VME/ATM 5215 Adapter
Users Guide

VME/ATM
5215 Adapter
Users Guide

Document No. UG05215-000, REVC

Print Date: May 12, 1998

Copyright Notice

Printed in the United States of America, 1998.

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." 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 Syncard® are registered trademarks and CellView™, (i)chip™, ADSLWatch™, ADSLEye™, SynWatch™, SynEye™, FibreView™, and the Interphase logo are trademarks of Interphase Corporation.

Microsoft®, MS-DOS®, Windows®, and Windows NT® are registered trademarks of Microsoft Corp.

Novell® and NetWare® are registered trademarks of Novell, Inc.

Solaris® and NFS® are registered trademarks and SunOS™ and ONC™ are trademarks of Sun Microsystems, Inc. Sun is a trademark or registered trademark of Sun Microsystems, Inc.

SPARC® is a registered trademark of SPARC International, Inc. SPARCstation™ and UltraSPARC™ are trademarks of SPARC International, Inc., licensed exclusively to Sun Microsystems, Inc.

LattisCell™, EtherCell™, Bay Networks™, and SAHI™ are trademarks of Bay Networks, Inc.

UNIX® is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Ltd.

AT®, IBM®, and OS/2® are registered trademarks and AIX™ and PowerPC™ are trademarks of International Business Machines Corporation.

HP-UX® is a registered trademark and Tachyon™ and Precision Bus™ are trademarks of Hewlett-Packard Company.

Intel® and Pentium® are registered trademarks and i386™, i486™, Intel386™, and Intel486™ are trademarks of Intel Corporation.

TI® is a registered trademark of Texas Instruments.

Compu-shield® is a registered trademark of Stewart Connectors Systems, Inc.

Tundra® is a registered trademark and Universe™ is a trademark of Tundra Semiconductor Corporation.

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) 654-5555
	Fax:	(214) 654-5500
	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

Anonymous FTP Server

ftp.iphase.com

5215 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.

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.

Interphase Fiber Products' Compliance

All Interphase fiber products comply with IEC regulations 825-1 and 825-2 for Class 1 laser devices.

Declaration of Conformity

(according to ISO/IEC Guide 22 and EN 45014)

Manufacturer's Name:	Interphase Corporation
Manufacturer's Address and Phone Number:	13800 Senlac Dallas, Texas 75234 U.S.A. 214/654-5000

declares, that the product:

Product Name: VME ATM

Model Number: 5215

conforms to the following Standards:

Safety:	EN 60950:1988 + A1, A2 IEC 825 -1 & -2 1993
EMC:	EN 55022:1988 class A EN 50082-1 Part 1 1992

Supplementary Information:

This product complies with the requirements of the **Low Voltage Directive 73/23/EEC** and the **EMC directive 89/336/EEC**.

Dallas, April 22, 1996

Mike Jobe, Quality Manager

European Contact:
Interphase European Headquarters, Centre d'affaires 10ème Avenue, 855, avenue Roger Salengro, 92370 Chaville - France
Phone: + 33 (0) 1 41 15 44 00; Fax: + 33 (0) 1 41 15 12 13

Contents

CHAPTER 7 5215 Diagnostics

CHAPTER 8 Host Adapter Ring Interface (HARI)

APPENDIX D Base Address Jumper Settings

APPENDIX E Sample Data Structures

APPENDIX F ATM Technology Overview

Introduction 1

Overview

This manual is intended for users who need to install the Integrated Driver 1.x (ID 1.x) for IRIX or who need to write a software driver for the V/ATM 5215 controller. It can also be used to install and/or test the 5215. Readers are assumed to have extensive knowledge of the following:

- ATM specifications
- C programming language, including experience writing and installing drivers
- Operating system of the host computer
- VMEbus specifications and hardware installation procedures

The level of expertise required will depend on the task to be performed.

Related Publications And Standards

The following publications are excellent sources of information related to this product:

- *UNI Version 3.0 Specification*
  ATM Forum
  Worldwide Headquarters
  480 San Antonio Road, Suite 100
  Mountain View, CA 94040-1219
  Phone: (415) 962-2585
  Fax: (415) 941-0849
- *VME64 Specification*
  VFEA International Trade Association
  10229 North Scottsdale Rd., Suite B
  Scottsdale, AZ 85253
  VMEbus specifications and terminology.
- *VMEbus Specification Rev. D, IEEE*
  IEEE Service Center
  Publications Sales Dept.
  445 Hoes Lane
  Picscataway, NJ 08854-4150
- *Writing a Unix® Device Driver*
  Janet I. Egan/Thomas J Teixeira
  John Wiley and Sons, Inc. 1988

  Writing a Unix® Device Driver is an excellent introduction to writing Unix drivers. The authors discuss how Unix performs input/output, how device drivers relate to the hardware I/O architecture, and how to incorporate the device driver into the Unix kernel. It also contains an excellent discussion of how to test drivers.

Icon Conventions

Icons draw your attention to especially important information:

NOTE

The Note icon indicates important points of interest related to the current subject.

CAUTION

The Caution icon brings to your attention those items or steps that, if not properly followed, could cause problems in your machine's configuration or operating system.

WARNING

The Warning icon alerts you to steps or procedures that could be hazardous to your health, cause permanent damage to the equipment, or impose unpredictable results on the surrounding environment.

Text Conventions

This section details many of the text conventions used throughout the manual. In addition, it provides many of the technical conventions.

- The terms *Controller*, *controller*, *board*, and *5215* are synonymous and are used interchangeably throughout this document.
- The term *short I/O* refers to a VMEbus-addressed block of memory in which only the lower 16 VMEbus address lines are used for transactions. The upper 16 VMEbus address lines are not used.
- The term *byte* represents 8 bits of data; *word* represents 16 bits (2 bytes); and *longword* represents 32 bits (2 words/4 bytes).
- Binary (single bit) data is represented as either *1* or *0*.
- When used in the context of a single bit of data, the term *set* means that the bit is a one (1). Similarly, the term *cleared* means that the bit is a zero (0).
- To represent hexadecimal numbers, the manual adopts the C language notation. Decimal numbers are shown as decimal digits. For example:

```
0x29 = 29 hex
41 = 41 decimal
```

- Many commands contain bits, bytes, or words which are marked RESERVED. Such reserved fields fall into two categories:
  - Reserved for future use, must be cleared (0, 0x00, or 0x0000)
  - Reserved for future use, do not write to this field

  In this manual, when a data structure contains reserved field(s), a statement about how the host should treat the reserved field(s) is included. The statement is placed immediately after the data structure.
- When showing binary representations of bytes or words, the diagrams may have many bits which do not have names. These are RESERVED. As an example:

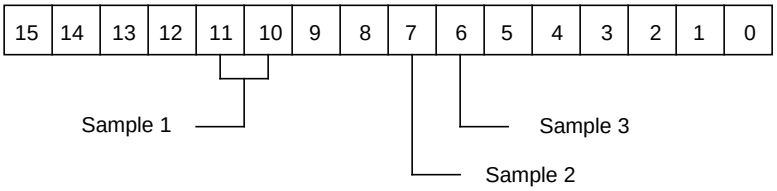


Figure 1-1. Example Byte Diagram

Bits 10 and 11 are called Sample 1, bit 7 is called Sample 2, and bit 6 is called Sample 3. All other bits are RESERVED.

Hardware Installation 2

Overview

This chapter provides instructions for installing the 5215 adapter in your system. It also describes and illustrates front panel LEDs and provides block diagrams of the V/ATM 5215 controller and daughtercard.

Installing the Adapter

To install the adapter, complete the following steps:

1. Do a visual inspection.
2. Power the system off.
3. Set on-board jumpers.
4. Install the board.
5. Power the system on.

CAUTION

Do not install or apply power to a damaged board. Failure to observe this warning could result in extensive damage to the board and/or system.

The 5215 is sensitive to electrostatic discharge (ESD), and the board can be damaged if handled improperly. Interphase ships the board enclosed in a special anti-static bag. Upon receipt of the board, take the proper measures to eliminate board damage due to ESD (that is, wear a wrist ground strap or other grounding device).

Visual Inspection

Before attempting the installation of this board, make sure you are wearing an anti-static or grounding device. Remove the 5215 board from the antistatic bag and visually inspect it to ensure no damage has occurred during shipment. If the board is undamaged and all parts are accounted for, continue with the installation.

Powering the System Off

Ensure that the host system and peripherals are turned OFF.

Setting On-board Jumpers

Set all on-board jumpers so that the 5215 is properly configured for operation within your system.

NOTE

Some jumpers default settings should not (or cannot) be altered. The other jumpers are used to help configure the board for your specific system.

On-board Jumper Functions

The following table lists the on-board jumpers and their functions:

Jumper	Function
JA1	Power On Self Test Diagnostic
JA2	XON/XOFF
JA3–JA5	Reserved
JA6	UART Baud Rate
JA7	CPU Frequency
JA8	CPU Cache
JA9	Reserved
JA10	PBUG (Reserved)
JA11	UART Enable/Disable
JA12	GDB Debug
JA13	Noisy DTCK Filter
JA14	Fairness
JA15	VME Arbitration Configuration
JA16–JA17	Not Installed
JA18	Short I/O size
JA19	VRAM Slave Enable
JA20	VRAM Slave Address Modifier
JA21	VRAM Slave Address (Va23)
JA22	VRAM Slave Address (Va22)
JA23	SRAM Short I/O Address Modifier
JA24	Short I/O Address (Va15)
JA25	Short I/O Address (Va14)

Jumper	Function
JA26	Short I/O Address (Va13)
JA27	Short I/O Address (Va12)
JA28	Short I/O Address (Va11)
JA29	Short /O Address (Va10)
JA30	Short I/O Address (Va9)
JA31–JA42	VMEbus Request Level

To locate the jumpers, see the board layout in Figure 2-1:

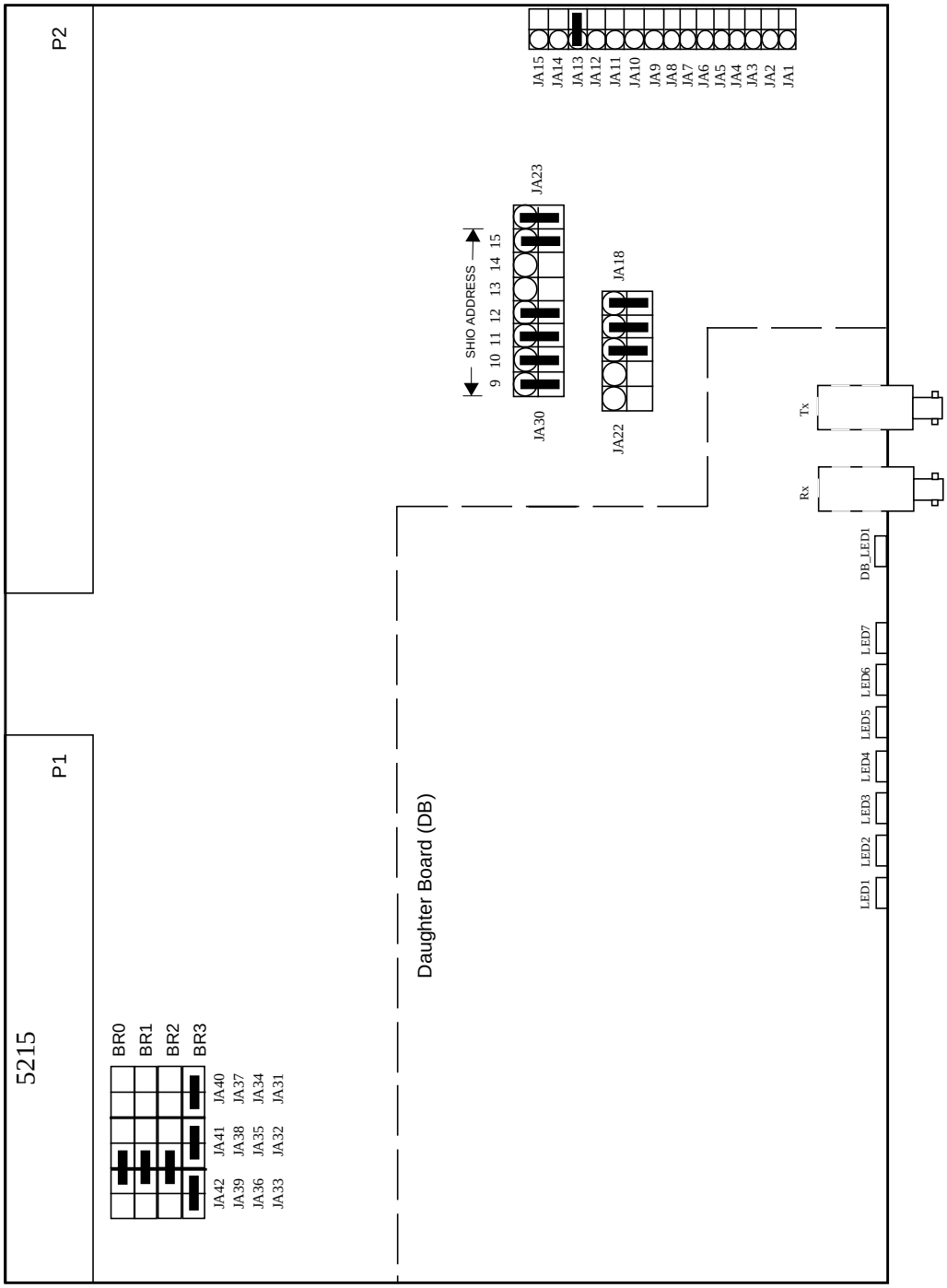


Figure 2-1. V/ATM 5215 Board Layout

On-Board Jumper Settings

JA1 Power On Self Test (POST). With this jumper Out, the power on self test (POST) is enabled and will execute on power-up. When the jumper is In, the POST is disabled.

JA2 XON/XOFF. When JA1 is In, XON/XOFF is enabled. When Out, XON/XOFF is disabled.

JA3-J5 Reserved.

JA6 UART Baud Rate. When JA6 is In, the UART baud rate is 9600. When Out, the UART baud rate is 38400.

JA7 CPU Frequency. When JA7 is In, the CPU frequency is 20 MHz. When Out, the frequency is 25 MHz.

JA8 CPU Cache. When JA8 is In, the CPU cache is disabled. When Out, it is enabled.

JA9 Reserved.

JA10 PBUG. Reserved.

JA11 UART Enable. When JA11 is In, the UART is enabled. When Out, the UART is disabled.

NOTE

When JA11 is In and no cable/RS232 port is connected to the board, the board will not come out of reset.

JA12 GDB Debug Enable. When JA12 is In, GDB is enabled. When Out, GDB is disabled.

JA13 Noisy DTACK Filter. If this jumper is IN (Default), DTACK is not filtered (that is, there is no additional immunity against noisy DTACK). This can save 20ns/cycle for systems that are dependent on the release of DTACK for the next cycle to start. If this jumper is OUT, DTACK is filtered. Some systems have considerable ringing on the trailing edge of DTACK. Since the 5215 VMEbus interface is an asynchronous design, the circuitry may interpret this noise as a response to the next transfer. Installing this jumper causes the trailing edge of DTACK to be sampled by a clock, eliminating the illegal transition.

JA14 Fairness. When JA14 is installed, the 5215 will not re-request the VME bus until it has seen its bus request level become inactive.

JA15 VME Arbitration Configuration. In, early release; Out, standard release. This jumper determines whether the 5215 will use Early or Standard release when releasing the bus during VME mastership. Early release will allow the 5215 to release the bus after the last VAS of ownership. Standard release will hold BBSY until all 5215 VME control signals are inactive.

JA16–JA17. Not Installed.

JA18 Short I/O Size. If In, 512 bytes; if Out, 2K. This jumper should always be installed.

JA19 VRAM Slave Enable. If JA19 is not installed, the entire 4MB internal VRAM buffer is available in the VME A24 space. This jumper should normally be installed.

NOTE

This option is not available on default hardware configurations. If you require VRAM access, contact Interphase for configuration information.

JA20 VRAM slave Address Modifiers. Installed, this jumper allows the VRAM address compare circuitry to respond to either privileged or non-privileged address modifiers (0x3D and 0x39). If not installed, this jumper allows only privileged address modifiers.

JA21 VRAM Slave Address. VME address 23.

JA22 VRAM Slave Address. VME address 22.

JA23 SRAM Short I/O Address Modifiers. When this jumper is installed, it allows the short I/O compare circuitry to accept either privileged or non-privileged address modifiers (0x29 and 0x2D). When this jumper is not installed, only privileged accesses (0x2D) are allowed.

JA24–JA30 Short I/O Base Address. Jumpers JA24 through JA30 set the base address of the 512 bytes of short I/O space RAM on the 5215. See the following table to determine the Base Address settings. (A more complete listing is provided in *Common Boot Error Codes* on page 159.)

Jumper	VMEbus Address Bit
JA24	A15
JA25	A14
JA26	A13
JA27	A12
JA28	A11
JA29	A10
JA30	A9

Removing a jumper sets the corresponding address bit to 1. Installing a jumper clears the address bit to 0. The short I/O base address must be a multiple of 0x200. The following table illustrates possible jumper settings for the Short I/O Base Address:

Jumper Settings							VMEbus Address							Base Address
JA24	JA25	JA26	JA27	JA28	JA29	JA30	A15	A14	A13	A12	A11	A10	A9	
IN	IN	IN	IN	OUT	IN	IN	0	0	0	0	1	0	0	0x0800

Jumper Settings							VMEbus Address							Base Address
IN	OUT	IN	OUT	OUT	IN	OUT	0	1	0	1	1	0	1	0x5A00
IN	OUT	OUT	IN	IN	IN	IN	0	1	1	0	0	0	0	0x6000
OUT	IN	OUT	IN	OUT	IN	OUT	1	0	1	0	1	0	1	0xAA00

JA31–JA42 VMEbus Request Level. Jumper fields JA30 through JA42 are used to set the 5215's VMEbus request priority. See Figure 2-2.

Figure 2-2 shows possible valid configurations for this jumper block:

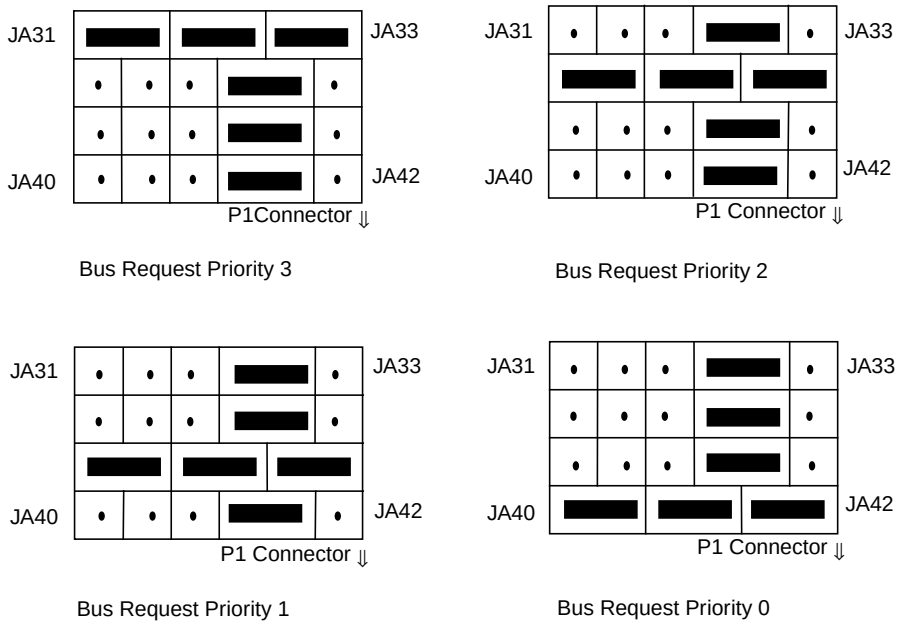


Figure 2-2. VMEbus Request Priority Jumper Setting

Installing the Board(s)

Before you install the 5215 in the VMEbus, configure the backplane jumpers, if required, for each board to be installed. See your system documentation for backplane configuration requirements. Carefully slide the 5215 into the VMEbus card slot. The board should slide in and seat without difficulty. If there is any difficulty, remove the board and check for obstructions.

Once the board(s) are properly seated in the slot(s), tighten the captive mounting screws on each end of the board front panel(s).

CAUTION

System power and peripherals power must be turned OFF before attempting to install the 5215. Failure to do so may result in severe damage to the board and/or system.

Powering on the System

Once the 5215 is installed in the chassis, turn the host on. **If your other system documentation (host/peripherals) contains precautionary statements about powering up the system, please follow these precautions.**

After powering up the system, observe LED1, which is located on the edge of the board opposite the VMEbus connectors. Once the power-up diagnostics complete successfully, the LED1 will stop toggling and the green light is visible. At this point, the 5215's Common Boot Interface is booted up and ready to accept commands. (See *Bootup and Reset Sequence* on page 55.) For more information on the LEDs, see *Front Panel LEDs* on page 13.

NOTES

Once the Common Boot interface is active, LED1, Status, turns green.

If one of the Power On Self Tests fails, the LED1 will be red and LEDs (ST0–ST2) will indicate the failure.

Alternately, if LED1 fails to turn green, the 5215 may not be fully seated in the chassis. Turn off the system and ensure that the 5215 is firmly inserted into the backplane slot. If the LED does not turn green after reapplying power to the system, call Customer Service at Interphase for assistance.

Front Panel LEDs

There are seven LEDs on the 5215 motherboard and two LEDs on the daughtercard. See Figure 2-1 on page 8 for the location of the LEDs.

Motherboard LEDs

LED1 BOARD OK (BOK). This LED is red at power-on reset or after a board failure. Under normal operating conditions, this LED is green. The status LEDs, ST0–ST2, offer additional controller status.

LED2 S-REQPEND. Host access to internal SRAM in progress.

LED3 D-REQPEND. Host access to internal DRAM in progress.

LED4 DMA_EN. VMEbus DMA transfer in progress.

LED5–LED7 (ST0–ST2). These are the 5215 status bits, 0–2. They offer additional status information in conjunction with the Board OK (BOK) LED. When the BOK LED is red, these LEDs indicate the failure mode, and when the BOK LED is green, they indicate which interface (CB or HARI) is active. When BOK is green and these LEDs are blinking, the Common Boot Interface is active. When Common Boot boot up completes, ST0–ST2 cycle up and down indicating that HARI is active.

As the Power On Self Test (POST) executes, ST0–ST2 indicate the test number being executed (same encoding as CB failure in table below). If the POST fails, BOK turns red and ST0–ST2 indicate the failure.

If POST completes successfully, ST0–ST2 blink on/off, indicating that Common Boot is ready for commands.

After the CB BOOT command completes, ST0–ST2 cycle up and down, indicating that the Host Adapter Ring Interface (HARI) is active.

Daughtercard LEDs

LINK 0—Optic Link Status. When on, the receive optics are detecting signals; otherwise this LED is OFF.

LINK 1—Front-End Status. When on, the daughtercard (front-end) Reset is active. When off, reset has been deasserted, but the front-end is uninitialized. When blinking on/off, the front-end has been initialized and is ready.

The front panel **ST0–ST2** LED encoding definitions when **BOK** is RED are as follows:

BOK = Red				
ST2	ST1	ST0	CBII	HARI
			blinking on/off	non-blinking
0	0	0	reserved	reserved
0	0	1	MEM diag failed	reserved
0	1	0	DMA diag failed	reserved
0	1	1	FE diag failed	reserved
1	0	0	reserved	F/W Panic
1	0	1	reserved	VME Bus Error
1	1	0	reserved	VME Bus Timeout
1	1	1	Power-on Reset	Power-on Reset

0=Off 1=On

The **LINK0–LINK1** encodings are as follows:

DB_LED1	Link Status
Link 0—Off	Fiber Optic link is down
Link 0—Green	Fiber Optic link is OK
Link 1—Yellow	Power On Reset
Link 1—Off	Power On Complete
Link 1—Blinking Yellow	Daughtercard is initialized and running

5215 Block Diagram

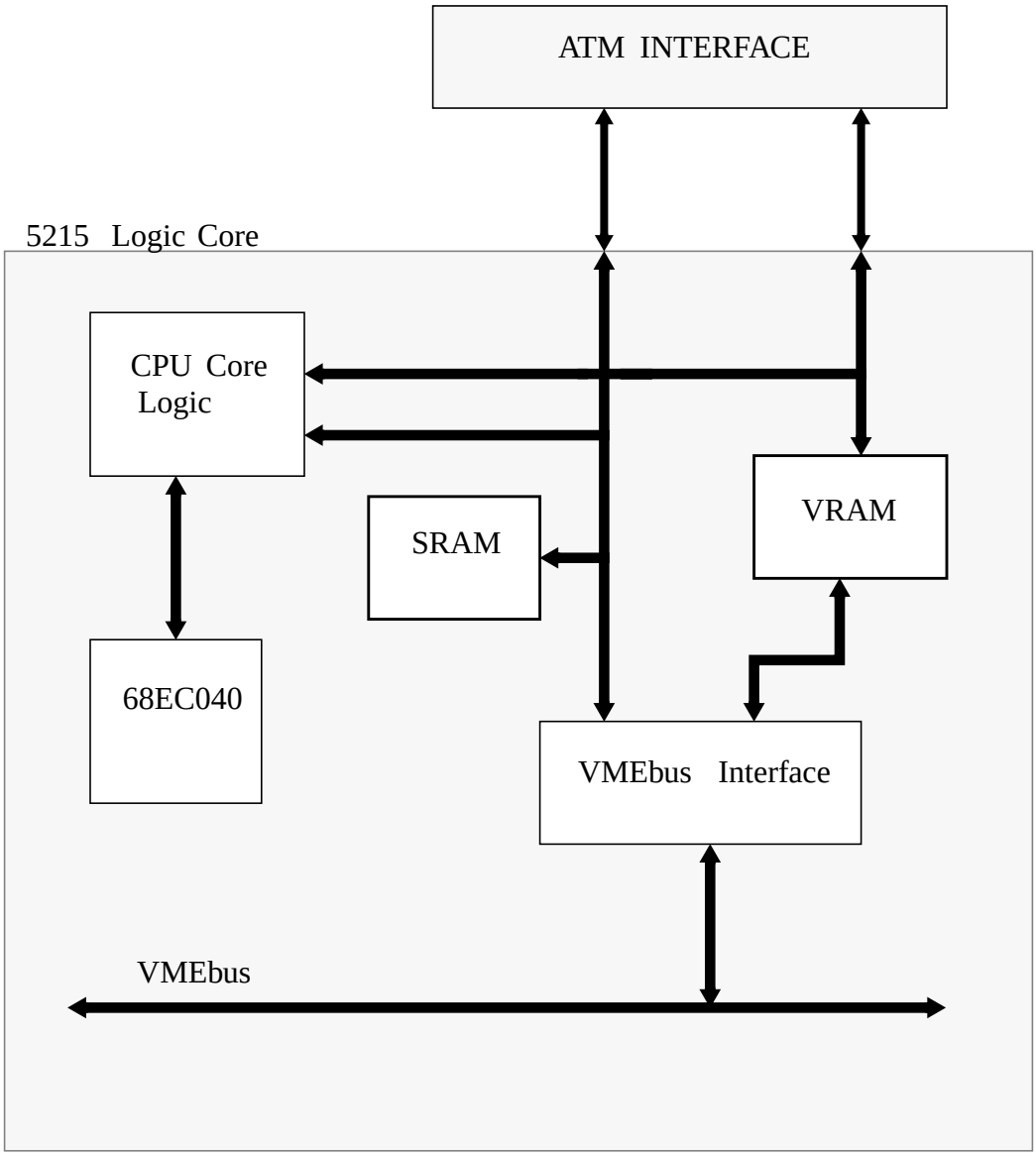


Figure 2-3. Block Diagram of the V/ATM 5215 Controller

5215 Daughtercard Block Diagram

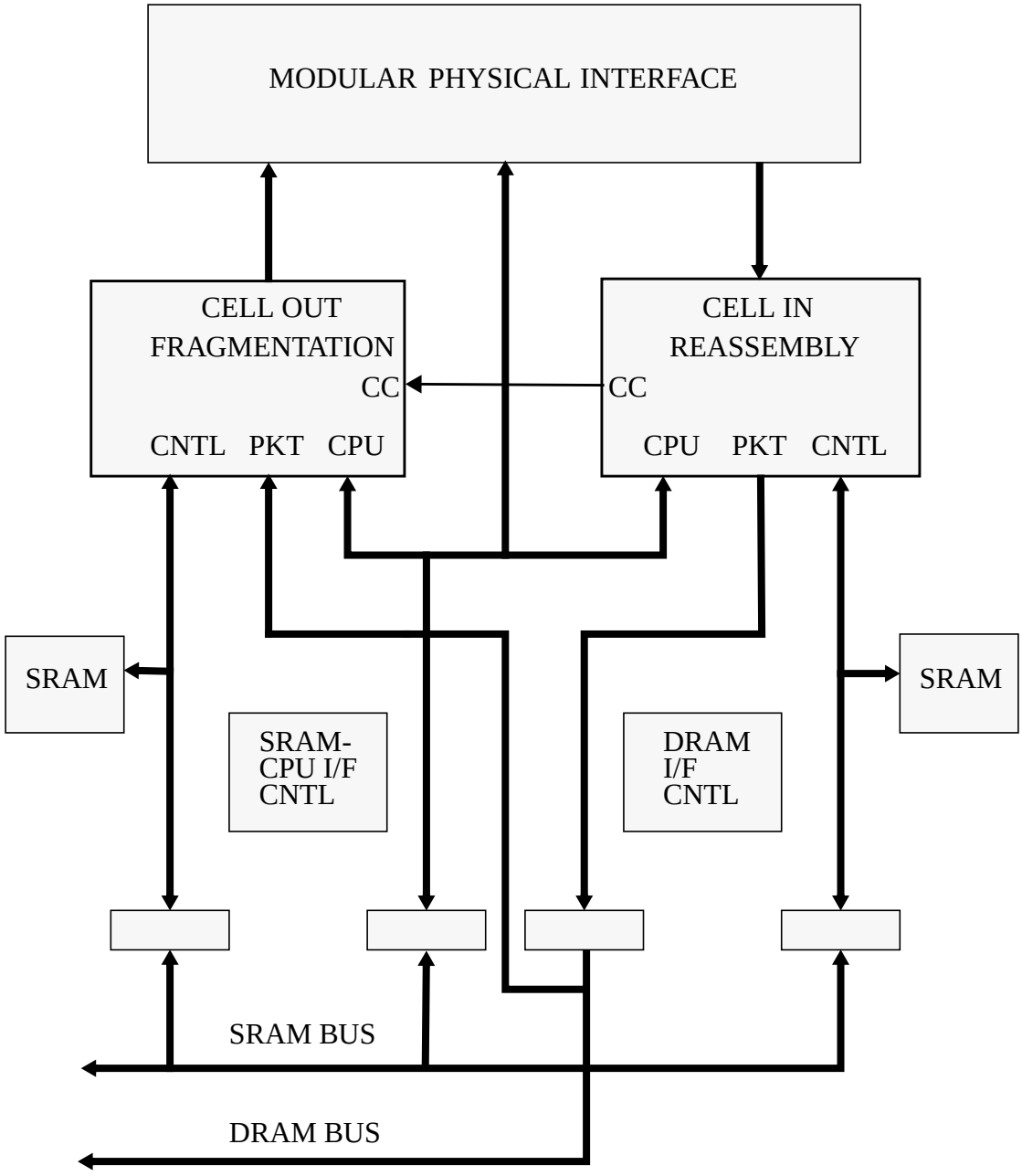


Figure 2-4. Block Diagram of the 5215 Daughtercard

Software Installation 3

Overview

This chapter contains the procedures for installing the ID1.x driver for IRIX version 5.3 and 6.2. Even though the installation procedures are the same for both versions, the correct driver must be loaded for the appropriate operating system. The drivers reside in their respective directories, `IRIX_5.3` and `IRIX_6.2`, on the installation CD-ROM.

The drivers provide broadcast/multicast addressing, allowing you to run legacy LAN (Ethernet, Token Ring, and FDDI) applications and protocols over an ATM network. Multiple Emulated LANs (ELANs) are supported in that a single board can join up to four different ELANs simultaneously.

The Interphase software supports the following:

- One VME ATM adapter per system
- User-Network Interface (UNI) 3.0/3.1 signalling specification for Switched Virtual Circuits (SVCs)
- Permanent Virtual Circuits (PVCs) based on RFC-1483
- ATM Forum LAN Emulation specification version 1.0
- Up to four LECs (LAN Emulation Clients) using both SVC and PVC communications
- One IP-over-ATM client (based on RFC-1577) using both SVC and PVC communications
- Optional Network Services
  - One LECS (LAN Emulation Configuration Server)
  - Up to eight LESs (LAN Emulation Servers)
  - One IP-over-ATM ARP server

Driver Requirements

Installing the driver requires a system with the following:

- IRIX 5.3 or 6.2 operating system
- Root password
- 1.0 MB space in /(`root`) file system
- 6.0 MB space in /`usr` file system
- CD-ROM drive
- IP address, IP name, broadcast address, and net mask for each client to be enabled

In addition, the following information must exist:

- Major device number for the `ia` hardware driver. The default setting is 60.
- Major device number for the `li` software driver. The default setting is 61.
- Short I/O address for the slot in which the adapter is installed. There are two slots. The address for slot #1 (the default) is 0xBF400000, and for slot #2 is 0xBF600000. See your hardware documentation for the location of the VME slots.

If any *Read Me First* documentation is in your installation kit, review it before installing the driver. Also, refer to the appropriate Silicon Graphics documentation as necessary during the installation process (for more information, see *Related Publications And Standards* on page 1). If you are not thoroughly familiar with ATM networking, see *ATM Technology Overview* on page 173.

If you have any questions about the installation that are not answered in this users guide and supporting documentation, see the assistance information at the front of this manual.

Configuring the Network Interface

We recommend that the IRIX network interface be configured prior to running the Interphase install script. The configuration procedures in this section are specific to the IRIX operating system and not to the Interphase driver. The instructions are generic in nature, and your setup may vary slightly. This manual assumes the person installing the driver is familiar with the administration of IRIX networking.

All the clients do not have to be activated. You only need to configure the actual clients that are going to be used on the network. Additional clients, if needed, can be configured at a later date without reinstalling the driver. For example, if you are going to use one LEC and the IP-over-ATM client, only two clients need to be activated at this time.

To configure the IRIX network interface for the clients, do the following:

1. For each client being activated (four LANE and one IP-over-ATM maximum), add the <*IP name*> and <*IP address*> to the hostname database file `/etc/hosts`.

   For example, a client with an IP address of *123.234.100.055* and an associated hostname of *irix-atm* is entered in the file as follows:

   ```
   123.234.100.055 irix-atm
   ```

2. Add the interface(s) to the kernel by editing the file `/etc/config/netif.options`

   Follow the directions provided in the file.

   The interface names used by the driver are:

   ```
   li0: LANE client 0
   li1: LANE client 1
   li2: LANE client 2
   li3: LANE client 3
   li32: IP-over-ATM client
   ```

Each client represents an individual entity that must be added to the system. For example, when editing the `/etc/config/netif.options` file, look for blank *if<number>* variables such as:

```
:if<X>name=
:if<X>addr=
```

where <X> is the next available interface number.

Set the variables to their network interface values, and then uncomment the lines by removing the colon, as shown in the following example:

```
if1name=li0
if1addr=irix-atm
```

where *li0* is from the above list of interface names for clients, and the *irix-atm* name is mapped to an IP address in the `/etc/hosts` file, edited in step **1**.

3. Create the interface configuration file.

The system boot procedure performs an `ifconfig` on all network interfaces specified in the `/etc/config/netif.options` file. This config uses command line options contained in the file `/etc/config/ifconfig-<X>.options` where <*X*> is the interface number used in step **2**.

For example, to create an ifconfig file for *if1* from the previous step, the file name is `/etc/config/ifconfig-1.options` with the content your network requires, such as:

```
broadcast 123.234.100.0 netmask \
0xffffff00 up
```

4. Repeat step **1** through step **3** for each network interface to be supported by the driver.

When the network interface is complete for all the clients you want to configure, continue with the next section.

Installing the Driver

The CD-ROM included with the adapter contains all of the files required to install the driver. With the network interface completed as explained in the previous section, install the driver as follows:

1. Log in as **`root`**.

You must know the current password.

2. Check the available disk space by entering:

```
df -k
```

There must be at least 1.0 MB of free space in the `/root` file system, and at least 6.0 MB of free space in the `/usr` partition.

3. There are two software packages for the VME ATM adapter. One package is for the IRIX 5.3 operating system, and the other package is for IRIX version 6.2. The packages are located in their respective directories, `IRIX_5.3` and `IRIX_6.2`, on the CD-ROM. If you are not sure which operating system is installed on your machine, enter the command:

   `uname -sr`

4. Determine whether you need to change the defaults for the installation script, `inatm`. The installation variables and default values are as follows:

Variable	Command	Description	Default
BINDIR	**`-b`** *(path)*	Directory for binaries	`/usr/sbin`
CFGDIR	**`-c`** *(path)*	Directory for config files	`/etc/config`
MANDIR	**`-m`** *(path)*	Directory for man pages	`/usr/share/catman/a_man/catl`
SHIO	**`-a`** *(addr)*	Adapter's short I/O address	`0xBF400000` for slot 1 (the short I/O address for slot 2 is `0xBF600000`)
IAMAJ	**`-i`** *(num)*	Major device number for ia hardware driver	`60`
LIMAG	**`-l`** *(num)*	Major device number for li software driver	`61`
IUSR	**`-u usr`**	Install *user*	`root`
IGRP	**`-g grp`**	Install *group*	`sys`

You can change the defaults in either of two ways:

– Permanently change the value of a variable by editing the `inatm` installation script.

– Temporarily override the default for one or more variables by using command line options.

For example, if you want to change the install script permanently to place the binary utility files in `/bin` instead of the default `/usr/sbin`, edit the `BINDIR` variable in the install script to read:

```
BINDIR = /bin
```

To override the same default as a one-time substitution during the current installation, use the command line option:

```
inatm -b /bin
```

Additional options are separated with a space. For example, if a value of 80 is needed for the IAMAJ variable, in addition to the BINDIR setting above, the multiple commands would be:

```
inatm -b /bin -i 80
```

5. Insert the CD-ROM in the local drive and enter one of the following:

   If IRIX version 5.3 is installed on your machine, enter:

   ```
   cd /CDROM/IRIX_5.3
   ```

   If IRIX version 6.2 is installed on your machine, enter:

   ```
   cd /CDROM/IRIX_6.2
   ```

6. Run the installation script to install the driver.

   a. Enter the command:

      ```
      ./inatm [option(s)]
      ```

      where **[options(s)]** is the command line options, if needed, as explained in step **4** on page 20.

   b. If an existing VME ATM driver is detected on your machine, the installation script detects it and prompts you to remove it.

      Enter **`yes`** if this occurs.

   c. You are prompted to confirm the parameters for installing the driver.

      Enter **`yes`** at the prompt. Otherwise, enter **`no`** and repeat this procedure beginning with step **4** on page 20.

   The installation script begins to execute and should run to completion without any interrupts or prompts. When complete, the following message appears:

```
The driver has now been installed
and the kernel reconfigured.
However, no ATM clients or
services have been enabled yet.
After rebooting the new kernel,
use CellView (/usr/sbin/cellview)
to fully configure, maintain, and
monitor your ATM environment.

You must reboot to use the new
driver/kernel

Reboot now (yes/no)?
```

Figure 3-1. Driver Installation Complete Message

7. Enter **`yes`** at the prompt.

8. Use CellView to configure the clients to run on the network, as described in *Configuring the Adapter Using CellView* on page 25. If the LECS and the LESs are to be on this adapter, you must enable their dialogs with the **Global** option from the main CellView dialog.

When all items are enabled and configured, exit the CellView utility and continue the installation with the procedures in the next section.

CAUTION

All configuration data for CellView is stored in the file /etc/atm/cvconf, for which an on-line man page is available. You are strongly encouraged to use the CellView utility to modify the parameters instead of editing them manually. If the parameters get out of sync, the driver may not work properly.

If for some reason the CellView utility will not run on your system, there are command line utilities (such as lec, les, and lecs) that can modify the software currently running. However, these commands do not change the permanent settings in the cvconf file. The effects of these commands are lost when the machine is turned off or restarted. Whatever the case, do not mix the running of command line and CellView utilities. Use one method consistently or unpredictable results may occur.

Automating Startup and Shutdown

The UNI signalling application must be started after the hardware device drivers have been loaded by the kernel. The application can be started manually after bootup, or the start procedure can be automated using the start-up script provided with the installation software. You can add both start-up and shutdown commands to the `/etc/init.d/network` file.

NOTE

The paths to the lec, sigd_start, and iadump files in the following sections must match the paths specified during the driver installation (the -b command line option). The paths to lec, signald, and iadump, if changed from the install script's default (/usr/sbin/), must be changed in the following examples, and also in the scripts: sigd_start and sigd_stop.

To automate the start-up and shutdown, edit the `/etc/init.d/network` file as follows:

1. Set the LANE client start-up by adding the following script ahead of the double semicolon (;;) that terminates the case `start` statement:

```
# ATM/ID1 startup
if $IS_ON atmcv && test -x /usr/sbin/cvconf; then
/usr/sbin/cvconf
fi
```

   The `inatm` install script creates the `/etc/config/atmcv` file with the contents set to `on`. This file is used by the $IS_ON test created above. To disable the configuration of the LANE clients at boot time, change the contents of the file to `off`. This change mandates that you configure each client manually after every boot or reboot.

2. Set the Signald startup by adding the following command lines to the end of the case `start` statement:

```
# ATM/SIGD startup
if $IS_ON atmsigd && test -x /usr/sbin/sigd_start; then
/usr/sbin/sigd_start
fi
```

   The `inatm` install script creates the `/etc/config/atmsigd` file with the contents set to `on`. This file is used by the $IS_ON test created above. To disable startup of the signaling application at boot time, change the contents of the file to **`off`**. This change mandates that you start the signald application manually after every boot or reboot.

3. Shut down the system.

   Most systems do not provide a hardware reset during a reboot. If the hardware is not reset during a reboot, the board remains active, and the kernel PANICs on receipt of the first packet.

   You can provide this hardware reset by adding the following script ahead of the double semicolon (;;) that terminates the case `stop` statement:

```
# ATM/LANE shutdown
if $IS_ON atmdown && test -x /usr/sbin/iadump; then
/usr/sbin/iadump shutdown
fi
```

CAUTION

If you do not add a shutdown command and the Kernel PANIC occurs, the only way to bring the workstation back up is to turn the power off, and then back on again.

   The `lane` install script creates the `/etc/config/atmdown` file with the contents set to `on`. This file is used by the $IS_ON test created above. To disable this hardware reset during shutdown, change the contents of the file to **`off`**. This change mandates that you reset the hardware manually prior to a reboot (or requires a power cycle before booting).

4. Reboot the system.

   A reboot is required to install, configure, and activate the adapter and driver. To reboot the system, do the following:

   a. Click the desktop manager's System Restart function, or enter:

   `/etc/reboot`

   b. Watch the console for messages and/or errors starting with `IASG` as the system reboots.

   These messages are copied to the system log file `/var/adm/SYSLOG`, where you can access them if needed.

The installation of the IRIX driver for the VME ATM adapter is complete.

Setting the European STM-1 Mode

To configure the adapter for European STM-1 mode, edit the file `/var/sysgen/master.d/li` as follows:

1. Locate the line:

   ```
   int ia_stm = 0;
   ```

2. Set the *ia_stm* variable to **1** instead of zero:

   ```
   int ia_stm = 1;
   ```

3. Run the command:

   ```
   /etc/autoconfig
   ```

4. Reboot the system.

Configuring the Adapter Using CellView

4

Overview

This chapter provides information about CellView, a cross-platform, GUI utility that enables you to configure clients and services for adapters across a wide range of operating systems. It explains how to use CellView on all compatible operating systems other than NetWare. (Your adapter might not support all of the operating systems that are compatible with CellView.)

NOTE

For ATM adapters that consist of a motherboard and 1 or 2 PMC ATM daughtercards, the term *adapter* in this chapter refers to the daughtercard.

This chapter describes the following:

- CellView features
- ATM address format
- Configuration requirements
- CellView startup procedure
- Setup procedures
- CellView statistics
- CellView global settings

Because of differences in operating systems, the example CellView dialogs shown in this chapter may vary slightly from the dialogs on your screen. However, the functions are the same. For example, you might use a button to enable a setting where the example shows a checkbox.

CellView Features

CellView enables you to:

- Configure signalling parameters
- Enable, disable, and configure up to four LAN Emulation Clients (LECs) and one IP-over-ATM client per adapter
- Set up PVC communications with other end stations
- Enable, disable, and configure up to eight LAN Emulation Servers (LESs) and one IP-over-ATM ARP server per adapter
- Enable, disable, and configure one LAN Emulation Configuration Server (LECS) per adapter

- Display operating statistics for:
  - SONET, signalling, and cell-level activity
  - ATM Adaptation Layer 5 (AAL5) activity
  - LEC membership and activity
  - IP-over-ATM activity
  - LECS and LES activity
- Set parameters for displaying and updating CellView dialogs

ATM Address

During configuration, you will need to know the end station's unique ATM address, as shown in the ATM Address field in Figure 4-1:

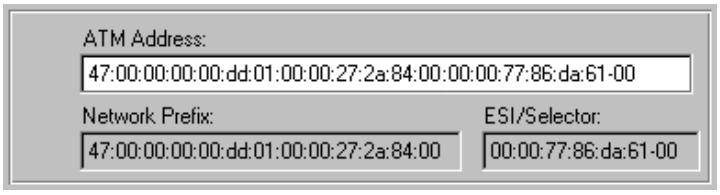


Figure 4-1. ATM Address

The Network Prefix and ESI/Selector fields show the separate ATM address components.

The network prefix is the 13-byte link address of the adapter, as set by the ATM switch.

The ESI/Selector consists of a unique end station identifier (ESI) and a pointer (selector) with the following values:

- The ESI value is the 6-byte MAC address of the adapter (00:00:77:86:da:61 in the example).
- The Selector value is a 1-byte pointer to clients and services located in the end station (00 in the example).

The Selector value for the end station is always 00. The Selector values for the clients and services located in the end station are as follows:

Item Number	Range of Selector Values
LEC1 – LEC4	0x00–0x03
LES1 – LES8	0x20–0x27
BUS1 – BUS8	0x40–0x47

Configuration Requirements

The degree of configuration needed for the adapter(s) depends largely on whether the services are already installed on the network. For example, if the LECS, LESs, and ARP server are running elsewhere on the network, you need to enable and configure only the local clients.

At least one LEC or IP-over-ATM client must be enabled and fully configured on each adapter before running applications on the network.

Before you begin the configuration procedures described in this chapter, the following prerequisite requirements must be met:

- At least one adapter is installed in the end station.
- The driver for your operating system is installed.
- The machine has been rebooted after the driver installation.

If any of these tasks are not complete, please complete them now. See the appropriate chapter(s) or addendum(s) for instructions.

Starting CellView

The startup procedure for CellView depends on the operating system you are running. The following table describes the startup procedure for each operating system:

Operating System	CellView Startup Procedure
AIX	Enter the command **`cellview`**
HP-UX	Enter the command **`cellview`**
Solaris	Enter the command **`cellview`**
UnixWare	Enter the command **`cellview`**
IRIX	Add to your path: **`/usr/sbin/`** Enter the command **`cellview`**
SCO	Add to your path: **`/etc/atm/bin`** Enter the command **`cellview`**
Windows 95	Select **Start.** Select **Programs.** Select **Control Panel.** Double-click the **CellView** icon.
Windows NT 3.51	Go to the **Main** Group. Open the **Program Manager.** Open the **Control Panel.** Double-click the **CellView** icon.
Windows NT 4.0	Select **Start.** Select **Settings.** Select **Control Panel.** Double-click the **CellView** icon.

When you start CellView, the main dialog appears:

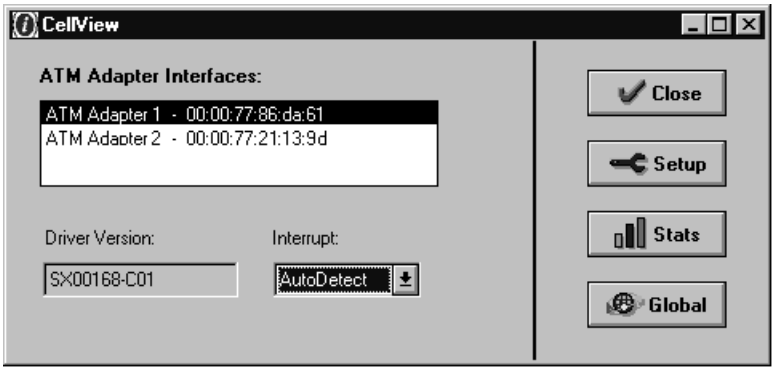


Figure 4-2. Main CellView Dialog

NOTE

If you are configuring a PCI or PMC ATM adapter under Windows NT, the main dialog contains a selection box for changing the PCI Interrupt value.

The main CellView dialog lists the adapter(s) installed in your computer and provides navigation buttons. All subsequent dialogs are *child* dialogs. The contents of each Setup and Statistics dialog apply to the adapter that is highlighted in the ATM Adapter Interfaces box.

The main CellView dialog provides the following selections:

Select...	To do this...
Interrupt (Windows NT only)	Set the PCI interrupt value for the adapter highlighted in the ATM Adapter Interfaces box. We recommend the default **AutoDetect** setting. If the driver fails to recognize the adapter with AutoDetect set, you might need to set a specific interrupt level.
Close	Exit CellView.
Setup	To configure: • Signalling • LECs • LESs (on the same adapter as the LECS) • LECS (on the same adapter as the LESs) • IP-over-ATM (1577) client and ARP server

Select...	To do this...
Stats	To display the status of: • Signalling • AAL5 • LECs • LESs • LECS • IP-over-ATM (1577) client and ARP server
Global	To configure: • Applications (parameters for Setup and Stats dialogs) • Connection limits for end stations (display only) • Debugging (enable tracking of API, Signalling, ILMI)

The setting **Enable Advanced Settings,** which is located in the Application dialog of Global settings, controls whether the LECS and LES tabs appear for editing and viewing in the Setup and Stats routines. If Enable Advanced Settings is set to **OFF** (the default), the tabs do not appear.

Setup

Use CellView Setup dialogs to configure your adapter(s) for use on the ATM network. You can configure up to four LECs and one IP-over-ATM client per adapter. The initial number of clients per adapter is specified during driver installation.

You can also configure one LECS and up to eight LESs per adapter for the ATM network. If you will be enabling the LECS and LESs, it is recommended that you configure these before configuring clients.

Following is an overview of the procedures to use to configure network services for each adapter in an end station. The topics in this section provide detailed information about each of these procedures.

1. On the main CellView dialog, highlight the adapter you are configuring, and select **Setup** to make dialogs available for setting up signalling, clients, and servers.
2. On the Signalling dialog, check that all signalling parameters conform to the capabilities of your ATM switch, and change settings, if needed.
3. **If the LECS and the LES(s) will be on this machine,** use the LECS and LES dialogs to configure these servers. (You might need to enable settings in the Global Application dialog to make these dialogs available.)
4. On the LEC dialog(s), configure the LECs specified for the adapter during driver installation.

5. On the 1577 dialog, configure the IP-over-ATM client and ARP server, if specified for the adapter during driver installation.
6. Exit CellView Setup.
7. **If you are running Windows NT,** exit CellView and reboot the machine.

The edits made with CellView are stored in a system file (`/etc/atm/cvconf`) on your hard disk. In Windows 95/NT systems, you must exit CellView and restart the machine to make these changes take effect. In UNIX-based systems, however, all changes to Setup dialogs, except UNI Signalling, are dynamic when you exit Setup. You can restart Signalling from the command line without rebooting the machine, as described on page 32, to make signalling changes take effect.

The following table shows where to find information about CellView Setup tasks:

For information about...	See page...
Setting Up UNI Signalling	30
Setting Up LAN Emulation Clients	32
Setting Up IP-over-ATM Client and Server	36
Setting Up a LAN Emulation Configuration Server	39
Setting Up a LAN Emulation Server	41

Setting Up UNI Signalling

The ATM adapter drivers are designed to operate with the UNI 3.0/3.1 signalling specification. All connections between two end stations are set up and torn down dynamically using SVC communications.

To set up UNI signalling parameters:

1. On the main CellView dialog, highlight the adapter you want to configure. Then select **Setup** to display the Signalling dialog:

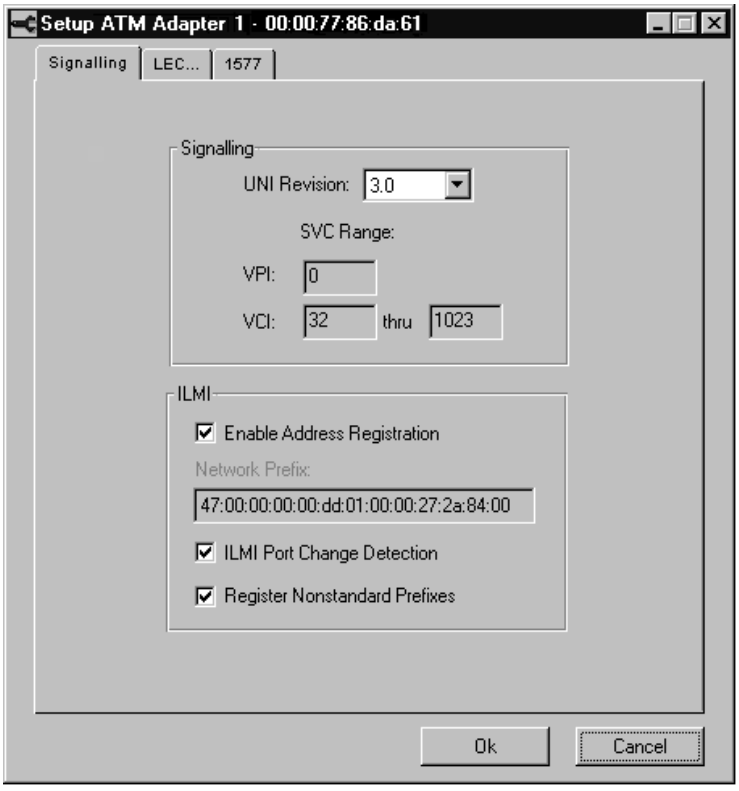


Figure 4-3. CellView Signalling Setup

2. Provide information in the Signalling dialog fields, as follows:

Use this field...	To do this...
UNI Revision	Select the UNI revision (UNI 3.0 or UNI 3.1).
SVC Range: VPI (Virtual Path Identifier)	Determine the VPI available to the driver for SVCs. The VPI is a one-byte field in the ATM cell header that, when combined with the VCI, forms an ATM address. Currently, only VPI 0 is supported.
SVC Range: VCI (Virtual Circuit Identifier)	Determine the range of VCIs available to the driver for SVCs. The range is fixed and cannot be edited. In the sample dialog (Figure 4-3 on page 31), circuits 0–31 are reserved for system communications. Circuits 32–1023 are available for station-to-station traffic.
Enable Address Registration	Enable the adapter to acquire its 13-byte network prefix at bootup from the switch. Disable this feature if your switch does not support address registration.
Network Prefix	Display or enter the adapter's 13-byte network prefix. If the Enable Address Registration feature is enabled, the network prefix is entered automatically at bootup, and is display-only. If the Enable Address Registration feature is disabled, obtain the network prefix from the switch and enter it manually.

Use this field...	To do this...
ILMI Port Change Detection	Enable the adapter to poll the ATM switch every 30 seconds, to advise that it is up and running. At bootup, the switch adds the host station to the active stations list. If the host station is turned off, polling stops, and the switch removes the host from this list. Disable this feature if your switch does not support ILMI polling.
Register Nonstandard Prefixes	Enable the ILMI to register the ATM Forum's well-known address for the LECS with the switch. The prefix in the well-known address usually differs from the ATM prefix for the adapter. Some switches might not support the registration of different prefixes through a single port. Disable this setting if your switch does not support mixed prefixes.

In UNIX-based systems, you must restart signalling for signalling setup changes to take effect.

- In IRIX operating systems, restart signalling with the command **`sigd_start`**
- In other UNIX operating systems, restart signalling with the command **`restart_signald`**

Setting Up LAN Emulation Clients

CellView provides a separate tabbed LEC dialog for each LEC setup. Using the LEC dialogs, you can set parameters for each LEC to establish communications with an LECS at bootup, or to contact a LES directly. You can also set up PVC links with other end stations.

To set up a LEC:

1. On the main CellView dialog, highlight the adapter you want to configure.
2. Select **Setup,** and then select the **LEC...** tab.
3. Select the **LEC<*x*>** tab (where <*x*> is the number of the LEC you are configuring) to display the appropriate LEC dialog:

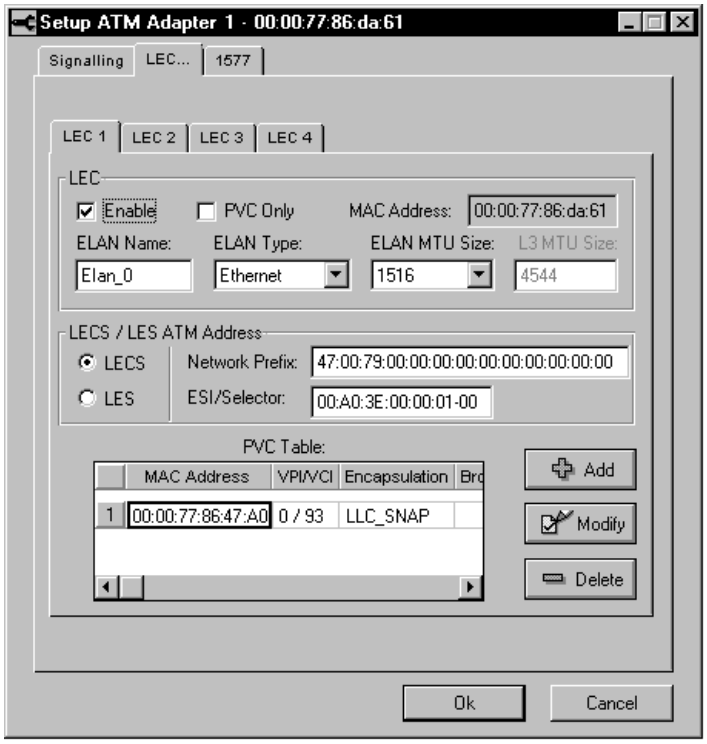


Figure 4-4. CellView LEC Setup

4. Enter information in the LEC dialog fields to configure the following:
   - LEC settings and LECS/LES ATM address settings
   - LEC PVC settings

The topics that follow provide information about configuring these settings.

LEC and LECS/LES ATM Address Settings

To configure LEC settings and LECS/LES ATM address settings, provide information in LEC dialog fields, as follows:

Use this field...	To do this...
Enable	Enable the LEC.
PVC Only	Set up PVC communications without enabling SVC activity. For PVC-only clients, the only other settings you need to configure are the L3MTU Size and the PVC Table.
MAC Address	View the MAC address of the LEC. LEC 1 is assigned the base address for the adapter. The remaining LECs use increments of the base address for their MAC addresses. If the LEC is enabled as Token Ring, the MAC Address is displayed in non-canonical format.

Use this field...	To do this...
ELAN Name	Identify the network name of the emulated LAN that you want the client to join. Some configuration servers require a name while others do not. The Interphase LECS and LESs require a name only in the following cases: • The Strict ELAN Name field is enabled on the LES dialog. • You need to differentiate between two or more LESs with the same MTU size and ELAN type. For exact requirements, see your server documentation. For information about setting up Interphase servers, see *Setting Up a LAN Emulation Configuration Server* on page 39.
ELAN Type	Select the ELAN type (Ethernet or Token Ring).
ELAN MTU Size	Select the requested ELAN MTU size. The client uses this entry when making a request to join a LES. The MTU returned to the client is equal to or less than the requested value. Recommended settings are: • ELAN with Ethernet edge device: **1516** • ELAN with Token Ring edge device: **4544** • ELAN with ATM end stations only: **9234** All nodes on the same logical subnet must use the same MTU size.
L3MTU Size	Force the MTU size to the entered value, rather than the value returned by the LANE server (LES). Done only for values other than zero (0). **If the client is in PVC-only mode (no LAN Emulation services), you must set the L3MTU size to the IP MTU of the network.** If the MTU is not based on the MAC layer, you must set the L3MTU size to the actual IP MTU of the network. For example, if the ELAN MTU Size of a Token Ring network is 4544, but the IP MTU is 2048, set the **L3MTU Size** to **2048.**
LECS/LES ATM Address	Select the mode of contact at bootup: • **LECS** contacts the LECS with the 20-byte ATM address in the Network Prefix and ESI/Selector fields. (The default address is the ATM Forum's well-known address.) • **LES** uses the 20-byte address in the ATM address fields to contact a LES directly.

LEC PVC Settings

Use the PVC Table in the LEC dialog to set up PVC connections between a client and other network end stations. Figure 4-5 illustrates the PVC Table for LECs:

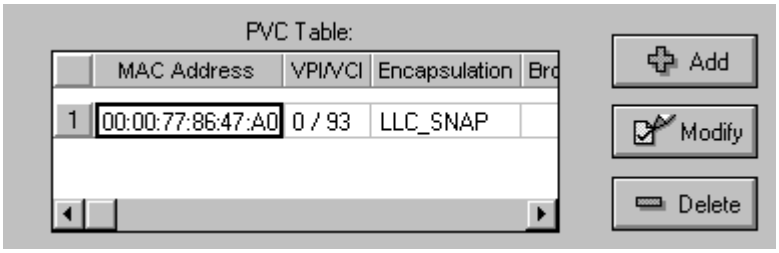


Figure 4-5. PVC Table in LEC Dialog

A 6-byte MAC address and a VCI number are associated with each PVC. Packets from the upper layers whose destination MAC addresses match an address in the PVC table are transmitted on that PVC.

In addition to providing setup information in the PVC Table, you must also consider the following when configuring PVCs for LECs:

- All PVCs must be configured in the adapters and the switch(es) manually.
- The MAC address of the target station must be mapped to a VCI in the local host.
- The MAC address of the local client must be mapped to a VCI in the target station.

You can use buttons in the PVC table block to add, modify, or delete PVC table entries, as described in the topics that follow.

Adding or Modifying LEC PVC Entries

To add or modify PVC table entries for the LEC:

1. Select **Add,** or highlight the appropriate entry and select **Modify.**

   The following dialog appears:

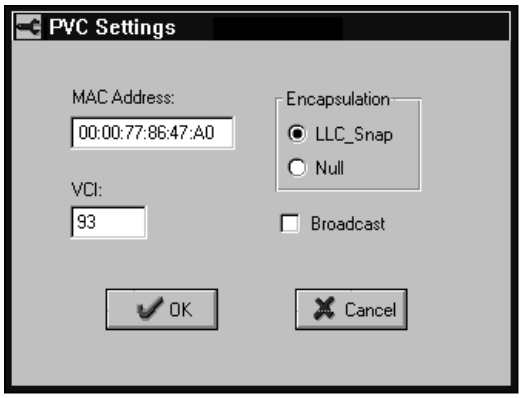


Figure 4-6. PVC Settings for LECs

2. Provide information in the PVC Settings dialog fields, as follows:

Use this field...	To do this...
MAC Address	Enter the 6-byte address of the target station. Use the natural format for the ELAN. For example, for a Token Ring ELAN, use non-canonical format.
VCI	Identify the VCI number that the LEC uses to communicate with the target. To determine the range of VCIs available, go to the Setup dialog for Signalling, and check the SVC Range in the VCI fields (Figure 4-3 on page 31). In most cases, the range is 32–1023. (The corresponding VPI cannot be edited and is fixed at a value of zero.)
Encapsulation	Select the encapsulation method (LLC_Snap or Null). The host and target must use the same encapsulation method. See *ATM Technology Overview* on page 173 for an explanation of these encapsulation methods.
Broadcast	Enable the entry as a Broadcast PVC. If enabled, all broadcast/multicast packets from the upper layers are transmitted on this PVC. A maximum of 16 Broadcast PVCs can be enabled on each LEC.

Deleting LEC PVC Entries

To delete a PVC table entry, highlight the entry, and select **Delete.**

Setting Up IP-over-ATM Client and Server

The IP-over-ATM client conforms to the RFC 1577 specification using Classical IP-over-ATM where no multicast or broadcast services are available.

Using the 1577 dialog, you can enable the Classical IP-over-ATM client and enable the host adapter as the Address Resolution Protocol (ARP) server for the network. You can also set up PVC links with other end stations.

To set up an IP-over-ATM client/ARP server configuration:

1. On the main CellView dialog, highlight the adapter you want to configure.
2. Select **Setup,** and then select the **1577** tab to display the 1577 dialog:

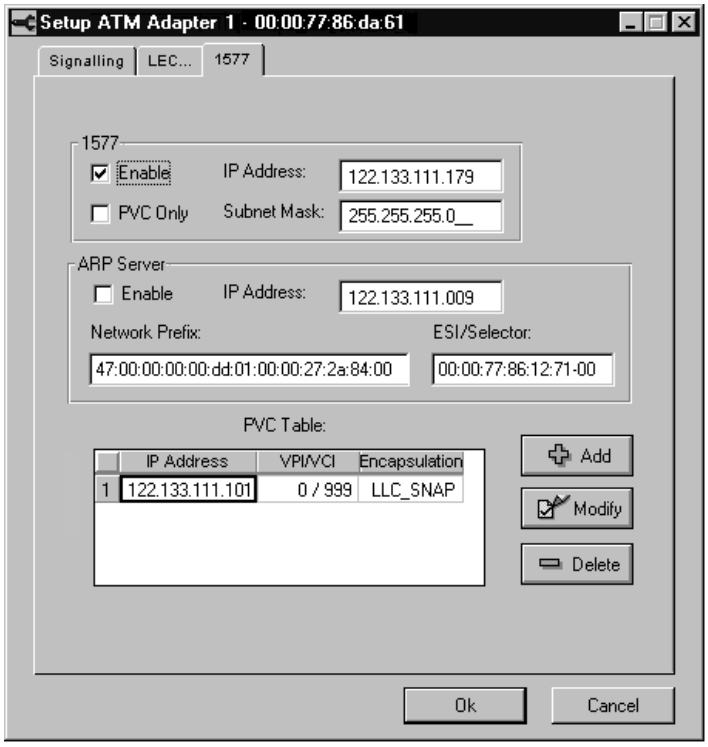


Figure 4-7. CellView IP-over-ATM Setup

3. Enter information in the 1577 dialog fields to configure the following:
   - IP-over-ATM client and ARP server settings
   - IP-over-ATM PVC settings

The topics that follow provide information about configuring these settings.

IP-over-ATM Client and ARP Server Settings

To configure IP-over-ATM client and ARP server settings, provide information in the 1577 dialog fields, as follows:

Use this field...	To do this...
Enable	Enable the Classical IP-over-ATM client.
PVC Only	Set up PVC communications without enabling the SVC activity. For PVC-only clients, the only other settings you need to configure are the PVC Table settings.
IP Address	Enter the client's IP address.
Subnet Mask	Enter the client's subnet mask identifier.
ARP Server: Enable	Enable the host adapter as the ARP server for your IP-over-ATM network.
ARP Server: IP Address	If the host adapter is not the ARP server, enter the IP address of the ARP server located elsewhere in the network.

Use this field...	To do this...
ARP Server: Network Prefix and ESI/Selector	If the host adapter is not the ARP server, enter the ARP server's ATM address, which consists of the Network Prefix and ESI/Selector. (For details about the address structure, see *ATM Address* on page 26.) The IP-over-ATM driver must know the ATM address of the ARP server. When the system is powered up, it calls the ARP server with this ATM address.

IP-over-ATM PVC Settings

Use the PVC Table in the 1577 dialog to set up PVC connections between the local client and other end stations in the IP-over-ATM subnetwork. Figure 4-8 illustrates the PVC Table for IP-over-ATM clients:

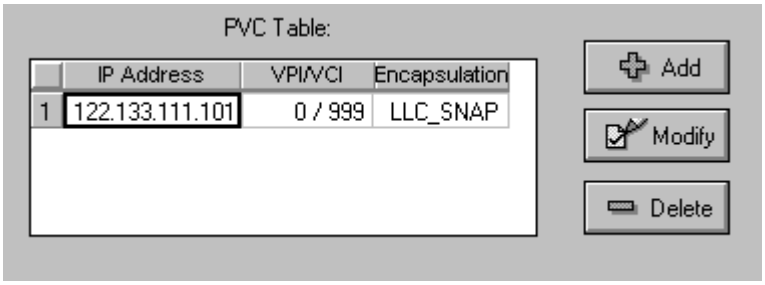


Figure 4-8. PVC Table in 1577 Dialog

In addition to providing setup information in the PVC Table, you must also consider the following when configuring PVCs for the IP-over-ATM client:

- All PVCs must be configured in the adapters and the switch(es) manually.
- The IP address of the target station must be mapped to a VCI in the local host.
- The IP address of the local client must be mapped to a VCI in the target station.

You can use buttons in the PVC table block to add, modify, or delete PVC table entries, as described in the topics that follow.

Adding or Modifying IP-over-ATM PVC Entries

To add or modify PVC table entries for an IP-over-ATM client:

1. Select **Add,** or highlight the appropriate entry and select **Modify.**

   The following dialog appears:

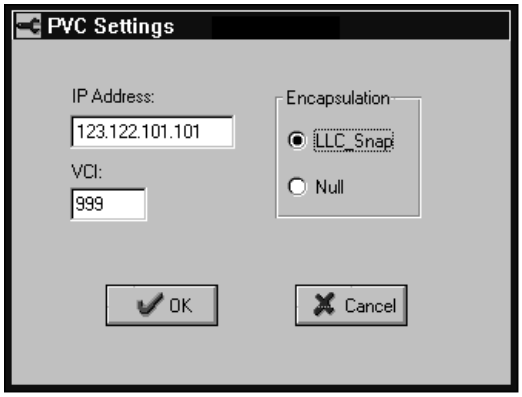


Figure 4-9. PVC Settings for IP-over-ATM Clients

2. Provide information in the PVC Settings dialog fields, as follows:

Use this field...	To do this...
IP Address	Enter the IP address of the target station.
VCI	Identify the VCI number that the IP-over-ATM client uses to communicate with the target. To determine the range of VCIs available, go to the Setup dialog for Signalling, and check the SVC Range in the VCI fields (Figure 4-3 on page 31). In most cases, the range is 32–1023. (The corresponding VPI cannot be edited and is fixed at a value of zero.)
Encapsulation	Select the encapsulation method (LLC_Snap or Null). The host and target must use the same encapsulation method. See *ATM Technology Overview* on page 173 for an explanation of these encapsulation methods.

Deleting IP-over-ATM PVC Entries

To delete a PVC table entry, highlight the entry, and select **Delete.**

Setting Up a LAN Emulation Configuration Server

To communicate on an ATM network, a LEC must join an emulated LAN (ELAN). The ELAN consists of a group of end stations that communicate among themselves with the same MTU size, the same frame format, and the same broadcast/multicast services. The LAN Emulation Server (LES) is the central server for the ELAN. The LAN Emulation Configuration Server (LECS) manages which LEC joins which LES.

The LECS must reside on the ATM adapter with the LESs it serves; it cannot communicate with LESs across the network. You must enable the LECS and associated LESs before configuring LECs that join LESs.

The LECS provides a central point of contact on the network for LECs at bootup. At bootup, the LEC sends the LECS a request for the type of ELAN it wants to join. The LECS responds with the current ATM address and the configuration parameters of the LES that serves the requested ELAN.

To set up the LECS:

1. If the LECS tab is not available, select **Global** from the main CellView dialog to display the Application Dialog. Then enable the **Enable Advanced Settings** field and click **OK.**
2. On the main CellView dialog, highlight the adapter you want to configure.
3. Select **Setup,** and then select the **LECS** tab to display the LECS dialog:

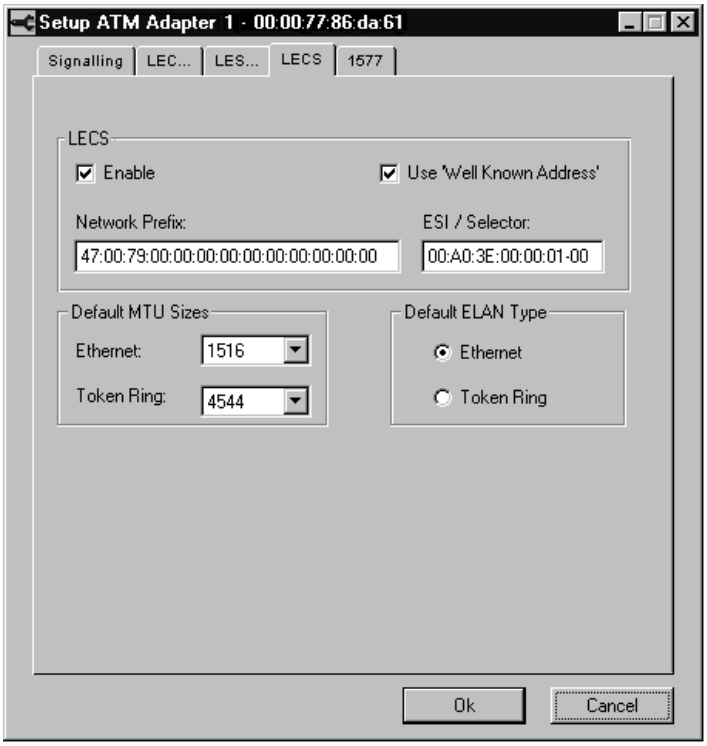


Figure 4-10. CellView LECS Setup

4. Provide information in the LECS dialog fields, as follows:

Use this field...	To do this...
Enable	Enable the LECS and all permanent mappings of LECs to specific LESs.
Use 'Well Known Address'	Enable the ATM Forum's well-known address to be loaded to the Network Prefix and ESI/Selector fields. When enabled, these address fields cannot be edited.

Use this field...	To do this...
Default MTU Sizes	Select the default Ethernet or Token Ring ELAN MTU size to assign to a LEC if the LEC's configuration request omits the MTU size. If the request also omits the ELAN type, the LECS will attempt to assign the LEC to a LES of the default ELAN type with an MTU size equal to or less than the requested size.
Default ELAN Type	Select a default ELAN type to assign to a LEC if the LEC's configuration request omits the ELAN type. The LECS will attempt to assign the LEC to a LES of the Default ELAN Type with an MTU size equal to or less than the requested size.

Setting Up a LAN Emulation Server

To communicate on an ATM network, a LEC must join an emulated LAN (ELAN). The ELAN consists of a group of end stations that communicate among themselves with the same MTU size, the same frame format, and the same broadcast/multicast services. The LAN Emulation Server (LES) is the central server for the ELAN. It provides address registration, address resolution, and broadcast services for all ELAN members. The LAN Emulation Configuration Server (LECS) manages which LEC joins which LES.

The driver allows up to eight LESs to be enabled per adapter at one time (allowing adapter support for up to eight ELANs). LESs must reside on the ATM adapter with the LECS that serves them; they cannot communicate with the LECS across the network. You must enable a LES and its associated LECS before configuring LECs to join the LES.

When a LEC joins a LES, the MAC-ATM relationship of the client is registered with the LES. The LES maintains a running account of these addresses to resolve queries from ELAN members. The connection between the LES and each LEC is a permanent, open connection as long as the client is running.

The LES maintains a table of client information that it uses to provide address resolution services to the ELAN. When an end station is disconnected or turned off, the LES removes client(s) from the table automatically. Therefore, relocating an end station is a matter of physically moving the hardware and connecting it to the same or another ATM switch. At bootup, the client(s) rejoin the previous LES with a new ATM address.

CellView provides a separate tabbed LES dialog for each LES-ELAN setup.

To set up a LES:

1. If the LES... tab is not available, select **Global** from the main CellView dialog to display the Application Dialog. Then enable the **Enable Advanced Settings** field, and click **OK.**
2. On the main CellView dialog, highlight the adapter you want to configure.
3. Select **Setup,** and then select the **LES...** tab.
4. Select the **LES<*x*>** tab (where <*x*> is the number of the LES you are configuring) to display the appropriate LES dialog:

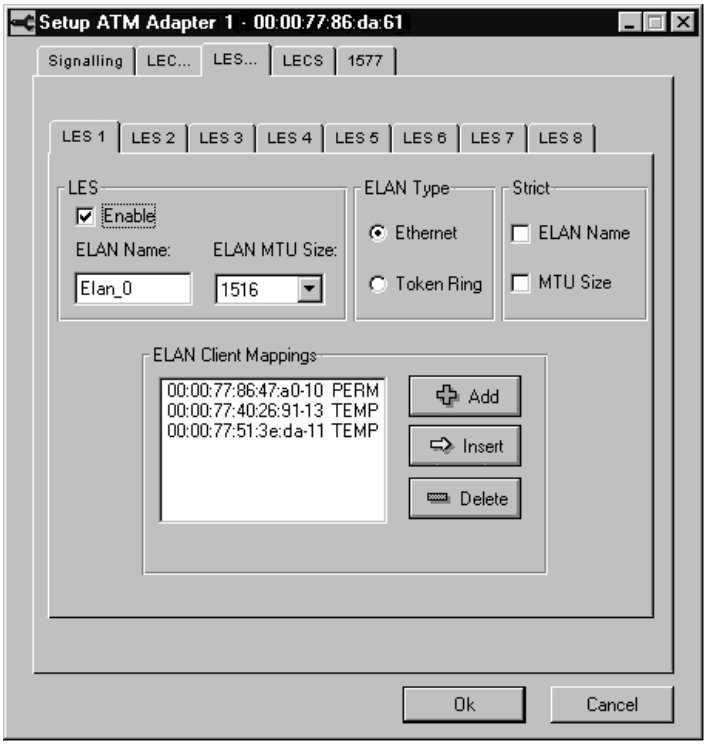


Figure 4-11. CellView LES Setup

5. Enter information in the LES dialog fields to configure the following:
   - LES settings
   - ELAN Client Mappings settings

The topics that follow provide information about configuring these settings.

LES Settings

To configure LES settings, provide information in the LES dialog fields, as follows:

Use this field...	To do this...
Enable	Enable the LES.
ELAN Name	Enter the name of the ELAN for which the LES provides services (not required for a LEC to join the group). If the ELAN Name setting in the Strict block is enabled, the ELAN name in the LEC request must exactly match this name (case-sensitive). Otherwise, the server prevents the LEC from joining the group.

Use this field...	To do this...
ELAN MTU Size	Select the MTU size for the ELAN for which the LES provides services. Recommended options are: • ELAN with Ethernet edge device: **1516** • ELAN with Token Ring edge device: **4544** • ELAN with ATM end stations only: **9234** If the MTU Size setting in the Strict block is enabled, the request from the LEC must exactly match the ELAN MTU Size. Otherwise, the server prevents the LEC from joining the group.
ELAN Type	Select the ELAN type (Ethernet or Token Ring) to use for all communications between end stations. If the ELAN includes an edge device to a legacy LAN, the ELAN type must be set to interface with that edge device.
Strict: ELAN Name	Specify that the ELAN Name in the LEC's request must exactly match the entry in the ELAN Name text field on this dialog (case-sensitive).
Strict: MTU Size	Specify that the ELAN MTU Size in the LEC's request must exactly match the entry in the ELAN MTU Size selection field on this dialog.

ELAN Client Mappings Settings

The ELAN Client Mappings block of the LES dialog shows the ESI/Selector value for each LEC currently assigned to the LES. The clients can be *permanently* or *temporarily* assigned to the LES, based on settings configured in this LES dialog or in the LEC dialog. Figure 4-12 illustrates the ELAN Client Mappings block:

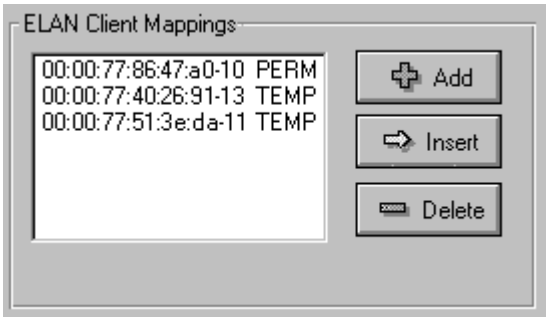


Figure 4-12. ELAN Client Mappings Table in LES Dialog

ELAN Client Mappings use two client classifications:

- **PERM** means the client is permanently assigned to the LES (using the Add or Insert button on this dialog). A permanently-assigned client can join only its assigned LES. This gives network administrators control in assigning clients to specific ELANs.
- **TEMP** means the client is temporarily assigned to the LES based on the configuration parameters requested by the client. This entry is removed when the client leaves the ELAN.

The ELAN Client Mappings table obtains the list of permanently-assigned clients from a table maintained by the LECS. When the LECS is enabled, it checks for a permanent status each time a client attempts to join the network, regardless of whether the client is configured to contact the LECS or directly contact a LES.

Use the ELAN Client Mappings block of the LES dialog to permanently assign or remove LES client assignments.

Both the Add and Insert buttons in the ELAN Client Mappings block enable you to permanently assign clients to the LES.

- Use the **Add** button to enter the client's ESI/Selector value.
- If the client is currently assigned or running on another ELAN, it is more convenient to use the **Insert** button to move the client to this LES.

You can permanently assign a client that is permanently assigned to another LES, temporarily assigned to another LES, or currently not a member of any ELAN.

Adding a Client

To add a client to a LES:

1. Select **Add** to display the following dialog:

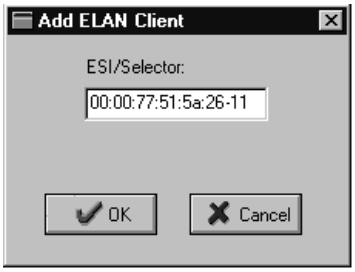


Figure 4-13. Add Permanent Clients to a LES

2. Enter the ESI/Selector value for the client.

   If you are not familiar with the structure of this address, see *ATM Address* on page 26.

3. Select **OK.**

This procedure permanently adds the new entry to the LES's ELAN Client Mappings table. Also, if the client was a member of another ELAN, this procedure removes the client from that ELAN's LES. When the client attempts to join the network again, it can join only the newly-assigned LES.

Inserting Clients

To move clients from another LES to the LES being configured:

1. Select **Insert** to display the following dialog:

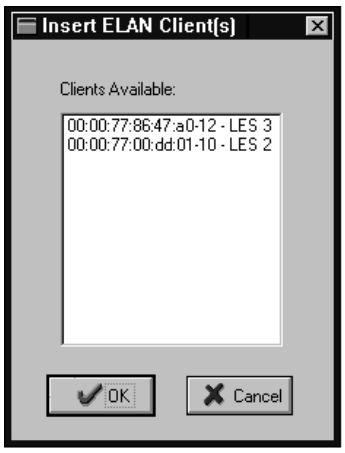


Figure 4-14. Insert Permanent Clients

   This dialog lists all permanent and temporary clients that are assigned to other LESs.

2. Highlight the clients you want to move to the LES being configured, and select **OK.**

This procedure permanently adds the new entries to the LES's ELAN Client Mappings table and removes the client from other LESs.

Deleting a Client

To remove a client from the LES, highlight the client in the ELAN Client Mappings block, and select **Delete.**

If the client was permanently assigned, it is removed from the ELAN served by the LES and from the table of permanent clients maintained by the LECS. It can attempt to join the network again any time after deletion.

If the client was temporarily assigned, it is removed from the ELAN served by the LES. When it attempts to join the network again, it will be assigned to a LES based on requested parameters. (This will be the same LES unless changed in the LEC Setup dialog since the last startup.)

Statistics Information

The dialogs in the CellView Statistics group provide statistics about installed adapters that can help you check for proper driver operation and troubleshoot network problems.

To see statistics about an installed adapter, select the adapter on the main CellView dialog, and then select **Stats.** To see statistics for an LECS or LESs located on an adapter, you might first need to make the LECS and LES tabs available in the Statistics group. To do so, on the main CellView dialog select **Global;** then select the **Application** tab and enable the **Enable Advanced Settings** feature.

Verifying Driver Installation

You can use information in the Statistics dialogs to verify that the driver is up and running properly. When the end station boots up, certain systems communications must occur between the client and the server (through the switch) so the client can log on to the network. Statistics routines monitor the state of client and server traffic, as well as signalling and AAL5 traffic.

To do a quick check of an adapter's operating statistics:

1. Start CellView.

   For instructions, see *Starting CellView* on page 27.

2. On the main CellView dialog, if more than one adapter is installed in the machine, select an adapter.

3. Select **Stats**, and check the following indicators in the Signalling dialog:
   - All three graphical LEDs in the Signalling State box should be green.
   - In the State Detail box, the ILMI, QSAAL, and Signalling field values should be similar to:

     `ILMI:` **`ILMI Registered`**

     `QSAAL:` **`Data transfer ready`**

     `Signalling:` **Signalling ready**
   - In the Signalling Statistics box, the Frames In and Frames Out columns should indicate traffic.

4. Select the **AAL5** tab, and check the following indicators:
   - All six graphical LEDs in the SONET box should be green.
   - The fields in the AAL5 Statistics box should indicate traffic.

5. For each LEC enabled on the adapter, select the **LEC** tab, and check the following indicators:
   - The LEC graphical LED should be green.
   - The Tx Packets and Rx Packets fields of the display boxes should indicate traffic.

6. If the IP-over-ATM client is enabled, select the **1577** tab, and check the following indicators:
   - The 1577 graphical LED should be green.
   - The 1577 Statistics fields should indicate traffic.
7. If more than one adapter is installed in the machine, repeat steps **2** through **6** for each additional adapter.

If you find an item in error, the driver might be incorrectly installed or configured.

Troubleshooting

You can use the Statistics dialogs to provide helpful information when you contact Interphase support personnel concerning ATM network problems. Statistics dialogs display network statistics gathered by CellView, such as signalling and traffic information, for the clients and servers on the adapter(s) installed in an end station.

CellView gathers network operation statistics for local Interphase clients and servers only. It does not gather statistics from switches or from other end stations.

Global Settings

Settings for global routines control CellView parameters that are common to all adapters. You are not required to restart end stations to make global settings take effect. Use Global settings to:

- Control application display settings
- Display signalling connection settings in drivers
- Produce information to use for debugging

The following table shows where to find information about CellView Global tasks:

For information about...	See page...
Controlling Application Display Settings	47
Displaying Signalling Connections Settings	49
Producing Debugging Messages	50

Controlling Application Display Settings

Use the Application dialog to control display settings for CellView Setup and Statistics routines, such as refresh rates and tabbed dialogs available for setup and statistics routines.

To control display settings:

1. From the main CellView dialog, select **Global** to display the Application dialog:

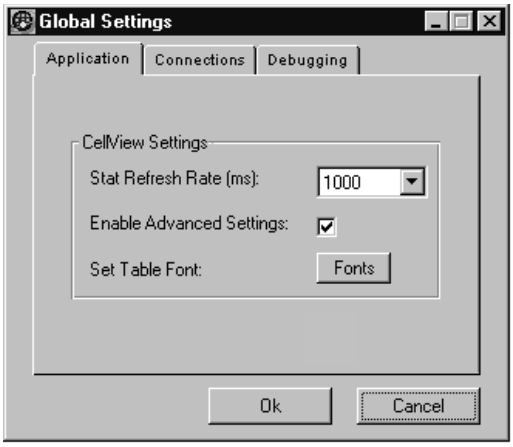


Figure 4-15. CellView Global Application

2. Provide information in Application dialog fields, as follows:

Use this field...	To do this...
Stat Refresh Rate	Control the update or refresh rate, in milliseconds, of the data fields in the Statistics dialogs.
Enable Advanced Settings	Activate the tabs for the LECS and the LES in the Setup and Statistics dialogs. This setting does not affect the enable/disable condition of the ELAN servers. The tabs can be hidden from view while the servers remain enabled.
Set Table Font	Select the font used for tables in dialogs.

Displaying Signalling Connections Settings

Use the Connections dialog to display the signalling connections configuration in adapter drivers. Currently, these settings cannot be changed; they are preset in the drivers.

To display the Connections dialog, select **Global** from the main CellView dialog, and then select the **Connections** tab:

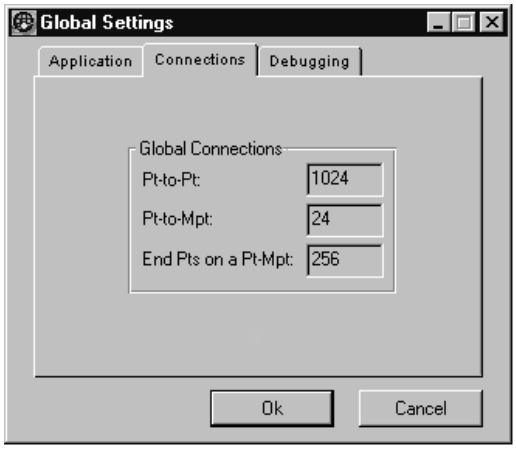


Figure 4-16. CellView Global Connections

The following table describes Connections dialog fields:

Use this field...	To do this...
Pt-to-Pt	Display the maximum number of SVC connections with other end stations that are allowed simultaneously.
Pt-to-Mpt	Display the maximum number of SVC multipoint connections that are allowed simultaneously. These are used primarily between LECs and a LES/BUS.
End Pts on a Pt-Mpt	Display the maximum number of end points on each multipoint connection. This affects the number of clients allowed on the same ELAN.

Producing Debugging Messages

Use the Debugging dialog as a troubleshooting aid for the adapter. If you have difficulty establishing connections with other network nodes or establishing ILMI connectivity with the switch, enable the appropriate debug options. The debug options produce signalling debug messages from various modules to help isolate the problem. Be sure to disable the options when debugging is complete.

To produce Debugging messages:

1. From the main CellView dialog, select **Global,** and then select the **Debugging** tab.

Figure 4-17. CellView Debugging Options

2. Select Debugging dialog options, as follows:

Use this field...	To do this...
API	Track the Signalling Application Program Interface (API). These are the routines used by the application software to manage SVCs.
Signalling	Track the Signalling PDUs exchanged by the host Signalling and the switch.
ILMI	Track the IME PDUs exchanged by the host Signalling and the switch. Useful in debugging ILMI Address Registration problems.

Functional Overview 5

Overview

The 5215 firmware interface consists of two major parts used for initializing and operating the controller. These are the Common Boot (CB) Interface and the Host Adapter Ring Interface (HARI) Interface. This chapter provides a brief overview of how the firmware interface works. It also describes the procedures and requirements for submitting commands that affect board operation.

Firmware Interface

The Common Boot Interface is a monitor that begins to run when the controller is first powered up or reset. A general-purpose protocol, it provides a deterministic start-up sequence for resident or downloadable host/board interfaces. It functions as a distinct entity in the controller's firmware architecture.

The Common Boot interface provides a mechanism to boot, initialize, and run controller diagnostics on the 5215. It is also used to download and execute optional firmware images and to permanently store images in *Flash Memory.* The host uses the Common Boot interface to submit the characteristics of the initial HARI channel pair needed to boot the HARI Interface on the 5215.

The host issues commands to the 5215 via the intelligent HARI Interface. This interface is implemented in the 512-byte *Short I/O* space on the 5215. The term *short I/O* refers to a block of on-board memory configured to operate in the VMEbus short supervisory I/O address space. This memory is shared across the bus between the host CPU and the 5215 firmware. Figure 3-1 depicts how short I/O and the HARI channel descriptors can be viewed as part of the Communications Buffer:

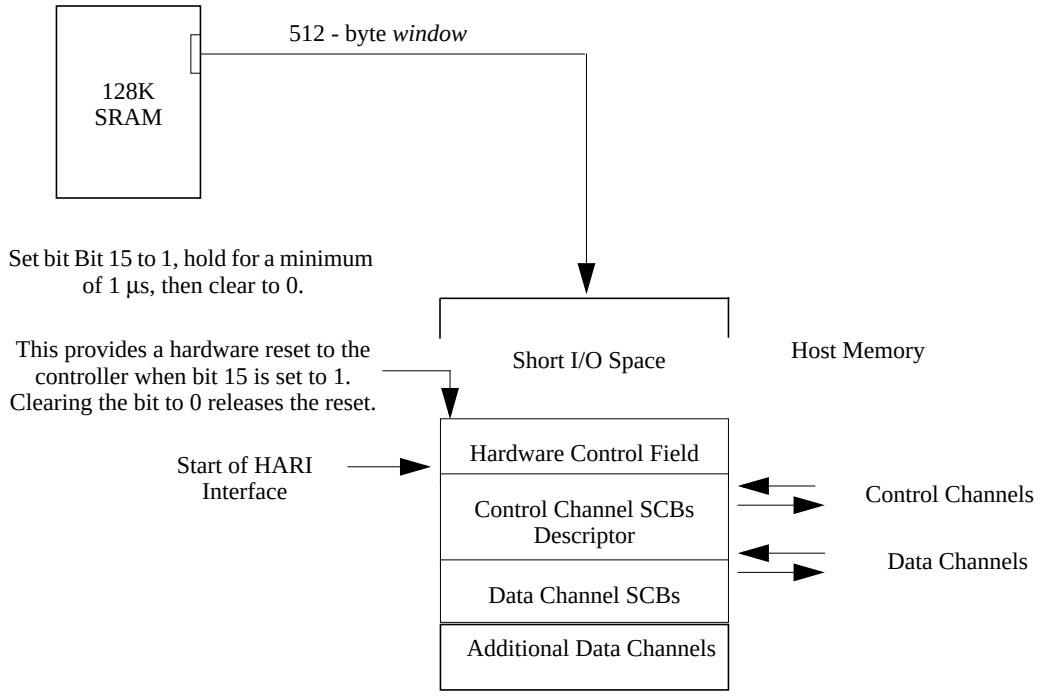


Figure 5-1. Short I/O as a Window into the 5215's Communications Buffer Address Space

System Requirements

To interface with the 5215, the host must meet the following requirements:

1. Although the 5215 supports 32-bit access to short I/O, some systems will only perform 16-bit transfers. If the system breaks short I/O into 16-bit transfers, the following requirements must be observed:
   - To transfer a 32-bit quantity to/from short I/O, the host must place the MSW (Most Significant Word) in the low-order address and the LSW (Least Significant Word) in the high address.
   - The host must have at least 16-bit access into the controller's on-board shared memory (*short I/O*) space. In addition, it must be able to read or write a 16-bit quantity in one operation without leaving an access window between bytes. The 5215 will allow the host to transfer a 32-bit quantity in one operation.
2. At least 512 bytes of VMEbus short I/O address space must be available on the system VMEbus for use by the HARI Interface.
3. The byte ordering of commands must be *big-endian* (that is, Most Significant Byte first).

The Common Boot Interface 6

Overview

The host uses the Common Boot interface to submit the characteristics of the initial HARI channel pair needed to boot the HARI Interface on the 5215. This initial channel pair is used to create additional channels after HARI is running. Once HARI boots successfully, the Common Boot interface exits until the board is reset. The host can perform a variety of other tasks when in Common Boot mode. These include:

- Performing extended diagnostics
- Downloading code for on-board execution. This feature is for special applications only. It is not needed to boot the native HARI Interface stored in controller firmware.

This chapter describes how to use the Common Boot interface to get the HARI Interface up and running on the 5215 controller. It also details various commands that you can use when booting the controller.

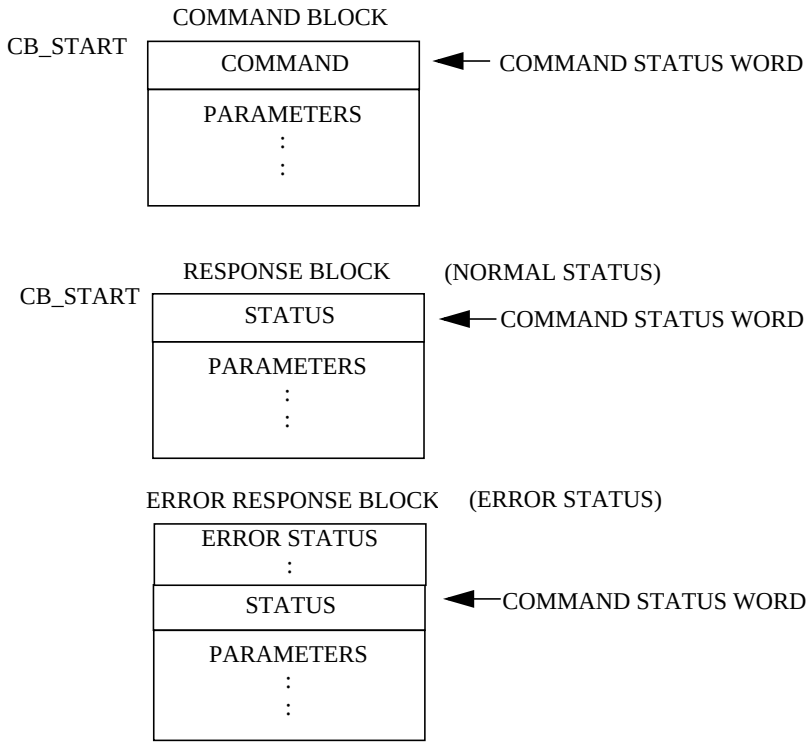


Figure 6-1. Common Boot Command/Response Format

After the Common Boot interface is active and the Command Status Word is *CBOK*, the host issues commands by writing over the Command Status Word (shown in Figure 6-1) with one of the Common Boot commands listed in *Common Boot Commands* on page 57.

If the command to be issued has parameters, these parameters must be written into shared memory before the command code is entered. After the command is issued, the controller processes the command. Meanwhile, the host polls the Command Status Word for CBOK or FAIL, indicating the controller has completed the command and returned the appropriate response.

While waiting for the host to issue commands, the controller also polls the Command Status Word. When it detects the Command Status Word has been overwritten, it will read the value and compare it with a list of valid commands. The Common Boot interface will record the command in the Error Status Block (shown in Figure 6-1 on page 53), then execute it. Recording the command in the Error Status Block provides a way to determine what command caused an error or failure. Commands execute until completion, and no timeout facility is provided by the controller. If the host must provide a timeout, such as during power-up, the following two-step approach is recommended:

1. The host first determines if the board is alive using the CB_HERALD pattern. Once this pattern is written in short I/O, the host can be confident of the correct address configuration, and the likelihood of either CBOK or FAIL after a command is almost certain. The CB_HERALD pattern will be presented in short I/O within 1.5 seconds from power-up.
2. Wait 10 seconds for CBOK or FAIL before timing out.

Some diagnostic commands may take several seconds to complete.

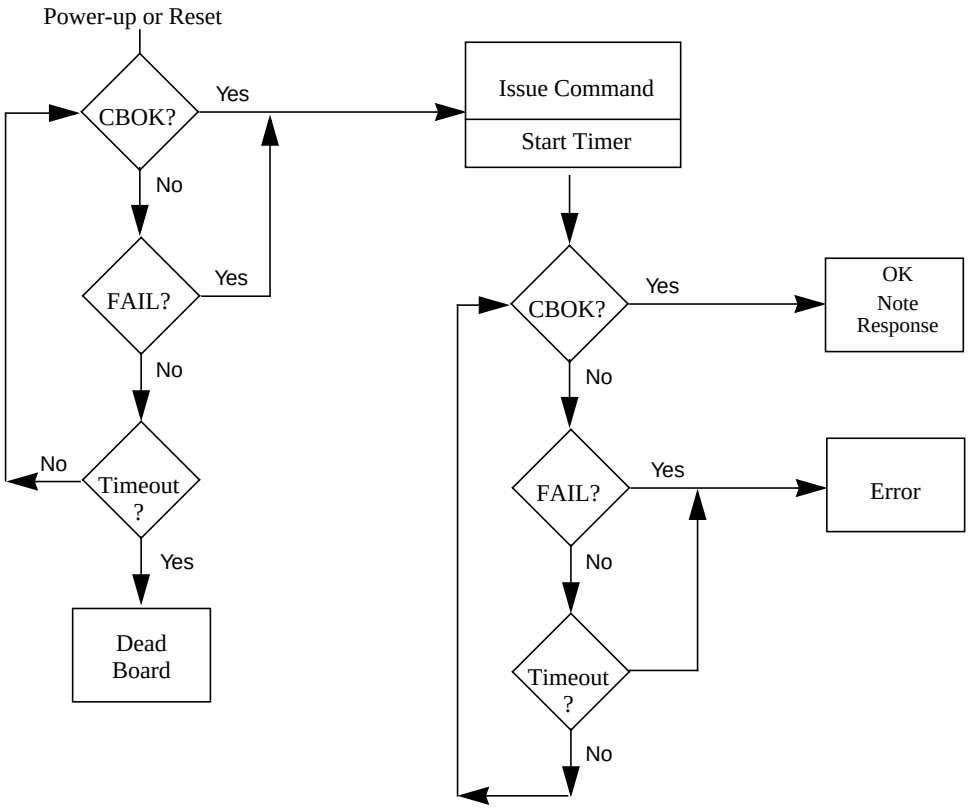


Figure 6-2. Polling the Command Status Word for CBOK, FAIL, or Response

Unless you need to perform diagnostics, or download code, the Common Boot may simply be used to boot the HARI Interface. In this scenario, the following occurs when the controller is powered up or reset:

- The Common Boot starts up and fills shared memory with CB_HERALD (see the following section, *Bootup and Reset Sequence*).
- The host issues the BOOT command to boot the HARI Interface.

The BOOT command allows you to create the initial control send and receive channels. These structures enable the host to interact with the controller using the HARI Interface. Additional channels may be established once HARI has booted.

Bootup and Reset Sequence

The following sequence of events occurs each time the 5215 is powered up or reset.

1. The 5215 executes its bootstrap code and performs a series of power-up diagnostics. Once the 5215 Common Boot interface is active, LED1 turns green.
2. The Common Boot Interface starts up and fills the 512-byte short I/O space with a 32-bit value called CB_HERALD. The CB_HERALD contains information about the start of the Common Boot interface and the total size of the controller's short I/O shared memory space. The format of CB_HERALD is as follows:

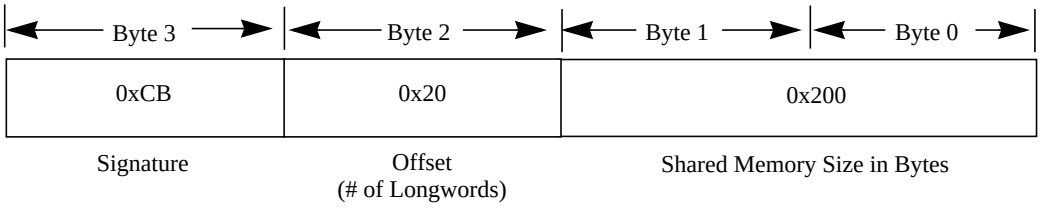


Figure 6-3. CB_HERALD Format

- Bytes 1 and 0 contain the size of 5215's 512 byte shared memory space.
- Byte 2 specifies the offset (in longwords) where the Common Boot interface starts in shared memory. The first longword of the Common Boot command area is referred to as the *Command Status Word*.
- Byte 3 contains the Common Boot Signature, which is a hex CB (0xCB) or 203.

If the power-up diagnostics completed successfully, *CBOK* will be written to the Command Status Word. CBOK is defined to be a 32-bit ASCII CBOK, or 0x43424F4B.

Figure 6-4 depicts the controller's shared memory space after the controller initializes successfully:

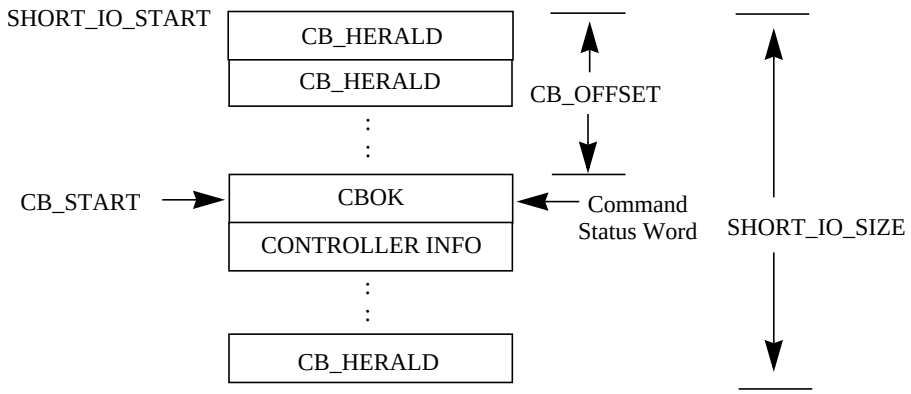


Figure 6-4. Format of Shared Memory after CBOK

If any power-up test fails, the controller writes the ASCII value **FAIL** (0x4641494C) to the Command Status Word, followed by a set of fields describing the failure. Figure 6-5 depicts the format of shared memory when FAIL occurs during power-up diagnostics:

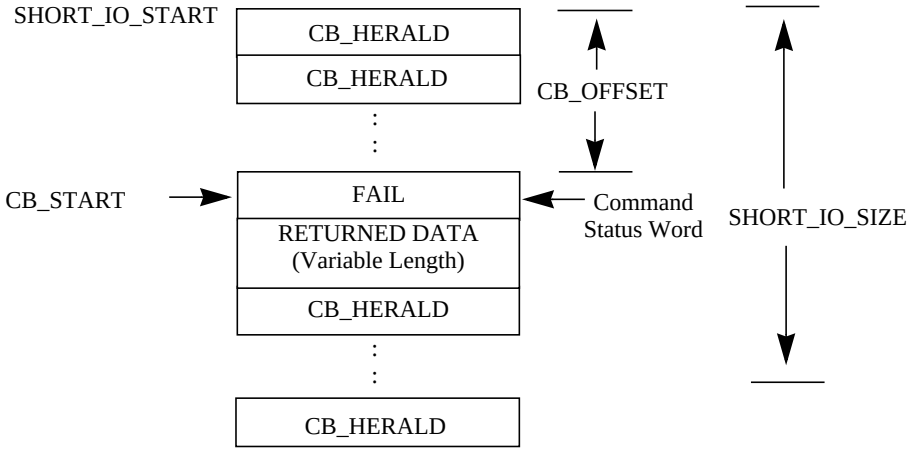


Figure 6-5. Format of Shared Memory after FAIL

3. The host reads the Command Status Word. It should contain one of the following 32-bit ASCII values: CBOK (0x43424F4B) or FAIL (0x4641494C). If it contains CBOK, the host may issue any Common Boot command. If it contains FAIL, the host may issue the DIAG command to identify the problem.

4. The board polls the Command Status Word. When it is overwritten with a command code, the board executes the command and writes the results (CBOK or FAIL) back to the Command Status Word. It then polls for the next command.

No interrupt is generated at the completion of a Common Boot command, so the host must poll the Command Status Word to determine whether the command has completed. This is done by checking that the board has executed the command and written CBOK or FAIL into the Command Status Word.

5. To boot the HARI Interface, the host issues the Common Boot command, *BOOT*. Once the HARI Interface boots up, status LEDs ST0–ST2 (LED 5–LED 7) will cycle up and down, and continue to *cylon* as long as HARI is running normally.
6. The HARI Interface starts up and creates the initial control send and received channel SCBs. Common Boot exits until the board is reset.

Resetting the 5215

To reset the 5215, toggle bit 15 of the Hardware Control field from 0 to 1, hold for a minimum of 1 microsecond, then clear the bit to 0. The Hardware Control field is a 16-bit field located at the start of short I/O (that is, bytes 0 and 1 of the 512-byte short I/O space). When the 5215 is reset, it goes through its bootup sequence as described in the previous section. The reset bit must hold the 1 for a minimum of 1 µsecs.

NOTE

After toggling the *reset* bit, the host should wait at least 500 ms before looking for CB_HERALD. Or the user may write a value in the command/status location (normally zero or 0xDEAD) and reset the controller. After power-up, the zero or 0xDEAD value will be replaced by CBOK or FAIL.

Common Boot Commands

This next sections of this chapter provide detailed information about the following V/ATM Common Boot (CB) commands:

Command	Description	See Page
BOOT	This command is used to boot an alternate firmware image.	59
DIAG	This command performs a series of tests whose definition is specific to the 5215.	61
FILL	This command allows the host to modify controller memory or hardware.	63
FLSH	This command allows the host to control the state of the controller's on-board LED.	64
GRNL	This command allows the host to control the state of the controller's on-board LED.	65
JUMP	This command allows the host to force the CPU to do an unconditional jump.	66
PEEK	This command allows the host to examine controller memory.	67
POKE	This command allows the host to modify controller memory.	68
DNLD	This command is used to download an image to the controller.	69

Command	Description	See Page
BURN	This command is used to store a saved downloadable image into flash EPROM to be used as the EXEC 0 image.	70
REDL	This command allows the host to control the state of the controller's on-board LED.	71
STUF	This command allows the host to modify controller memory or hardware.	72
HGET	This command allows the host to examine controller memory or hardware.	73
HPUT	This command allows the host to modify controller memory or hardware.	75
INFO	This command allows the host to identify a controller as well as obtain various controller specific information.	77

Common Boot commands support the following functions:

- Boot resident and downloadable interface
- Controller LED commands
- Controller memory read and write
- Controller diagnostic commands
- Burn image into FLASH memory

The following `define` statements describe the Common Boot command set, along with the CBOK and FAIL responses:

```
#define MAKE_LONG(a,b,c,d) (((u32)a<<24) | ((u32)b<<16) | ((u32)c<<8) | (u32)d)
#define CB_BOOT  MAKE_LONG('B','O','O','T')
#define CB_EXEC  MAKE_LONG('E','X','E','C')
#define CB_DIAG  MAKE_LONG('D','I','A','G')
#define CB_FILL  MAKE_LONG('F','I','L','L')
#define CB_FLSH  MAKE_LONG('F','L','S','H')
#define CB_GRNL  MAKE_LONG('G','R','N','L')
#define CB_JUMP  MAKE_LONG('J','U','M','P')
#define CB_PEEK  MAKE_LONG('P','E','E','K')
#define CB_POKE  MAKE_LONG('P','O','K','E')
#define CB_DNDL  MAKE_LONG('D','N','L','D')
#define CB_BURN  MAKE_BURN('B','U','R','N')
#define CB_REDL  MAKE_LONG('R','E','D','L')
#define CB_STUF  MAKE_LONG('S','T','U','F')
#define CB_CBOK  MAKE_LONG('C','B','O','K')
#define CB_FAIL  MAKE_LONG('F','A','I','L')
```

All of these `define` statements are unique in both the upper and lower words. This enables the host to make both 16-bit and 32-bit accesses to shared memory when using Common Boot, provided that the controller supports both types of accesses.

Downloading Code

A download mechanism is provided on Interphase controllers that support the Common Boot interface. This feature allows the host to download code to the 5215 firmware for execution. Applications of the feature include:

- Booting an interface other than the native HARI Interface
- Making firmware upgrades in the field

Interphase provides utilities that automate the download process. These utilities are tied to specific controller types (for example, 5211, 5215, etc.) and hardware/firmware revision levels. They typically use the Common Boot commands POKE and BOOT to download and execute the code.

BOOT

Group: Common Boot

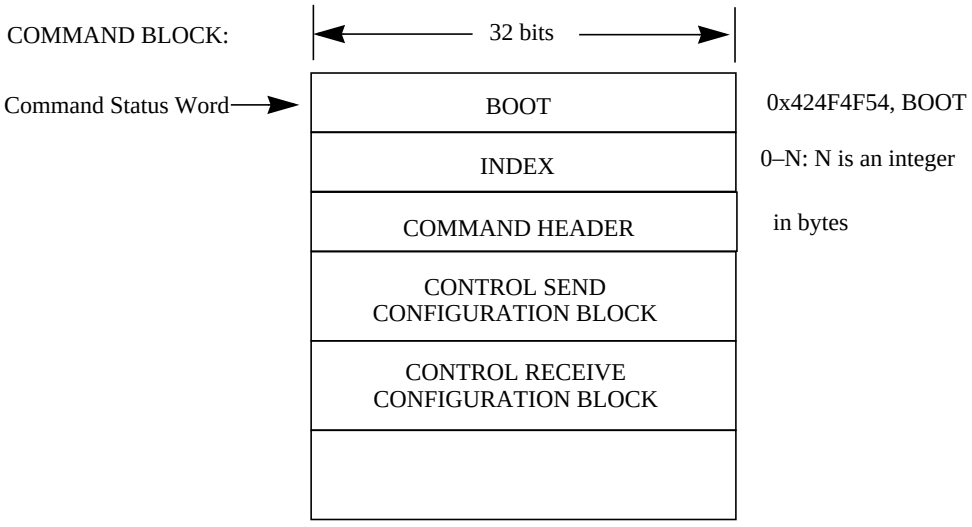


Figure 6-6. BOOT Command

Function

The BOOT command will boot a native or downloaded interface. This section describes the format of the BOOT command when booting the controller's native HARI Interface (that is the default interface stored in EPROM.) For information on booting a downloaded interface, see *Downloading Code* on page 59. The specifics of Boot are discussed in Appendix B on page 147.

If you are booting the native HARI Interface, the BOOT command includes the following HARI command:

```
Configure Control Channels
```

This command establishes the initial pair of HARI channels needed to issue commands to, and obtain responses from the controller.

Code

0x424F4F54 (ASCII BOOT)

Parameters

Index

Boot Entry Point Index (0,1).
With a supplied 0, the native HARI Interface will boot.
With a supplied 1, the downloaded image is moved to the execution RAM and the control is passed to the new image. No other parameters are required for index 1.

Total Length of Remaining Parameters

Specifies the aggregate length, in bytes, of all HARI commands the 5215 may expect to find in shared memory following the Total Length field in this command

Control Send Configuration Block

Establishes the operating parameters of the control send channel.

Control Receive Configuration Block

Establishes the operating parameters of the control receive channel.

Programming Sequence

If CBOK, supply the parameters, then supply the BOOT command. When CBOK occurs or the booted interface responds, the Common Boot Interface *goes away* until the host performs a hardware reset on the board.

Error Returns

If FAIL, the CB ERROR status block will be filled in.
BOOT_0_ATTEMPT (0x0D)

DIAG

Group: Common Boot

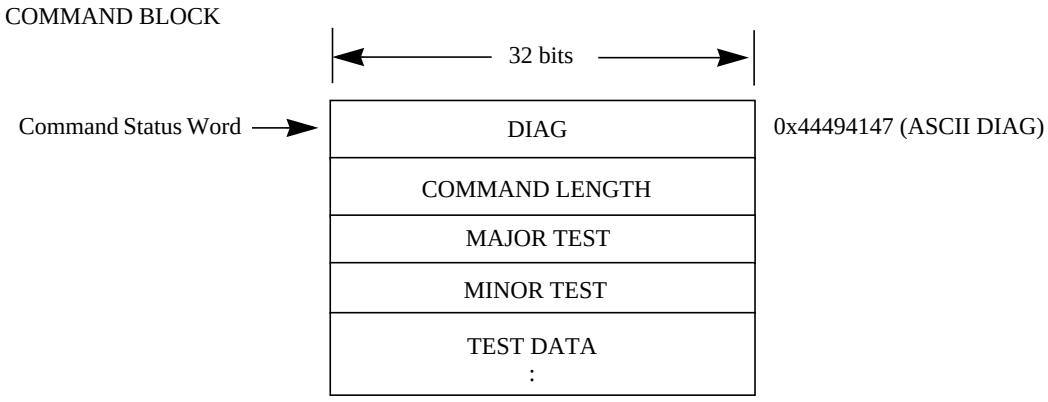


Figure 6-7. Format of the DIAG command

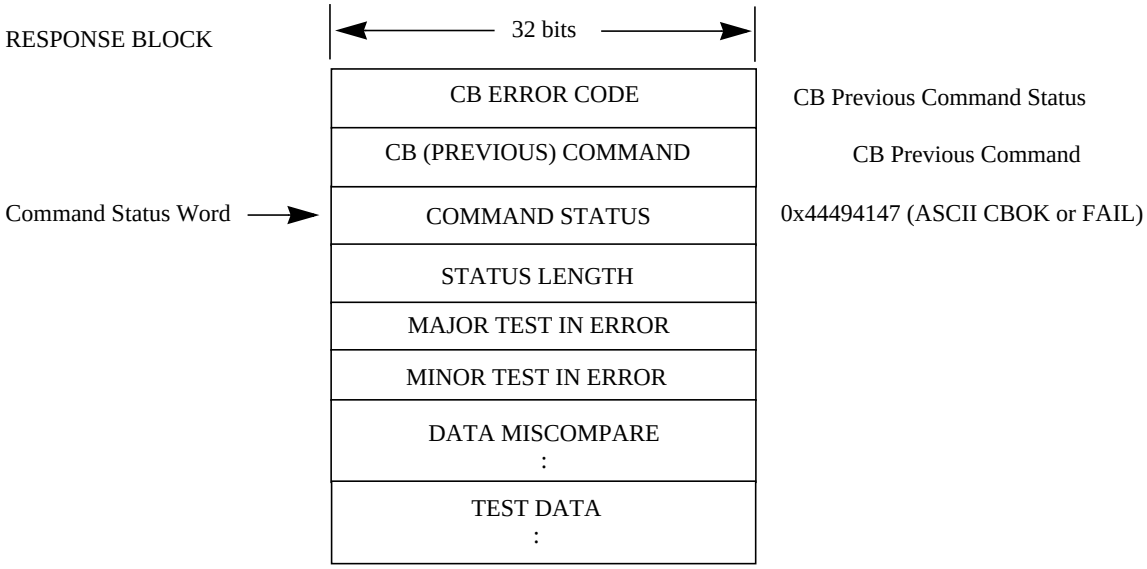


Figure 6-8. FAIL Response Block

Function

Execute Diagnostics. This command performs a series of tests whose definition is specific to the 5215. For a description of how the fields are defined, refer to *5215 Diagnostics* on page 79.

Code

0x44494147 (ASCII DIAG)

Parameters

Command Length

This field contains the total number of bytes in the test command, including itself, but excluding the Command Status Word (which contains the ASCII DIAG).

Major Test

Major test to be executed.

Minor Test

Minor test (if any) to be executed.

Test Data

Test-specific data to be used in the test.

Programming Sequence

If CBOK, supply the parameters, then supply the DIAG command. When CBOK occurs, the test has been successfully completed. If FAIL occurs, the 5215 returns the response structure shown in Figure 6-8 on page 61.

Response Block

The Response Block contains the Command Status (CBOK or FAIL). If the response is normal, the board will place CBOK in the Command Status Word field and the other fields are irrelevant (that is, Don't Cares). If the response is FAIL, additional information is provided in the fields as shown in Figure 6-8 on page 61 and described below:

CB Error Code

This field gives the error code of previous CB command.

CB Previous Command

Previously issued Common Boot command.

Command Status

Status (CBOK or FAIL) of the issued command.

Status Length

This field contains the total number of bytes in the status block, including itself, but excluding the Command Status Word (which is the ASCII FAIL). Contains 0x10 if no error data is returned.

Major Test In Error

This reports the major test number of the failed test.

Minor Test In Error

This reports the minor test number, if any, of the failed test.

Data Miscompare

Error data (if any).

Test Data

This field contains test-specific information about the error. Some errors have data associated with them, while others do not.

FILL

Group: Common Boot

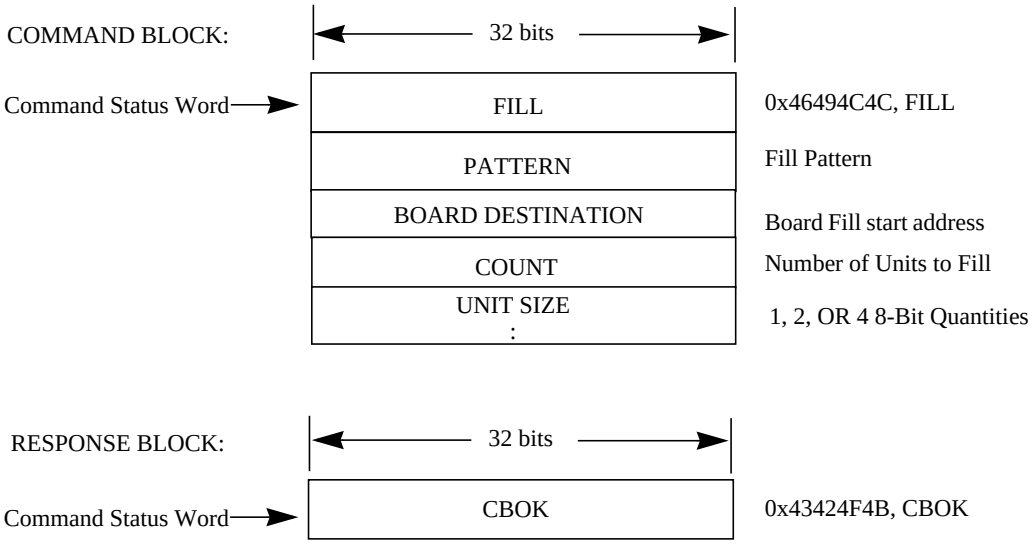


Figure 6-9. FILL Command and Response Format

Function

Fill area with pattern.

Starting at the provided DESTINATION ADDRESS, this function will write to the contents of memory the value contained in the command parameter PATTERN. This transfer will be made in data widths provided by the UNIT SIZE parameter. The range of memory checked will be determined by the parameter UNIT COUNT.

Code

0x46494C4C (ASCII FILL)

Parameters

Pattern

The pattern used for filling memory.

Board Destination

The starting board address for pattern filling.

Count

The number of units to fill with pattern.

Unit Size

The size of unit (1=8, 2=16, or 4=32 bits).

Programming Sequence

If CBOK, supply the parameters, then supply the FILL command. When CBOK occurs, the pattern has been successfully laid down.

Error Returns

If FAIL, then CB ERROR block will be filled in.
BAD_SIZE (0x0B)

FLSH

Group: Common Boot

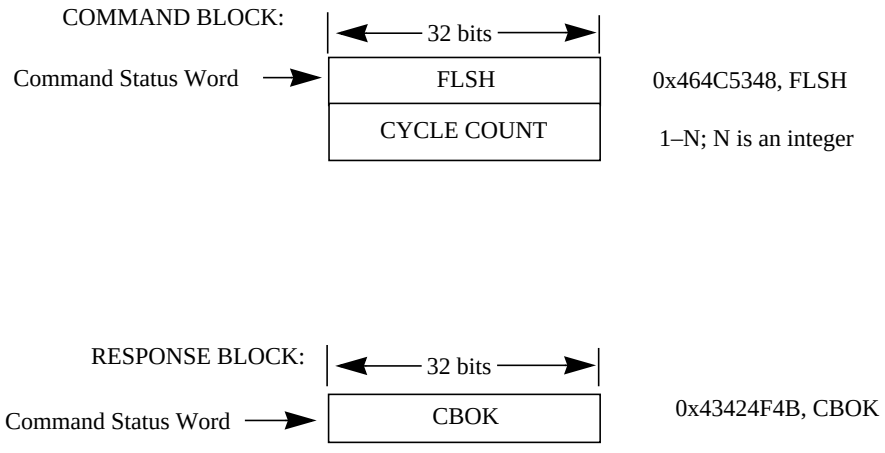


Figure 6-10. FLSH Command and Response Format

Function

Flash the LED(s).

Code

0x464C5348 (ASCII FLSH)

Parameters

Cycle Count

The number of times an LED red-to-green transition occurs.

Programming Sequence

If CBOK, then supply the FLASH cycle parameter, then supply the FLSH command. When CBOK occurs, FLSH is finished.

GRNL

Group: Common Boot

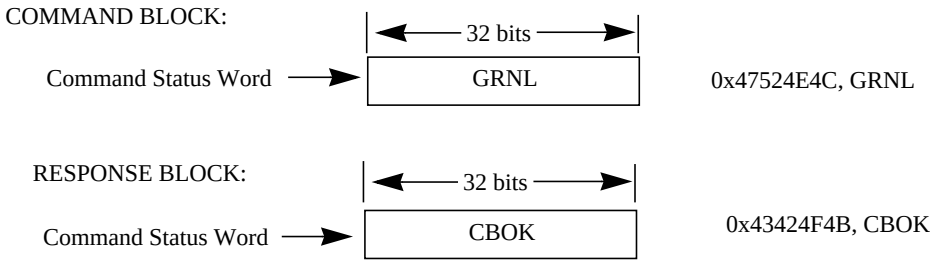


Figure 6-11. GRNL Command and Response Format

Function

Turn on the green LED.

Code

0x47524E4C (ASCII GRNL)

Parameters

None required.

Programming Sequence

If CBOK, supply the GRNL command. When CBOK occurs or the LED turns green, GRNL is finished.

JUMP

Group: Common Boot

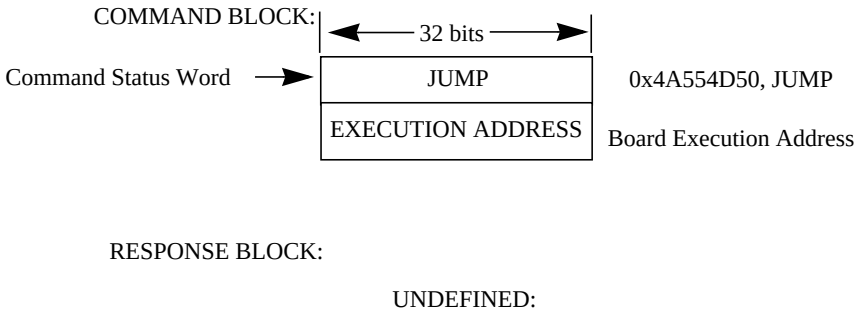


Figure 6-12. JUMP Command and Response Format

Function

Transfer execution to address.

Code

0x4A554D50 (ASCII JUMP)

Parameters

Execution Address

Board execution address.

Programming Sequence

If CBOK, supply the target execution address, then supply the JUMP command.

PEEK

Group: Common Boot

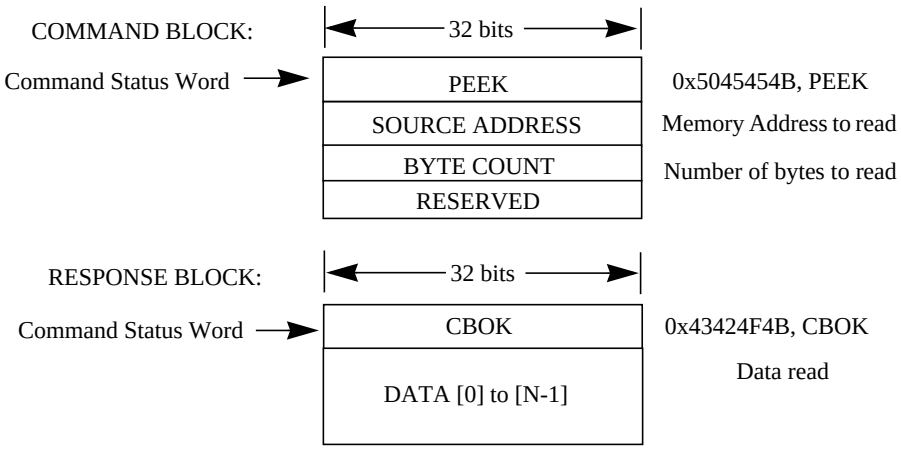


Figure 6-13. PEEK Command and Response Format

Function

Examine controller memory.

Code

0x5045454B (ASCII PEEK)

Parameters

Source Address

Board memory address to read.

Byte Count

Number of units to read.

Reserved

This field is reserved and should be 0.

Programming Sequence

If CBOK, supply the parameters, then supply the PEEK command. When CBOK occurs, PEEK is finished and memory will be dumped at the location immediately following CB_START + 4.

POKE

Group: Common Boot

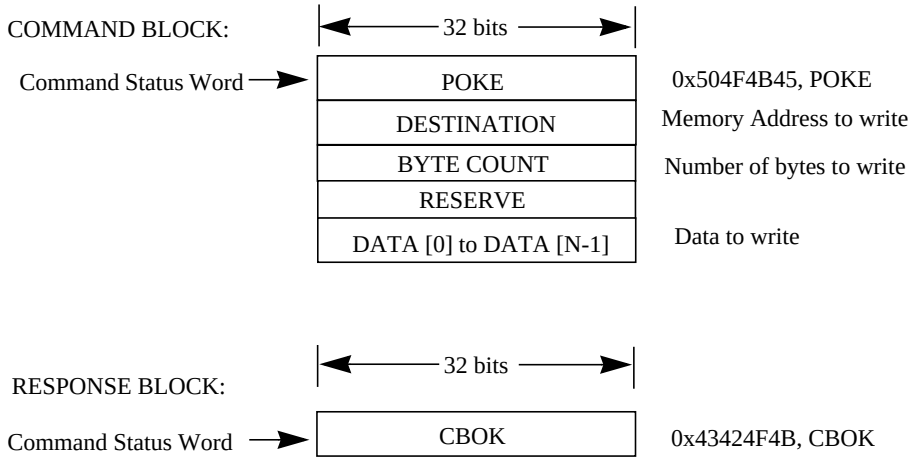


Figure 6-14. POKE Command and Response Format

Function

Modify controller memory.

Code

0x504F4B45 (ASCII POKE)

Parameters

Destination Address

Where data is to be written.

Byte Count

Number of bytes to write.

Reserved

This field is reserved and should be 0.

Programming Sequence

If CBOK, supply the parameters, then supply the POKE command. The data to be poked immediately follows the RESERVED field. When CBOK occurs, POKE is finished and written memory exists on the controller at the destination address.

DNLD

Group: Common Boot

The CB Download command DNLD allows the host to download new code to the controller.

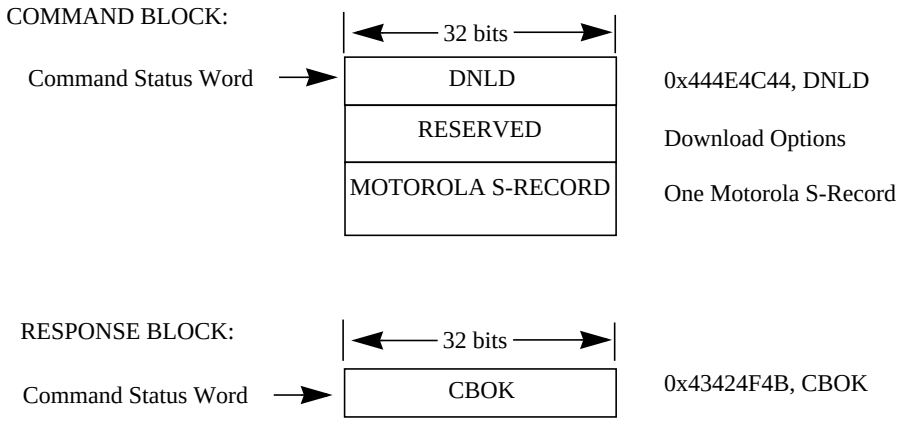


Figure 6-15. DNLD (DOWNLOAD) Command and Response Format

Function

This command allows the host to download Motorola S-Records, one record at a time, into the unit's download memory. The function will track the SD-Record type (0–9), convert the input from ASCII to binary format, checksum each record, track if the image being downloaded, and download the image into buffer RAM with the appropriate offset.

NOTE

The function will do an overall checksum of the header image and keep a record of the expected program image size and checksum.

Code

0x444E4C44 (ASCII DNLD)

Parameters

None

Motorola S_record

This entry consists of a single Motorola S-Record. Record types S0, S1, S2, S3, S7, S8 and S9 are supported.

Programming Sequence

If CBOK, supply the parameters, then supply the DNLD command. The S-Record to be downloaded immediately follows the RESERVED field. When CBOK occurs, DNLD is finished and the next DNLD command can be issued until all the S-Records are downloaded.

If the data was downloaded with the FLASH option, the HOST may issue an BOOT 1 to start executing the code or a BURN command to save the code in FLASH EPROM.

Error Returns

If FAIL, the CB ERROR status block will be filled in:

```
CB_SREC_BAD_TYPE(0x01)  CB_IMG_CHKSUM(0x02)
CB_SREC_CHKSUM(0x03)    CB_BUFFER_OVERRUN(0x04)
CB_IMG_OVERRUN(0x05)    CB_BAD_SORE(0x06)
CB_BAD_LENGTH(0x08)     CB_NOT_SREC(0x09)
IHE_DHRCHKSUM(0x17)     CB_DNLD_SEQC(0x23)
```

BURN

Group: Common Boot

The CB BURN command is used to update FLASH EPROM with a downloaded image that was saved in RAM using the CB DNLD command.

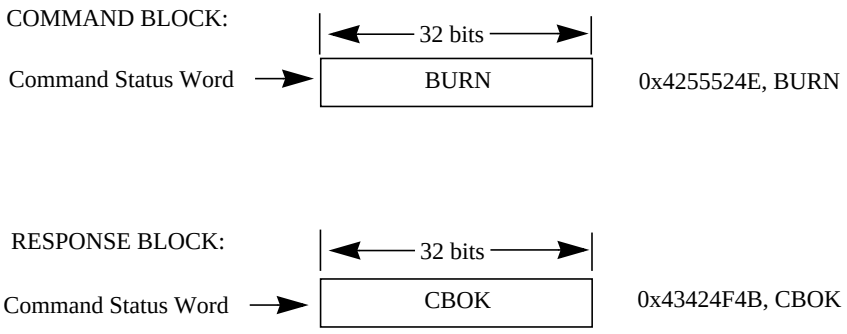


Figure 6-16. BURN Command and Response Format

Function

This function allows for the updating of FLASH EPROM with new program images.

Code

0x4255524E (ASCII BURN)

Programming Sequence

If CBOK, supply the parameters, then the BURN command. The data to be burned into FLASH EPROM should have been previously downloaded using the CB DNLD or CB POKE command.

Error Returns

If FAIL, the CB ERROR status block will be filled in.

REDL

Group: Common Boot

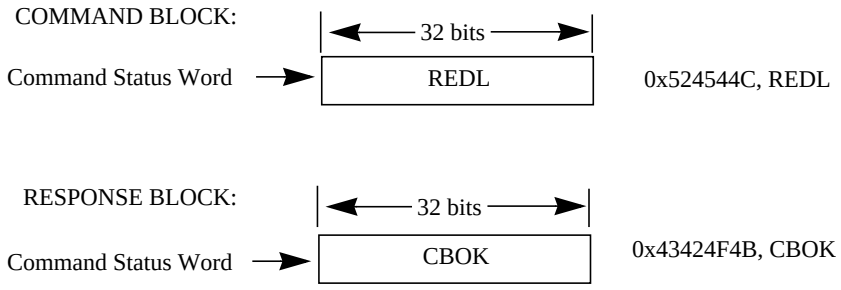


Figure 6-17. REDL Command and Response Format

Function

Turn on the red LED (if present on controller)

Code

0x5245444C (ASCII REDL)

Parameters

None

Programming Sequence

If CBOK, then supply the REDL command. When CBOK occurs or the LED turns red, REDL is finished.

STUF

Group: Common Boot

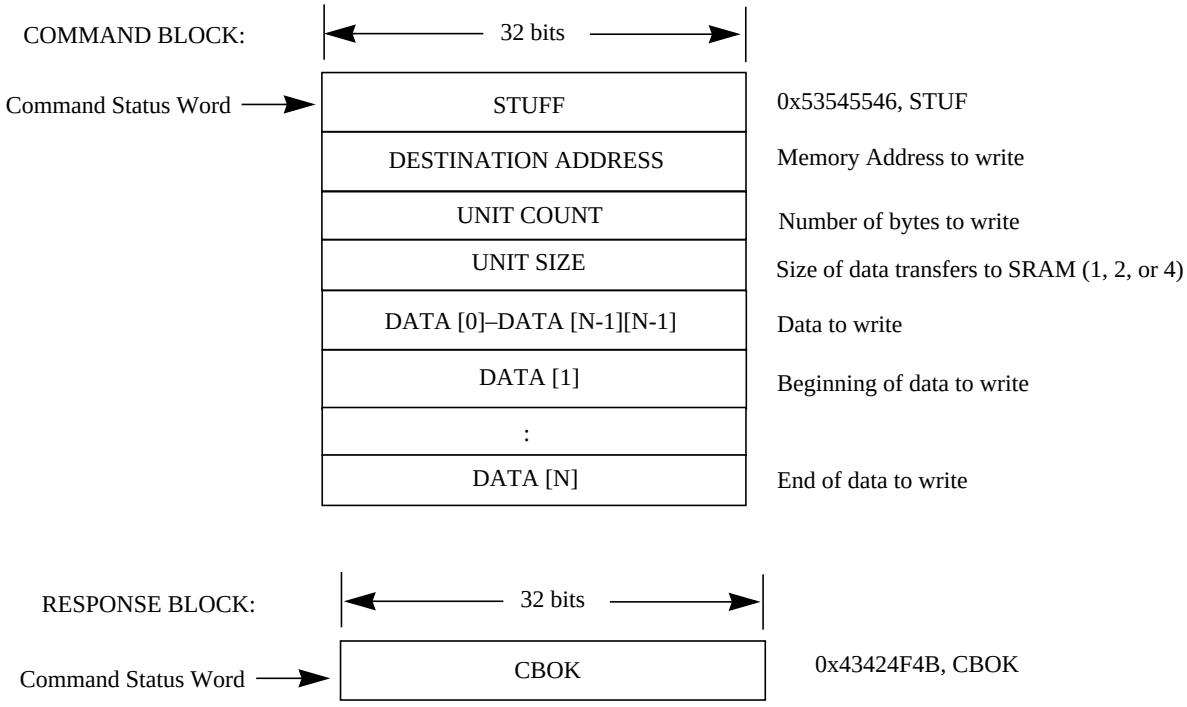


Figure 6-18. STUF Command and Response Format

Function

Modify controller memory by stuffing multiple data units at the same destination address. This routine is used to stuff data into a single, non-incrementing address location. The data is transferred into the address location in the size requested. The data to be written is read from shared memory in the format of (byte, word, or longword) the parameter UNIT SIZE.

Code

0x53545546 (ASCII STUF)

Parameters

Destination Address

Where data is written. This address does not increment. This value is an absolute address.

Unit Count

Number of units to write. The value placed in UNIT COUNT should not exceed the size in Longword.

Unit Size

Size in bytes of units (1, 2, or 4).

Programming Sequence

If CBOK, supply the parameters and then supply the STUF command. The data to be stuffed immediately follows the UNIT SIZE field. When CBOK occurs, STUF is finished and the specified data exists on the controller at the destination address.

Error Returns

If FAIL, the CB ERROR status block will be filled in.
BAD_SIZE (0x0B)

HGET

Group: Common Boot

The CB HGET command is used to examine a series of address locations.

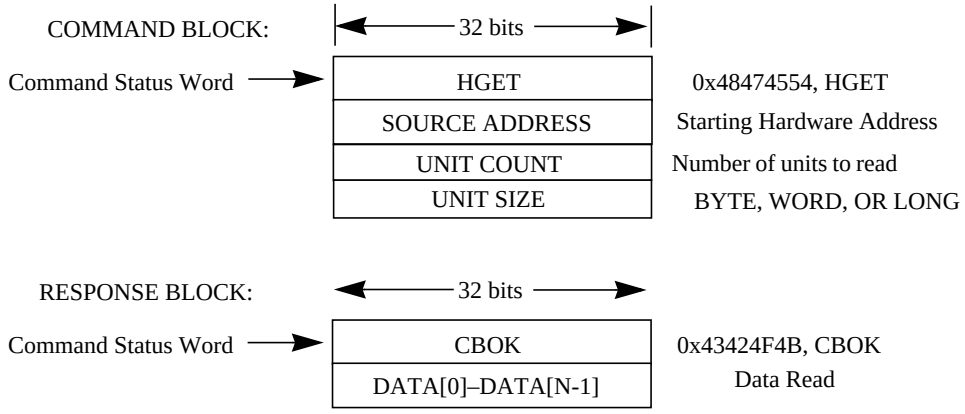


Figure 6-19. HGET Command and Response Format

Function

Examine a series of address locations.

The function simply transfers data from the source addresses to the addresses of the shared memory data fields. Both the shared memory and the source address are accessed and incremented as given by the UNIT SIZE input variable.

Code

0x48474554 (ASCII HGET)

Parameters

Source Address

Where data is to be read from.

Unit Count

Number of units to read.

Unit Size

BYTE, WORD, or LONG

This parameter determines the width in bytes of the CPU reads to the controller hardware (BYTE (8 bits), WORD (16 bits), or LONG (32 bits)).

It is represented as a 32 bit ASCII value. BYTE 0x42595445, WORD 0x574F5244, LONG 0x4C4F4E47.

Example

For memory source with the first for longword addresses containing the following: A0=0x00112233 A1=0x44556677 A2=0x8899AABB A3=0xCCDDEEFF.

Assuming a UNIT COUNT = 4 and a VMEbus address of COMMAND STATUS WORD = 0xff4080, the following results will appear in the controller's short I/O memory:

Unit Size = BYTE

0xff4084 = 0x00 0xff4085 = 0x11 0xff4086 = 0x22 0xff4087 = 0x33

Unit Size = WORD

0xff4087 = 0x0011 0xff4086 = 0x2233 0xff4088 = 0x4455 0xff408a = 0x6677

Unit Size = LONG

0xff4084 = 0x00112233 0xff4088 = 0x44556677 0xff408c = 0x8899AABB 0xff490 0xCCDDEEFF

Programming Sequence

If CBOK, supply the parameters, then supply the HGET command.

Error Returns

IF FAIL, the CB ERROR status block will be filled in.
BAD_SIZE(0x0B)

HPUT

GROUP: Common Boot

The CB HPUT command is used to modify a series of address locations.

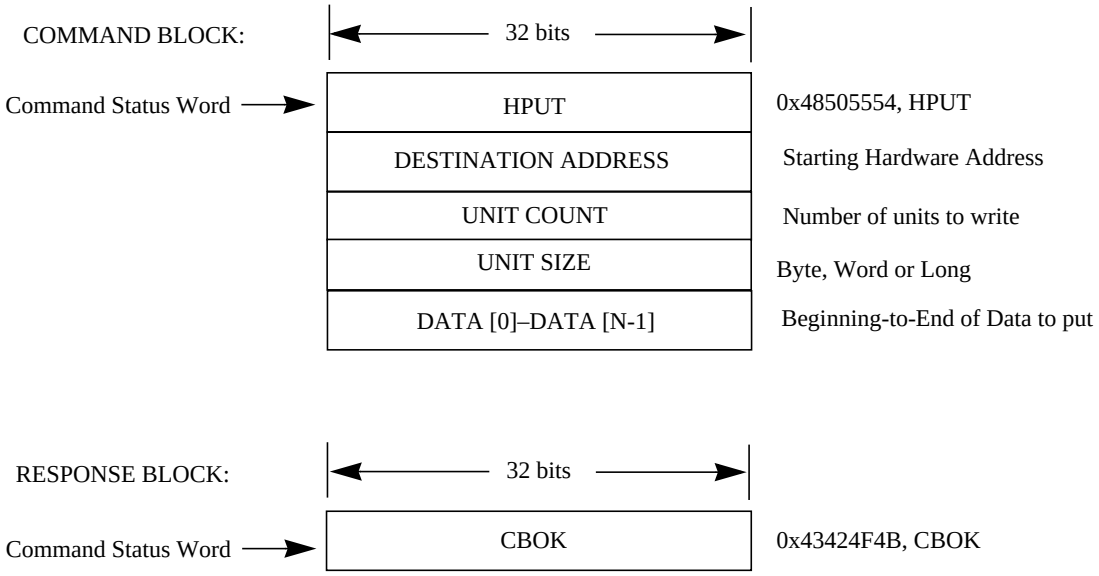


Figure 6-20. HPUT Command and Response Format

Function

Modify a series of address locations.

The function simply writes to the destination address the contents of the shared memory data fields. Both the shared memory and the source address are accessed and incremented as given by the UNIT SIZE input variable.

Code

0x48505554 (ASCII HPUT)

Parameters

Destination Address

Starting address where data is to be written.

Unit Count

Number of units to write.

Unit Size

BYTE, WORD, or LONG

This parameter determines the width in bytes of the CPU writes to the controller hardware (Byte (8 bits), Word (16 bits), or Long (32 bits)).

It is represented as a 32 bit ASCII value. Byte 0x42595445, Word 0x574F5244, Long 0x4C4F4E47.

Example

For a HPUT command with the first four longword data parameters containing the following: D0 = 0x00112233 D1 = 0x44556677 D2 = 0x8899AABB D3 = 0xCCDDEEFF.

Assuming a UNIT COUNT = 4 and a DESTINATION ADDRESS of 0x0c00000 the following results will appear in the controller's memory:

Unit Size = BYTE

xc00000 = 0x00 0xc00001 = 0x11 0xc00002 = 0x22 0xc00003 = 0x3.

Unit Size = WORD

0xc00000 = 0x0011 0xc00002 = 0x2233 0xc00004 = 0x4455 0xc00006 = 0x6677.

Unit Size =LONG

0xc00000 = 0x00112233 0xc00004 = 0x44556677 0xc00008 = 0x8899AABB 0xc0000c = 0xCCDDEEFF.

Programming Sequence

If CBOK, supply the parameters, then supply the HPUT command. The data to be put immediately follows the UNIT SIZE field.

Error Returns

IF FAIL, the CB ERROR status block will be filled in.
BAD_SIZE(0x0B)

INFO

The CB board information command (INFO) allows the host to identify the controller.

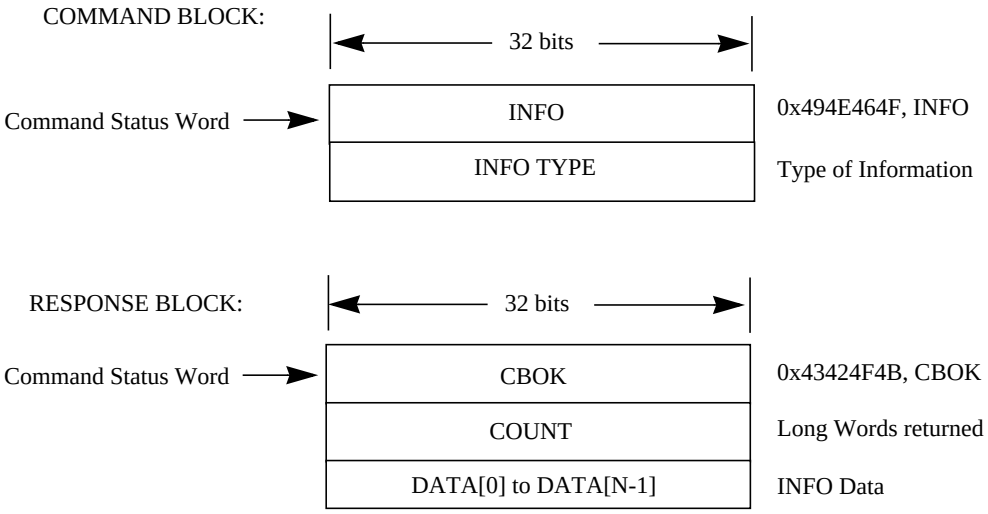


Figure 6-21. INFO Command and Response Format

Function

Function returns board identification, boot image information or controller specific information.

Code

0x494E464F (ASCII INFO)

Parameters

Info Type

Specifies which information the controller makes available to the host:

0 **CB information:** Setting this value to 0 tells the controller to return CB information.

1 **EXEC 0 ID:** Setting this value to 1 tells the controller to return the EXEC 0 header information.

2 **Controller specific information:** Setting this value to 2 tells the controller to return the Controller Info structure (see *Controller Info (0x00)* on page 125).

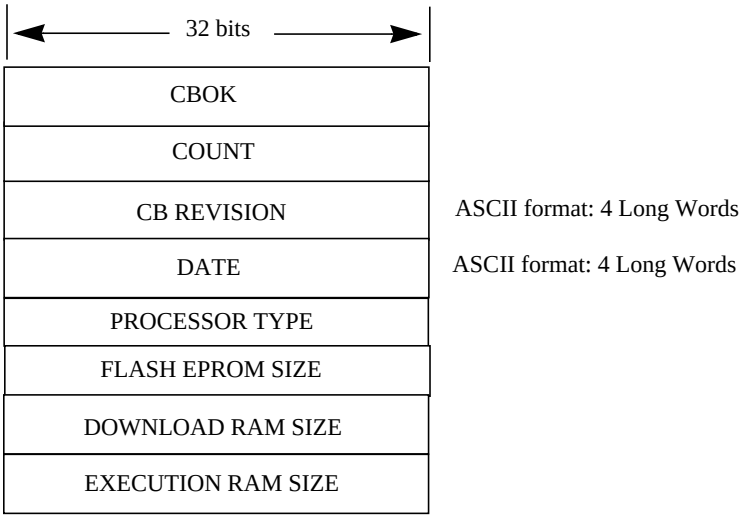


Figure 6-22. Information Structure Type 0 and Type 1

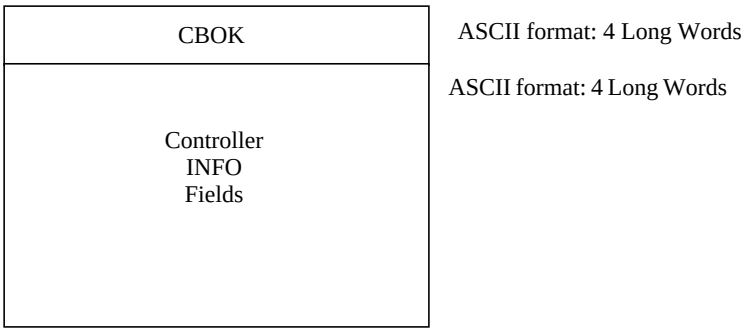


Figure 6-23. Information Structure Type 2

Programming Sequence

If CBOK, supply the INFO TYPE; then supply the INFO command. When CBOK occurs, the host may examine the information.

Error Returns

IF FAIL, the CB ERROR status block will be filled in.
BAD_SIZE(0x0B)

5215 Diagnostics 7

Overview

The 5215 supports a variety of power-up and host-controlled diagnostics. Power-up diagnostics execute automatically each time the board is reset. Host-controlled diagnostics may be invoked individually while the Common Boot interface is running (that is, before the host has booted the HARI Interface using the BOOT command).

- In power-up mode, memory, front-end, and DMA diagnostic tests are performed. Memory is zeroed out if all tests pass. With host-controlled diagnostics, extended memory tests are available. Refer to the Power-up test (PUP_TEST) in each major test category (Memory, Front-end, DMA).
- VME DMA tests are not executed in power-up mode, but can be run in host-controlled mode.
- External loopback tests are not executed during power-up diagnostics, but can be invoked by the host after power-up.
- The VMEbus is not accessed during the power-up DMA test, but can be invoked by the host after power-up.

The 5215 diagnostics provide the following controller tests:

- Major Test 0x1—Memory Diagnostics
- Major Test 0x2—ATM Front End Diagnostics
- Major Test 0x3—VME DMA Diagnostics

This chapter provides general information about the hardware memory map, diagnostic test categories, and diagnostic command formats and results. It also provides detailed information about Memory, ATM Front End, and DMA diagnostics.

Memory Map

The following table shows a memory map of the 5215 hardware:

Value	Description
0x20000000	PRAM—program space 512K
0x36000000	SRAM—static ram 128K
0x38000000	VRAM—video ram 1M to 4M (board option)
0x9A000000	General output register (MBC)
0x9A800000	Iport register—status of onboard interrupts (1 = set)
0xB4000000	Addgen base address
0x3601FF40	DLE space for the addgen (4 * 20)
0x3601FFF0	SGE space for the DLE translation (1 * 10)

Common Boot Diagnostic Tests

The following table provides a quick-reference listing of the 5215 Common Boot major and minor diagnostic tests:

Major Test	Major Test Type	Minor Test	Possible Errors
0x1	Memory Tests	0x0—PUP_TEST 0x1—PRAM 0x2—SRAM 0x3—BRAM 0x4—FLASH	0x1 Test Failed
0x2	ATM Front-end Tests	0x0—PUP_TEST 0x1—Internal Loopback 0x4—External Loopback 0x5—F-FRED SRAM 0x6—R-FRED SRAM 0x8—F-FRED CPU Bus 0x9—R-FRED CPU Bus	0x2E DIAG_ERROR is written to the CB Last Command Status, and FAIL is written to the Command Status Word, followed by the appropriate error data.
0x3	VME DMA Diagnostics	0x0—PUP_TEST 0x2—Standard I/O DMA 0x4—RAW DMA 0x5—READ BUFFER 0x6—WRITE BUFFER 0x7—VIRQ 0x8—DUMP ASIC REG	See *DMA Diagnostic Error Codes* on page 99. 0x2E DIAG_ERROR is written to the CB Last Command Status, and FAIL is written to the Command Status Word, followed by the appropriate error data.

Diagnostic Command

Diagnostic commands are issued through the short I/O space. The procedure for issuing diagnostic commands is as follows:

1. Build the command block directly behind the Command Status Word in short I/O.
2. If the current Command Status Word is ASCII CBOK (0x43424F4B), write the ASCII word DIAG (0x44494147) or TEST (0x54455354) over the Command Status Word.
3. Wait for the Command Status Word to read either CBOK or FAIL.

Diagnostic Command Structure

Most of the commands in the Common Boot command set are generic. Their definition is the same for any controller that supports the Common Boot interface. The DIAG command is an exception. It is used to perform a variety of controller-specific tests. Therefore, the definition of its fields varies depending on the controller and on the type of test being performed. Figure 7-1 shows the format of the DIAG command:

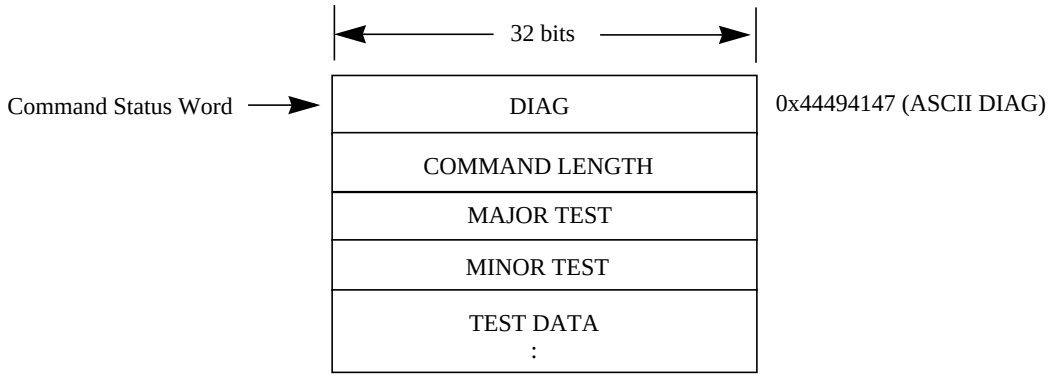


Figure 7-1. Format of the DIAG command

Diagnostic Command Fields

Command Length

This field contains the total number of bytes in the test command, including itself, but excluding the Command Status Word (which contains the ASCII value DIAG).

Major Test

Each major hardware section of the 5215 has a separate test, which is assigned a number. A major test number of 0x0 executes all of the major tests. See *Diagnostic Test Data* on page 82 for major test numbers.

Minor Test

Some of the major tests are divided into a number of minor tests. A minor test number of 0x0 will execute all the minor tests in the associated major test.

NOTE

If the Major Test field contains 0x0, the Minor Test field is ignored.

If no minor tests are defined for the major test, the host may enter either 0x0 or 0x1 in the minor test field. If the 5215 returns a status block, the returned Minor Test field will contain the value 0x1.

Diagnostic Test Data

Only VMEbus DMA tests require test-specific data. These fields are identified in the VMEbus DMA test description in *VME DMA Diagnostics — Major Test Code 0x3* on page 90. Following are the test number codes for major diagnostic tests:

Code	Test Type
0x0	Memory Test
0x2	ATM Front-end
0x1	DMA Test

Test Results

If the test completes successfully, the Command Status Word will contain the ASCII value CBOK (0x43424F4B). No other data is returned.

Failed Test Result Structure

If the test fails, the Command Status Word will contain the ASCII value FAIL and the 5215 returns the structure shown in Figure 7-2:

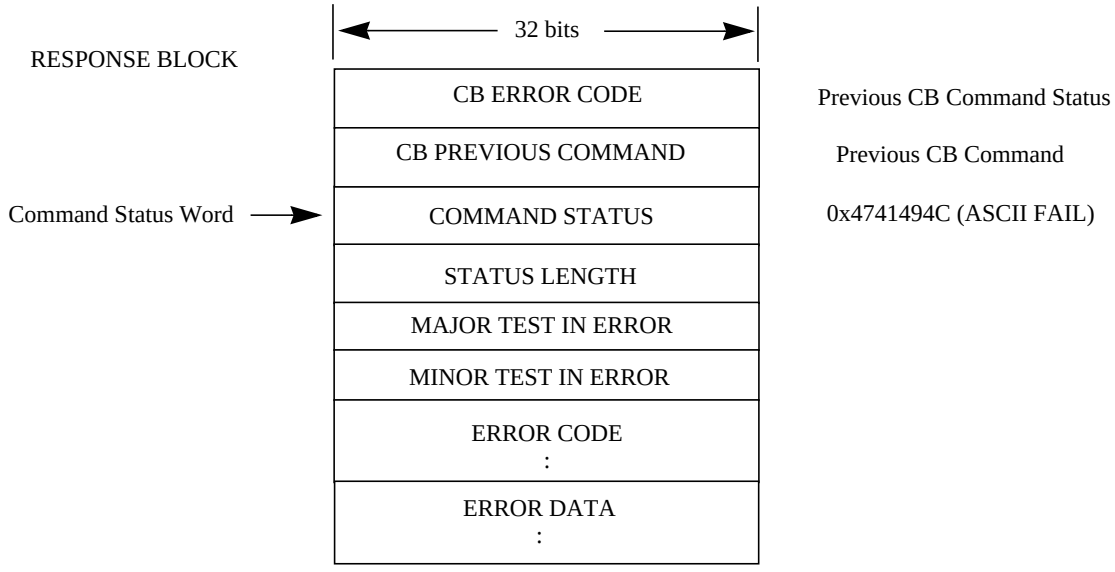


Figure 7-2. FAIL Returned Status Block

NOTE

If a diagnostic test fails, please make a note of the returned data. This will enable Interphase to provide you with better, faster technical assistance.

Failed Test Result Fields

CB Error Code

This field reports the status of the previously issued CB command.

CB Previous Command

This reports the previously issued CB command that failed.

Status Length

This field contains the total number of bytes in the status block, including itself, but excluding the Command Status Word (which contains the ASCII FAIL code 0x4741494C).

Major Test in Error

This reports the major test number of the failed test.

Minor Test in Error

This field reports the minor test number of any failed test. If there is no minor test, this field contains 0x1.

Error Code

This field reports the test error code. *DMA Diagnostic Error Codes* on page 99 lists the diagnostic error codes. See Appendix C on page 159 for listings of the Common Boot error codes.

Error Data

This field contains test-specific information about the error. Some errors have data associated with them, while others do not. If a given error does not include error data, the returned structure will end after the Error Code field. It will consist of four 32-bit fields (Status Length, Major Test in Error, Minor Test in Error, and Error Code).

Memory Diagnostics — Major Test Code 0x1

There are three distinct memory areas on the 5215. The CPU operates from 512K bytes of static RAM, referred to as PRAM. The board uses another set of static RAM, referred to as SRAM. The size of this bank of memory is variable up to 1 Mbyte. onboard buffer space and onboard shared memory (short I/O space) are implemented in video RAM chips, abbreviated BRAM. This is also referred to as buffer RAM. The BRAM size is variable up to 4 Mbytes.

There are four tests written to test memory. One is a quick test run at power-up or reset, and the other three are extended diagnostics, which can be run by the host using Common Boot commands.

Following is a memory test command example:

Name	Value in Hex
Length	0x10
Major	0x1
Minor	0x2
Options	0x0

Memory Test Options

Figure 7-3 shows the Options field:

31	. . .						2	1	0
rsvd	rsvd	rsvd	rsvd	rsvd	rsvd	rsvd	rsvd	rsvd	option

Figure 7-3. Options Field

Bit 0

If bit 0 is set (bit 0 = 1) then controller memory is cleared when the test is complete. If bit 0 is cleared (bit 0 = 0) then the test patterns remain in memory after the test is complete.

Bits 1–31

These bits are reserved and should be 0.

Minor Test Codes

The following table lists the minor tests supported by the memory diagnostic test:

Minor Test Code	Name	Function
0x0	PUP_TEST ONLY	Quick check of all 3 memory types
0x1	PRAM	Tests the 512K CPU Static Ram
0x2	SRAM	Tests up to 1 Mbyte of Static Ram
0x3	BRAM	Tests up to 4 Mbytes of Video Ram

If the test completes successfully, the 5215 will move the CB **DIAG** command to the **Previous Command** position in short I/O, and write **CBOK** into the **Command Status Word**. See Figure 7-2 on page 82.

0x0—PUP_TEST

An example of a power-up test command is:

Name	Value in Hex
Length	0x10
Major	0x1
Minor	0x0
Options	0x0

0x1—PRAM

An example of a PRAM test command is:

Name	Value in Hex
Length	0x10
Major	0x1
Minor	0x1
Options	0x0

0x2—SRAM

An example of an SRAM test command is:

Name	Value in Hex
Length	0x10
Major	0x1
Minor	0x2
Options	0x0

0x3—BRAM

An example of a BRAM test command is:

Name	Value in Hex
Length	0x10
Major	0x1
Minor	0x1
Options	0x0

Memory Test Error Codes

The only error code returned by the memory tests is 0x1. Along with this error code, the test will report where the error was found, the data read from that location, and the data that was expected. Common Boot defines a location for the previous command (in this case DIAG) and a location for a status for that command. When reporting an error from a diagnostic routine, that status will indicate the CB error code, DIAG_ERROR (0x2e). See Appendix B for CB Error Codes.

The DIAG error code and results will be found in the normal data structure for CB diagnostics as follows:

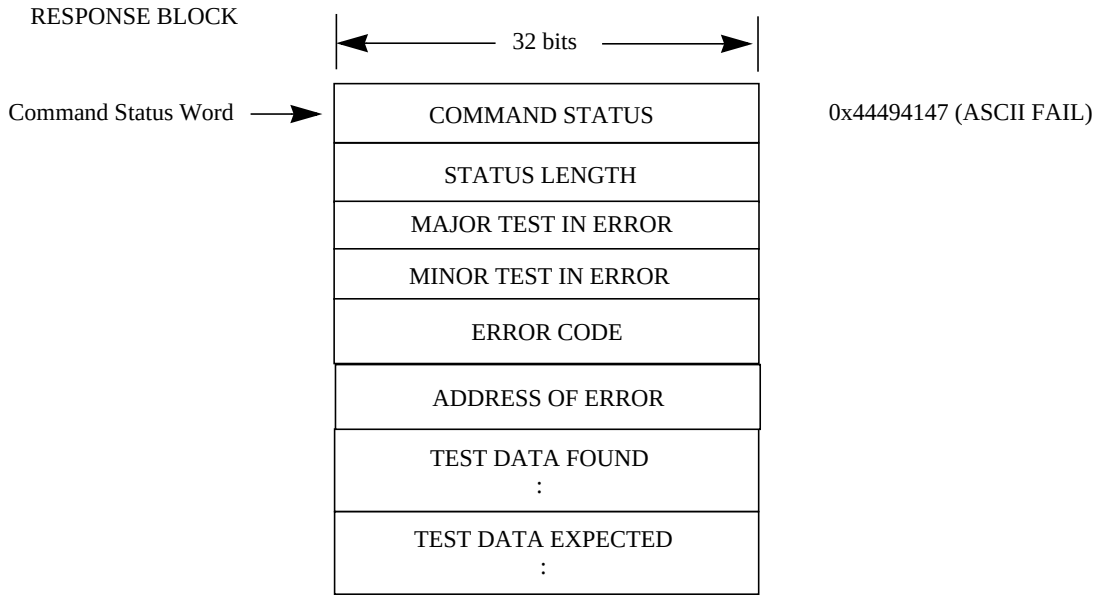


Figure 7-4. CB Diagnostic Error Data Structure

Power-up Diagnostic Test

Power-up memory diagnostics are run at power-up or system reset. This is an abbreviated test of all three segments of memory. The first test checks the integrity of the data bus. If no problem is discovered, BRAM (buffer RAM), SRAM, and then the PRAM (program RAM) are tested. The address bus to each memory space is exercised to find address lines that are stuck, shorted, or disconnected.

Extended Diagnostic Test

Each of the extended memory tests perform the same operations. They write each longword with its address, and then return and check that each is correct. If no problem is discovered, a walking 1's test is performed.

ATM Front-End Diagnostics — Major Test Code 0x2

The format of the command block for the front-end diagnostics is shown in Figure 7-1 on page 81. The format of the response block is shown in Figure 7-2 on page 82.

The front-end diagnostics perform tests on the fragmentation (F-FRED) and reassembly (R-FRED) devices. These minor tests include the Fragmentation FRED (F-FRED) SRAM, the Reassembly FRED (R-FRED) SRAM, configuration registers in the F-FRED and the R-FRED, and internal and external loopback tests.

Minor Test Codes

Six minor test codes can be issued in conjunction with the front-end major test code 0x2. These codes are listed in the following table:

Code	Name	Function
0x0	PUP_TEST	Runs all minor tests except the external loopback.
0x1	Internal Loopback	Tests the F-FRED and R-FRED chips.
0x4	External Loopback	Tests the entire data path. **NOTE:** Requires a loopback plug.
0x5	F-FRED SRAM Test	Performs a Data=Address test of the F-FRED SRAM. Writes data to the F-FRED SRAM, reads the data back, and compares the two sets of data.
0x6	R-FRED SRAM Test	Writes to the R-FRED SRAM, reads the data back and compares it with the data that was written.
0x8	F-FRED CPU Register Test	Performs a walking 0's and walking 1's test of the F-FRED CPU registers.
0x9	R-FRED CPU Register Test	Performs a walking 0's and walking 1's test of the R-FRED CPU registers.

0x0—PUP_TEST

The power-up test performs all minor tests except for the external loopback. An example of a PUP_TEST command is:

Name	Value in Hex
Length	0x10
Major	0x2
Minor	0x1

0x1—Internal Loopback Test

The internal loopback verifies that the FRED chips are functioning properly. The test transmits a data pattern that is fragmented by the F-FRED and then internally wrapped back to the R-FRED. The R-FRED reassembles the frames into a data pattern. The reassembled data pattern is compared to the transmitted pattern. If the patterns do not match, a failure is reported. The optics are not tested by the internal loopback test. If another station is connected to the board when the internal loopback test is executed, it will see frames on the wire.

An example of an internal loopback command is:

Name	Value in Hex
Length	0x10
Major	0x2
Minor	0x1

0x4—External Loopback Test

The external loopback requires a loopback plug. The external loopback test verifies that the FRED chips and optics are functioning properly. The test is similar to the internal loopback. However, the frames are externally wrapped back through the loopback plug to the R-FRED and reassembled. The reassembled data pattern is compared to the transmitted pattern. If the patterns do not match, the DIAG_ERROR code (0x2E) is written to the CB Previous Command Status, and FAIL is written to the Command Status Word. (See *Test Results* on page 90.)

An example of an external loopback command is:

Name	Value in Hex
Length	0x10
Major	0x2
Minor	0x4

0x5—F-FRED SRAM Test

The F-FRED SRAM test writes data to the SRAM, reads the data back, and then compares the two sets of data. If the data does not match, the DIAG_ERROR code 0x2E is written to the CB Previous Command Status, and FAIL is written to the Command Status Word. (See *Test Results* on page 90.)

An example of an F-FRED SRAM command is:

Name	Value in Hex
Length	0x10
Major	0x2
Minor	0x5

0x6—R-FRED SRAM Test

The R-FRED SRAM test writes data to the SRAM, reads the data back, and then compares the two sets of data. If the data does not match, the DIAG_ERROR code 0x2E is written to the CB Previous Command Status, and FAIL is written to the Command Status Word. (See *Test Results* on page 90.)

An example of an R-FRED SRAM command is:

Name	Value in Hex
Length	0x10
Major	0x2
Minor	0x6

0x8—F-FRED Register Test

The F-FRED configuration registers are tested with a walking 0's and walking 1's test. After the registers are tested, they are reset. If the test fails, the DIAG_ERROR code 0x2E is written to the CB Previous Command Status, and FAIL is written to the Command Status Word. (See *Test Results* on page 90.)

An example of an F-FRED Register command is:

Name	Value in Hex
Length	0x10
Major	0x2
Minor	0x8

0x9—R-FRED Register Test

The R-FRED configuration registers are tested with a walking 0's and walking 1's test. After the registers are tested, they are reset. If the test fails, the DIAG_ERROR code 0x2E is written to the CB Previous Command Status, and FAIL is written to the Command Status Word. (See *Test Results* on page 90.)

An example of an R-FRED Register command is:

Name	Value in Hex
Length	0x10
Major	0x2
Minor	0x9

Test Results

The diagnostic command issued is moved to the CB Previous Command Word. (See Figure 7-2 on page 82.) If the test runs successfully, CBOK is written into the Command Status Word. On a failure, the DIAG_ERROR code 0x2E is written to the CB Previous Command Status, and FAIL is written to the Command Status Word, followed by error data appropriate to the test failure.

VME DMA Diagnostics — Major Test Code 0x3

This section describes the available DMA diagnostic commands, options, error codes, and power-up diagnostics.

Command Block

The format of a typical command block for the DMA Diagnostic command is as shown in the following table. The format of the response block is shown in *Response Block* on page 92.

Name	Value in Hex
Length	0x20
Major	0x3
Minor	0x5
Sys address	0x100000
DMA count	0x200
Transfer options	023D
Data seed	0x55555555
Data option 0	0x80023
Data option 1	0x1200

NOTE

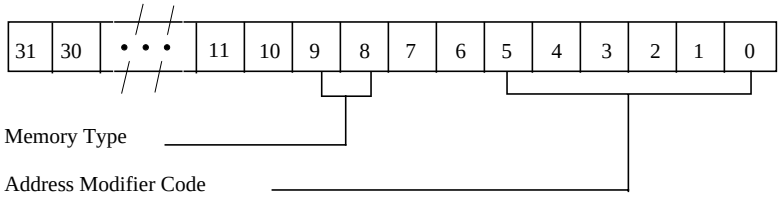
Not all DMA diagnostic commands use all of the fields shown in the command block.

Transfer Options Word

The Transfer Options Word is used to specify the type of VMEbus transaction used for data transfers. The bits in the Transfer Options Word indicate whether the transfer should be 16, 32, or 64 bits wide, and which address modifier should be used.

The Transfer Options Word is defined as a 32-bit unsigned integer, even though only the least significant word of the field (bits 0–15) is used to specify options. The most significant word in the field (bits 16–31) is a reserved 16-bit field.

Following is the format of the Transfer Options Word:



Note: All unused bits are reserved and must be cleared to 0.

Figure 7-5. Transfer Options Word for Data Transfers

Bits 0–5 Address Modifier

This field contains the VMEbus address modifier used by the controller for all VMEbus data transfers associated with this data transfer. It is not modified in any way by the controller.

Bits 6–7 Reserved

These bits are reserved and must be cleared to 0.

Bits 8–9 Memory Type

This two-bit field specifies the width of the data transfer. Permitted values are shown in the following table:

Bit 9	Bit 8	Memory Type
0	0	(reserved)
0	1	16-bit transfers

Bit 9	Bit 8	Memory Type
1	0	32-bit transfers
1	1	64-bit transfers

Bits 10–31 Reserved

These bits are reserved and must be cleared to 0.

Response Block

The following fields are a part of the response block for the DMA Diagnostic test:

Name	Value in Hex
Length	0x2C
Major Test Code	0x3
Minor Test Code	*
Status Error	*
Buffer Address	0x38000000
Data Offset	0x100
Expected Value	0x55555
Found Value	0x5554
Addgen Status Register	0x4001
Addgen IOP Status	0x1
Interrupt Port Read	0x8

The Buffer Address, Data Offset, Expected Value, and Found Value fields are for miscompares only. The Addgen Status Register is discussed in the next section.

ADDGEN Status Registers

The ADDGEN is a DMA state machine and SRAM /DRAM controller. The ADDGEN status registers provide information as part of the DMA diagnostic response block.

Figure 7-6 shows the ADDGEN General Status Register:

31–16	15–14	13–12	11–9	8–7	6–4	3	2	1	0
rsvd	rev = 0x01	rsvd	dma chanl has work	rsvd	dma chanl busy	bus tmo intr	bus error intr	count exhst intr	dma done intr

Figure 7-6. ADDGEN General Status Register—High True Values

Figure 7-7 shows the ADDGEN IOP Status Register:

31–21	20	19–18	17–15	14–13	12–10	9–8	7–5	4–3	2–0
rsvd	tick timer active	rsvd	dma chanl tmo intr	rsvd	dma chanl bus error	rsvd	dma chanl exhst intr	rsvd	dma chanl done intr

Figure 7-7. Addgen IOP Status Register—High True Values

Figure 7-8 shows the MBC Interrupt Port:

31–19	18	17	16	15	14	13	12	11	10
rsvd	Big timer	uart A break	uart A xmit	uart A rcv	uart B break	uart B xmit	uart B rcv	acfail	dbug abort

Figure 7-8. MBC Interrupt Port—Low True Values

Minor Test Codes

The following table lists the minor test codes for the DMA Diagnostics:

Minor Test Code	Name	Function
0x0	PUP_TEST	Quick power-up diagnostic
0x2	STANDARD DMA	Executes a VME write followed by a VME read. With auto data build and compare.
0x3	RESERVED	N/A
0x4	RAW DMA	Executes a VME read followed by a VME write of the same buffer. No data manipulation.
0x5	READ BUFFER	Executes a VME write with the specified onboard buffer. Builds data according to user parameters.

Minor Test Code	Name	Function
0x6	WRITE BUFFER	Executes a VME read into the specified onboard buffer. Compares data according to the user parameters.
0x7	VIRQ	Issues a VME interrupt on the given level and with the given vector.
0x8	DUMP ASIC REGS	Reads and prints all registers of the specified ASIC. UART output only.

0x0—PUP_TEST

The power-up test (PUP_TEST) executes a Read Buffer using the ADDGEN special mode of no VME bus involvement from the first page of VRAM. A data pattern of 0x55555555 with the incrementing/inverting option is built. This is followed by a Write Buffer to the next page of VRAM using the same data and data options to compare the data.

This test can be executed from the host/UART also.

Parameters

None; only the major and minor test codes are needed.

Following is a Power-up test command example:

Name	Value in Hex
Length	0xC
Major	0x3
Minor	0x0

0x2—Standard DMA

Builds a byte-wide incrementing pattern in the first page of VRAM and does a VME write DMA followed by a VME read DMA to the first page of SRAM and compares the data.

Parameters

System address (physical). DMA count, in bytes.

Transfer options. Standard usage. (See *Transfer Options Word* on page 91.)

Following is a Standard DMA test command example:

Name	Value in Hex
Length	0x18
Major	0x3
Minor	0x2

Name	Value in Hex
Sys address	0x100000
DMA count	0x200
Transfer options	023D

Response

See Appendix C on page 159 for Common Boot error codes. See *DMA Diagnostic Error Codes* on page 99 for DMA Error codes.

0x3—This Test Is Reserved

Test 0x3 is not supported by the 5215. A response is not applicable.

0x4—Raw DMA

Executes a VME read DMA followed by a VME write DMA. No data is built or compared.

Parameters (same as the Standard DMA test)

Following is a Raw DMA test command example:

Name	Value in Hex
Length	18
Major	03
Minor	04
Sys address	100000
DMA count	200
Transfer options	023D

Response

See Appendix C on page 159 for Common Boot error codes. See *DMA Diagnostic Error Codes* on page 99 for DMA Error codes.

0x5—READ Buffer

Builds an onboard buffer according to the host supplied parameters and executes a VME write DMA.

Parameters

System address (physical). DMA count, in bytes. If this value is set to 0x0 then the special ADDGEN mode will be used where no host bus state machine (HBSM) or VME bus involvement will take place.

A 2k buffer will be will be transferred from the random access side of VRAM to the serial bank. All normal addgen interrupts and handshake still take place. This is useful for determining if the failure is in the CPU-ADDGEN area or the ADDGEN-HBSM-VME area.

Transfer options. Standard usage. (see *Transfer Options Word* on page 91.)

Data seed. Seed value for the data pattern to build.

Data options. The bit structures for the READ Buffer Data Option 0 and Data Option 1 are shown in the following figures:

Option Word	31–16	15–10	9	8	7–4	3–0
0	buffer #	reserved	reset on fail	ram select	data width	data pattern

Figure 7-9. Bit Structure of READ Buffer Data Option 0

Option Word	31–16	15–12	11–9	8–0
1	reserved	buffer offset	burst control	reserved

Figure 7-10. Bit Structure of READ Buffer Data Option 1

Buffer #. This value is for the onboard buffer offset in 512 byte increments. (That is, if the value was 0x8, then the onboard buffer used would be (0x8 * 0x200) + base address of the ram selected.)

Reserved. This value should be set to 0x0.

Reset on fail. If this bit is set to 0x1 then the hardware will NOT be reset after a failure condition. This allows the hardware to be interrogated upon encountering a failure condition.

RAM select. This bit defines whether to use VRAM or SRAM for the onboard ram. A 0x0 selects VRAM and a 0x1 selects SRAM.

Data width. These bits define how the data is to be built or compared by the processor. This is useful for determining where data bit failures are between the CPU and buffers. Values are 0x0, 0x1 and 0x2 for byte, word and long.

Data pattern. These bits define how data is to be built or compared. The following table assumes that the data seed is 0x55, the data width is bytes and the length is 4 bytes.

Reserved. This value should be set to 0x0.

Buffer offset. This 4-bit field is used to determine on which of the first 16 bytes of the selected buffer to start. This allows changing the byte boundary of the internal buffer being used.

Burst control. If these 3 bits are cleared, no burst control is requested; otherwise, the value in the burst control field * 16 is selected. (For example, burst control = 2 causes a burst value of 32 transfers to be selected.)

Reserved. This value should be set to 0x0.

Following are the data pattern values for the READ Buffer command:

Value	Function	Example
0x0	Solid	0x55555555
0x1	Increment	0x55565758
0x2	Inverted	0x55aa55aa
0x3	Increment and Invert	0x55aa56a9
0xF	No manipulation	Buffer used as is

Following is a Read Buffer command example:

Name	Value in Hex
Length	0x20
Major	0x3
Minor	0x5
Sys address	0x100000
DMA count	0x200
Transfer options	023D
Data seed	0x55555555
Data option 0	0x80023
Data option 1	0x1200

Response

See Appendix C on page 159 for Common Boot error codes. See *DMA Diagnostic Error Codes* on page 99 for DMA Error codes.

0x6—Write Buffer

Executes a VME read DMA to the onboard buffer location specified and compares data according to the host supplied parameters.

Parameters

Same as Read buffer except for the 0x0 byte count mode. In the special 0x0 byte count mode the serial bank of the VRAM is transferred to the random access side of VRAM and compared according to the supplied parameters.

Command example: same as Read buffer except for the Minor code.

Response

See Appendix C on page 159 for Common Boot error codes. See *DMA Diagnostic Error Codes* on page 99 for DMA Error codes.

0x7—VIRQ

A VME interrupt will be generated on the level and with the vector supplied in the command.

Parameters

The following figure shows VIRQ parameter bits:

31–11	10–8	7–0
Reserved	IRQ level	IRQ vector

Reserved—Bits 11–31. Bits 11–31 should be set to 0x0.

IRQ level—Bit 8–10. For the IRQ level, legal values are 0x1–0x7.

IRQ vector—Bits 0–7. Bits 0–7 indicate the bus vector supplied during IACK.

Following is a VIRQ command example:

Name	Value in Hex
Length	0x10
Major	0x3
Minor	0x7
IRQ options	0x0155

Response

See Appendix C on page 159 for Common Boot error codes. See *DMA Diagnostic Error Codes* on page 99 for DMA Error codes.

0x8—Dump ASIC Registers

Used as a UART command only, this command will dump the register contents of the specified ASIC. The format of the dump is register # = value (that is 10 = 1234).

Parameters

Selected ASIC. The values for the selected ASIC are:

1 = Addgen
2 = Megamux
3 = MBC

Following is a Dump ASIC Registers command example:

Name	Value in Hex
Length	0x10
Major	0x3
Minor	0x8
ASIC select	0x1

Response

See Appendix C on page 159 for Common Boot error codes. See *DMA Diagnostic Error Codes* on page 99 for DMA Error codes.

DMA Diagnostic Error Codes

The following table lists the VME DMA diagnostic error codes. See Appendix C on page 159 for more information on Common Boot error codes:

Value	Description
0x0	PASS
0x1	Data miscompare
0x2	VME bus time out reported by the addgen
0x3	VME buserr reported by the addgen
0x4	Addgen interrupt time-out, no done, error, or tick interrupt occurred
0x5	Premature interrupts, interrupts present before the test started
0x6	The addgen tick timer elapsed before the DMA done or error interrupts
0x7	Addgen status register lacked sanity
0x8	Addgen spurious interrupt, and interrupt occurred when not expected
0x9	VIRQ time-out, the VIRQ inport bit was never seen
0xA	VIRQ IACK time-out, the IACK inport bit was never seen
0xB	Premature VIRQ interrupt, the VIRQ inport bit was present before the test was started
0xFE	Bad command parameter supplied
0xFF	Illegal test code

Host Adapter Ring Interface (HARI) 8

Overview

The VME ATM software interface is the Host Adapter Ring Interface (HARI). This interface is a series of channels that facilitate communication between the host system and the 5215 controller. This chapter describes the Host Adapter Ring Interface and the functions of the channels that comprise the interface.

NOTE

The functions described in this chapter require firmware revision A02 or later.

Host Adapter Ring Interface Functions

The Host Adapter Ring Interface (HARI) provides three basic functions to the host system:

1. Provides the mechanism for the host to issue commands to the controller that do the following:
   a. Configure the channels and alter operating characteristics
   b. Configure and manage the ATM Virtual Connections
   c. Allocate on-board resources
2. Provides the mechanism for the controller to return asynchronous event indications back to the host.
3. Provides the mechanism for system-to-front-end data transfers.

These functions are accomplished through three basic blocks:

- A **Short I/O Control Block**, which is a shared memory data structure between the host and controller
- **Rings**, which contain a set of pointers to system memory locations
- **Memory Buffers**, which contain commands to be issued to the board, data to be transmitted, or represent host memory resources to contain incoming data.

A host system communicates with the 5215 controller through a series of **channels**. Each channel provides a specific function and consists of a Short I/O Control Block (SCB), a Control Ring, and Command/Data Buffers. There are two fundamental types of channels, **send** and **receive**. Send channels provide host-to-controller command/data transfers, and receive channels provide controller-to-host indication/data transfers.

All controller maintenance is carried out through the **control send** channel, and all status change indications are reported through the **control receive** channel. All data to be output on the front-end passes through a **data send** channel, and all incoming data passes through a **data receive** channel. ATM raw and OAM cells are returned through the **raw receive** channel.

HARI Channel Types

There are five types of channels in the Host Adapter Ring Interface:

- Control Send (CTLS)
- Control Receive (CTLR)
- Data Send (HIPS/LOPS)
- Data Receive (SMLR/LRGR)
- Raw/OAM Cell Receive (RAWR)

Basic Blocks Of a Channel

Regardless of the specific function performed, each channel operates exactly the same and contains the three basic blocks (shown in Figure 8-1 on page 103):

- Short I/O Control Block (SCB)
  - The Short I/O Control Block (SCB) controls accesses to, and provides operational status for, its respective ring. Each SCB contains interrupt status, a host ring pointer, and a controller ring pointer. The host pointer indicates the next available element to be submitted by the host, while the controller pointer specifies the next element to be processed by the controller.
  - All SCBs must reside in controller Short I/O, and are fixed in length. Their location is specified by the host at configuration.
- Rings & Ring Elements
  - Each ring is a sequential (circular) array of elements, where each element specifies a buffer to be exchanged between the host and controller. The channel with which the ring is associated defines the type of data contained in the buffer and the direction of the transfer.
  - Rings may reside anywhere in VME accessible memory. Ring size and location are specified by the host at configuration.
- Buffers
  - Buffers may contain command, status, or front-end data. Only those buffers associated with either of the control channels (control send or control receive) are processed by the controller. Buffers associated with a data channel are transparent to the controller and are moved directly between the system and front-end.
  - Buffers may reside anywhere in VME accessible memory. Buffer size and location are specified by the host at configuration and/or run-time.

Figure 8-1 illustrates the three basic blocks of the host adapter ring interface:

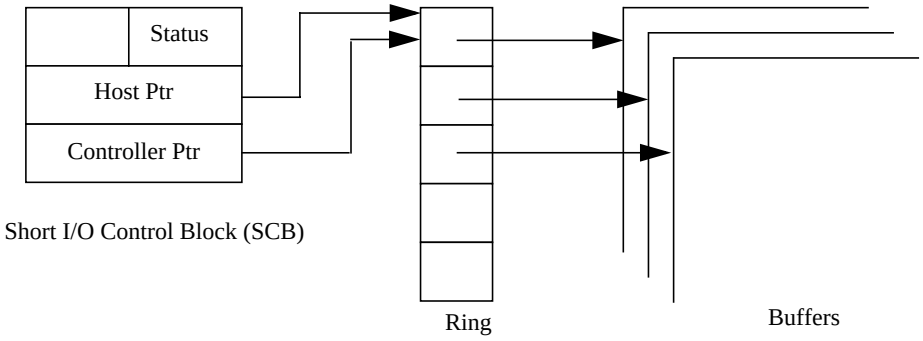


Figure 8-1. The Three Basic Blocks of the Host Adapter Ring Interface

Ring Operation

At boot time, the short I/O control blocks are initialized by an intermediate boot loader, such as the Common Boot interface, which supplies the parameters to configure the channels. Each channel is supplied with an interrupt level and interrupt vector, ring address, ring length, and DMA controls, and various other operating parameters.

Upon initialization, the pointers are set to point to the first ring element as illustrated in Figure 8-2a.

```
ctlr ptr = host ptr = starting address of first ring element
```

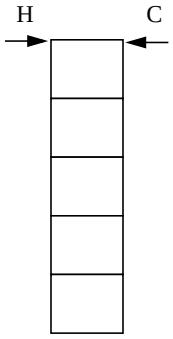


Figure 8-2a. Empty Ring

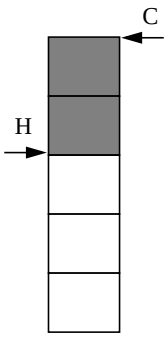


Figure 8-2b. Ring Partially Full

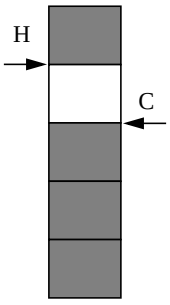
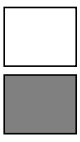


Figure 8-2c. Ring full and Wrapped

H = Host Pointer
C = Controller Pointer
H = C = Empty Ring
H + 1 = C = Ring Full



= Empty Ring Element

= Full Ring Element

Figure 8-2. The Three Basic Ring States

Ring Empty State

When the ring is empty (Figure 8-2a on page 103), it indicates that there is no data to be processed by the controller, and no host memory space is required to service incoming data.

Ring Active State

When the ring is partially full (shown in Figure 8-2b on page 103), it indicates that the host has initiated a transaction, and the controller has data to process. Once the controller processes the data, it will return the buffer status information, and then advance its pointer to indicate that the transfer is complete.

Figure 8-2b illustrates the condition where the host has added two items to be processed by the controller. Figure 8-2c on page 103 illustrates the condition where the host has added six items for the controller to process, and the controller has completed the transfer of two items. This can be seen by noting that the ring length is five elements. The host has submitted five items, filling up the ring. During this time the controller has completed two of the items (the top two elements shown in Figure 8-2c); also, the host pointer has wrapped back to the first element and submitted another item to be processed, for a total of six elements submitted by the host.

Ring Full State

The ring is full when there is only one empty ring element between the host pointer and the controller pointer. There must be at least one empty ring element to prevent ring overflow. By definition, the ring is full when the host pointer plus one element equals the controller pointer. In other words:

```
Host Pointer + 1 = Controller Pointer
```

When ring overflow occurs, the host pointer and controller pointer point to the same element. This condition is defined as an empty ring. In this scenario the controller does not recognize that there is data to process.

NOTE

The host pointer and the controller pointer must always point to the beginning of a valid ring element.

Structure Definitions of Basic Channel Blocks

This section provides a detailed description of the following HARI software structures:

- Channel Short I/O Control Block
- Control Ring Element
- Data Send Ring Element
- Data Receive Ring Element
- Raw/OAM Receive Ring Element

Short I/O Control Block

The handshake between host and controller for each channel is performed through a shared memory structure located in the controller's Short I/O controller block. This field is initialized by the controller when the channel is configured by the host. Afterwards, both the host and controller maintain respective pointers that indicate ownership of the ring elements. Interrupt status and control is also interchanged here.

Structure

Figure 8-3 shows the Short I/O Control Block structure:

Offset	Description 31 . . . . . 16 15 0	
0x0	Reserved	Interrupt Timer
0x4	Host Pointer	
0x8	Controller Pointer	
0xC	Reserved	

Figure 8-3. Channel Short I/O Control Block (SCB) Structure

Fields

The following topics describe the fields in the channel short I/O control structure.

Interrupt Timer

The Short I/O Control Block has a 16-bit Interrupt timer as part of its channel descriptor. The host can modify this timer to meter its interrupt load on a per-channel basis. When the interrupt timer field is cleared to zero (that is, all bits are cleared to zero), the controller can post an interrupt. When the interrupt timer field is set to ones (that is, all bits are set to one) this indicates that an interrupt is pending.

The host sets the timer by writing any value from 0 through 0xFFFE into this field. The controller interprets this value as units of 1 milliseconds (1 ms). The controller decrements this timer once every millisecond. Once the specified amount of time expires, the controller will interrupt the host if (or as soon as) the channel contains pending messages.

Writing 0xFFFF to the interrupt timer field causes interrupts to be suspended from the associated channel. This value is used by both the controller and the host as follows:

- Prior to interrupting the host, the controller writes 0xFFFF to the timer of the channel that caused the interrupt. This suspends further interrupts from the channel until the interrupt service routine resets the timer.
- After servicing the interrupt, the host sets the timer field to a value from 0 to 0xFFFE to enable the interrupt to acknowledge future interrupts.

Reserved

This field is reserved for use by the controller.

Host Pointer

This pointer holds the address of the next ring element available to the host, and is maintained by the host. It must always point to the beginning of a valid element.

Controller Pointer

This pointer holds the next ring element to be processed by the controller, and is maintained by the controller.

Reserved

This field is reserved for use by the controller.

Short I/O Allocation Example

Figure 8-4 shows an example of how a host can allocate the controller's short I/O. This is a recommended layout only — it is **not** the only way to allocate the controller's short I/O.

Offset	0x0	0x4	0x8	0xC
0x00	MSR			
0x10	Control Send Channel SCB			
0x20	Control Receive Channel SCB			
0x30	Data Send Channel (1) SCB			
0x40	Data Send Channel (2) SCB			
0x50	Small-Buffer Data Receive Channel SCB			
0x60	Large-Buffer Data Receive Channel SCB			
0x70	Raw Receive Channel SCB			
0x80	CBOK			
: :	: :			
0x13C : : : 0x1FC	Host Statistics Block			

Figure 8-4. Example Controller Short I/O Layout

Control Ring Element

The control send ring is used by the host to issue configuration and maintenance commands to the controller. The control ring element points to the host buffer that contains the command to be processed. Upon completion of the command, the result is returned to the command buffer, the EXEC field is updated in the ring element, the controller updates its SCB pointer, and, if enabled, an interrupt is generated to the host.

The control receive ring is used to report any asynchronous status change indications (that is, link up/down) to the host. The host must *anticipate* any such event by making buffers available prior to any actual occurrence. Upon any event, the message is written to the buffer, the EXEC field is updated in the ring, the controller updates its SCB pointer, and, if enabled, an interrupt is generated to the host.

Structure

Figure 8-5 shows the control ring element structure:

Offset	Description 31 . . . . . 16 15 0	
0x0	EXEC	DMAC
0x4	VME Address	
0x8	Buffer Length	
0xC	Host Usable Tag	

Figure 8-5. Control Ring Element Structure

Fields

The following topics describe the fields in the Control Ring Element Structure.

EXEC

The Execution Control (EXEC) Word contains the control and completion status for the ring element. Execution options are provided by the host, and completion status is returned by the controller. The bits are defined as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC	ER	EX	0	0	FB	IE	EP	BER	BTO	0	0	0	0	0	0

Figure 8-6. EXEC Word Definition

Bit 15: CC—Command complete. This bit should be cleared to 0 by the host prior to releasing the ring element to the controller. Upon completion of the element, the controller will set this bit to 1. All other returned status bits should be considered invalid until this bit is set to 1.

Bit 14: ER—Command completed with error. This bit should be cleared to 0 by the host prior to releasing the ring element to the controller. Upon completion of the element, if an error occurred, the controller will set this bit to 1. The BER and BTO bits indicate VME transfer errors and the control send command buffers contain further error definitions.

Bit 13: EX—Command completed with exception. This bit should be cleared to 0 by the host prior to releasing the ring element to the controller. Upon completion of the element, the controller will set this bit to 1 if an exception event occurred (currently there are no exception events defined).

Bits 12-11: Reserved. These bits are reserved and should be cleared to 0.

Bit 10: FB—FRED (Fragmentation/Reassembly Device chipset) bug-bytes. The segmentation hardware has a bug when transmitting AAL5 packets. It decrements the length field by 52 instead of 48 for each cell of the packet, requiring adjustment to allow for the bug.

If set to 1, the firmware will do this conversion. If cleared to 0, the host must provide the conversion.

Bit 9: IE—Interrupt Enable. When this bit is set to 1, it enables the host completion interrupt to be generated.

Bit 8: EP—End-of-Packet When this bit is cleared to 0, it indicates that this element is to be combined with the following elements until the EP bit is set to 1, forming a single packet. When this bit is set to 1, it indicates the end of a packet.

For data send channels, host data buffers are packed (gathered) into internal buffers until the EP bit is set to1.

For data receive channels, the internal packet buffer consumes (scatters) as many host buffers as are required. The EP bit is set to 1 in the last element and cleared to 0 for all others. This scatter function does require some initialization and impose some restrictions. (See the Channel Configuration Block's RSE flag and restrictions.)

Bit 7: BER—VME Bus Error. This bit should be cleared to 0 by the host prior to releasing the ring element to the controller. Upon completion of the element, the host will set this bit to 1 if a VME Bus Error was detected during the VME transfer. The ER bit will be set to 1 also.

Bit 6: BTO—VME Bus Time Out. This bit should be cleared to 0 by the host prior to releasing the ring element to the controller. Upon completion of the element, the host will set this bit to 1 if a VMEbus timeout was detected while attempting the VME transfer. The ER bit will be set to 1 also.

Bits 5-0: Reserved. These bits are reserved and should be cleared to 0.

NOTE

The SCB's Host Pointer should never point to an element that does not have the EP bit set.

DMAC

The DMA Control (DMAC) Word contains bit options that control VME transfers. It consists of the following bits:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	User Defined Address Modifier						AM	0	BUS		ADR		BM	SU	RW

Figure 8-7. DMAC Word Definition

Under normal circumstances, the host should only use bits 0–6 to describe the optimal transfer options. The controller will adjust the actual transfer performed based on buffer alignment and sizes (providing the appropriate Address Modifier Codes and transfer protocols).

Bits 8–14 should be used only when addressing devices that require vendor-unique address modifier codes.

Bit 15: This bit is reserved and must be cleared to 0.

Bits 14-9: User defined Address Modifier. The field is only used when the AM bit is set to 1 (that is, only if bit 8 is set). This field specifies a user defined source for the VMEbus Address Modifier code. When bit 8 is set, bits 9-14 are forced onto the VMEbus Address Modifier code lines.

Bit 8: AM—Address Modifier Code source. When this bit is cleared to 0, the controller dynamically determines and adjusts the address modifier code from the information in the SU/BM/ADR/BUS options, buffer alignment and size, and internal state information.

If set to 1, bits 9–14 specify the Address Modifier Code to be presented throughout the VME transfer.

Bit 7: This bit is reserved and must be cleared to 0.

Bits 6-5: BUS—Maximum VMEbus Size. These bits are used by the controller to determine the maximum data width for the VMEbus transfer (D8, D16, D32, D64). The bits are defined as follows:

Bit 6	Bit 5	VMEbus Data Size
0	0	8-bit maximum data size
0	1	16-bit maximum data size
1	0	32-bit maximum data size
1	1	64-bit maximum data size

Bits 4-3: ADR—Address Size. These bits are used by the controller to set the VMEbus address modifier code to correctly indicate the selected address size (A16, A24, A32).

NOTE

All 32 bits of address are always driven during the address portion of a cycle regardless of the setting of the Address Size bits.

The Address Size bits are defined as follows:

Bit 4	Bit 3	Address Size
0	0	16-bit Address
0	1	24-bit Address
1	0	32-bit Address
1	1	Illegal

Bit 2: BM—Block mode. When this bit is set to 1, block mode transfers are enabled where possible. When cleared to 0, block mode is disabled.

Bit 1: SU—User/Supervisor (1—super, 0—user). When this bit is set to 1, the VMEbus address modifier code will be set to supervisory mode. When this bit is cleared, the VMEbus address modifier code will be set to non-privileged mode.

Bit 0: RW—Direction (0—VME write, 1—VME read). When this bit is cleared to 0, the data transfer is from the controller to the VME host system. If set to 1, the transfer is from the VME host system to the controller.

VME Address

This field specifies the physical address of the command/data buffer containing data to be transferred.

Buffer Length

This field specifies the total number of bytes in the data buffer.

Host Usable Tag

If needed for the application, this field may contain a host-generated command tag. Command tags are typically generated by the host's device driver and can be used to notify host processes associated with a command. HARI does not use the command tag in any way; it simply returns the value in the response.

Data Send Ring Element

The data send ring(s) are used to transfer data packets from host system buffers to the ATM media. Each data send ring element points to the buffer to be transmitted onto the media. A virtual circuit (VC) table entry must have been initialized prior to issuance of a transmit. Upon completion, the EXEC field is updated, the controller updates its SCB pointer, and, if enabled, an interrupt is generated to the host.

Structure

Figure 8-8 shows the data send ring element structure:

Offset	Description 31 . . . . . 16 15 0	
0x0	EXEC	DMAC
0x4	VME Address	
0x8	Buffer Length	
0xC	Host Usable Tag	
0x10	MODE	VCT Index

Figure 8-8. Data Send Ring Element Structure

Fields

The following topics describe the fields in the data send ring element structure.

EXEC

The Execution Control (EXEC) Word contains the control and completion status for the ring element. Execution options are provided by the host, and completion status is returned by the controller. The bits are defined as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC	ER	EX	0	0	FB	IE	EP	BER	BTO	0	0	0	0	0	0

Figure 8-9. EXEC Word Definition

Bit 15: CC—Command complete. This bit should be cleared to 0 by the host prior to releasing the ring element to the controller. Upon completion of the element, the controller will set this bit to 1. All other returned status bits should be considered invalid until this bit is set to 1.

Bit 14: ER—Command completed with error. This bit should be cleared to 0 by the host prior to releasing the ring element to the controller. Upon completion of the element, if an error occurred, the controller will set this bit to 1. The BER and BTO bits indicate VME transfer errors and the control send command buffers contain further error definitions.

Bit 13: EX—Command completed with exception. This bit should be cleared to 0 by the host prior to releasing the ring element to the controller. Upon completion of the element, the controller will set this bit to 1 if an exception event occurred (currently there are no exception events defined).

Bits 12-11: Reserved. These bits are reserved and should be cleared to 0.

Bit 10: FB—FRED (Fragmentation/Reassembly Device chipset) bug-bytes. The segmentation hardware has a bug when transmitting AAL5 packets. It decrements the length field by 52 instead of 48 for each cell of the packet, requiring adjustment to allow for the bug.

If set to 1, the firmware will do this conversion. If cleared to 0, the host must provide the conversion.

Bit 9: IE—Interrupt Enable. When this bit is set to 1, it enables the host completion interrupt to be generated.

Bit 8: EP—End-of-Packet. When this bit is cleared to 0, it indicates that this element is to be combined with the following elements until the EP bit is set to 1, forming a single packet. When this bit is set to 1, it indicates the end of a packet.

For data send channels, host data buffers are packed (gathered) into internal buffers until the EP bit is set to1.

For data receive channels, the internal packet buffer consumes (scatters) as many host buffers as are required. The EP bit is set to 1 in the last element and cleared to 0 for all others. This scatter function does require some initialization and impose some restrictions. (See the Channel Configuration Block's RSE flag and restrictions.)

Bit 7: BER—VME Bus Error. This bit should be cleared to 0 by the host prior to releasing the ring element to the controller. Upon completion of the element, the host will set this bit to 1 if a VME Bus Error was detected during the VME transfer. The ER bit will be set to 1 also.

Bit 6: BTO—VME Bus Time Out. This bit should be cleared to 0 by the host prior to releasing the ring element to the controller. Upon completion of the element, the host will set this bit to 1 if a VMEbus timeout was detected while attempting the VME transfer. The ER bit will be set to 1 also.

Bits 5-0: Reserved. These bits are reserved and should be cleared to 0.

NOTE

The SCB's Host Pointer should never point to an element that does not have the EP bit set.

DMAC

The DMA Control (DMAC) Word contains bit options that control VME transfers. It consists of the following bits:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	User Defined Address Modifier						AM	0	BUS		ADR		BM	SU	RW

Figure 8-10. DMAC Word Definition

Under normal circumstances, the host should only use bits 0–6 to describe the optimal transfer options. The controller will adjust the actual transfer performed based on buffer alignment and sizes (providing the appropriate Address Modifier Codes and transfer protocols).

Bits 8–14 should be used only when addressing devices that require vendor-unique address modifier codes.

Bit 15: This bit is reserved and must be cleared to 0.

Bits 14-9: User defined Address Modifier. The field is only used when the AM bit is set to 1 (that is, only if bit 8 is set). This field specifies a user defined source for the VMEbus Address Modifier code. When bit 8 is set, bits 9-14 are forced onto the VMEbus Address Modifier code lines.

Bit 8: AM—Address Modifier Code source. When this bit is cleared to 0, the controller dynamically determines and adjusts the address modifier code from the information in the SU/BM/ADR/BUS options, buffer alignment and size, and internal state information.

If set to 1, bits 9–14 specify the Address Modifier Code to be presented throughout the VME transfer.

Bit 7: This bit is reserved and must be cleared to 0.

Bits 6-5: BUS—Maximum VMEbus Size. These bits are used by the controller to determine the maximum data width for the VMEbus transfer (D8, D16, D32, D64). The bits are defined as follows:

Bit 6	Bit 5	VMEbus Data Size
0	0	8-bit maximum data size
0	1	16-bit maximum data size
1	0	32-bit maximum data size
1	1	64-bit maximum data size

Bits 4-3: ADR—Address Size. These bits are used by the controller to set the VMEbus address modifier code to correctly indicate the selected address size (A16, A24, A32).

NOTE

All 32 bits of address are always driven during the address portion of a cycle regardless of the setting of the Address Size bits.

The Address Size bits are defined as follows:

Bit 4	Bit 3	Address Size
0	0	16-bit Address
0	1	24-bit Address
1	0	32-bit Address
1	1	illegal

Bit 2: BM—Block mode. When this bit is set to 1, block mode transfers are enabled where possible. When cleared to 0, block mode is disabled.

Bit 1: SU—User/Supervisor (1—super, 0—user). When this bit is set to 1, the VMEbus address modifier code will be set to supervisory mode. When this bit is cleared, the VMEbus address modifier code will be set to non-privileged mode.

Bit 0: RW—Direction (0—VME write, 1—VME read). When this bit is cleared to 0, the data transfer is from the controller to the VME host system. If set to 1, the transfer is from the VME host system to the controller.

VME Address

This field specifies the address of the data buffer to be transferred.

Buffer Length

This field specifies the total number of bytes in the data buffer.

Host Usable Tag

If needed for the application, this field may contain a host-generated command tag. Command tags are typically generated by the host's device driver and can be used to notify host processes associated with a command. HARI does not use the command tag in any way, it simply returns the value in the response.

Mode

This field contains the ATM MODE Word. The following bits in the ATM Mode Control Word specify the packet segmentation to be performed:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTI Value			0	EM	CR	PCKTYP		0	0	0	0	0	0	0	0

Figure 8-11. ATM Mode Word Definition

Bits 15-13: PTI—Optional PTI field for OAM cell header. These bits are to be used as an optional source for the OAM cell header's PTI field when the packet type is 11. When the packet type is 10, the entire ATM cell header is derived from the Virtual Circuit (VC) Table.

Bit 12: Reserved. This bit is reserved and must be cleared to 0.

Bit 11: EOM—End of Message. When bit 11 is set to 1, it indicates that the segmentation hardware must set the EOM bit in the ATM header by forcing the PTI field to xx1 for EOM and SSM cells for AAL3/4 and AAL5 cells.

Bit 10: CRC32 Enable. When set to 1, this bit indicates that the segmentation hardware must generate a 32-bit frame check sequence (FCS) for the frame. Note that a bit can alternately be set in the VC table for similar effect. This bit provides FCS control on a per-frame basis where the bit in the VC table provides FCS control on a per-connection basis.

Bits 9-8: PCKTYP—Packet Type. These two bits indicate the segmentation type to be performed: AAL3/4, AAL5, or OAM (with the source of the PTI bits option):

Bit 9	Bit 8	Algorithm
0	0	AAL3/4 segmentation performed
0	1	AAL5 segmentation performed
1	0	An OAM cell is transmitted using PTI values from the VC table
1	1	An OAM cell is transmitted using PTI values from bits 15-13

Bits 7-0: Reserved. These bits are reserved and must be cleared to 0.

VCT Index

This field is a reference to the Virtual Circuit (VC) Table entry that controls the transmission of this data packet. The corresponding VC Table entry defines the ATM header (which includes the Virtual Path/Virtual Circuit ID), rate of transmission, and the AAL packet type. Data cannot be successfully sent until the table entry specified by the VCT Index has been properly initialized.

Data/Raw Receive Ring Element

The data receive ring(s) are used to transfer incoming packets from the ATM media to host data buffers. Each ring element points to a host buffer to receive the incoming packet. The host must provide buffers before receiving incoming packets (a VC Table entry must have been initialized to accept packets from the VCI). Once a packet has been reassembled, it is transferred into the host buffer, the EXEC/Status/VCI words are updated, the controller updates its SCB pointer, and, if enabled, an interrupt is generated to the host.

Structure

Figure 8-8 shows the data receive ring structure:

Offset	Description 31 . . . . . .23 . . . . . 16 15 0		
0x0	EXEC		DMAC
0x4	VME Address		
0x8	Buffer Length		
0xC	Host Usable Tag		
0x10	Reserved	Status/Error	VCI

Figure 8-12. Data Receive Ring Element Structure

Fields

The following topics describe the fields in the data receive ring element structure.

EXEC

The Execution Control (EXEC) Word contains the control and completion status for the ring element. Execution options are provided by the host, and completion status is returned by the controller. The bits are defined as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC	ER	EX	0	0	FB	IE	EP	BER	BTO	0	0	0	0	0	0

Figure 8-13. EXEC Word Definition

Bit 15: CC—Command complete. This bit should be cleared to 0 by the host prior to releasing the ring element to the controller. Upon completion of the element, the controller will set this bit to 1. All other returned status bits should be considered invalid until this bit is set to 1.

Bit 14: ER—Command completed with error. This bit should be cleared to 0 by the host prior to releasing the ring element to the controller. Upon completion of the element, if an error occurred, the controller will set this bit to 1. The BER and BTO bits indicate VME transfer errors and the control send command buffers contain further error definitions.

Bit 13: EX—Command completed with exception. This bit should be cleared to 0 by the host prior to releasing the ring element to the controller. Upon completion of the element, the controller will set this bit to 1 if an exception event occurred (currently there are no exception events defined).

Bits 12-11: Reserved. These bits are reserved and should be cleared to 0.

Bit 10: FB—FRED (Fragmentation/Reassembly Device chipset) bug-bytes. The segmentation hardware has a bug when transmitting AAL5 packets. It decrements the length field by 52 instead of 48 for each cell of the packet, requiring adjustment to allow for the bug.

If set to 1, the firmware will do this conversion. If cleared to 0, the host must provide the conversion.

Bit 9: IE—Interrupt Enable. When this bit is set to 1, it enables the host completion interrupt to be generated.

Bit 8: EP—End-of-Packet. When this bit is cleared to 0, it indicates that this element is to be combined with the following elements until the EP bit is set to 1, forming a single packet. When this bit is set to 1, it indicates the end of a packet.

For data send channels, host data buffers are packed (gathered) into internal buffers until the EP bit is set to1.

For data receive channels, the internal packet buffer consumes (scatters) as many host buffers as are required. The EP bit is set to 1 in the last element and cleared to 0 for all others. This scatter function does require some initialization and impose some restrictions. (See the Channel Configuration Block’s RSE flag and restrictions.)

Bit 7: BER—VME Bus Error. This bit should be cleared to 0 by the host prior to releasing the ring element to the controller. Upon completion of the element, the host will set this bit to 1 if a VME Bus Error was detected during the VME transfer. The ER bit will be set to 1 also.

Bit 6: BTO—VME Bus Time Out. This bit should be cleared to 0 by the host prior to releasing the ring element to the controller. Upon completion of the element, the host will set this bit to 1 if a VMEbus timeout was detected while attempting the VME transfer. The ER bit will be set to 1 also.

Bits 5-0: Reserved. These bits are reserved and should be cleared to 0.

NOTE

The SCB's Host Pointer should never point to an element that does not have the EP bit set.

DMAC

The DMA Control (DMAC) Word contains bit options that control VME transfers. It consists of the following bits:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	User Defined Address Modifier						AM	0	BUS		ADR		BM	SU	RW

Figure 8-14. DMAC Word Definition

Under normal circumstances, the host should only use bits 0–6 to describe the optimal transfer options. The controller will adjust the actual transfer performed based on buffer alignment and sizes (providing the appropriate Address Modifier Codes and transfer protocols).

Bits 8–14 should be used only when addressing devices that require vendor-unique address modifier codes.

Bit 15: This bit is reserved and must be cleared to 0.

Bits 14-9: User defined Address Modifier. The field is only used when the AM bit is set to 1 (that is, only if bit 8 is set). This field specifies a user defined source for the VMEbus Address Modifier code. When bit 8 is set, bits 9-14 are forced onto the VMEbus Address Modifier code lines.

Bit 8: AM—Address Modifier Code source. When this bit is cleared to 0, the controller dynamically determines and adjusts the address modifier code from the information in the SU/BM/ADR/BUS options, buffer alignment and size, and internal state information.

If set to 1, bits 9–14 specify the Address Modifier Code to be presented throughout the VME transfer.

Bit 7: This bit is reserved and must be cleared to 0.

Bits 6-5: BUS—Maximum VMEbus Size. These bits are used by the controller to determine the maximum data width for the VMEbus transfer (D8, D16, D32, D64). The bits are defined as follows:

Bit 6	Bit 5	VMEbus Data Size
0	0	8-bit maximum data size
0	1	16-bit maximum data size
1	0	32-bit maximum data size
1	1	64-bit maximum data size

Bits 4-3: ADR—Address Size. These bits are used by the controller to set the VMEbus address modifier code to correctly indicate the selected address size (A16, A24, A32).

NOTE

All 32 bits of address are always driven during the address portion of a cycle regardless of the setting of the Address Size bits.

The Address Size bits are defined as follows:

Bit 4	Bit 3	Address Size
0	0	16-bit Address
0	1	24-bit Address
1	0	32-bit Address
1	1	illegal

Bit 2: BM—Block mode. When this bit is set to 1, block mode transfers are enabled where possible. When cleared to 0, block mode is disabled.

Bit 1: SU—User/Supervisor (1—super, 0—user). When this bit is set to 1, the VMEbus address modifier code will be set to supervisory mode. When this bit is cleared, the VMEbus address modifier code will be set to non-privileged mode.

Bit 0: RW—Direction (0—VME write, 1—VME read). When this bit is cleared to 0, the data transfer is from the controller to the VME host system. If set to 1, the transfer is from the VME host system to the controller.

VME Address

This field specifies the address of the data buffer to be transferred.

Buffer Length

This field specifies the total number of bytes in the buffer.

Host Usable Tag

This field may contain a host-generated command tag, if needed for the application. Command tags are typically generated by the host's device driver and can be used to notify host processes associated with a command. HARI returns the value in the host-generated command tag.

VCI

This field specifies the Virtual Circuit Identifier (from the ATM cell header) from which the packet was received.

Status/Error

Bits 16–23 indicate any error that occurred during the data transfer:

23	22	21	20	19	18	17	16
RSE	CNG	EOF/ CF	PEP	CER/ CCE	PTE/ CSE	OFL/ SQE	CPE

Figure 8-15. Status/Error Word Definition

Bit 16: CPE is used to qualify the error conditions on bits 17-19, and 21. When a cell error is encountered, this bit is set to 1; otherwise, it is cleared to 0.

0—Bits 17-19 and bit 21 are packet error bits
1—Bits 17-19 and bit 21 are cell errors

Bit 17: If CPE is 0, then bit 17 is OFL; otherwise, bit 17 is SQE.

OFL is set when a packet overflows the packet buffer in the packet memory, and hence, is terminated.

SQE is set when a cell is received with an out-of-order AAL3/4 sequence number. The active packet from the VC where this cell was received is terminated.

0—No error
1—Buffer overflow error/cell sequence error for AAL3/4

Bit 18: If CPE is 0, then bit 18 is PTE; otherwise, bit 18 is CSE.

PTE is set when a packet has not completed within the pre-programmed amount of time, and hence has terminated.

CSE is set when a cell is received with a cell size that violates the size definition of AAL 3/4. The active packet on the VC on which this cell was received is terminated.

0—No error
1—Packet timeout/cell size error for (AAL 3/4)

Bit 19: If CPE is 0, then bit 19 is CER; otherwise, bit 19 is CCE.

CER is set when a packet CRC error occurs. CER should be ignored if packet CRC is not being used on this VC.

CCE is set when a cell is received with an erroneous payload CRC (10-bit CRC) belonging to the active packet (for AAL 3/4). The packet is then terminated.

0—No error detected
1—Packet CRC Error/cell payload CRC Error (AAL3/4)

Bit 20: PEP is set if a parity error is encountered on the link interface in the payload of the packet. If this bit is cleared it indicates that there was no parity error.

0—No parity error detected
1—Parity error in cell payload

Bit 21: If CPE is 0, then bit 21 is EOF; otherwise, it is CF.

EOF is set when an end-of-packet error occurs. This happens when the last cell of a packet is not received and the packet is terminated.

CF is set when a cell that may belong to the active packet was flushed due to errors on the link. The packet is terminated upon this error condition.

0—No error detected
1—End of Packet error/cell flushed

Bit 22: CNG is set when the reassembly hardware 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 experienced by one or more cells of this packet during transit.

Bit 23: RSE indicates roll over sequence error at packet boundaries, that is the first cell of this packet did not have the successive sequence number from the last cell received on this circuit.

0—No roll over sequence error
1—Roll over sequence error detected.

Reserved

These bits are reserved for use by the controller. The data returned in this field should be ignored by the host.

Command Definitions

This section defines the command codes and structures issued via the control send channel. All of the command codes share a common *generic* header (shown in Figure 8-16). This header specifies the length of the command, the command code issued, and the command status.

The commands perform diagnostics, configure the channels (data, send, and receive), and configure the virtual circuit table, the rate queues, and the congestion control.

Control Send Command Structures

All control send command blocks begin with the same generic header containing the command length, command code and command status. The data to be sent is appended to this generic header.

Generic Structure

Figure 8-16 shows the generic control send command structure:

Offset	Description 31 0
0x0	Length
0x4	Code
0x8	Status
0xC	Command Specific Data

Figure 8-16. Control Send Generic Header Structure

Generic Fields

The following topics describe the generic fields in the Control Send command structure.

Length

This field specifies the length of the command, in bytes, including this header.

Code

This field specifies the Control Send command code issued. The available command codes are shown in *Control Send Command Codes* on page 124.

Status

This field provides information on the result of the operation performed by the issued command. The host should invalidate this field prior to releasing it to the controller. The returned status codes are shown in *Control Send Status Codes* on page 124.

Data

This field contains command specific data. The data is not a part of the generic header; it is appended to the generic header prior to issuing the command.

Control Send Command Codes

The following HARI command codes are available (the full functional descriptions are defined later in this section):

Code	Description
0x00	Controller Info
0x01	Force Control Receive
0x10	Configure Control Send/Receive Channels
0x11	Configure VATM Data Channels
0x12	Configure Data Channels
0x13	Set VME Burst Count
0x14	Configure Raw/OAM Receive Channel
0x20	Configure Virtual Circuit Table Index
0x21	Report Virtual Circuit Table Index Configuration
0x30	Configure Rate Queue
0x31	Report Rate Queue Configuration
0x40	Write FLASH Sector
0x41	Read FLASH Sector
0x60	Configure Congestion Control
0x61	Report Congestion Control Configuration

Control Send Status Codes

Following are the completion status codes returned by the controller:

Code	Status	Description
0x00	OK	Command completed normally
0x01	Code	Invalid Command Code
0x02	Length	Invalid Command Length
0x03	Spec	Error in the Command Specific Data

Code	Status	Description
0x04	Max Channel	Receive Count or Send Count too large
0x1B	ESC	UART <ESC> key received
0x50	Flash Count	Byte count too large
0x51	Burn Failed	An Attempt to burn the Flash has failed
0x52	Illegal Command	Illegal command for attached daughterboard

Controller Info (0x00)

The Controller Info command is diagnostic, and returns some basic operating parameters.

Structure

The Controller Info command (0x00) returns the following information:

Offset	Description
0x00	Generic Header
0x0C	Version
0x1C	Date
0x2C	CPU
0x30	Flash
0x34	VRAM
0x38	VCs
0x3C	Jumper Field

Figure 8-17. Controller Info Structure

Following is an example of the information returned by the Controller Info Command:

Information	Value
version[16]	Current firmware version (for example, `5215 V/ATM X06`)
date[16]	Date and time of the current firmware version (for example, `03/04/94 15:00`)
CPU type	0x00068040
Flash size	0x20000
VRAM size	0x400000
Number of VCs	0x800
Jumper Field	0x9fff

Fields

Generic Header

This field contains the command length, command code, and command status. See Figure 8-16 on page 123 for the command structure.

Version/Date

This field is a 32-byte ASCII string that contains the product identification, current version of the controller firmware, and the time/date stamp.

CPU

This field contains the CPU processor type.

Flash

This field contains the size of the flash, in bytes. This number is given in hexadecimal notation.

VRAM

This field contains the size of the VRAM, in bytes. This number is given in hexadecimal notation.

VCs

This field contains the maximum number of virtual circuits allowed. This number is given in hexadecimal notation.

Jumpers

The jumper field is shown below:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Daughterboard ID				JA1	JA2	JA3	JA4	JA5	JA6	JA7	JA8	JA9	JA10	JA11	JA12

31-19	18	17	16
Reserved	media ID	media ID	media ID

Figure 8-18. Jumper Field Returned by Controller Info Command

Bits 31-19: Reserved. Bits 31-19 are reserved.

Bits 18-16: When the daughterboard ID bits (bits 15-12) are 0101, bits 18-16 identify the type of media interface on the controller. The interface types are defined as follows:

Bit 18	Bit 17	Bit 16	Media Interface Type
0	0	0	Reserved
0	0	1	Reserved
0	0	1	Reserved
0	1	1	25.9 Mbps Desktop
1	0	0	155 Mbps UTP
1	0	1	25.6 Mbps UTP
1	1	0	155 Mbps SONET
1	1	1	100 Mbps TAXI

Bits 15-12: Daughterboard ID. Bits 15-12 identify the attached daughterboard and are encoded as follows:

Bit 15	Bit 14	Bit 13	Bit 12	Daughterboard
0	1	0	1	Embedded Motherboard
1	0	0	0	TAXI 100 Mbit OC1
1	0	0	1	SONET 155 Mbit OC3
-	-	-	-	All other combinations are reserved

Bits 11–0: Firmware Jumper settings. Bits 11–0 indicate the current settings of the firmware jumpers, and the function of each. The settings are as follows (the default setting for each is shown):

Bit	Jumper	0 = In	1 = Out	Default
11	JA1	POST disabled	POST enabled	out
10	JA2	XON/XOFF enabled	XON/XOFF disabled	out
9	JA3	reserved		out
8	JA4	reserved		out
7	JA5	reserved		out
6	JA6	9600 baud	38400 baud	out
5	JA7	reserved		out
4	JA8	CPU Cache disabled	CPU Cache enabled	out
3	JA9	reserved		out
2	JA10	PBUG enabled	PBUG disabled	out
1	JA11	UART enabled	UART disabled	out
0	JA12	GDB enabled	GDB disabled	out

Force Control Receive (0x01)

The Force Control Receive command (0x01) is a diagnostic command that returns the response in both the control send ring/buffer and the control receive ring/buffer. A control receive element must be available for this command to post the return in the receive ring.

Structure

Figure 8-19 shows the structure for the Force Control Receive Command Structure:

Offset	Description
0x00	Generic Header

Figure 8-19. Force Control Receive Command Structure

Fields

Generic Header

This field contains the command length, command code, and command status. See Figure 8-16 on page 123 for the command structure.

Configure Control Send/Receive Channels (0x10)

The Configure Control Send/Receive Channels command (0x10) is issued at boot time to configure both the control send and control received channels. Upon boot completion, all further commands are issued through the control send channel. Status/event indications are reported via the control receive channel.

Structure

Figure 8-20 shows the structure for the Configure Control Send/Receive Channels command:

Offset	Description
0x00	Generic Header
0x0C	Send Channel Configuration Block
0x28	Receive Channel Configuration Block

Figure 8-20. Configure Control Send/Receive Channel Command Structure

Fields

The following topics describe the fields in the Control Send/Receive Channel command.

Generic Header

This field contains the command length, command code, and command status. See Figure 8-16 on page 123 for the command structure.

Send Channel Configuration Block

This block contains configuration information for the control send channel. See Figure 8-31 on page 142 for the block structure.

Receive Channel Configuration Block

This block contains configuration information for the control receive channel. See Figure 8-31 on page 142 for the block structure.

NOTE

The RW bit in the DMAC control word is ignored for the control send channel.

Configure VATM Data Channels (0x11)

The Configure VATM Data Channels command (0x11) is used to configure the default data channels. The default data channels consist of two send channels and two receive channels.

NOTE

This command is archaic, and although still supported, it is recommended to use the Configure Data Channels command (0x12) instead.

Structure

Figure 8-21 shows the configuration structure for the Configure VATM Data Channels command:

Offset	Description
0x0	Generic Header
0xC	Fixed-Rate Send Channel Configuration Block
0x28	Variable-Rate Send Channel Configuration Block
0x34	Small-Buffer Receive Channel Configuration Block
0x40	Large-Buffer Receive Channel Configuration Block

Figure 8-21. VATM Data Channel Command Structure

Fields

The following topics describe the fields in the VATM Data Channel command.

Generic Header

This field contains the command length, command code, and command status. See Figure 8-16 on page 123 for the command structure.

Fixed-Rate Send Channel Configuration Block

This field contains information about the configuration of the fixed-rate send channel. See Figure 8-31 on page 142 for the block structure.

Variable-Rate Send Channel Configuration Block

This field contains information about the configuration of the variable-rate send channel. See Figure 8-31 on page 142 for the block structure.

Small-Buffer Receive Channel Configuration Block

This field contains information about the configuration of the small-buffer receive channel. See Figure 8-31 on page 142 for the block structure.

Large-Buffer Receive Channel Configuration Block

This field contains information about the configuration of the large-buffer receive channel. See Figure 8-31 on page 142 for the block structure.

Configure Data Channels (0x12)

The Configure Data Channels command (0x12) is used to configure the user specified data channels. Up to 8 send channels and 2 receive channels may be configured. A minimum of 1 each must be configured to transmit and receive packets via the ATM media.

The sole intent of the multiple send channels is to provide a logical grouping function for the host. For example, the host may want to associate a channel with a rate queue, which eliminates slower rate packet completions from blocking a higher rate completion. Or, for the same reason, group by packet size or priority.

The intent of the multiple receive channels is to help with buffer management. Each channel may define different buffer maximum transfer units (MTUs), one small (that is, 256 bytes) and one large (that is, 9180 bytes). The controller will return incoming packets to the appropriate channel based on the size of the packet. A small packet will be returned in the large ring if no small buffer elements are available.

Structure

Figure 8-22 shows the structure for the Configure Data Channels command:

Offset	Description
0x00	Generic Header
0x0C	Send Channel Count
0x10	Receive Channel Count
0x14–[offset of n]	Send Configure Block 1–n
[offset of Blocks 1–n]	Receive Configure Block 1–n

Figure 8-22. Configure Data Channels Command Structure

Fields

The following topics describe the fields in the Configure Data Channels command.

Generic Header

This field contains the command length, command code, and command status. See Figure 8-16 on page 123 for the command structure.

Send Channel Count

This field indicates the number of data send channels to configure. The maximum number is 8.

Receive Channel Count

This field indicates the number of data receive channels to configure. The maximum number is 2.

Send Configure Block 1–n

This field specifies the configuration blocks for the send channel. See Figure 8-31 on page 142 for the block structure. This field is repeated SEND COUNT times.

Receive Configure Block 1–n

This field specifies the configuration blocks for the receive channel. See Figure 8-31 on page 142 for the block structure. This field is repeated RECEIVE COUNT times. The last field is always a **receive configure** field.

Set VME Burst Count (0x13)

The Set VME Burst Count command (0x13) is used to set the burst count, that is, the number of transactions performed before the controller relinquishes the VMEbus. The burst count determines how long the controller will hold on to the VMEbus before releasing it. After completing a VME burst, the controller relinquishes the bus for use by other devices on the bus, and immediately requests the bus for another burst.

Structure

Figure 8-23 shows the structure for the Set VME Burst Count command:

Offset	Description
0x0	Generic Header
0xC	Burst Count

Figure 8-23. Set VME Burst Count Command Structure

Fields

The following topics describe the fields in the Set VME Burst Count command.

Generic Header

This field contains the command length, command code, and command status. See Figure 8-16 on page 123 for the command structure.

Burst Count

This field indicates the number of 8-bit, 16-bit, 32-bit, or 64-bit transactions to be performed before the controller relinquishes the VMEbus. The data size (8-bit, 16-bit, 32-bit, or 64-bit transfer mode) is determined by bits 5 and 6 of the DMAC Word (See *DMAC* on page 109 for the word definition).

Valid values for the burst count are 0 to 0x3FF, inclusive (0 = full packet transfers).

Configure Raw Receive Channel (0x14)

The Configure Raw Receive Channel command (0x14) is used to enable and specify the operating parameters for the Raw/OAM Receive Channel. The Raw and OAM type ATM cells are returned to the host as 64-byte data packets; the first 4 bytes are the cell header, followed by the 48-byte payload field, and padded out to 64 bytes with unknown data.

To receive Raw or OAM type ATM cells, a VC Table entry must be configured, specifying which VCI to accept and setting the packet type for raw or OAM cells.

Structure

Figure 8-24 shows the structure for the Configure Raw Receive Channel command:

Offset	Description
0x0	Generic Header
0xC	Channel Configuration Block

Figure 8-24. Configure Raw Receive Channel Command Structure

Fields

The following topics describe the fields in the Configure Raw Receive Channel command.

Generic Header

This field contains the command length, command code, and command status. See Figure 8-16 on page 123 for the command structure.

Channel Configuration Block

This field contains the Channel Configuration Block for this channel. See Figure 8-31 on page 142 for the block structure.

Configure/Report Virtual Circuit Table Index (0x20/0x21)

The Configure Virtual Circuit (VC) Table Index command is used to configure one or more consecutive virtual circuit table entries. A table entry defines and controls the transmission and reception of data packets (or raw/OAM cells) to and from a virtual connection.

The maximum number of available entries is a hardware build option and is made available in the Controller Info structure. This structure is available at power-up (appended to CBOK), through the Common Boot INFO type 2 command, and the HARI Controller Info command.

The host is responsible for the allocation and deallocation of table entries. The host can configure all table entries at initialization, or dynamically allocate and reuse on an as-needed basis. The table index does not have to match the VPI/VCI of the ATM cell header. These are specified in the ATM header bytes of the configuration block.

See Appendix B on page 147 for further information on Circuit Management.

Structure

Figure 8-25 shows the structure for the Configure Virtual Circuit (VC) Table Index command:

Offset	31.....24	23.....16	15.....8	7.....0
0x00	Index			
0x04	Count			
0x08	ATM Mode		0	
0x0C	hdr byte 0	hdr byte 1	hdr byte 2	hdr byte 3
0x10	mode		0	
0x14	0	Cell Quota	0	

Figure 8-25. Configure Virtual Circuit Table Index Command Structure

Fields

The following topics describe the fields in the Virtual Circuit (VC) Table Index.

Generic Header

This field contains the command length, command code, and command status. See Figure 8-16 on page 123 for the command structure.

Index

This field specifies the starting index number.

Count

This field specifies the number of sequential indexes. Fields 0x08– 0x14 are duplicated COUNT times, once for each index configured.

ATM Mode

This field contains the ATM MODE Word and specifies the packet type (AAL3/4, AAL5, OAM), CRC and EOM generation. The bits in the ATM Mode Control Word specify the packet segmentation to be performed.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTI Value			0	EM	CR	PCKTYP		0	0	0	0	0	0	0	0

Figure 8-26. ATM MODE Word Definition

Bits 15–13: PTI—Optional PTI field for OAM cell header. These bits are to be used as an optional source for the OAM cell header's PTI field when the packet type is 11. When the packet type is 10, the entire ATM cell header is derived from the VC Table.

Bit 12: Reserved. This bit is reserved and must be cleared to 0.

Bit 11: EOM—End of Message. When set to 1, this bit indicates that the segmentation hardware must set the EOM bit in the ATM header by forcing the PTI field to xx1 for EOM and SSM cells for AAL3/4 and AAL5 cells.

Bit 10: CRC32 Enable. When set to 1, this bit indicates that the segmentation hardware must generate a 32-bit frame check sequence (FCS) for the frame. Note that a bit can alternately be set in the VC table for similar effect. This bit provides FCS control on a per-frame basis where the bit in the VC table provides FCS control on a per-connection basis.

Bits 9–8: PCKTYP—Packet Type. These two bits indicate the segmentation type to be performed: AAL3/4, AAL5, or OAM (with the source of the PTI bits option):

Bit 9	Bit 8	Algorithm
0	0	AAL3/4 segmentation performed.
0	1	AAL5 segmentation performed.
1	0	An OAM cell is transmitted using PTI values from the VC table.
1	1	An OAM cell is transmitted using PTI values from bits 15–13.

Bits 7–0: Reserved. These bits are reserved and must be cleared to 0.

hdr byte 0 / hdr byte 1 / hdr byte 2 / hdr byte 3

The header (hdr) byte, hdr byte 0, hdr byte 1, hdr byte 2, hdr byte 3, compose the ATM cell headers of packets sent on the corresponding virtual circuit. The packet-to-cell segmentation hardware uses these header bytes as the ATM cell header, without modification (with the optional exception of the payload type field). These fields should be set to conform to ATM standards. The segmentation hardware transmits hdr0 followed by hdr1, hdr2, hdr3, and then computes and appends the header error checksum (HEC) on these bytes when the cell is transmitted.

Mode

The bits in this field specify the transmission rate, congestion control mode, and FCS generation for all packets sent to this table index. The bits are described below:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FCS	CC Mode			Rate Queue			Reserved								

Bit 15: FCS Enable. When this bit is set, a packet-level FCS (CRC-32) is generated and included for all AAL3/4 and AAL5 frames. This bit provides a per connection FCS control.

NOTE

This bit can alternately be set in the ring element's Mode word to provide FCS control on a per-packet basis.

Setting the bit here will cause FCS generation regardless of the state of the ATM Mode word's bit.

Bits 14–12: Congestion Control Mode. These 3 bits define the rate of cell recovery after the virtual circuit has been subjected to congestion control. Recovery takes place when the congestion level maintained by the segmentation hardware is changed to the next lower level as a result of virtual circuit congestion. These 3 bits define the number of cell transfers that must take place without encountering a congestion notification cell to cause recovery of the cell rate.

If the mode is set to flush packets (111) queued for transmission on this virtual circuit, the segmentation hardware will terminate the packets.

The recovery modes are as follows:

Bit 14	Bit 13	Bit 12	Recovery Action
0	0	0	Recovery after every one cell transfer
0	0	1	Recovery after every two cell transfers
0	1	0	Recovery after every four cell transfers
0	1	1	Recovery after every eight cell transfers

Bit 14	Bit 13	Bit 12	Recovery Action
1	0	0	Recovery after every sixteen cell transfers
1	0	1	Invalid
1	1	0	Invalid
1	1	1	Flush packets on this virtual circuit

Bits 11–9: Rate Queue. This 3-bit field links the virtual circuit to one of eight different rate queues. These queues are as follows:

Bit 11	Bit 10	Bit 9	Rate Queue
0	0	0	High priority bank (A) queue 0
0	0	1	High priority bank (A) queue 1
0	1	0	High priority bank (A) queue 2
0	1	1	High priority bank (A) queue 3
1	0	0	Low priority bank (B) queue 0
1	0	1	Low priority bank (B) queue 1
1	1	0	Low priority bank (B) queue 2
1	1	1	Low priority bank (B) queue 3

Bits 8–0: Reserved.

Cell Quota

This 6-bit field is used to determine the maximum number of credits that a VC can accumulate (cell quota) and therefore sets the limit on the maximum number of cells that can be burst at peak rate. It is defined as multiples of 32 cells.

Configure/Report Rate Queues (0x30/0x31)

The Configure Rate Queues command (0x30) is used to configure the eight available rate queues. These are the rate queues required in the Virtual Circuit Table entries. A rate queue must be initialized and enabled prior to any transmissions using the respective queue. Any or all queues may be used. Any or all may be initialized and enabled/disabled at any time. The INFO structure's jumper field, which specifies the attached daughterboard ID, can be used to determine the maximum transmission rate.

The Report Rate Queue command (0x31) will report the rates of the respective queue.

Structure

Figure 8-27 shows the structure for the Report Rate Queue command:

Offset	Description
0x00	Generic Header
0x0C	Queue A0 Control Word
0x10	Queue A1 Control Word
0x14	Queue A2 Control Word
0x18	Queue A3 Control Word
0x1C	Queue B0 Control Word
0x20	Queue B1 Control Word
0x24	Queue B2 Control Word
0x28	Queue B3 Control Word

Figure 8-27. Configure/Report Rate Queue Command Structure

Fields

The following topics describe the fields in the Configure/Report Rate Queue Command structure.

Generic Header

This field contains the command length, command code, and command status. See Figure 8-16 on page 123 for the command structure.

Queue A0–B3 Control Word

For the Configure command, any field cleared to 0 will leave the respective queue rate unchanged:

31 13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	Rsrvd	RQ_XOFF_En	RQ_Enable	Prescaler		RQ_Preload							

Figure 8-28. Rate Queue Control Word Structure

Bits 31–12: Reserved and must be cleared to 0.

Bit 11: RQ_XOFF_Enable.

0—Ignore XON input while processing this rate queue.
1—Rate queue is processed only if XON input is true.

Bit 10: RQ_Enable.

0—Disable servicing of the rate queue.
1—Enable servicing of the rate queue.

Bits 9–8: Prescaler.

00—Divide CLK by 4 and use as the rate counter clock.
01—Divide CLK by 16 and use as the rate counter clock.
10—Divide CLK by 64 and use as the rate counter clock.
11—Divide CLK by 256 and use as the rate counter clock.

Bits 7–0: RQ_Preload. This is the internal rate counter value (preload value). Values written to bits 0–7 determine the peak segmentation rate of frames linked to that particular rate queue. The value is determined by the following formula:

R = | (255–(424/Peak_rate) x (1000/Tclk) x (1/Prescaler)) |

where:

- R is the integral *preload* value programmed into the queue rate register
- Peak_rate is the peak segmentation rate desired in Mbps
- Tclk is the segmentation hardware's CLK period in nanoseconds (50ns)
- Prescaler is one of 4, 16, 64, or 256

The following table shows example rate queue preload values, based on the prescaler, for some of the possible transmission rates (the segmentation hardware has a TCLK period of 50ns):

Peak Rate	Program value			
	Prescaler /4	Prescaler /16	Prescaler /64	Prescaler /256
1 Mbps	Invalid	Invalid	122	221
5 Mbps	Invalid	149	228	248
10 Mbps	43	202	241	251
20 Mbps	149	228	248	Invalid
45 Mbps	207	243	252	Invalid
50 Mbps	212	244	Invalid	Invalid
100 Mbps	233	249	Invalid	Invalid
155 Mbps	241	251	Invalid	Invalid

Write/Read Flash Sector (0x40/0x41)

The controller provides a 256-byte block of FLASH EPROM storage to the host. The Write Sector Command (0x40) is used to burn data into flash, and the Read Sector command (0x41) is used to read the flash sector.

Structure

Figure 8-29 shows the structure for the Write/Read Flash Sector command:

Offset	Description
0x0	Generic Header
0xC	Data

Figure 8-29. Write/Read Flash Sector Command Structure

Fields

The following topics describe the fields in the Write/Read Flash Sector.

Generic Header

This field contains the command length, command code, and command status. See Figure 8-16 on page 123 for the command structure.

Data

This field contains the Flash Sector data.

NOTE

The flash PROM can be written to a maximum of 10,000 times.

Configure/Report Congestion Control (0x60/0x61)

The Configure Congestion Control command (0x60) is used to enable/disable and initialize any or all of the three methods of congestion control provided. Setting any of the flag bits to 1 will enable the respective congestion recognition method.

The Report Congestion Control Configuration command (0x61) is used to interrogate the current configuration.

See Appendix A for further information on Congestion Recognition and Notification.

Structure

Figure 8-30 shows the structure for the Configure Congestion Control command:

Offset	31.....24	23.....16	15.....8	7.....0
0x00	Generic Header			
0x0C	Flags			
0x10	rsvd 0	rsvd 1	oam_mask	oam_compare

Figure 8-30. Configure Congestion Control Structure

Fields

The following topics describe the fields in the Configure Congestion Control command.

Generic Header

This field contains the command length, command code, and command status. See Figure 8-16 on page 123 for the command structure.

Flags

Bits 31–4: Reserved. Bits 31–4 are reserved and should be cleared to 0.

Bit 3: CRC. Bit 3, when set to 1, enables the CRC generation/checking for the OAM cells.

Bit 2: OAM. Bit 2, when set to 1, will enable the reassembly hardware to recognize a definable OAM cell as a congestion notification from the VPI/VCI, and back off the transmission rate for that VCI. The congestion recovery method in the respective VC Table entry specifies when to increase the transmission rate.

The reassembly hardware uses the cell header PTI field and the first byte of the cell payload to recognize congestion notification. The payload byte to use is defined by the host using the oam_compare and oam_mask fields (as defined below).

If the Raw Receive Channel is active and an element is available, this OAM cell will be returned to the host.

Bit 1: GFC. Bit 1, when set to 1, will enable the reassembly hardware to recognize the cell header GFC bit as a congestion notification from the VPI/VCI, and back off the transmission rate for that VCI. The congestion recovery method in the respective VC Table entry specifies how to recover the transmission rate.

The status bit CNG in the receive ring element may be set.

Bit 0: XON/XOFF. Bit 0, when set to 1, enables a vendor-specific symbol for XOFF to disable all transmission of all cells until receipt of the XON symbol.

OAM_MASK

These bits, if set to 1, enable the corresponding bits of the payload byte to be compared to the oam_compare bits. If cleared to 0, the bits are don't care.

OAM_COMPARE

These bits define the value to compare the first payload byte for congestion recognition, and is used in conjunction with the oam_mask byte. For example, if the most significant three bits must equal a binary 101 and the least significant five bits are don't care, the oam_mask byte would be a binary 11100000 and the oam_compare byte would be a binary 101xxxxx (where x = don't care).

Channel Configuration Block (CCB)

This structure is used to initialize the operating parameters for all channel types and must be provided for each channel opened.

This initialization requires the host to perform two resource allocations:

- Where, in Short I/O, to locate the channel's SCB
- Where, in host memory, to locate the channel's ring

Structure

Figure 8-31 shows the structure for the Channel Configuration Block:

Offset	31.....24	23.....16	15.....8	7.....0
0x00	SCB Pointer			
0x04	Ring VME Address			
0x08	Ring Length			
0x0C	Reserved		Ring DMAC	
0x10	Data MTU			
0x14	Reserved			
0x18	ILVL	IVCTR	Throttle	Flags

Figure 8-31. Channel Configuration Block (CCB) Structure

Fields

The following topics describe the fields in the Channel Configuration Block.

SCB Pointer

This field specifies the Short I/O Control Block (SCB) offset into Short I/O. The controller will initialize the SCB fields upon acceptance of this structure. This field is masked by the firmware with the Short I/O size, so the actual physical address may be supplied here if it accurately reflects the offset into Short I/O.

Ring VME Address

This field specifies the physical starting VME address of the ring.

Ring Length

This field specifies the length of the ring, in bytes. If this field is cleared to 0, the channel is disabled.

Reserved

This field is reserved.

Ring DMAC

This field specifies the DMA Control Word for the ring. See *DMAC* on page 109 for the word definition.

Data MTU

This field specifies the maximum transfer unit (MTU) for data.

Reserved

This field is reserved.

ILVL

This field specifies the VMEbus interrupt level to assert for this channel.

IVCTR

This field specifies the interrupt vector to present for this channel during the interrupt acknowledge.

Throttle

This field specifies the maximum number of ring elements to service at any one time. This aids in balancing the work load between channels, preventing a channel from starving off others. Valid entries are from 0 to 0xff, 0 specifying all pending ring elements.

Flags

This field contains the following channel configuration block flag bits:

7	6	5	4	3	2	1	0
0	0	0	0	0	RSE	TCM	RSP

Figure 8-32. Channel Configuration Block Flag Bits

Bits 7–3: Reserved. Bits 7–3 are reserved for future use and must be cleared to 0.

Bit 2: RSE—Receive Scatter Enable. Bit 2, when set to 1, enables the *scattering* of a receive packet into multiple host buffers. Each receive packet will consume consecutive ring elements until the entire packet has been DMA'd into host memory. The ring element's EP bit set to 1 designates the end of packet.

This bit is only applicable to data receive channels.

NOTE

This option can ONLY be used with ONE receive channel active/initialized.

Bit 1: TCM—Transmit Completion Mode. Bit 1 defines when the controller reports a data send channel ring element as complete. When cleared to 0 (TCM0), the element is reported as completed after the DMA transfer of the host data buffer to the controller memory and the host buffer is free for reuse. When set to 1 (TCM1), the element is not reported as completed until the entire packet has been transmitted out to the media.

NOTE

This bit is only applicable to data send channels.

Bit 0: RSP—Ring Response Enable. Bit 0,when set to 1, specifies that the ring elements are to be DMA transferred back into system memory upon completion. This is required for all receive channels, and channels for which the host must be informed of completion status. However, for transmit channels using TCM0, the ring completion status is of little significance, and disabling this extra DMA transfer can increase total system throughput. When cleared to 0, the response transfers are disabled.

Adapter Specifications

A

Operating Environment

Temperature	0–55° C / 32–131° F
Relative humidity	10–95% noncondensing
Altitude	0–15,000 feet

Storage Environment

This section assumes the adapter is stored in its original anti-static bag and box.

Temperature	-20–85° C / -4–185° F
Relative humidity	LAN and storage adapters: 10–95% noncondensing WAN adapters: 5–95% noncondensing
Altitude	0–50,000 feet

Boot Initialization B

Overview

This appendix describes how the Common Boot interface functions to initialize the controller to use the HARI interface. It also describes how HARI is initialized with the desired operating parameters.

After completing the power-up reset, the 5215 enters the Common Boot (CB) host interface mode. The Common Boot interface is generic across multiple controllers and provides mechanisms for diagnostic testing, downloading and burning the FLASH, and booting the controller specific interface, which in this case, is the Host Adapter Ring Interface (HARI). See ***The Common Boot Interface*** on page 53 for details about the Common Boot interface.

The host must perform the following steps to boot the HARI:

1. Reset the Controller.

   To reset the 5215, simply toggle bit 15 of the Hardware Control field from 0 to 1 and then back to 0. The Hardware Control field is a 16-bit field located at the start of short I/O (that is, bytes 0 and 1 of the 512-byte short I/O space). When the 5215 is reset, it goes through its bootup sequence. The reset bit must hold the logic *1* for a minimum of 1 µsec.

NOTE

After toggling the reset bit (bit 15), the host should wait at least 500 milliseconds before looking for CB_HERALD. Or the user may write a value in the command/status location (normally zero or 0xDEAD) and reset the controller. The zero or 0xDEAD value will be replaced by CBOK or FAIL after power-up.

2. Locate the Common Boot Command Pointer.

   After the CB interface successfully starts up, it fills the 512-byte short I/O with a 32-bit value called CB_HERALD. The host must locate the CB_START pointer (command pointer) prior to issuing CB commands. The second byte of the herald specifies the longword (32 bit) offset of the command block. CBOK must be present at this location for CB to accept further commands.

   In a typical CB_HERALD (0xCB200200), the byte (0xCB) is strictly an identifier code, the next byte (byte 2, shown in Figure 6-3 on page 55) specifies the longword offset of CB_START (longword 0x20 = byte offset 0x80), and the last 16 bits indicate the size of Short I/O (0x200 bytes).

3. Allocate System Resources.

   At boot time the host must configure the channels for both Control Send and Control Receive. This requires the host to allocate VME system memory for both the send and receive rings, and the short I/O space for the each ring's short I/O control block (SCB). The configuration of each ring is left to the discretion of the host. Typically, each ring configuration is 16 elements with data maximum transfer units (MTUs) of 256 bytes.

4. BOOT 0.

   The host must configure both control channels using the Configure Control Send/Receive Channels command block as the parameters for the CB BOOT 0 command. BOOT should be written into the CB command field *after* writing the configuration parameters.

5. Wait for CBOK.

   The host must poll for BOOT completion. While the host is polling for BOOT completion, no interrupt is generated. Successful completion is indicated by CBOK, otherwise FAIL is returned, which generally indicates bad configuration parameters.

   When CBOK is returned, the Short I/O Control Blocks (SCB) have been initialized by the controller the timer = 0, host pointer = ctlr pointer = start of ring.

   At this point, HARI replaces Common Boot as the host interface; however, only the control channels are active. Further configuration is required to define the controllers operating environment and to activate the data channels. This is done via the Control Send Channel.

Initialization

Prior to sending or receiving ATM frames, HARI must be initialized with the desired operating characteristics. All further maintenance and configuration commands are now issued via the Control Send channel. All commands, with the exception of channel configurations, are dynamic in nature and may be issued at the host's discretion. Refer to Chapter 6 for further details on command specifics.

1. Allocate System Resources.

   The host must allocate system resources for each Send and Receive channel desired. HARI allows for as many as eight transmit channels, two receive channels, and one raw/OAM cell receive channel; however, a minimum of one send and one receive channel is required for proper operation. Multiple send channels are provided strictly for host convenience and in no way affect the transmission of ATM packets.

NOTE

The host directly controls the onboard buffer allocations. For each ring element of each data channel, the 5215 attempts to allocate 1 onboard VRAM buffer, the size of each buffer being the respective channel's MTU (modulo 2K bytes). Allocation errors are fatal and require a controller reset and re-initialization. Both CB and the Controller Info Command report the total amount of VRAM available.

For host systems that can only perform D16 transfers to controller Short I/O space, rings must be allocated to NOT to cross a 64K page boundary. This prevents having to debounce the SCB's *host* and *controller* pointers (only the least significant 16 bits change).

2. Configure Data Channels.

   The host may now issue the Configure Data Channels command to initialize and activate the desired data channels. Upon successful completion, the controller will have initialized all the specified Short I/O Control Blocks (SCBs), and is nearly ready for the transmission and reception of ATM packets.

3. Configure Rate Queues.

   The 5215 provides eight programmable transmission rate queues, of which one or more may be used. The transmission rate of any circuit (connection) is specified by the rate queue field of the corresponding Virtual Circuit Table entry. (See Step 5).

4. Set VME Burst Count.

   The Set VME Burst Count is a throttling mechanism provided for VMEbus usage. The Set VME Burst Count specifies the maximum number of VME transactions per bus request/grant.

5. Configure Virtual Circuit Table Index(es).

   The 5215 allows for a maximum of 2048 active circuits. Each circuit is defined and controlled by an entry in the Virtual Circuit Table. The host must configure a table entry for each circuit requiring either transmission or reception. The host is solely responsible for the allocation, configuration, and maintenance of each table entry. Table entries may be used and re-used at the host's discretion.

Connection Management

Prior to actual transmission or reception of packets, the host must define the operating characteristics of each circuit (connection). ATM allows for multiple active circuits, each with an individual packet type, transmission rate, and other characteristics.

The host must provide the following parameters (using the Configure VC Table Index command) prior to sending and receiving packets on a virtual circuit:

- Packet type (AAL3/4, AAL5, raw, OAM)
- The 4 byte ATM cell header
- FCS (CRC-32) enable/disable
- Transmission Rate
- Congestion Recovery Mode

Table entries may be allocated, configured, and deallocated on an as-needed basis. If the environment is known, all table entries may be configured once at initialization, with no further maintenance required.

NOTE

Unless otherwise noted, when a Virtual Circuit Index is mentioned in this manual, it refers the Virtual Circuit Table Index number, and NOT the VCI contained within the ATM cell header.

Transmits

The 5215 provides the segmentation of data packets into the appropriate ATM cells, and multiplexes them onto the media with all active packets. The host needs only to ensure that all packets are less than or equal to the respective send channel's MTU, which is specified at configuration.

To initiate a transmit, the host selects which channel to use and allocates the next available ring element (the SCB's *host* field). After providing the appropriate buffer description in the ring element, the host simply increments the SCB's *host* field to the next ring element (honor ring wrap). Upon completion, the controller will increment the SCB's controller pointer, and, if enabled, will generate an interrupt to the host.

Completion Modes

The 5215 provides two modes (TCM0 and TCM1) for reporting transmit completions:

- **TCM0**—reports completion when the data has been successfully DMA'd onto the board and the host buffer is free for reuse. This mode returns ring elements in the same order as issued, and may be used with ring responses (RSP) disabled. Any transmission failures will not be reported in this mode.
- **TCM1**—reports completion when the data has been transmitted on the media. Due to possible different transmission rates, the ring elements may be returned *out-of-order* and the ring responses (RSP) must be enabled.

The desired mode is specified at channel configuration time. All elements within the ring report completions in the same mode.

Gathers

The 5215 packs two or more discontiguous data blocks into a packet. Gather operations are provided using the End-of-Packet (EP) bit in the ring control word. When cleared to 0, this element is to be combined with all subsequent elements with EP = 0, into one contiguous data packet until an element with the EP bit set to 1 is encountered.

NOTE

Because these blocks are packed into 1 onboard buffer, the combined byte counts of all blocks must be less than or equal to the channel's maximum transfer unit (MTU).

The Short I/O Control Block's *host* pointer must never define (point to) a ring element without the EXEC Word's EP bit set to 1.

Receives

The 5215 provides the reassembly of ATM cells into data packets. The host needs only to ensure that the respective receive channel's MTU is sufficient to accommodate the incoming packets. The data receive mechanism differs from the data transmit mechanism. For receives the host must *anticipate* incoming data. Host receive buffers must be provided before any data can be received.

In addition to providing data buffers, the host must configure entries in the Virtual Circuit Table, specifying which VCIs to listen for and what type of packets to expect.

To *anticipate* a receive, the host allocates the next available receive ring element (the SCB's host field). After providing the appropriate buffer description in the ring element, the host simply bumps the SCB's *host* field to the next ring element (honor ring wrap). Upon

reception of a data packet, the controller will transfer the packet into the specified host buffer, update the ring element's EXEC, length, VCI, and status fields, bump the SCB's controller pointer, and, if enabled, will generate an interrupt to the host.

Small/Large Buffers

Due to the MTU of ATM packets (9180 bytes), the 5215 provides a mechanism to assist in host buffer requirements. The host can initialize two receive channels, specifying a different MTU for each (that is, 256 and 9180 bytes). The controller will return reassembled packets, based on size, to the appropriate channel.

NOTE

The 5215 must internally allocate the larger MTU size buffers for both rings (each buffer modulo 2K) for packet reassembly. Because the host has indirect control of onboard resource allocations, this must be taken into consideration when specifying the number of elements (ring length) in each ring.

Small packets (those less than the small MTU) will be returned in the large receive channel if no small buffers are available but large buffers are.

Scatters

For systems that cannot provide the large buffers required by ATM, the 5215 supports *scattering* of packets into multiple host buffers. The controller will consume as many consecutive ring elements as required to hold the data packet. All ring elements will have the EXEC word's EP bit cleared to 0 with the exception of the last buffer, which will have the EP bit set to 1.

The controller will hold the packet until sufficient host buffers are available. Ring status and/or host interrupts are not updated/generated until the entire packet has been transferred to the host system.

NOTE

This option requires special initialization (see Channel Configuration Block option flag RSE) and can ONLY be used when ONE data receive channel is initialized.

Statistics

The Host Statistics Block contains current operational statistics for the controller. This block is organized into three basic blocks:

- Individual channel statistics
- Controller interrupts
- Receive exceptions

Channel Statistics

The first group contains counters for each possible channel (up to a maximum of 13). A channel block is assigned as channels are initialized by the host, that is, blocks 0 and 1 are assigned to the Control Send and Control Receive channels, respectively, at bootup. As data/raw channels are configured, statistic blocks are allocated accordingly.

Unused channel slots do not affect the other two group locations in Short I/O.

OffSet	Field Description
0x0	Number of elements issued by host
0x4	Number of elements returned by the controller
0xc	Number of interrupts generated to the host

Controller Interrupts

The second group indicates the total number of VME interrupts (VMEbus IRQ) generated by the controller and acknowledgments (VMEbus IACKs) received from the host.

Offset	Field Description
0x0	Interrupts generated by the controller
0x4	Interrupt acknowledges received from the host

Receive Exceptions

The third group provides receive exceptions encountered during packet reassembly. These exceptions are not reported to the host via the receive or control channels, and do not consume host data buffers.

Offset	Exception Description
0x00	Receive Exception—no errors
0x04	Receive Exception—out of sequence COM cell received
0x08	Receive Exception—out of sequence EOM cell received
0x0c	Receive Exception—reserved
0x10	Receive Exception—No buffers available (small packet dropped)

Offset	Exception Description
0x14	Receive Exception—No buffers available (large packet dropped)
0x18	Receive Exception—Invalid VCI (cell received on invalid VC)
0x1c	Receive Exception—Invalid VPI (cell received on invalid VP)

Host Statistics Block Layout

The Host Statistics Block (HSB) resides in the upper most locations of controller Short I/O. The three groups of statistics described above, are allocated from upper memory backwards, starting with the receive exceptions, the interrupt counters, and then the channel statistics as follows (shown for 512 byte Short I/O):

Offset	Statistics Description
0x13c	Control Send—elements issued/returned/interrupts
0x148	Control Rcv—elements issued/returned/interrupts
0x154	Data Channel 0—elements issued/returned/interrupts
0x160	Data Channel 1—elements issued/returned/interrupts
0x16c	Data Channel 2—elements issued/returned/interrupts
0x178	Data Channel 3—elements issued/returned/interrupts
0x184	Data Channel 4—elements issued/returned/interrupts
0x190	Data Channel 5—elements issued/returned/interrupts
0x19c	Data Channel 6—elements issued/returned/interrupts
0x1a8	Data Channel 7—elements issued/returned/interrupts
0x1b4	Data Channel 8—elements issued/returned/interrupts
0x1c0	Data Channel 9—elements issued/returned/interrupts
0x1cc	Data Channel 10— elements issued/returned/interrupts
0x1d8	Total Interrupt requests
0x1dc	Total Interrupt acknowledges
0x1e0	Receive Exception—no errors
0x1e4	Receive Exception—out of sequence COM cell received
0x1e8	Receive Exception—out of sequence EOM cell received
0x1ec	Receive Exception—reserved
0x1f0	Receive Exception—No buffers available (small packet dropped)
0x1f4	Receive Exception—No buffers available (large packet dropped)
0x1f8	Receive Exception—Invalid VCI (cell received on invalid VC)
0x1fc	Receive Exception—Invalid VPI (cell received on invalid VP)

Receive Congestion Recognition Methods

The reassembly (receive) hardware is capable of recognizing and reacting to any or all of three possible congestion *notification* methods:

- **XON/XOFF**

  This *symbol* recognition is vendor specific. When XOFF is received, ALL transmissions are ceased until a subsequent XON is received.

  No notification of the XOFF is presented to the firmware or passed back to the host.

- **BECN** (GFC bit 7)

  This is recognized when a cell header with the most significant bit of the first byte (MSB of the GFC field) is set. When such a cell is received, the reassembly hardware will request the segmentation hardware to *back off* transmissions on the congested VC only. The recovery algorithm is specified in the VC table entry (use the Configure VC Table Index command).

  The status bit CNG in the receive ring element may be set.

- **OAM F5 Congestion Cell**

  The host may *define* how the first byte of an OAM F5 cell is to be recognized. When such a cell is received, the reassembly hardware will request the segmentation hardware to *back off* transmissions on the congested VC only. The recovery algorithm is specified in the VC Table entry (use the Configure VC Table Index command).

  If the Raw receive channel has been configured and a ring element is available, this cell will be returned to the host.

These three methods are mutually exclusive, and are enabled by setting their respective flag bits (bits 3-0).

Transmit Congestion Notification

The segmentation (transmit) hardware is capable of sending either BECN or OAM F5 cells, as well as any other type of raw cell, as follows:

1. Allocate and configure a VC Table entry, setting the 4-byte header, mode bits, etc. to conform to approved ATM standards. (See the Configure VC Table Index command).
2. Allocate a Data Send ring element and set:
   - The VCT Index (configured in step 1).
   - The ATM Mode word with packet type bits as either 10 or 11 (if 11, supply PTI bits This should match the ATM Mode word used in configuring the VCT entry in step 1).
   - The buffer length to 48 bytes (0x30).
3. Build the cell payload in the data buffer specified in the ring element.
4. Send the packet (release the ring element to the controller).

Segmentation Rate Queues

The segmentation hardware contains eight rate queues. Each rate queue is programmed for the peak rate at which the packets on that queue are segmented. The rate queues are organized in two banks—a high priority bank and a low priority bank, as shown in the following table:

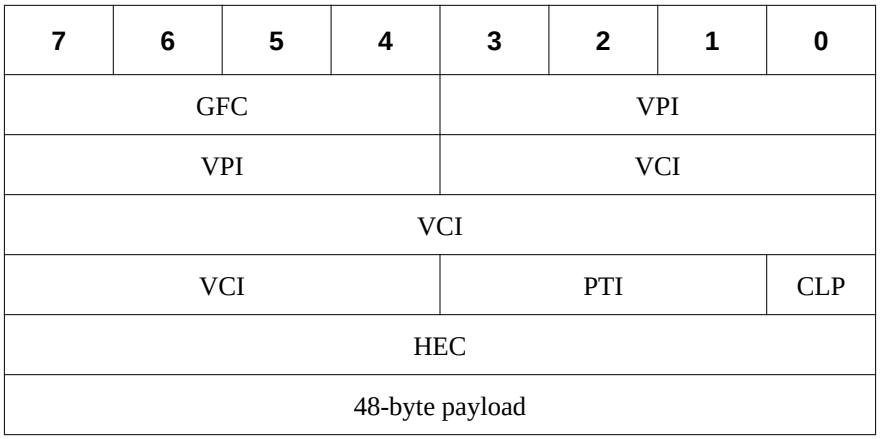
High Priority Bank Rate Queue Registers	Low Priority Bank Rate Queue Registers
RQ_REG_A0	RQ_REG_B0
RQ_REG_A1	RQ_REG_B1
RQ_REG_A2	RQ_REG_B2
RQ_REG_A3	RQ_REG_B3

Requests are serviced in a round-robin fashion within the high priority bank before servicing any requests within the low priority bank.

Each rate queue has an internal 8-bit rate counter. When this rate counter overflows, it generates a queue service request for its queue. When a rate queue is being serviced, one cell from each actively linked packet is transferred to the cell interface before another rate queue service request is processed. Continuous service requests from queues in the high priority rate queue will keep service requests by low priority rate queues from being serviced.

ATM Cell

For reference, the ATM cell format is as follows:



GFC - Generic Flow Control
VPI - Virtual Path Identifier
VCI - Virtual Circuit Identifier
PTI - Payload Type Identifier
CLP - Cell Loss Priority
HEC - Header Error Check

Figure B-1. ATM Cell Format

The PTI bit encoding is as follows:

3	2	1	Interpretation
0	0	0	User data cell, congestion not experienced, SDU-type = 0
0	0	1	User data cell, congestion not experienced, SDU-type = 1
0	1	0	User data cell, congestion experienced, SDU-type = 0
0	1	1	User data cell, congestion experienced, SDU-type = 1
1	0	0	Segment OAM F5 flow related cell
1	0	1	End-to-end OAM F5 flow related cell
1	1	0	Reserved for further traffic control & resource management
1	1	1	Reserved for future functions

Common Boot Error Codes C

Overview

This appendix lists the various Common Boot (CB) Error Status Codes. Error codes are shown for CB Error Status Codes I, CB Error Status Codes II, memory test power-up codes, CPU test power-up codes, EPROM test power-up codes, hardware critical power-up codes, and illegal interrupt power-up codes.

Common Boot Error Status Codes I

The following table lists the CB error status codes:

Codes	Descriptions
01	Bad S-record type
02	Image checksum error
03	S-record checksum error
04	Download buffer overrun
05	Image overrun
06	Bad S0 S-record
07	Image header checksum error
08	Bad length
09	Not a valid S-record
0A	Bad compare
0B	Bad Size—Bad UNIT size
0C	Bad INFO type
0D	Boot 0 attempted
0E	Invalid BOOT type
0F	Major test unknown
10	Minor test unknown
11	RAM type unknown
12	Memory test failure
13	Unknown option
14	Illegal return
15	Minimum execution RAM not found

Common Boot Error Status Codes II

The following table lists CB Error Codes generated during the DNLD, EXEC and BURN commands:

Codes	Descriptions
16	Bad MAGIC NUMBER in image header
17	Error in image checksum
18	Image header length error
19	Overlap in text, data or BSS addresses
1A	Text address bad
1B	Data address bad
1C	BSS address bad
1D	Stack address bad
1E	Entry address error
1F	No text section
20	Regions overlap
21	Regions lengths do not sum
22	Image checksum is bad
23	Sequence error
24	Header overrun
25	Burn algorithm field
26	Burn did not verify
27	Burn image is too large
28	Invalid offset for burn
29	No EXEC 0 image
2A	Invalid type on INFO command
2B	No BOOT 0 image available
2C	Execution type unrecognized
2D	Bad copy of execute image
2E	DIAG command failed

Common Boot Fail Status Block Format

Following is the format for a CB FAIL Status Block for the power-up/reset test sequence:

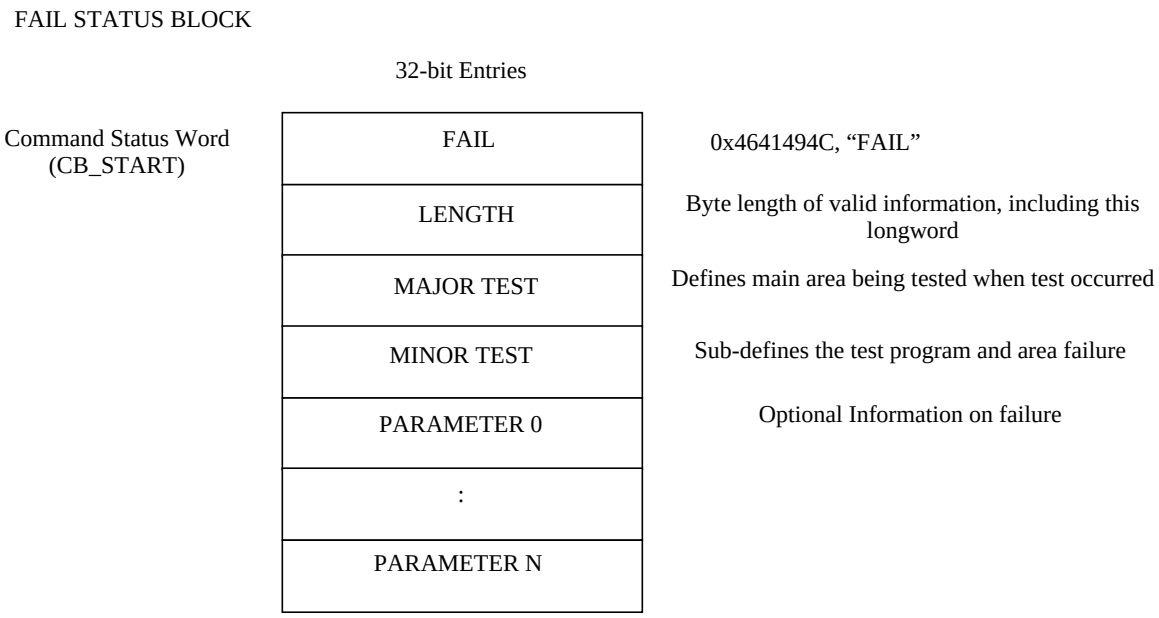


Figure C-1. CB Fail Status Block

This status block allows the host to examine the cause of the power-up/reset.

NOTE

The TEST command uses the same error codes as the MEMORY TEST section described below.

Memory Test Power-up Codes

MEMORY MAJOR TEST NUMBER	
MEMORY_TEST	0x00000001
MEMORY MINOR TEST NUMBER	
PROGRAM_RAM	0x00000001
STATIC_RAM	0x00000002
BUFFER_RAM	0x00000003
TYPES OF MEMORY TESTS (goes into error code field)	
EXIST_TEST	0x00000001
ADDRESS_FAULTS_TEST	0x00000002
MARCH_TEST	0x00000003
SLIDER_TEST	0x00000004
BLOCK_TEST	0x00000005
ADDR_PTR_TEST	0x00000006
SIZING_TEST	0x00000007

CPU Test Power-up Codes

CPU MAJOR TEST NUMBERS	
CPU_TEST	0x00000002
CPU MINOR TEST NUMBERS	
REG_TEST	0x000000001

EPROM Test Power-up Codes

EPROM MAJOR TEST NUMBERS	
EPROM_TEST	0x00000003
EPROM MINOR TEST NUMBERS	
CHECK_SUM	(0x01)

Hardware Critical Power-up Codes

HARDWARE CRITICAL MAJOR TEST NUMBERS	
HW_CRITICAL	0x00000004
HARDWARE CRITICAL MINOR TEST NUMBERS	
SHADOW	0x00000001

Illegal Interrupt Power-up Codes

ILLEGAL INTERRUPT MAJOR TEST NUMBER	
ILLEGAL_INTERRUPT	0x00000005
ILLEGAL INTERRUPT MINOR TEST NUMBERS	
STDBUS_ERROR	2 BUS ERROR
STD_ADDRESS_ERROR	3 ADDRESS ERROR
STD_ILLEGAL_INSTRUCTIONS	4 ILLEGAL INSTRUCTION
STD_ZERO_DIVIDE	6 CHK AND CHK2 INSTRUCTIONS
STD_CHK_CHK2	7 TRAPV INSTRUCTION
STD_PRIVILEGE	8 PRIVILEGE VIOLATION
STD_TRACE	9 TRACE
STD_EMUL_A	10 EMULATE A
STD_EMUL_B	11 EMULATE B
STD_HDW_BREAKPOINT	12 HARDWARE BREAKPOINT
STD_COPROCESSOR_VIOLATE	13 COPROCESSOR VIOLATION

STD_FORMAT_ERROR	14 FORMAT ERROR
STD_UNINITIALIZED_INT	15 UNINITIALIZED INTERRUPT
STD_RESERVED	16 RESERVED INTERRUPT 16-23, 59-63
STD_SPURIOUS	24 SPURIOUS INTERRUPT
STD_LEVEL_1_AUTO	25 LEVEL 1 AUTOVECTOR
STD_LEVEL_2_AUTO	26 LEVEL 2 AUTOVECTOR
STD_LEVEL_3_AUTO	27 LEVEL 3 AUTOVECTOR
STD_LEVEL_4_AUTO	28 LEVEL 4 AUTOVECTOR
STD_LEVEL_5_AUTO	29 LEVEL 5 AUTOVECTOR
STD_LEVEL_6_AUTO	30 LEVEL 6 AUTOVECTOR
STD_LEVEL_7_AUTO	31 LEVEL 7 AUTOVECTOR
STD_TRAP_0	32 TRAP 0
STD_TRAP_1	33 TRAP 1
STD_TRAP_2	34 TRAP 2
STD_TRAP_3	35 TRAP 3
STD_TRAP_4	36 TRAP 4
STD_TRAP_5	37 TRAP 5
STD_TRAP_6	38 TRAP 6
STD_TRAP_7	39 TRAP 7
STD_TRAP_8	40 TRAP 8
STD_TRAP_9	41 TRAP 9
STD_TRAP_10	42 TRAP 10
STD_TRAP_11	43 TRAP 11
STD_TRAP_12	44 TRAP 12
STD_TRAP_13	45 TRAP 13
STD_TRAP_14	46 TRAP 14
STD_TRAP_15	47 TRAP 15
STD_COPR_RESERVED	48 COPROCESSOR RESERVED 48-58
/* See STD_RESERVED for 59-63 */	
STD_USER	64 USER INTERRUPTS 64-255

Base Address Jumper Settings

D

Overview

Jumper fields 24 through 30 are used to set the base address of the 5215's short I/0 space. The following table shows the jumper settings for all possible base addresses. To locate the jumper fields, refer to the 5215 board diagram in Figure 2-1 on page 8.

Address	JA24	JA25	JA26	JA27	JA28	JA29	JA30
0x0000	In	In	In	In	In	In	In
0x0200	In	In	In	In	In	In	Out
0x0400	In	In	In	In	In	Out	In
0x0600	In	In	In	In	In	Out	Out
0x0800	In	In	In	In	Out	In	In
0x0A00	In	In	In	In	Out	In	Out
0x0C00	In	In	In	In	Out	Out	In
0x0E00	In	In	In	In	Out	Out	Out
0x1000	In	In	In	Out	In	In	In
0x1200	In	In	In	Out	In	In	Out
0x1400	In	In	In	Out	In	Out	In
0x1600	In	In	In	Out	In	Out	Out
0x1800	In	In	In	Out	Out	In	In
0x1A00	In	In	In	Out	Out	In	Out
0x1C00	In	In	In	Out	Out	Out	In
0x1E00	In	In	In	Out	Out	Out	Out
0x2000	In	In	Out	In	In	In	In
0x2200	In	In	Out	In	In	In	Out
0x2400	In	In	Out	In	In	Out	In
0x2600	In	In	Out	In	In	Out	Out
0x2800	In	In	Out	In	Out	In	In
0x2A00	In	In	Out	In	Out	In	Out
0x2C00	In	In	Out	In	Out	Out	In
0x2E00	In	In	Out	In	Out	Out	Out
0x3000	In	In	Out	Out	In	In	In
0x3200	In	In	Out	Out	In	In	Out
0x3400	In	In	Out	Out	In	Out	In

Address	JA24	JA25	JA26	JA27	JA28	JA29	JA30
0x3600	In	In	Out	Out	In	Out	Out
0x3800	In	In	Out	Out	Out	In	In
0x3A00	In	In	Out	Out	Out	In	Out
0x3C00	In	In	Out	Out	Out	Out	In
0x3E00	In	In	Out	Out	Out	Out	Out
0x4000	In	Out	In	In	In	In	In
0x4200	In	Out	In	In	In	In	Out
0x4400	In	Out	In	In	In	Out	In
0x4600	In	Out	In	In	In	Out	Out
0x4800	In	Out	In	In	Out	In	In
0x4A00	In	Out	In	In	Out	In	Out
0x4C00	In	Out	In	In	Out	Out	In
0x4E00	In	Out	In	In	Out	Out	Out
0x5000	In	Out	In	Out	In	In	In
0x5200	In	Out	In	Out	In	In	Out
0x5400	In	Out	In	Out	In	Out	In
0x5600	In	Out	In	Out	In	Out	Out
0x5800	In	Out	In	Out	Out	In	In
0x5A00	In	Out	In	Out	Out	In	Out
0x5C00	In	Out	In	Out	Out	Out	In
0x5E00	In	Out	In	Out	Out	Out	Out
0x6000	In	Out	Out	In	In	In	In
0x6200	In	Out	Out	In	In	In	Out
0x6400	In	Out	Out	In	In	Out	In
0x6600	In	Out	Out	In	In	Out	Out
0x6800	In	Out	Out	In	Out	In	In
0x6A00	In	Out	Out	In	Out	In	Out
0x6C00	In	Out	Out	In	Out	Out	In
0x6E00	In	Out	Out	In	Out	Out	Out
0x7000	In	Out	Out	Out	In	In	In
0x7200	In	Out	Out	Out	In	In	Out
0x7400	In	Out	Out	Out	In	Out	In
0x7600	In	Out	Out	Out	In	Out	Out
0x7800	In	Out	Out	Out	Out	In	In

Address	JA24	JA25	JA26	JA27	JA28	JA29	JA30
0x7A00	In	Out	Out	Out	Out	In	Out
0x7C00	In	Out	Out	Out	Out	Out	In
0x7E00	In	Out	Out	Out	Out	Out	Out
0x8000	Out	In	In	In	In	In	In
0x8200	Out	In	In	In	In	In	Out
0x8400	Out	In	In	In	In	Out	In
0x8600	Out	In	In	In	In	Out	Out
0x8800	Out	In	In	In	Out	In	In
0x8A00	Out	In	In	In	Out	In	Out
0x8C00	Out	In	In	In	Out	Out	In
0x8E00	Out	In	In	In	Out	Out	Out
0x9000	Out	In	In	Out	In	In	In
0x9200	Out	In	In	Out	In	In	Out
0x9400	Out	In	In	Out	In	Out	In
0x9600	Out	In	In	Out	In	Out	Out
0x9800	Out	In	In	Out	Out	In	In
0x9A00	Out	In	In	Out	Out	In	Out
0x9C00	Out	In	In	Out	Out	Out	In
0x9E00	Out	In	In	Out	Out	Out	Out
0xA000	Out	In	Out	In	In	In	In
0xA200	Out	In	Out	In	In	In	Out
0xA400	Out	In	Out	In	In	Out	In
0xA600	Out	In	Out	In	In	Out	Out
0xA800	Out	In	Out	In	Out	In	In
0xAA00	Out	In	Out	In	Out	In	Out
0xAC00	Out	In	Out	In	Out	Out	In
0xAE00	Out	In	Out	In	Out	Out	Out
0xB000	Out	In	Out	Out	In	In	In
0xB200	Out	In	Out	Out	In	In	Out
0xB400	Out	In	Out	Out	In	Out	In
0xB600	Out	In	Out	Out	In	Out	Out
0xB800	Out	In	Out	Out	Out	In	In
0xBA00	Out	In	Out	Out	Out	In	Out
0xBC00	Out	In	Out	Out	Out	Out	In

Address	JA24	JA25	JA26	JA27	JA28	JA29	JA30
0xBE00	Out	In	Out	Out	Out	Out	Out
0xC000	Out	Out	In	In	In	In	In
0xC200	Out	Out	In	In	In	In	Out
0xC400	Out	Out	In	In	In	Out	In
0xC600	Out	Out	In	In	In	Out	Out
0xC800	Out	Out	In	In	Out	In	In
0xCA00	Out	Out	In	In	Out	In	Out
0xCC00	Out	Out	In	In	Out	Out	In
0xCE00	Out	Out	In	In	Out	Out	Out
0xD000	Out	Out	In	Out	In	In	In
0xD200	Out	Out	In	Out	In	In	Out
0xD400	Out	Out	In	Out	In	Out	In
0xD600	Out	Out	In	Out	In	Out	Out
0xD800	Out	Out	In	Out	Out	In	In
0xDA00	Out	Out	In	Out	Out	In	Out
0xDC00	Out	Out	In	Out	Out	Out	In
0xDE00	Out	Out	In	Out	Out	Out	Out
0xE000	Out	Out	Out	In	In	In	In
0xE200	Out	Out	Out	In	In	In	Out
0xE400	Out	Out	Out	In	In	Out	In
0xE600	Out	Out	Out	In	In	Out	Out
0xE800	Out	Out	Out	In	Out	In	In
0xEA00	Out	Out	Out	In	Out	In	Out
0xEC00	Out	Out	Out	In	Out	Out	In
0xEE00	Out	Out	Out	In	Out	Out	Out
0xF000	Out	Out	Out	Out	In	In	In
0xF200	Out	Out	Out	Out	In	In	Out
0xF400	Out	Out	Out	Out	In	Out	In
0xF600	Out	Out	Out	Out	In	Out	Out
0xF800	Out	Out	Out	Out	Out	In	In
0xFA00	Out	Out	Out	Out	Out	In	Out
0xFC00	Out	Out	Out	Out	Out	Out	In
0xFE00	Out	Out	Out	Out	Out	Out	Out

Sample Data Structures

E

Overview

Following are the control structures with example settings used in configuring and operating the 5215 HARI interface. These structures use 32-bit Big-endian byte ordering.

```
typedef unsigned int          u32;  /* unsigned 32 bits        */
typedef unsigned short        u16;  /* unsigned 16 bits        */
typedef unsigned char         u8;   /* unsigned  8 bits        */


typedef struct hcfg_control { /* Configure Send/Rcv Channels   */

                              /* command header               */
      u32   length   = 0x;          /* length                  */
      u32   code     = 0x10;        /* code/controls           */
      u32   status   = 0xbad;       /* status                  */
                              /* control send channel CCB     */
      u32   scb      = 0x10;        /* SCB offset              */
      u32   ring     = 0x0;         /* ring address            */
      u32   rlen     = 0x100;       /* 16 elements             */
      u32   rngctl   = 0x56;        /* A32/D32 Block mode      */
      u32   mtu      = 0x100;       /* 256 byte MTU            */
      u32   rsvd     = 0x0;         /* most be 0               */
      u8    ilvl     = 0x1;         /* interrupt level         */
      u8    ivctr    = 0x60;        /* interrupt vector        */
      u8    thrtl    = 0x0;         /* do all available        */
      u8    flags    = 0x1;         /* RSP=1                   */
                              /* control receive channel CCB  */
      u32   scb      = 0x20;        /* SCB offset              */
      u32   ring     = 0x100;       /* ring address            */
      u32   rlen     = 0x100;       /* 16 elements             */
      u32   rngctl   = 0x56;        /* A32/D32 Block mode      */
      u32   mtu      = 0x100;       /* 256 byte MTU            */
      u32   rsvd     = 0x0;         /* most be 0               */
      u8    ilvl     = 0x1;         /* interrupt level         */
      u8    ivctr    = 0x61;        /* interrupt vector        */
      u8    thrtl    = 0x0;         /* do all available        */
      u8    flags    = 0x1;         /* RSP=1                   */
};
```

```
typedef struct hcfg_data { /* Configure Data Channels          */

                           /* command header                */
      u32   length   = 0x;          /* length                */
      u32   code     = 0x12;        /* code/controls         */
      u32   status   = 0xbad;       /* status                */
                           /* parameters                    */
      u32   send     = 0x1;         /* 1 data send channels  */
      u32   rcv      = 0x1;         /* 1 data rcv channels   */
                           /* data send channel CCB         */
      u32   scb      = 0x30;        /* SCB offset            */
      u32   ring     = 0x200;       /* ring address          */
      u32   rlen     = 0x500;       /* 64 elements           */
      u32   rngctl   = 0x76;        /* A32/D64 Block mode    */
      u32   mtu      = 0x2400;      /* 9180(+36) MTU         */
      u32   rsvd     = 0x0;         /* most be 0             */
      u8    ilvl     = 0x1;         /* interrupt level       */
      u8    ivctr    = 0x62;        /* interrupt vector      */
      u8    thrtl    = 0x0;         /* do all available      */
      u8    flags    = 0x0;         /* RSP=0, TCM=0          */
                           /* data rcv channel CCB          */
      u32   scb      = 0x40;        /* SCB offset            */
      u32   ring     = 0x700;       /* ring address          */
      u32   rlen     = 0x500;       /* 64 elements           */
      u32   rngctl   = 0x56;        /* A32/D32 Block mode    */
      u32   mtu      = 0x2400;      /* 9180(+36) MTU         */
      u32   rsvd     = 0x0;         /* most be 0             */
      u8    ilvl     = 0x1;         /* interrupt level       */
      u8    ivctr    = 0x63;        /* interrupt vector      */
      u8    thrtl    = 0x0;         /* do all available      */
      u8    flags    = 0x1;         /* RSP=1, RSE=0          */
};


typedef struct hcfg_raw { /* Configure Raw/OAM Receive Channel */

                           /* command header                */
      u32   length   = 0x;          /* length                */
      u32   code     = 0x14;        /* code/controls         */
      u32   status   = 0xbad;       /* status                */
                           /* raw receive channel CCB       */
      u32   scb      = 0x50;        /* SCB offset            */
      u32   ring     = 0xc00;       /* ring address          */
      u32   rlen     = 0x140;       /* 16 elements           */
      u32   rngctl   = 0x56;        /* A32/D32 Block mode    */
      u32   mtu      = 0x40;        /* 64 byte MTU           */
      u32   rsvd     = 0x0;         /* most be 0             */
      u8    ilvl     = 0x1;         /* interrupt level       */
      u8    ivctr    = 0x64;        /* interrupt vector      */
      u8    thrtl    = 0x0;         /* do all available      */
      u8    flags    = 0x1;         /* RSP=1                 */
};
```

```
typedef struct hcfg_rqs { /* Configure Rate Queues              */

                          /* command header                    */
      u32   length   = 0x;          /* length                  */
      u32   code     = 0x30;        /* code/controls           */
      u32   status   = 0xbad;       /* status                  */
                          /* rate queue controls               */
      u32   a0       = 0xcf1;       /* RQ_EN, 155Mbs           */
      u32   a1       = 0x0;         /* queue disabled          */
      u32   a2       = 0x0;         /* queue disabled          */
      u32   a3       = 0x0;         /* queue disabled          */
      u32   b0       = 0xcf1;       /* RQ_EN, 155Mbs           */
      u32   b1       = 0x0;         /* queue disabled          */
      u32   b2       = 0x0;         /* queue disabled          */
      u32   b3       = 0x0;         /* queue disabled          */
};

typedef struct hcfg_vct { /* Configure VCT Indexes              */

                          /* command header                    */
      u32   length   = 0x;          /* length                  */
      u32   code     = 0x20;        /* code/controls           */
      u32   status   = 0xbad;       /* status                  */
                          /* parameters                        */
      u32   index    = 0x0;         /* start at index 0        */
      u32   count    = 0x2;         /* config 2 entries        */
                          /* VCT 0, VC=5, raw/OAM cells        */
      u16   amode    = 0x0200;      /* EM=0, FCS=0, OAM        */
      u16   rsvd1    = 0x0;         /* must be 0               */
      u8    hdr0     = 0x0;         /* GFC=0; VPI=0            */
      u8    hdr1     = 0x0;         /* VPI/VCI=0               */
      u8    hdr2     = 0x0;         /* VCI=0                   */
      u8    hdr3     = 0x58;        /* VCI=5, PTI=OAM          */
      u16   mode     = 0x4000;      /* FCS=0, CC=16, RQ=A0     */
      u16   rsvd2    = 0x0;         /* must be 0               */
      u8    rsvd3    = 0x0;         /* must be 0               */
      u8    quota    = 0x1;         /* quota=1                 */
      u16   rsvd4    = 0x0;         /* must be 0               */
                          /* VCT 1, VC=5, AAL5 packets         */
      u16   amode    = 0x0d00;      /* EM=1, FCS=1, AAL5       */
      u16   rsvd1    = 0x0;         /* must be 0               */
      u8    hdr0     = 0x0;         /* GFC=0; VPI=0            */
      u8    hdr1     = 0x0;         /* VPI/VCI=0               */
      u8    hdr2     = 0x0;         /* VCI=0                   */
      u8    hdr3     = 0x50;        /* VCI=5, PTI=data         */
      u16   mode     = 0xc800;      /* FCS=1, CC=16, RQ=B0     */
      u16   rsvd2    = 0;           /* must be 0               */
      u8    rsvd3    = 0;           /* must be 0               */
      u8    quota    = 0x1;         /* quota=1                 */
      u16   rsvd4    = 0;           /* must be 0               */
};
```

```
typedef struct hset_burst { /* Set VME burst count              */

                              /* command header                 */
      u32   length   = 0x;          /* length                   */
      u32   code     = 0x13;        /* code/controls            */
      u32   status   = 0xbad;       /* status                   */
                                    /* parameters               */
      u32   burst    = 0x0;         /* full packet              */
};

=============

typedef struct csnd_ele { /* Control Send Ring Element          */
      u32   control  = 0x02000077;  /* IE=1, A32/D64, VMERD     */
      u32   address  = 0x0;         /* VME address              */
      u32   length   = 0x100;       /* 256 byte buffer          */
      u32   tag      = 0x0;         /* host usable tag          */
};

typedef struct crcv_ele { /* Control Receive Ring Element       */
      u32   control  = 0x02000076;  /* IE=1, A32/D64, VMEWR     */
      u32   address  = 0x0;         /* VME address              */
      u32   length   = 0x100;       /* 256 byte buffer          */
      u32   tag      = 0x0;         /* host usable tag          */
};

typedef struct dsnd_ele { /* Data Send Ring Element             */
      u32   control  = 0x07000077;  /* FB,EP,IE, D64, VMERD     */
      u32   address  = 0x0;         /* VME address              */
      u32   length   = 0x420;       /* 1024(%48) bytes          */
      u32   tag      = 0x0;         /* host usable tag          */
      u16   mode     = 0x0d00;      /* EM=1, FCS=1, AAL5        */
      u16   vci      = 0x1;         /* VCT=1                    */
};

typedef struct drcv_ele { /* Data Receive Ring Element          */
      u32   control  = 0x02000076;  /* IE=1, A32/D64, VMEWR     */
      u32   address  = 0x0;         /* VME address              */
      u32   length   = 0x2400;      /* 9180(+36) byte MTU       */
      u32   tag      = 0x0;         /* host usable tag          */
      u16   status   = 0xbad;       /* returned status          */
      u16   vci      = 0xbad;       /* returned ATM VCI         */
};

typedef struct rrcv_ele { /* Raw Receive Ring Element           */
      u32   control  = 0x02000076;  /* IE=1, A32/D64, VMEWR     */
      u32   address  = 0x0;         /* VME address              */
      u32   length   = 0x40;        /* hdr + cell + pad         */
      u32   tag      = 0x0;         /* host usable tag          */
      u16   status   = 0xbad;       /* returned status          */
      u16   vci      = 0xbad;       /* returned ATM VCI         */
};
```

ATM Technology Overview

F

Introduction to ATM

Asynchronous Transfer Mode (ATM) is a switched, connection-oriented technology that allows voice, data, images, and video to be sent over the same network media. With significantly higher bandwidths than legacy LANs such as Ethernet and Token Ring, ATM supports applications such as multimedia and teleconferencing with a quality of service that is unavailable from other networks.

In addition, ATM can be scaled down to interface with a wide range of existing network configurations, as shown in Figure F-1. This flexibility is a result of ATM's inherent scalability, which extends from desktops to supercomputers.

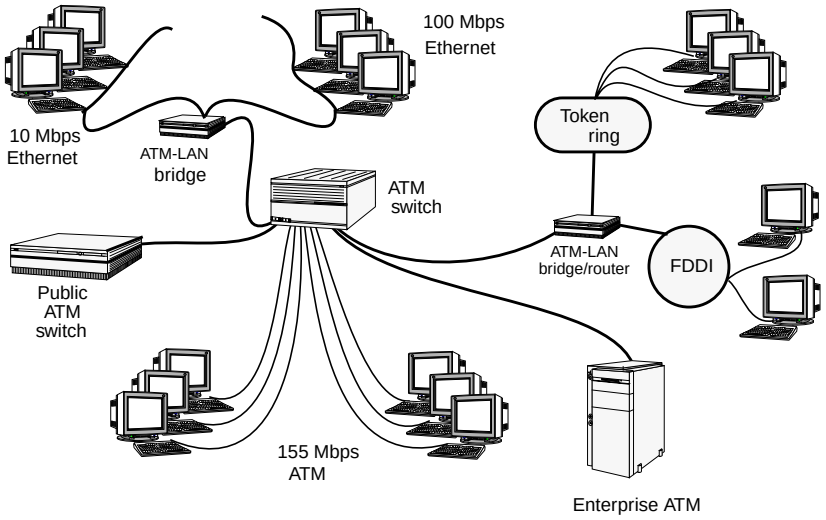


Figure F-1. ATM Network Topology

The core concept of ATM is cell switching. Cell switching combines the benefits of packet switching, as used in traditional data networks, with circuit switching, as used in voice communications.

With ATM, all data, voice, and video information is converted into small, fixed-length packets, called *cells*, of 53 bytes each. These cells provide a standard way to combine the transmission rates required by bursty, variable-length packets of data and images with the constant, average rates required by voice and full-motion video. Transmitting fixed-length cells virtually eliminates the traditional overhead delays encountered with variable-length packets.

Virtual Circuits

ATM uses *virtual circuits (VCs)* to create individual communication links between network nodes for transporting the 53-byte fixed-length cells. These virtual circuits carry all transmissions between nodes. They maintain the correct cell sequence throughout the length of the transmission and provide a defined Quality of Service (QoS).

ATM networks use two types of virtual circuits:

- Permanent virtual circuits (PVCs)
- Switched virtual circuits (SVCs)

The path or connection used by virtual circuits is the *virtual circuit connection* (VCC).

Virtual Circuit Connections

A virtual circuit connection (VCC) is a path or connection between any source and any destination in the ATM network. A VCC between two stations is configured manually for PVCs, and dynamically, via signalling, for SVCs. All communication between the two stations proceeds along the same VCC.

The ATM cell header's virtual circuit identifier (VCI) is assigned per network VCC link, that is, end station-to-switch, switch-to-switch, switch-to-end station, and so on.

The *virtual path* (VP) creates groups of the VCs carried between the ATM entities, such as the ATM switches shown in Figure F-2:

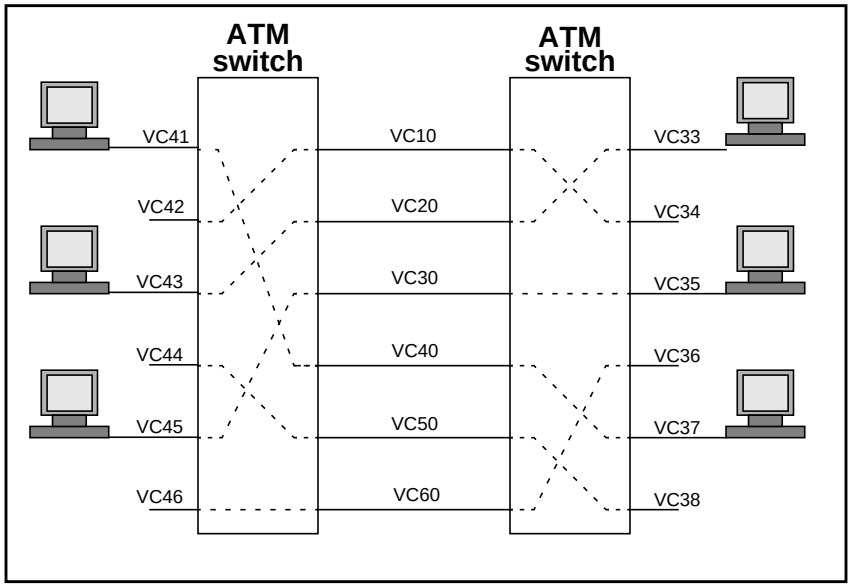


Figure F-2. Virtual Circuits

The VCs associated with a VP are globally switched without unbundling or processing the individual VCs. Thus, the cell-carrying capacity of each VC is preserved, and the quality of service of the VP can be scaled to extremely high speeds.

Permanent Virtual Circuits

Permanent virtual circuits (PVCs) are permanent, static connections between network nodes. With PVCs, the nodes operate as if they are connected with a dedicated physical line.

PVCs must be configured manually. To implement PVCs in an ATM network, you must create individual tables at each station and a master table at the switch. The table for each station must list the VCIs that identify the first steps in the paths leading to other stations with which the station is to communicate. At the switch, the master table must link all the information in the individual station tables.

When you move a station, you must recreate its table and also recreate its information in the switch's master table so that the station can reconnect to the network.

Switched Virtual Circuits

Switched virtual circuits (SVCs) are similar to voice telephone network connections; temporary connections between any two end points on the network are created dynamically for each session and torn down when the information exchange is complete. SVCs are created by signalling software in the end stations and the switches. This dynamic configuration eliminates the manual work required to implement PVCs.

With SVCs, the ATM switch only has to identify the output port to which to route a received cell and identify the new VPI/VCI on the output link.

Fixed-Length Cells

Fixed-length cells of 53 bytes each are transmitted over the VC established between the sending node and the receiving node. As shown in Figure F-3, each cell contains the following:

- A 5-byte header for connection identification, flow control, routing, and error control
- 48 bytes of data

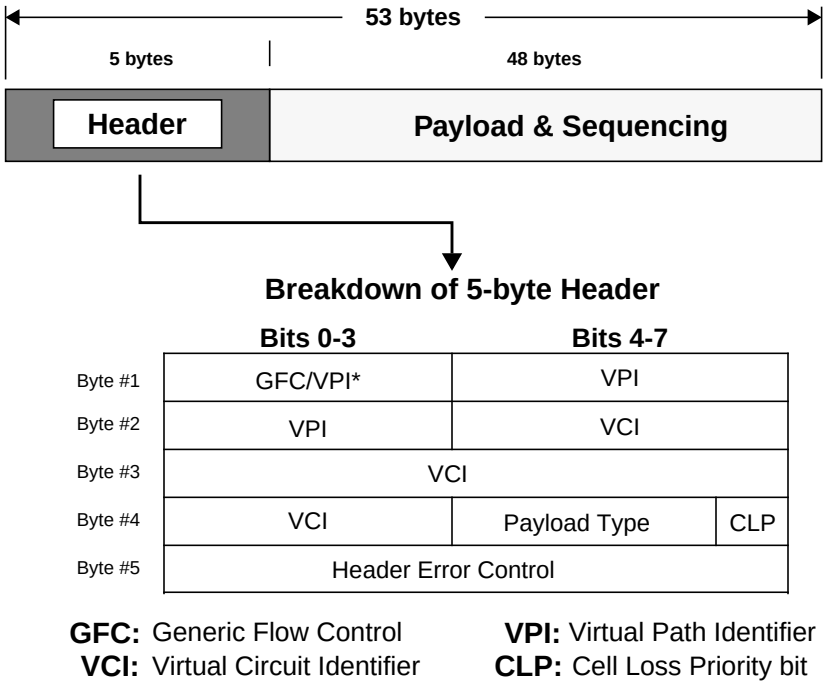


Figure F-3. The ATM Cell

The fixed-byte count allows fast hardware switching, lowers processing requirements, and practically eliminates the overhead associated with processing variable-length packets. As a result, ATM cells can be transmitted through multiple, contiguous connections and still maintain the time-sensitive quality of service required for audio and video applications.

Scalability

A major benefit of ATM is its inherent ability to support many different access and transmission speeds on different physical media. This is referred to as *scalability.*

For example, a cell generated on a 155 Mbps ATM LAN can be carried over a 45 Mbps asynchronous DS3 line to a network in a different location. From there, the cell can be switched into a 2.4 Gbps SONET (synchronous) transport system.

ATM networks can allocate a guaranteed amount of bandwidth for fixed-bit-rate, delay-sensitive transmissions such as video and voice data. And they can allocate a variable amount of bandwidth for the balance of network data that is less delay-sensitive.

ATM Layers

The B-ISDN (Broadband Integrated Services Digital Network) model defines three ATM layers:

- Physical Layer
- ATM Layer
- ATM Adaptation Layer (AAL)

In comparison with the OSI model, the three ATM layers are similar to the Physical Layer (Layer 1) and a portion of the Data Link Layer (Layer 2):

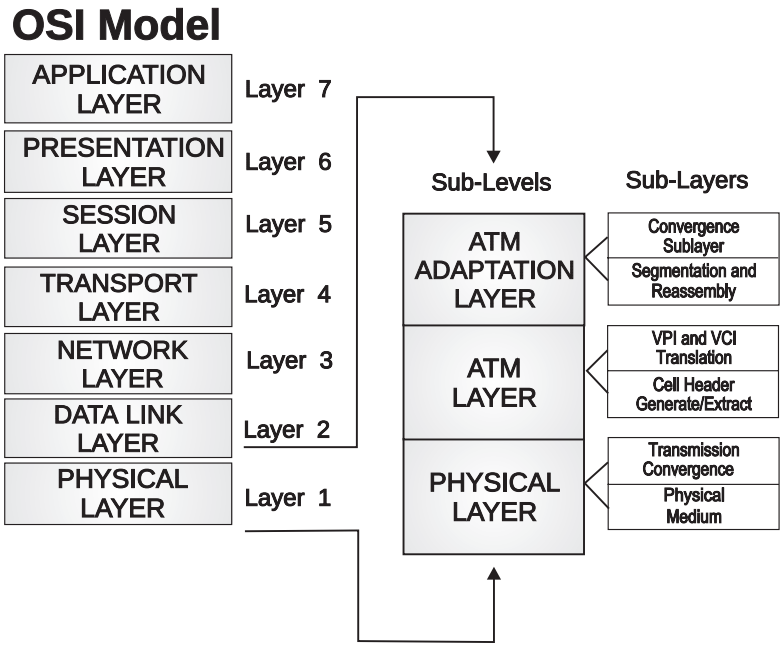


Figure F-4. ATM Compared to the OSI Model

Physical Layer

The ATM Physical layer defines the interface and framing protocol for the ATM network. The ATM Physical layer has two sublayers:

- The PMD sublayer provides bit transfer functions that are specific to the medium used in the network.
- The TC sublayer controls the transmission of frames over the physical medium.

ATM Layer

The ATM Layer defines the cell structure and how cells move between connections in an ATM network. This layer creates the cells and controls the setup and teardown of virtual connections and virtual paths in the network.

ATM Adaptation Layer

The ATM Adaptation layer (AAL) enhances the services provided by the ATM layer to better serve the next higher layer's functions. It provides timing and sequencing information tailored to the upper layer it is working with. It converts this information and the data being sent to ATM cells according to its rules. It then converts the ATM cells back to data after they reach their destination.

AAL is classified into four classes of service. For each class, AAL consists of two sublayers. The different service classes vary in the form of data transfer they support and in their sublayer structures.

AAL Service Classes

The currently defined AAL service classes are:

- **AAL1**
- **AAL2**
- **AAL3/4**
- **AAL5**

AAL1

AAL1 service supports the transfer of constant bit rate (CBR) data and provides mechanisms for maintaining timing and structure information for that data. It is primarily intended for transporting small data samples, such as voice.

AAL2

AAL2 service is not yet standardized. Its definition is currently under development by the ITU.

AAL3/4

AAL3/4 service supports the transfer of data that is sensitive to cell loss, but less sensitive to delay than AAL1-type data. AAL3/4 segments upper-layer data into cells, each of which consists of:

- A 2-octet header
- 44 octets of protocol data unit (PDU) payload
- A 2-octet trailer

The header and trailer provide message- and cell-level sequencing, as well as per-cell CRC. This cell-level management comes at the cost of the additional overhead octets in each cell. AAL3/4 is rarely used.

AAL5

AAL5 service, like AAL3/4, supports the transfer of data that is less sensitive to delay than AAL1 data. AAL5 is a simpler and more space-efficient protocol than AAL3/4. It relies on higher-level services to handle cell loss and other errors.

AAL5 encapsulates upper-layer SDUs into PDUs. Each PDU has a single trailer, which contains the upper-layer SDU control information, SDU length, and a CRC to guarantee the integrity of the entire PDU. AAL5 then segments the PDU into cells, each of which uses the entire 48-octet cell payload to deliver user data.

AAL Sublayers

The AAL for all service classes consists of two sublayers:

- **The Segmentation and Reassembly (SAR) sublayer c**onverts data into cells, and cells back to data.
- **The Convergence Sublayer (CS)** meets the specific service requirements of the next higher layer.

In AAL3/4 and AAL5, the Convergence Sublayer consists of two further sublayers:

- **The Service Specific Convergence Sublayer (SSCS)** handles details of the higher layer's specific services.
- **The Common Part Convergence Sublayer (CPCS)** converts upper-level SDUs to AAL PDUs.

Figure F-5 illustrates the protocol model for AAL3/4 and AAL5 sublayers:

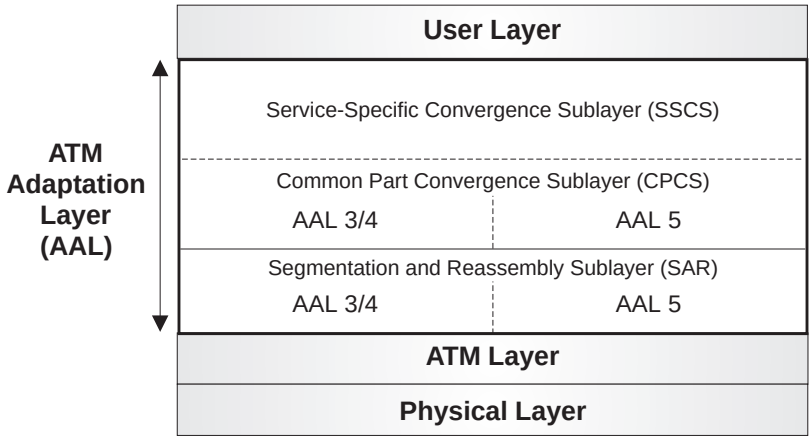


Figure F-5. Protocol Model for AAL3/4 and AAL5

Figure F-6 illustrates how the AAL5 sublayers handle data segmentation:

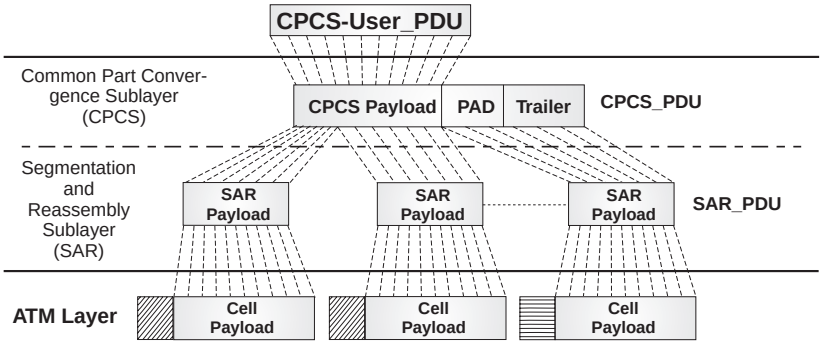


Figure F-6. AAL5 Sublayers

SONET/SDH

SONET (Synchronous Optical Network) is a North American multiplexing standard that defines a signal used in optical fiber networks. For ATM over fiber, OC-3c, with a base rate of 155.520 Mbps, is the most widely-used signal in the US.

SDH (Synchronous Digital Hierarchy) is the European equivalent of SONET. STM-1 (Synchronous Transport Mode, Level 1) is the European equivalent of OC-3c. Like OC-3c, STM-1 has a base rate of 155 Mbps, but STM-1 has slightly different framing information than the OC-3c signal.

Integrated Local Management Interface

Integrated Local Management Interface (ILMI) is the ATM Forum specification for incorporating network management into ATM networks. ILMI is based on the Simple Network Management Protocol (SNMP). The Interface Management Entities (IMEs) on the end station and the switch use ILMI to manage the physical link connecting them.

A subset of ILMI, known as *address registration,* is used when a station is connected to the ATM network. Address registration lets the end station and the switch dynamically build a unique ATM End Station Address (AESA). The AESA consists of two parts, the end station identifier (ESI) and the network prefix, which are concatenated together to define the end station's unique ATM network address.

- The ESI is the ATM adapter's unique 6-byte identifier.
- The network prefix is the 13-byte link address, as determined by the ATM switch.

LAN Emulation

LAN Emulation (LANE) is designed to allow existing network applications and network protocols to run over ATM networks. It supports ATM as a backbone for connecting *legacy* networks such as Ethernet, FDDI, and Token Ring. It supports efficient interaction for both directly-attached ATM end systems (in their own subnet) and legacy end systems attached through Layer 2 bridging devices.

LAN Emulation allows multiple emulated LANs (ELANs) to exist on the same physically interconnected ATM network. For example, using an appropriate ATM network interface card, an ATM end station can communicate with all of the following:

- An Ethernet segment on one ELAN
- A Token Ring segment on a second ELAN
- An FDDI segment on a third ELAN
- A fourth ELAN comprised of ATM nodes only (taking full advantage of high-speed ATM communications)

LAN Emulation Multiple Protocol Support

LAN Emulation supports a wide variety of protocols primarily because the functionality of the LANE interface is defined at the MAC layer. This allows for protocol-independent data transfer between devices attached to the ELAN and to other legacy-attached devices, as shown in Figure F-7:

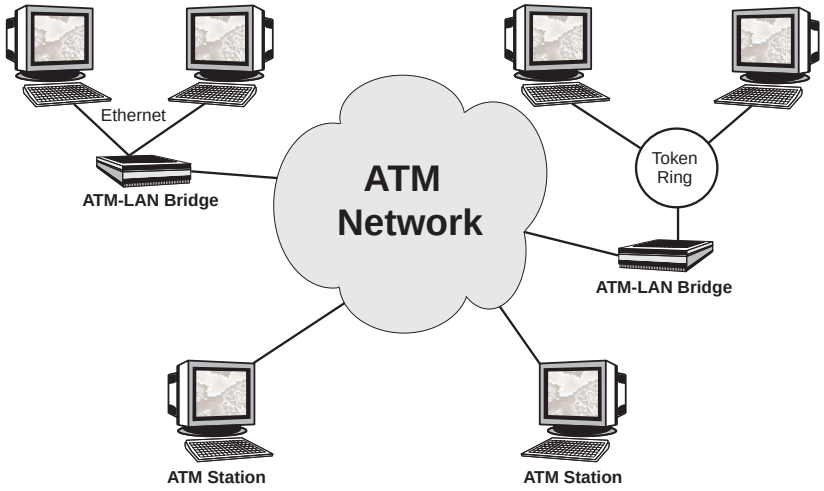


Figure F-7. ATM LAN Emulation

MAC-Dependent Protocol Support

However, LAN Emulation does not support protocols or applications that are dependent on MAC layer protocols, such as SMT/Token Management and CSMA/CD. These MAC-dependent applications are designed to deal with specific media issues such as ring arbitration and collision detection. In addition, because each ATM link carries traffic destined only for the one station attached to that link, a given station cannot receive all of the ELAN's frames like it would on a shared media network.

Therefore, removing the old equipment and installing LAN Emulation-based ATM equipment does not solve existing Ethernet/Token Ring/FDDI bridging problems.

Legacy LAN Protocol Support

LAN Emulation enables legacy LAN applications to operate over ATM while hiding the complexities of ATM point-to-point connections. With LAN Emulation, an ATM switch arranges ATM stations into groups that can implement and receive each other's broadcast/multicast requests. LAN Emulation supports requests in Ethernet and Token Ring frame formats, enabling legacy applications to run without modification over ATM networks. (LANE Emulation also supports FDDI, which uses Ethernet frames.)

LAN Emulation implements this frame format support with true MAC-level bridging. Multiple protocols, such as IP, IPX, and NetBIOS, can communicate across the ATM-LAN bridge. To illustrate how LAN Emulation works, Figure F-8 shows how an ATM host, an ATM-LAN bridge, and a legacy LAN host fit into the same network.

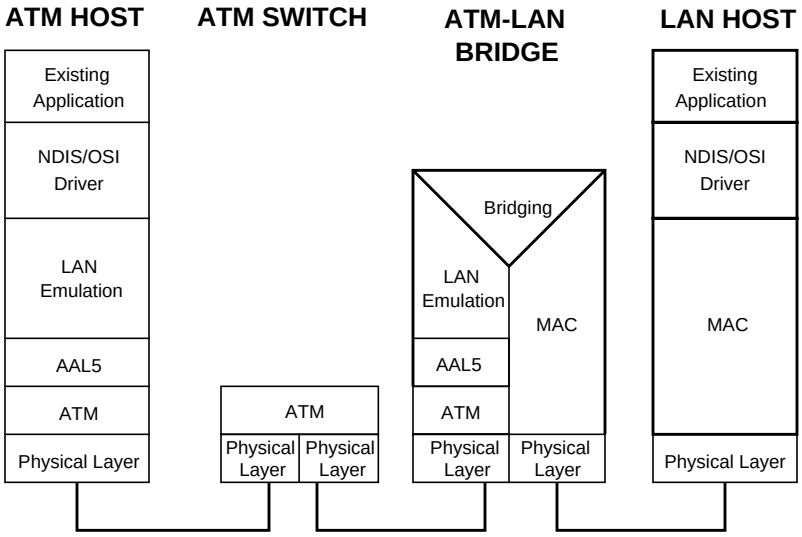


Figure F-8. LAN Emulation Protocol Stack

The goal of LAN Emulation (on the ATM side of the bridge) is to convince the upper layer protocols that a MAC layer of a legacy LAN lies in the lower layers, when actually it does not. The only discernible difference is that ATM's emulated LAN interface operates at much higher transmission rates.

All LAN Emulation connections (as defined by the ATM Forum) use the AAL5 ATM Adaptation layer to reassemble and segment the PDUs of legacy LANs into 53-byte ATM cells. The ATM-LAN bridge implements the MAC layer emulation to and from the legacy LAN. As illustrated by the LAN Host diagram in Figure F-8 on page 182, existing software requires no changes to run over ATM networks. ATM switches transport the cells just as they would for any other connection.

LAN Emulation Services

The LAN Emulation Client (LEC) defines the interface between a MAC layer entity of an ATM node and the emulated LAN. The LAN Emulation Service defines how the protocol interoperates with multiple LECs. It provides functions for initialization, address registration and resolution, and data forwarding. The three components that implement these functions are:

- LAN Emulation Configuration Server (LECS)
- LAN Emulation Server (LES)
- Broadcast and Unknown Server (BUS)

Some vendors provide these services in one or more switches, while others provide them in end station or edge devices.

LAN Emulation Configuration Server

The LECS provides a central point of contact for each LEC on the ATM network. It provides each LEC with configuration information, including the address of the LES that serves a specific ELAN. At bootup, LECs initially communicate with the LECS to locate the appropriate LES for their emulated LAN. The connection between the LEC and the LECS can be terminated when this transfer of information is complete.

Network personnel implement and maintain LECS information. Depending on the vendor, the LECS implementation might also provide numerous features such as support for multiple ELANs, ELAN management based on MAC addresses, and other network management features.

LAN Emulation Server

The LES provides LAN Emulation address resolution. This is a critical function because MAC addresses are 6-byte entities and ATM addresses are 20-byte entities. Each LEC notifies the LES of its MAC-ATM address bindings. The LES stores the MAC-ATM relationship so it can resolve queries from other clients for these MAC addresses.

All LECs on an ELAN connect to the same LES. All LECs on the same ELAN must log in with the LES before attempting to communicate with any other LEC in the group. They also must use the same Maximum Transfer Unit (MTU) size. An ELAN's MTU size depends on what type of legacy edge devices (if any) are being supported.

If the network interface card at the ATM node supports multiple LECs, each LEC on the card can be configured to a different ELAN and have a different MTU size than the other LECs. For example, three different LECs on the card might be configured as follows to communicate with nodes on three different ELAN types:

- LEC1 is configured to use a 1.5K MTU size to communicate with nodes on Ethernet subnets.
- LEC2 is configured to use a 4.5K MTU size to communicate with nodes on Token Ring and FDDI subnets.

- LEC3 can use a 9K (or maximum rate) MTU size to communicate with all ATM-only nodes on the ELAN.

If clients require more than one ELAN with the same MTU size, network personnel can configure differentiating characteristics (for example, logical names) in the LECS.

The connection between the LES and each LEC is open as long as the LEC is up and running. The LES maintains information about all LECs currently connected to the server. When a LEC is disconnected or turned off, the LES removes all information about the LEC.

Therefore, ATM networks using LAN Emulation support relocation of end stations with ease. Moving a station is a matter of physically moving the hardware to a new location, connecting it to the same or another ATM switch in the network, and turning on the power. At bootup, each client contacts the LECS; the LECS supplies the ATM address for the appropriate LES; the client joins the LES, and can then start communicating with other clients on that ELAN.

Broadcast and Unknown Server

The Broadcast and Unknown Server (BUS) provides a simplified mechanism to support both broadcasting and bridged network functionality. Most protocols use the network topology's shared media broadcast capabilities to implement the mechanism for translating network addresses to MAC addresses.

Because ATM is inherently non-shared, additional work must be done to enable every station on an ELAN to receive the broadcast and multicast frames. The BUS is a centralized server for the ELAN that accepts broadcast traffic from all clients in the group and then forwards that traffic to all the clients on a point-to-multipoint connection.

LAN Emulation Protocol

This section describes how the various phases of the LAN protocol accomplish the goal of getting multiple stations to communicate in the ATM network environment. The LAN Emulation protocol phases are:

1. Initialization
2. Configuration
3. Joining
4. Registration and BUS initialization
5. Data movement

Initialization

At power-up or reset, during initialization, a LEC's first action is to establish a connection with an LECS. The protocol defines a multi-step procedure to ensure that the LEC can reach the LECS in a variety of environments.

The first mechanism involves getting the LECS address(es) via ILMI. If no addresses are listed, or if none of the addresses work, the LEC uses the *well-known* ATM address (defined by the ATM Forum). An error occurs if this address does not work.

Additionally, as an out-of-band mechanism for obtaining the required configuration, network personnel can enter the address of the appropriate LES directly in the LEC setup.

Configuration

After the LEC establishes a connection to the LECS, the LEC transmits a message to the LECS. The message provides the LEC's ATM address, MAC address, LAN type, and frame size (MTU) request. When the LECS validates the request, it returns configuration information about the appropriate LES, including the LES's ATM address and the LAN type and frame size for the LEC to use for the session.

Joining

When the LEC gets the ATM address of the LES, it attempts to join the ELAN by creating a permanent connection to the LES. When it connects, the LEC transmits a message to the LES providing the configuration information obtained from the LECS and, optionally, a MAC address to register. A separate registration message can be used to register additional MAC addresses. The join process culminates in the LES adding the client as a leaf to its point-to-multipoint tree.

Registration and BUS Initialization

Upon joining the LES, the LEC requests the ATM address for the *all 1's* broadcast MAC address. The LES responds with the ATM address for the Broadcast and Unknown Server.

The LEC then initiates a Multicast Send connection to the BUS and accepts the incoming Multicast Forward multipoint VCC from the BUS.

Data Movement

After BUS connections are established, the LEC can begin forwarding frames. When a data frame needs to be transmitted, the LEC checks an internal table to see if a connection already exists for the destination MAC address. If so, it transmits the frame on that VCC. Otherwise, it queries the LES for the ATM address that corresponds with the MAC address. While waiting for a response, it forwards the frames destined for this MAC address to the BUS.

For a device on the legacy side of a bridge, the bridge learns the device's MAC address when the device starts transmitting. Then future requests to resolve the MAC address result in the bridge responding with its own ATM address.

Connections time out after periods of 20 minutes of inactivity in both directions. MAC entries received from the bridges are also timed out and reverified. This is done to support station movement on the legacy network where the edge devices do not have definitive knowledge of station movement activity.

LAN Applications

The advantage of LAN Emulation is that existing applications and protocols can be adapted to ATM merely by installing new ATM interfaces and drivers. The disadvantage is that these driver interfaces do not utilize any of the benefits of ATM, such as a varying quality of service, full ATM addressing, and other non-LAN protocol applications.

For future software designs, the ATM Forum has developed a Native Service Interface, which defines how new applications can be adapted to ATM networks, multiple vendors, and WinSock 2 support. This development provides an evolutionary solution where existing applications can continue to work as before, while new applications can take full advantage of ATM capabilities. The two interfaces are able to coexist in the same end systems.

Multiprotocol Encapsulation over ATM AAL5

The Internet Engineering Task Force (IETF) defines Multiprotocol Encapsulation over ATM AAL5 in RFC-1483. Multiprotocol Encapsulation over ATM AAL5 provides methods for encapsulating existing network protocol frames so that they can be transported over ATM in AAL5 frames. This encapsulation enables higher-layer protocols to use ATM networks to interconnect and exchange existing frame types.

RFC-1483 defines two encapsulation methods:

- The first method, *LLC encapsulation*, allows different frame types to be multiplexed over the same ATM VC. LLC encapsulation uses IEEE 802.2 Logical Link Control (LLC) headers to differentiate frame types. This is the primary method used in most environments.
- The second method, known as *VC Based Multiplexing* or *Null Encapsulation*, allows one frame type to exist on the VC because no encapsulation is used.

There are two classes of protocol frames:

- Routed protocols

  An example of a routed protocol is IP, which is the encapsulation defined in RFC 1577 to carry IP data over ATM networks.
- Bridged protocols

  Examples of bridged protocols are Ethernet and FDDI, which many ATM PVC services use to interconnect like legacy networks.

Specific encapsulation rules are spelled out for each class. In general, the LLC header identifies the encapsulated payload.

Classical IP over ATM

Classical IP (CIP) over ATM is designed to allow existing unicast IP network applications to run over ATM networks. The Internet Engineering Task Force (IETF) defines Classical IP in RFC 1577. Classical IP over ATM uses LLC Encapsulation for Routed Protocols, as defined in RFC-1483, for data encapsulation.

Figure F-9 on page 187 illustrates a Classical IP-over-ATM network configuration. Figure F-10 on page 187 illustrates the Classical IP-over-ATM protocol stack.

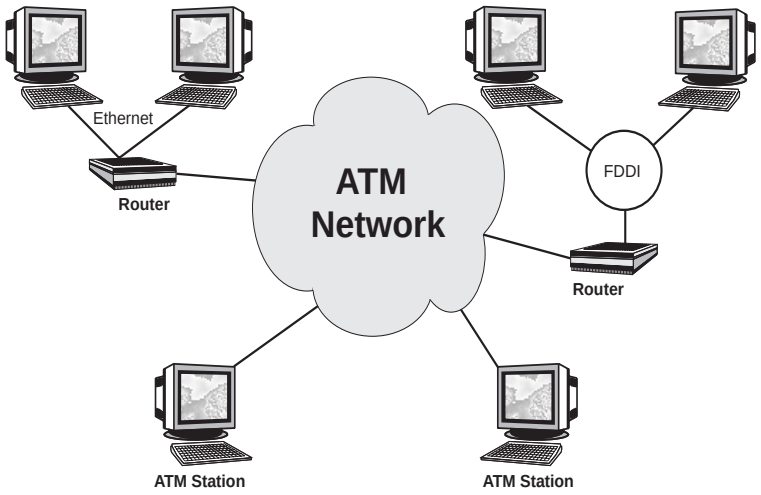


Figure F-9. Classical IP over ATM

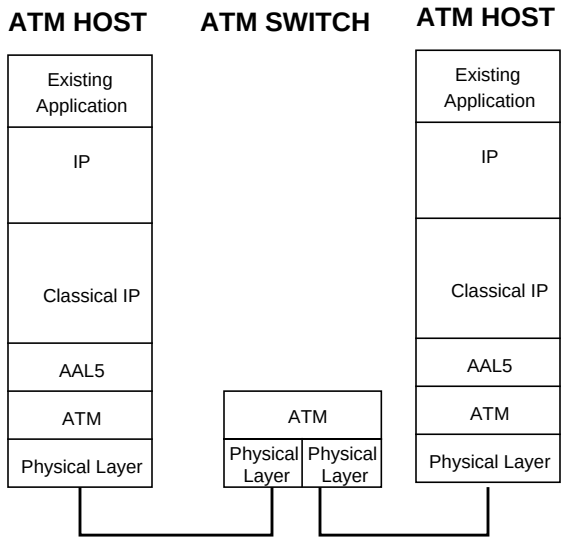


Figure F-10. Classical IP Protocol Stack

Classical IP Client Interface

The CIP Client defines the interface between an IP layer entity of an ATM node and the Logical IP Subnetwork (LIS). The standard supports only the delivery of unicast IP packets over ATM networks. To transfer a unicast IP packet, the CIP client must translate the IP address to an ATM address, create an SVC to that host, and then pass LLC encapsulated IP data over that VC.

The ATM ARP Server provides address resolution service to CIP clients. The ATM Address Resolution Protocol (ATMARP) defines how a CIP client can obtain the ATM address of an IP client it wants to connect to. This is done with ARP on broadcast enabled networks.

Because the ARP protocol requires a broadcast medium, IP-over-ATM networks use an additional protocol, the Inverse ATM Address Resolution Protocol (InATMARP), which is based on the Inverse Address Resolution defined in RFC 1293.

ATM ARP Servers use InATMARP to discover client address information. IP clients connected by PVC use InATMARP to discover address information about the remote end.

Most vendors provide ATM ARP Server services in one or more switches or ATM-attached devices such as routers.

Classical IP ARP Server

The ARP Server provides address resolution functions for clients on a Logical IP Subnet. Each client connects to the server via an SVC as it comes up on the ATM network or when it needs to resolve the address of another IP attached device. When the client connects, the server acquires the client's ATM to IP address mapping by means of InATMARP and builds a table of all the IP addresses on the LIS. Then when a client asks for the mapping via ATMARP, it can return the ATM address the client needs to create an SVC to the device.

Data Movement

When the ARP Server connection is established, the CIP client resolves IP to ATM address mappings and directly connects to other CIP Clients. When it needs to transmit a data frame, the client checks an internal table to see if a connection already exists for the destination IP address. If so, it encapsulates that frame with an LLC/SNAP header and transmits the frame on that VCC.

Otherwise, it queries the ARP Server for the ATM address that corresponds to the IP address. The Client then sets up an SVC to that ATM address and transmits the frame. For a device on the legacy (such as Ethernet) side of a router, the router's IP address must be configured as the gateway for the legacy subnet or as the default gateway for all unknown IP addresses not resolved by the ARP Server.

CIP Applications

The advantage of CIP is that existing IP unicast applications can be adapted to ATM merely by installing new ATM interfaces and drivers. They also perform better than other types of applications because of the larger MTU size supported by CIP.

The disadvantage is that the ATM driver interfaces do not utilize any of the benefits of ATM beyond its bandwidth and larger MTU support. The driver interfaces also do not support broadcast or multicast applications under CIP.

For More Information

For more information about ATM networking, see `http://www.atmforum.com`

Glossary

The ATM Forum's glossary can be found on the Internet at `http://www.atmforum.com/atmforum/glossary/glosspage.html`

The following glossaries may also be helpful:

- Networking Glossary: `http://www.ctcnet.com/tips/glossary.htm`
- Computing Dictionary: `http://wfn-shop.Princeton.EDU/foldoc/`

802.2 IEEE ◆ Standards that govern the LLC 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.

A16 ◆ A module that uses address lines A01 through A15.

A24 ◆ A module that uses address lines A01 through A23.

A32 ◆ A module that uses address lines A01 through A31.

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.

ACFAIL (AC Failure) ◆ Indicates AC input to the power supply is no longer being provided.

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 which 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.

AM0-AM5 (Address Modifier bits 0-5) ◆ Used to broadcast address size, cycle type, and master identification.

ANSI (American National Standards Institute) ◆ An organization which coordinates, develops, and publishes standards used in the United States.

Arbitration Bus ◆ Coordinates use of the DTB.

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.

AS (Address Strobe) ◆ Indicates a valid address on the address bus.

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.

BBSY (BUS Busy) ◆ Driven true by the current master to indicate it is using the bus.

BCLR (BUS Clear) ◆ Requests that the current master release the DTB.

BERR (BUS Error) ◆ Indicates to the current master that the data transfer was not completed.

best effort ◆ A QoS class where no specific traffic parameters and no attempts are made to guarantee no cell loss or delay variation.

BG0IN-BG3IN (BUS Grant 0-3 In) ◆ "Bus grant in" and "bus grant out" signals form bus grant daisy chains. The "bus grant in" signal indicates, to the board receiving it, that it can use the DTB.

BG0OUT-BG3OUT (BUS Grant 0-3 Out) ◆ Indicates to the next board in the daisy-chain that it can use the DTB.

B-ISDN (Broadband ISDN)

block read cycle ◆ Used to transfer a block of 1 to 1024 bytes from a slave to a master (256 transfers in 8, 16, or 32 bit width). The master broadcasts only one address and address modifier at the beginning of the cycle. The slave then increments this address on each transfer.

block write cycle ◆ A DTB cycle used to transfer a block of 1 to 1024 bytes from a master to a slave (256 transfers in 8, 16, or 32 bit width). The master broadcasts only one address and address modifier at the beginning of the cycle. The slave then increments this address on each transfer.

BR0-BR3 (BUS Request 0-3) ◆ Indicates that a master needs to use the DTB.

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 which 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 which 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 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 192.

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.

D00-D31 (Data Bus) ◆ Bidirectional data lines used to transfer data between masters and slaves.

data transfer ◆ Devices transfer data over the Data Transfer Bus (DTB).

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.

DS0, DS1 (Data Strobe 0, 1) ◆ Used in conjunction with LWORD and A01 to indicate how many data bytes are being transferred (1, 2, 3, or 4).

DS3 (Digital Standard 3) ◆ This ANSI standard defines the format of asynchronous data sent at the rate of 44.736 Mbps.

DTACK (Data Transfer Acknowledge) ◆ Signal generated by a slave. Indicates that valid data is available on the data bus during a read cycle, or that data has been accepted from the data bus during a write cycle.

DTB (Data Transfer Bus) ◆ The Data Transfer Bus allows masters to direct the transfer of data between themselves and slaves.

DTB arbitration ◆ The Arbitration Bus modules (requesters and arbiter) that coordinate and control data transfer.

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 193.

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.

ESI (End Station Identifier) ◆ Six-byte unique identifier that the end station concatenates to the network prefix during address registration to generate an ATM end station address (ESI).

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.

IACK (Interrupt Acknowledge) ◆ Used by an interrupt handler acknowledging an interrupt request.

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.

interrupt acknowledge cycle ◆ Initiated by an interrupt handler that reads status/ID information from an interrupter.

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.

IRQ1-IRQ7 (Interrupt Request 1-7) ◆ Generated by an interrupter.

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 realtime 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 190.

KB (Kilobytes) ◆ One kilobyte is equivalent to 1024 bytes when referring to memory size, and 1000 bytes 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).

LIS (Logical IP Subnet) ◆ IP subnet, which is fully contained in an ATM network and served by a Classical IP Server.

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 193.

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 193.

Master ◆ Initiates DTB cycles.

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 192.

module ◆ A functional board (master or slave) on the VMEbus.

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.

network ◆ An interconnection of multiple stations or systems that are able to send messages to and receive messages from one another.

network prefix ◆ First 13 bytes of an ATM End Station Address. Assigned by the switch during address registration. Used to route SVC signalling calls in the ATM network.

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 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.

priority interrupt ◆ Interrupt requests can be assigned one of seven priority values.

Priority Interrupt Bus ◆ Allows interrupter modules to send interrupt requests to interrupt handlers.

protocol ◆ A set of rules and conventions that govern the exchange of information between communicating parties.

PVC (Permanent/Provisioned Virtual Circuit) ◆ 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 196.

Q.93B ◆ Early draft of the signalling specification now known as Q.2931, on which UNI 3.0 was based.

Q.2110 ◆ ITU's *B-ISDN ATM Adaptation Layer- Service Specific Connection Oriented Protocol*, which is the protocol used to reliably transport signalling PDUs over a point-to-point link (such as between an end station and a switch).

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.

Q.SAAL (Signalling ATM Adaptation Layer) ◆ Original ITU draft of the specification of signalling's lower level service interface and transport protocol. Q.SAAL is now technically defined as *Q.2110*, but is still used to refer to this protocol layer.

read cycle ◆ Used to transfer 1, 2, 3, or 4 bytes from a slave to a master.

read-modify-write cycle ◆ Used to both read from and write to a slave location without permitting any other master to access that location. Useful in multiprocessing systems.

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 191.

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 196.

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 (approximately 8.5 micron) optical fiber through which only one signal can propagate. Length constraint for the device used in Interphase products is 20 kilometers.

Slave ◆ Detects DTB cycles initiated by a master.

SLIP (Serial Line Internet Protocol) ◆ A protocol for transmitting and receiving IP datagrams via a serial interface.

slot ◆ A position where a board can be inserted in a VMEbus backplane.

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 196.

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 196.

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 signaling element used by the MAC sublayer. The symbol set consists of sixteen data symbols and sixteen nondata 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.

SYSFAIL (System Fail) ◆ Indicates that a failure has occurred in the system. Can be generated by any board on the VMEbus.

SYSRESET (System Reset) ◆ Causes boards in slots to be reset.

system controller board ◆ A board which resides in slot 1 of a VMEbus backplane.

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 signaling. 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.

utilities ◆ These include a system reset line, a system fail line, and an AC fail line.

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 190 and *CAT-5 (Category 5 UTP)* on page 190.

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.

VME (Versa Module Eurocard)

VMEbus backplane ◆ A printed circuit board (PCB) with 96-pin connectors and signal paths that bus the connector pins.

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.

write cycle ◆ Used to transfer 1, 2, 3, or 4 bytes from a master to a slave.

workstation ◆ A networked computer typically reserved for end-user applications.

Index

When using this index, keep in mind that a page number indicates only where referenced material begins. It may extend to the page or pages following the page referenced.

R

S

T

U

V

W

X

Product Registration Card

Please take a minute to register your Interphase product. This will enable us to notify you about software updates and product enhancements.

Name _____

Title _____

Company Name _____

Company Address _____

City _____

State _____ Zip _____ Country _____

Province _____

Telephone ( ) _____

Fax ( ) _____

E-mail Address _____

Which product did you purchase? _____

Serial Number _____

Where did you purchase this product (company name)? _____

Date Purchased _____

Number of client nodes at this site: _____

Operating System(s) being used with this product:

❑ Solaris ❑ SunOS ❑ AIX ❑ NetWare

❑ HP-UX ❑ IRIX ❑ Mac OS ❑ Windows NT

❑ PC NFS ❑ DEC Ultrix ❑ Other _____

Operating System Version: _____

Network Protocol(s) in use:

❑ TCP/IP ❑ IPX ❑ AppleTalk ❑ Other _____

We welcome your comments, ideas, and suggestions. You can fax additional comments to 214/654-5500, or send them to intouch@iphase.com

_____

_____

_____

_____

If you are interested in receiving information on other products from Interphase, please check the appropriate boxes:

❑ ATM ❑ FDDI ❑ SCSI ❑ Fibre Channel ❑ 100 BaseT

If you are mailing this card from outside the United States, please add postage.

11/96

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 5784 DALLAS TEXAS

POSTAGE WILL BE PAID BY ADDRESSEE

ATTN: WARRANTY DEPARTMENT
INTERPHASE CORPORATION
13800 SENLAC DR
DALLAS TX 75234-9600