ One of the service types for transmitting traffic which is timing- and control-dependent and is characterized by having both average and peak cell rates. VBR-RT is suitable for carrying traffic such as packetized (compressed) video and audio.

VBR-NRT (Variable Bit Rate—Non-Real Time) ◆ One of the service types for transmitting traffic which is not timing-critical and is characterized by having both average and peak cell rates. VBR-NRT is well-suited to long data packet transfers.

VC (Virtual Channel/Connection/Circuit) ◆ A logical transmission path or connection between two network endpoints.

VCI (VC Identifier) ◆ A 16-bit identifier in an ATM cell header which, when combined with the VPI, identifies the VC to the next ATM device.

VLAN (Virtual LAN) ◆ A networking environment where users on physically independent LANs are interconnected in such a way that it appears they are in the same LAN workgroup.

VP (Virtual Path) ◆ A logical pipe which can contain a group of VCs that connect network devices.

VPI (VP Identifier) ◆ A field in an ATM cell header which, when combined with the VCI, identifies the VC to the next ATM device.

WAN (Wide Area Network) ◆ A network spanning a large geographical area that provides communications among devices on a regional, national, or international basis.

Workstation ◆ A networked computer typically reserved for end-user applications.