V/ATM 5215

High Performance Modular ATM Adapter for VMEbus System

User's Guide

V/ATM 5215

High Performance Modular ATM Adapter for VMEbus Systems

User's Guide

Document No. UG05215-000,REVB
Release Date: January 21, 1996

February 24, 1998

Copyright Notice

INTERPHASE CORPORATION
13800 Senlac
Dallas, Texas 75234
Phone: (214) 654-5000
FAX: (214) 654-5500

Disclaimer

Information in this user document supersedes any preliminary specifications, data sheets, and/or any other documents that may have been made available. Every effort has been made to supply accurate and complete information. However, Interphase Corporation assumes no responsibility or liability for its use. In addition, Interphase Corporation reserves the right to make product improvement without prior notice.

Trademark Acknowledgments

All terms used in this manual that are known to be trademarks or service marks are listed below. In addition, terms suspected of being trademarks have been appropriately capitalized. Use of a term in this manual should not be regarded as affecting the validity of any trademark or service mark.

- Interphase is a registered trademark of Interphase Corporation.
- UNIX is a registered trademark of UNIX Systems Laboratories, Inc.

Assistance

If You Purchased This Product From A Reseller

If you purchased this product from a reseller you must contact the reseller or distributor who sold you this product if:

- You need assistance with ordering or servicing
- You need technical assistance
- You have received the wrong product
- You have received a damaged product

If You Purchased Directly From Interphase Corporation

If you purchased product(s) directly from Interphase Corporation and need assistance with any of the items listed above, call Interphase Corporation. Please have your purchase order (P.O.) number and serial number ready.

In the United States:

Telephone	(214) 654-5555
Fax	(214) 654-5500
E-mail	intouch@iphase.com

In the United Kingdom:

Telephone	+44-1869-321222
Fax	+44-1869-247720

In the Pacific/Rim:

Telephone (Sales)	(81) 3-3445-7781
Telephone (Support)	(81) 3-3445-7783
Fax	(81) 3-3445-7782

Table of Contents

Chapter 5

Chapter 6

List of Figures

List of Tables

Notes:

CHAPTER 1
INTRODUCTION

OVERVIEW

This manual is intended for users 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.

HOW THIS MANUAL IS ORGANIZED

Chapter 1, Introduction — describes the V/ATM 5215 product related publications and the conventions and standards used in this manual are also discussed.

Chapter 2, Installation — describes the hardware, jumper configuration, and how to install the controller. It is recommended that you read this chapter thoroughly before attempting the installation.

Chapter 3, Functional Overview — describes how the 5215's interface works. This chapter also discusses the procedures for submitting commands and specific facts about the VMEbus and ATM technology that affect board operation.

Chapter 4, Common Boot Interface — describes the Common Boot Interface command set. The common Boot interface is used to perform extended diagnostic functions and to download code for on-board execution of special applications.

Chapter 5, Diagnostics — describes the 5215 extended and power-up diagnostics.

Chapter 6, Host Adapter Ring Interface (HARI) — describes the Host Adapter Ring Interface (HARI) and the function of the channels that comprise the interface.

Appendix A, Boot — describes the function of the Common Boot interface to initialize the controller to use the HARI interface. This appendix also describes how HARI is initialized with the desired operating parameters.

Appendix B, CB Error Codes — lists the various Common Boot power-up error codes.

Appendix C, Base Address Jumper Settings — contains a table of VME short I/O base address settings.

Appendix D, Sample Data Structures — provides sample control data structures used to configure and operate HARI. These structures use 32-bit, big-endian byte ordering.

RELATED PUBLICATIONS AND STANDARDS

The following publications are excellent resources 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

- The 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 discusse 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.

CONVENTIONS

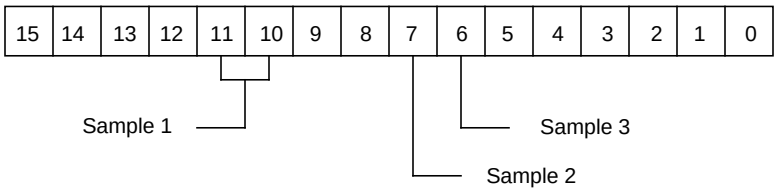
This section details many of the writing conventions used throughout the manual. In addition, it gives many of the technical conventions.

- 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:



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.

Notes:

CHAPTER 2
INSTALLATION

INSTALLATION OVERVIEW

Before attempting installation, read this chapter to ensure the safe and proper installation of the 5215 into your system. If you have any questions regarding installation which are not answered in this guide, please contact Interphase Customer Service.

The 5215 is installed into the VMEbus system using the following six steps:

Step 1. Visual Inspection

Step 2. Power System OFF

Step 3. Set On-Board Jumpers

Step 4. Install Board

Step 5. Power ON the system

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 (i.e., 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, proceed with the installation.

POWER THE SYSTEM OFF

Ensure that the host system and peripherals are turned OFF.

SET THE 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.

Table 2-1 on the following page lists the jumpers and their respective function.

The on-board jumpers are summarized in the following table. To locate the jumpers, refer to the board layout in Figure 2-1

Table 2-1. V/ATM 5215 On-Board Jumpers

On-Board Jumpers	
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)
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

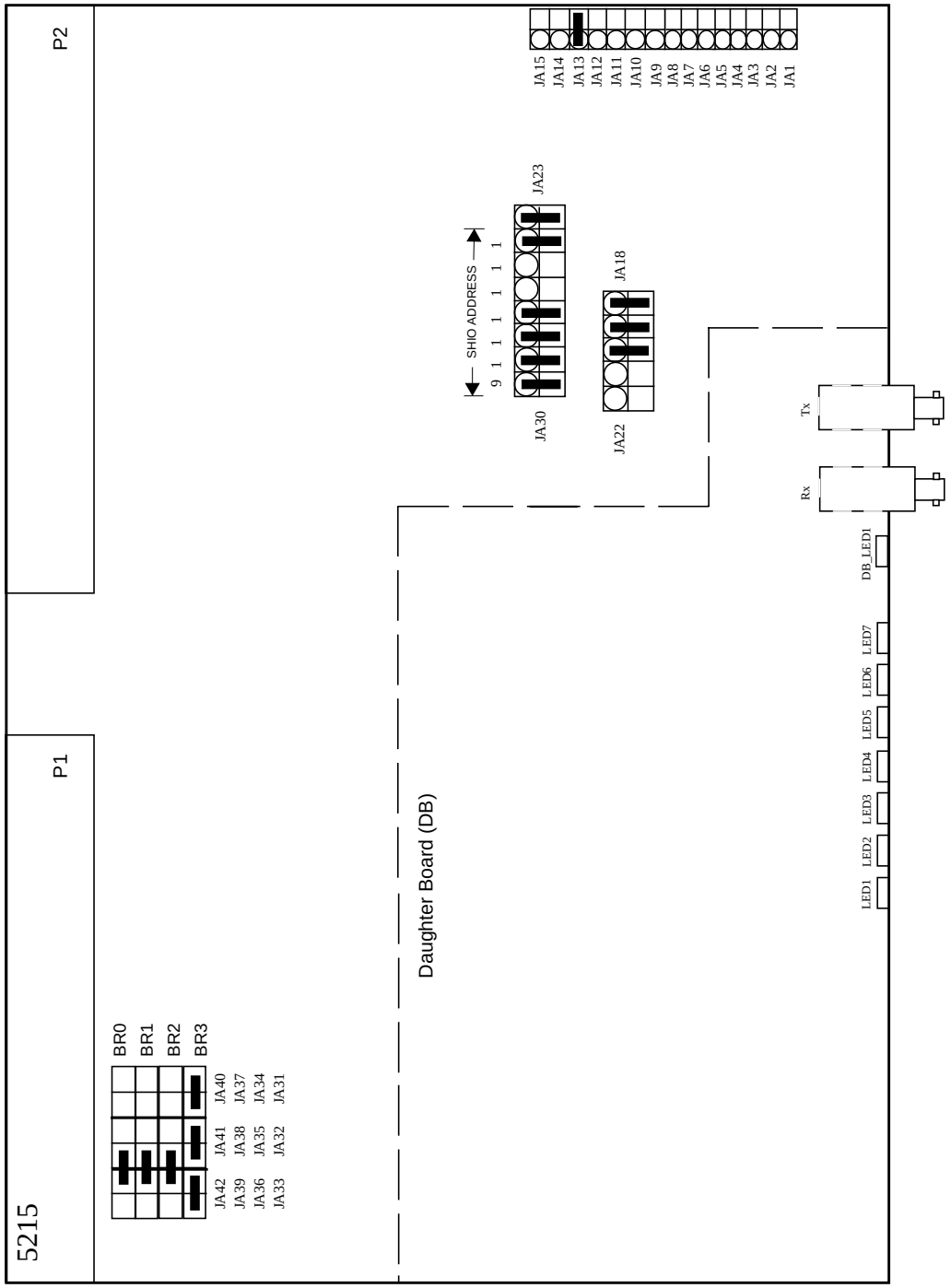


Figure 2-1. V/ATM 5215 Board Layout

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. If 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 (i.e., 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 it will cause the 5215 to 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 table below to determine the Base Address settings. A more complete listing is provided in **APPENDIX C**.

Table 2-2. Short I/O Base Address Jumpers:

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 Short I/O Base Address settings.

Table 2-3. Jumper settings for 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
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.

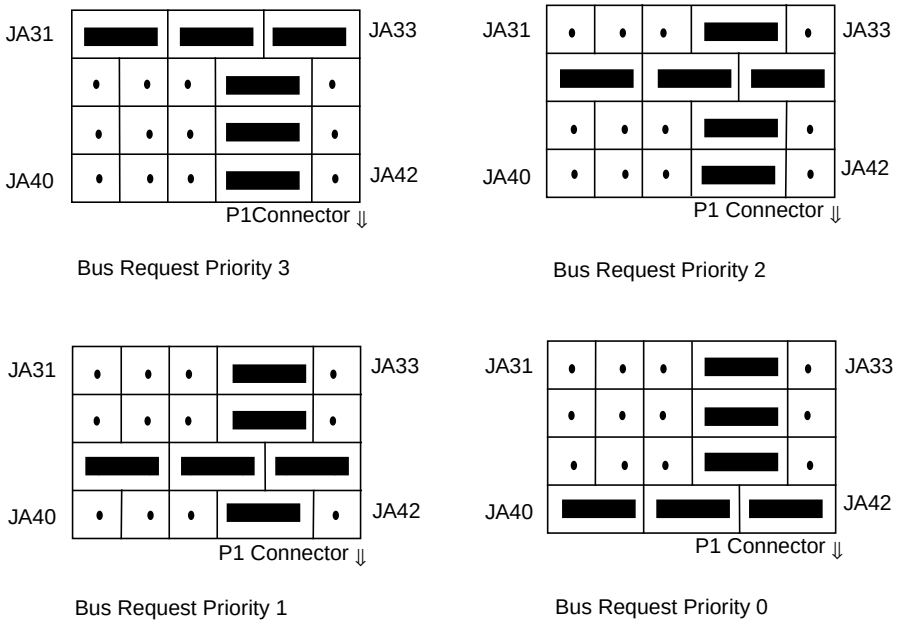


Figure 2-2. VMEbus Request Priority Jumper Setting

FRONT PANEL LEDs

There are seven LEDs on the 5215 motherboard and two LEDs on the daughterboard. Refer to the board layout diagram, Figure 2-1 for the location of the LEDs. The LEDs are:

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 to 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.

DAUGHTERBOARD 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 daughterboard (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:

Table 2-4. ST0-ST1 LEDs - Board OK Status

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:

Table 2-5. Link 0-Link 1 LED Status

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	Daughterboard is initialized and running

5215 BLOCK DIAGRAM

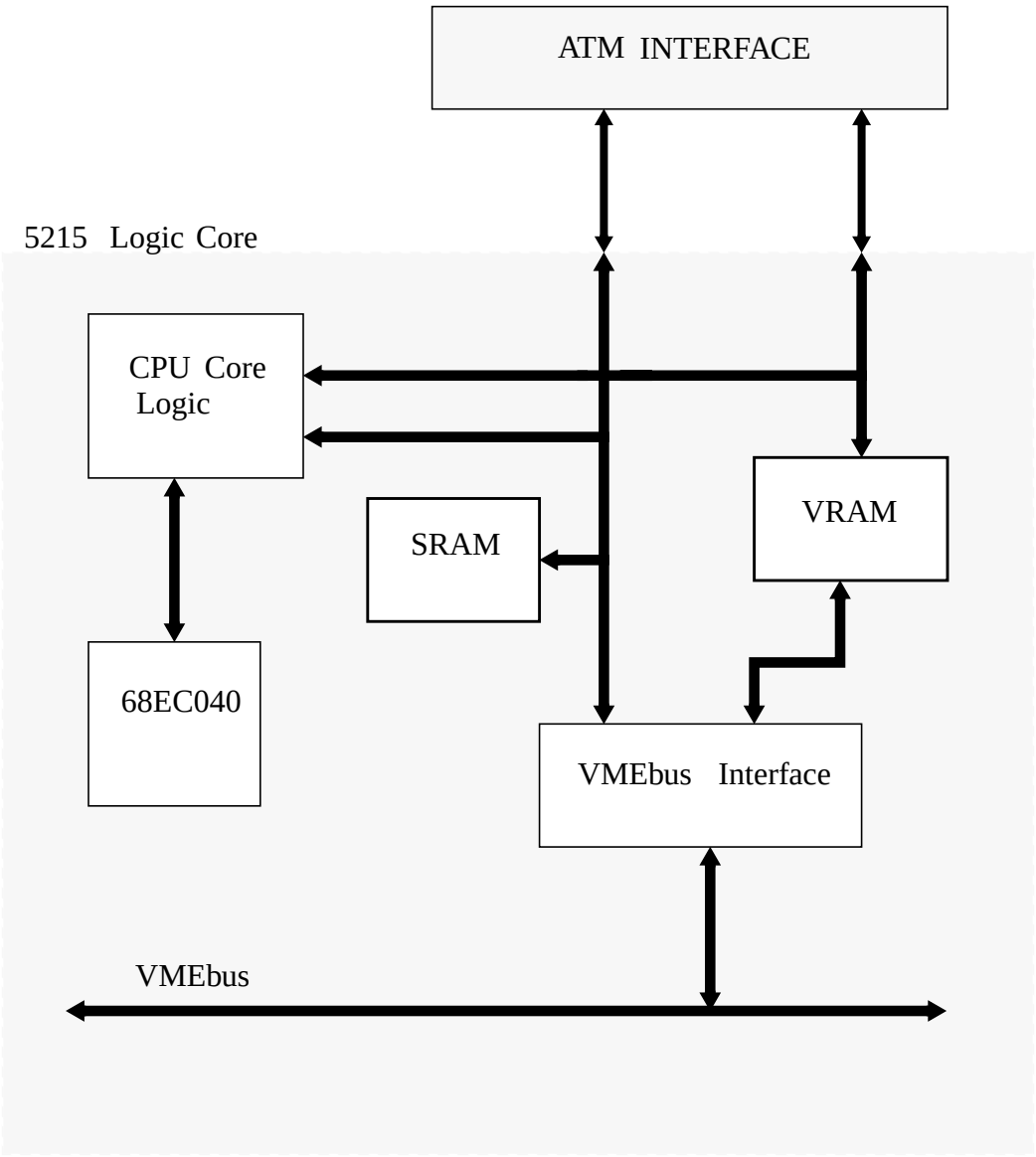


Figure 2-3. Block Diagram of the V/ATM 5215 Controller

5215 DAUGHTERBOARD BLOCK DIAGRAM

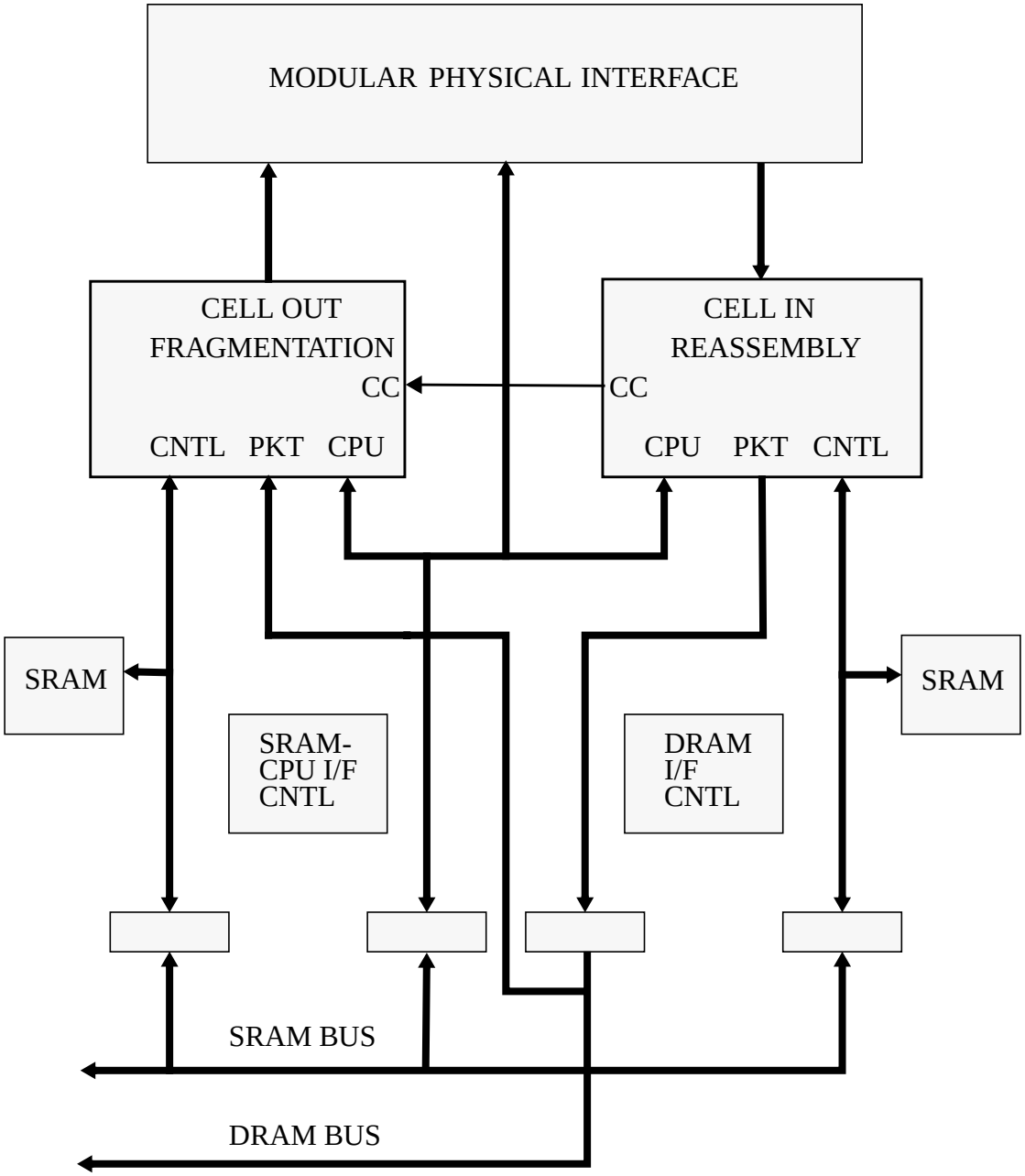


Figure 2-4. Block Diagram of the 5215 Daughterboard

INSTALLING THE BOARD(S)

Prior to installing the 5215 in the VMEbus configure the backplane jumpers, if required, for each board to be installed. Consult 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.

POWER 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 the 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”, page 4-4.) For more information on the LEDs, see page 2-9.

NOTE

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.

CHAPTER 3
FUNCTIONAL OVERVIEW

OVERVIEW OF THE 5215 INTERFACE

The 5215 firmware interface consists of two major parts used for initializing and operating the controller. These are called the Common Boot (CB) Interface and the Host Adapter Ring Interface (HARI) Interface.

The Common Boot Interface is a monitor which 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 for permanently storing 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.

.

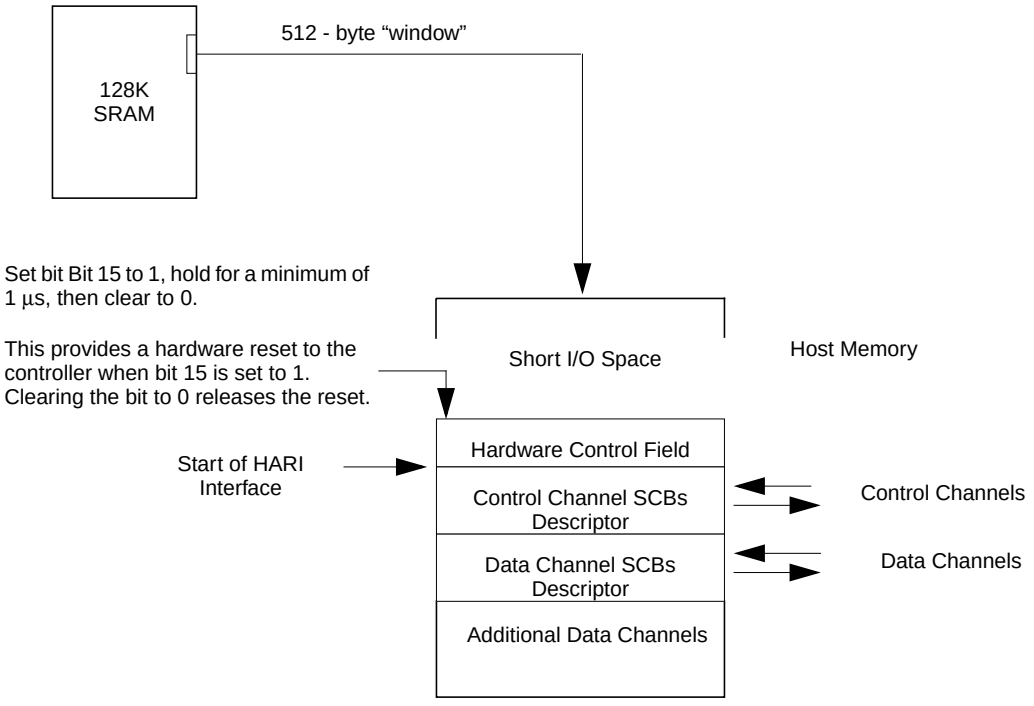


Figure 3-1. Short I/O as a Window into the 5215's Communications Buffer Address Space

SYSTEM REQUIREMENTS

To interface with the 5215, the host must meet 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 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” (i.e., Most Significant Byte first).

CHAPTER 4
THE COMMON BOOT INTERFACE

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 section 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.

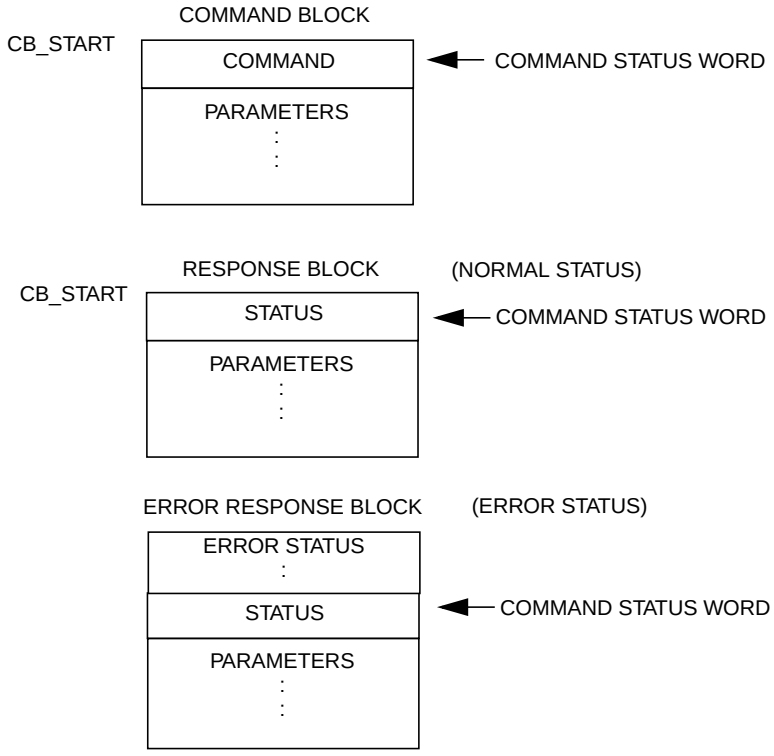


Figure 4-1. Common Boot Command/Response Format

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 4-1) with one of the Common Boot commands listed on page 4-7. 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 (See Figure 4-1), 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.

.

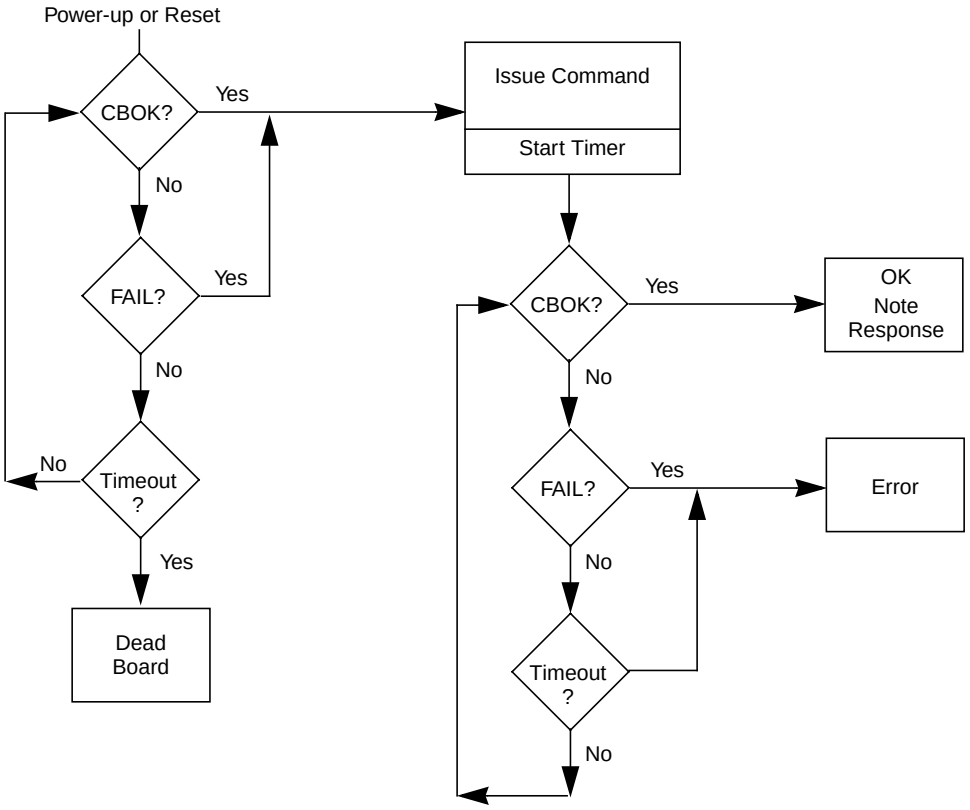


Figure 4-2. Polling the Command Status Word for CBOK, FAIL, or Response

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 BOOTUP AND RESET SEQUENCE on page 4-4).
- 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:

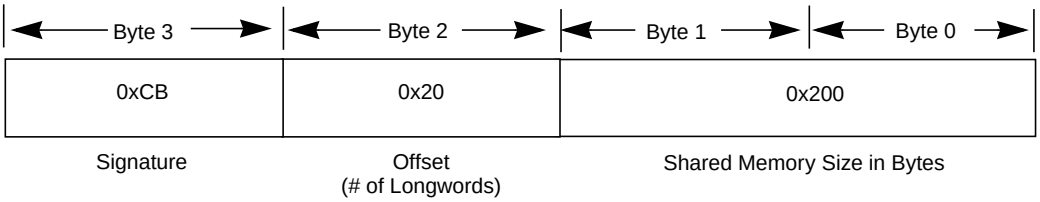


Figure 4-3. CB_HERALD Format

- Bytes 1 and 0 contain the size of 5215's 512 byte shared memory space.
- Byte 2 specifies the offset (in longwords) at which 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 4-4 depicts the controller's shared memory space after the controller initializes successfully.

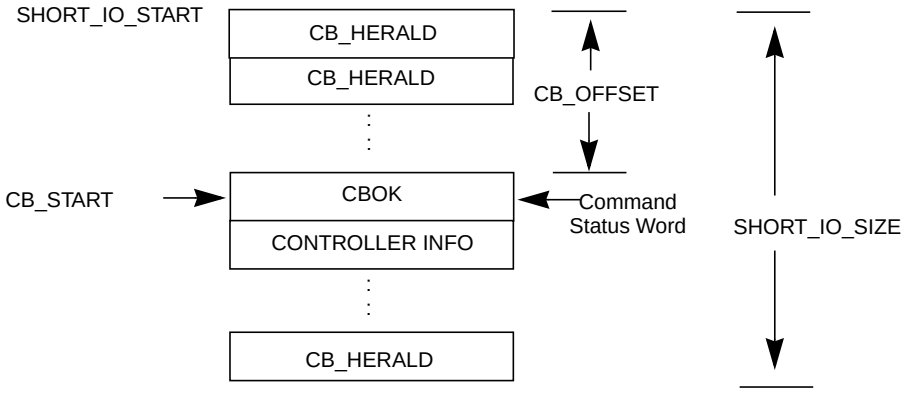


Figure 4-4. Format of Shared Memory after CBOK

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 4-5 depicts the format of shared memory when FAIL occurs during power-up diagnostics.

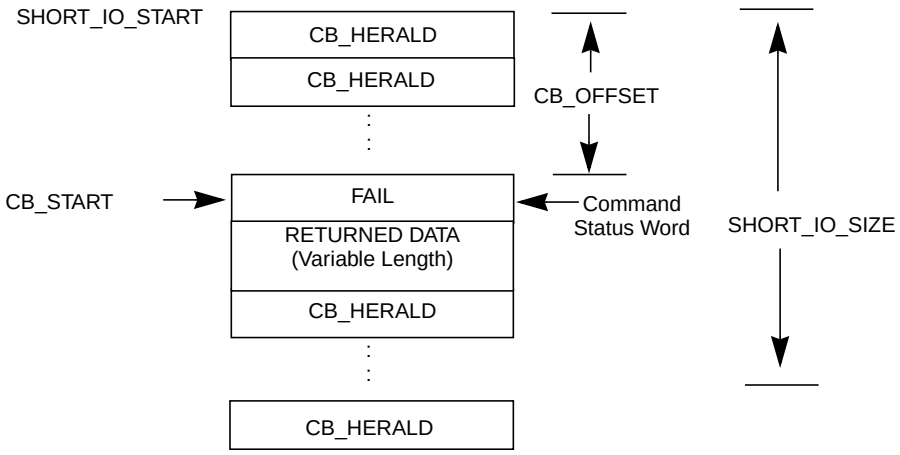


Figure 4-5. Format of Shared Memory after FAIL

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 (i.e., 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 section describes the Common Boot (CB) commands. The commands are listed in Table 4-1.

Table 4-1. V/ATM 5215 Common Boot Commands

COMMAND	DESCRIPTION	SEE PAGE
BOOT	This command is used to boot an alternate firmware image	4-9
DIAG	This command performs a series of tests whose definition is specific to the 5215.	4-11
FILL	This command allows the host to modify controller memory or hardware	4-13
FLSH	This command allows the host to control the state of the controller's on-board LED	4-15
GRNL	This command allows the host to control the state of the controller's on-board LED	4-16
JUMP	This command allows the host to force the CPU to do an unconditional jump	4-17
PEEK	This command allows the host to examine controller memory	4-18
POKE	This command allows the host to modify controller memory	4-19
DNLD	This command is used to download an image to the controller	4-20
BURN	This command is used to store a saved downloadable image into flash EPROM to be used as the EXEC 0 image	4-22
REDL	This command allows the host to control the state of the controller's on-board LED	4-23
STUF	This command allows the host to modify controller memory or hardware	4-24
HGET	This command allows the host to examine controller memory or hardware	4-26
HPUT	This command allows the host to modify controller memory or hardware	4-28
INFO	This command allows the host to identify a controller as well as obtain various controller specific information	4-30

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 the above-listed 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.

A description of the Common Boot command set appears later in this chapter.

DOWNLOADING CODE

A download mechanism is provided on Interphase controllers which 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 which automate the download process. These utilities are tied to specific controller types (e.g. 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

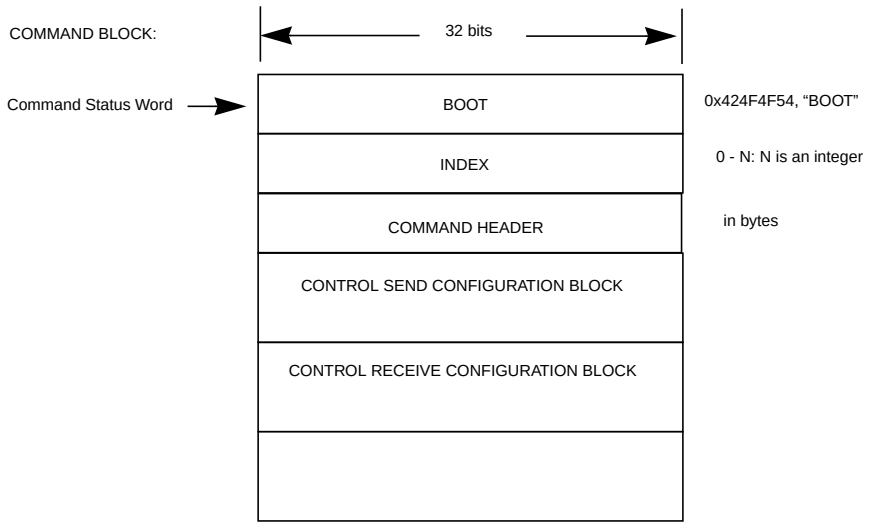


Figure 4-6. BOOT Command

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 (i.e. the default interface stored in EPROM.) For information on booting a downloaded interface, see DOWNLOADING CODE on page 4-8. The specifics of Boot are discussed in **APPENDIX A**.

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 which 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.

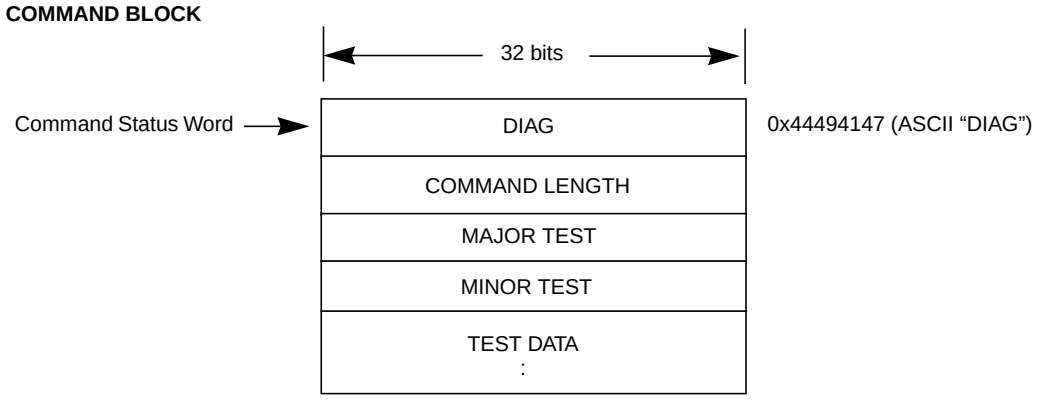


Figure 4-7. Format of the DIAG command

.

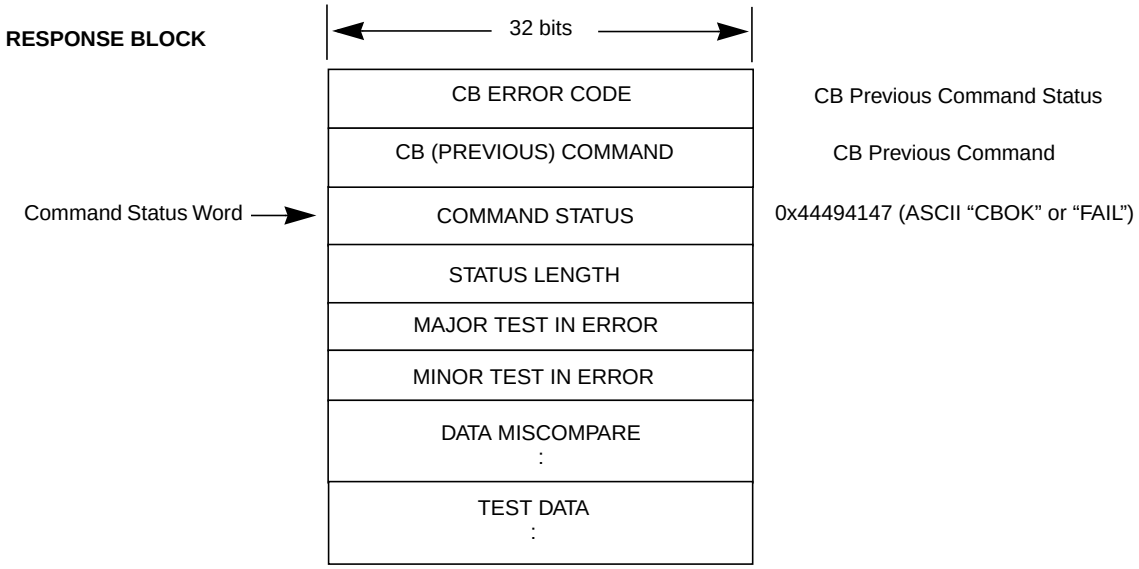


Figure 4-8. “FAIL” Response Block

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 **CHAPTER 5** for more information on diagnostics.

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 4-8.

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 (i.e. Don’t Cares). If the response is “FAIL,” additional information is provided in the fields as shown in Figure 4-8 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

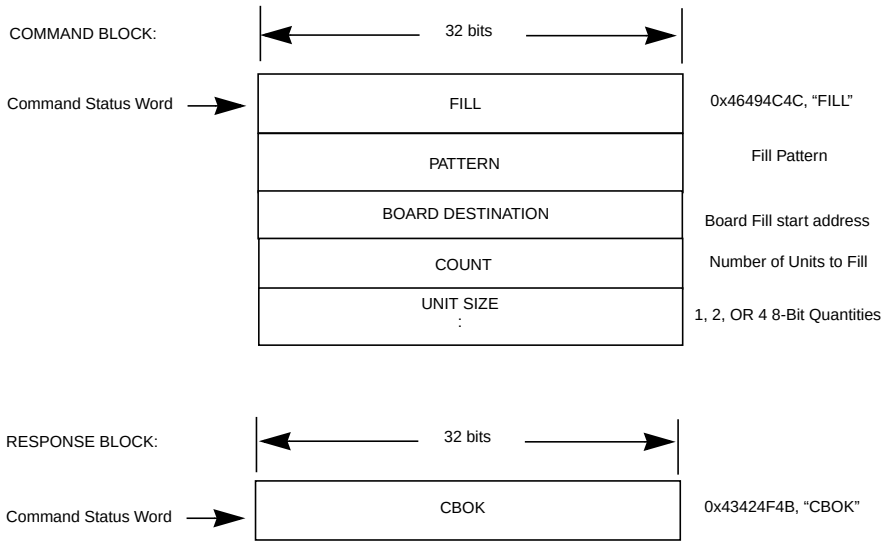


Figure 4-9. FILL Command and Response Format

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

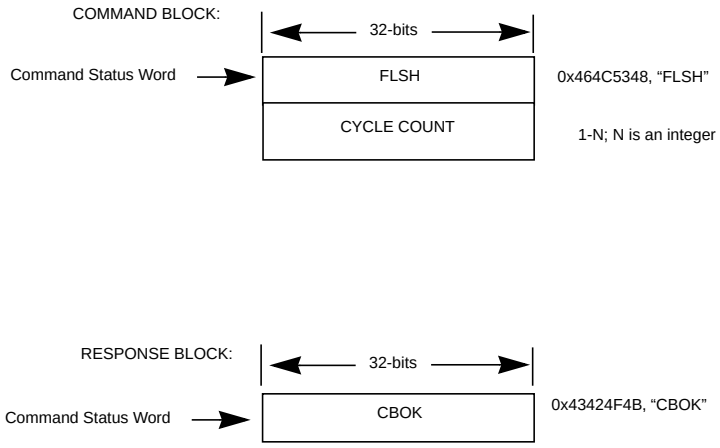


Figure 4-10. FLSH Command and Response Format

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.

NEVER FAILS

GRNL

Group: Common Boot

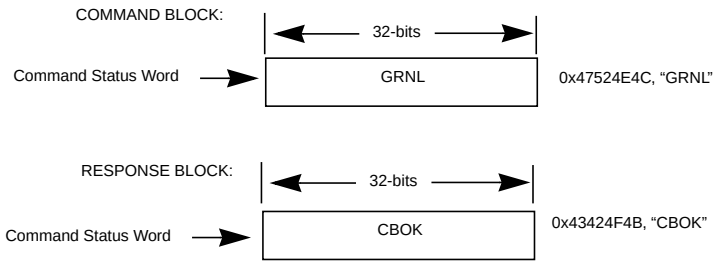


Figure 4-11. GRNL Command and Response Format

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.

NEVER FAILS

JUMP

Group: Common Boot

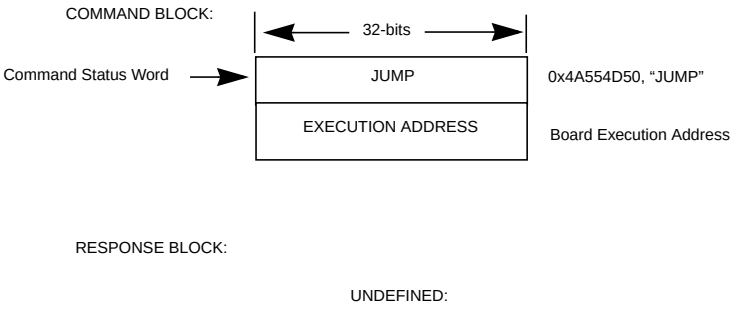


Figure 4-12. JUMP Command and Response Format

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

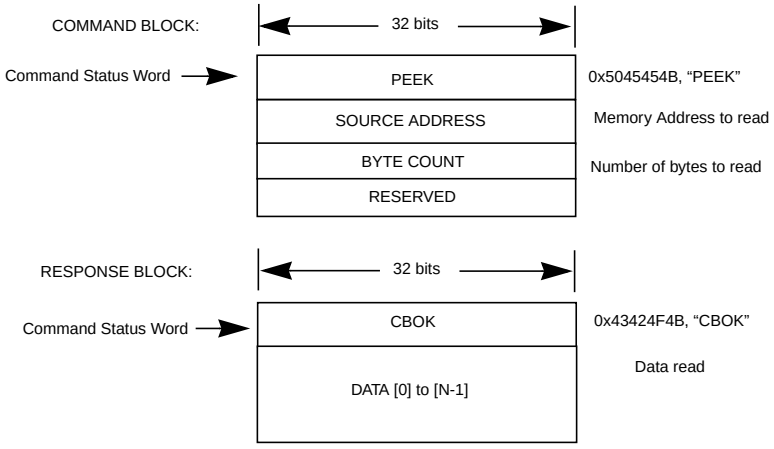


Figure 4-13. PEEK Command and Response Format

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

NEVER FAILS

POKE

Group: Common Boot

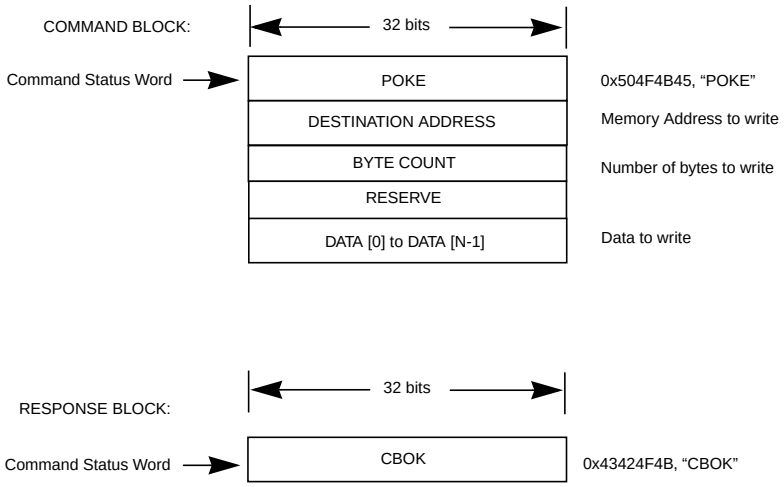


Figure 4-14. POKE Command and Response Format

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.

NEVER FAILS

DNLD

Group: Common Boot

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

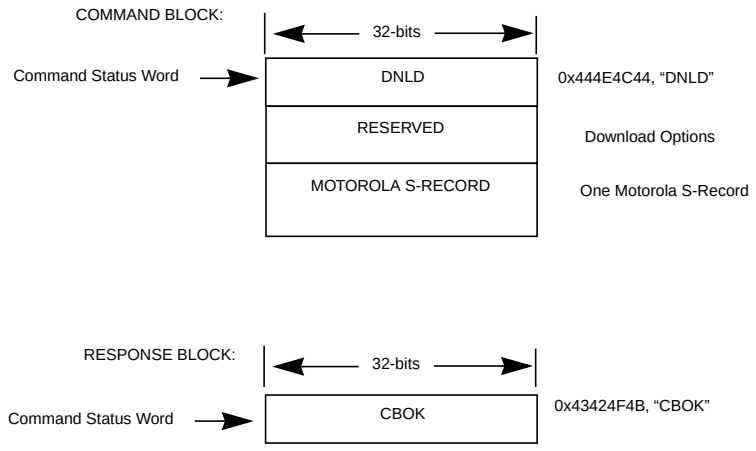


Figure 4-15. DNLD (DOWNLOAD) Command and Response Format

FUNCTION

This command allows the host to download Motorola S-Records, one record at a time, into the units download memory. The function will track the SD-Record type (0-9), will convert the input from ASCII to binary format, will checksum each record, will track if the image being downloaded will 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 down loaded image that was saved in RAM using the CB DNLD command.

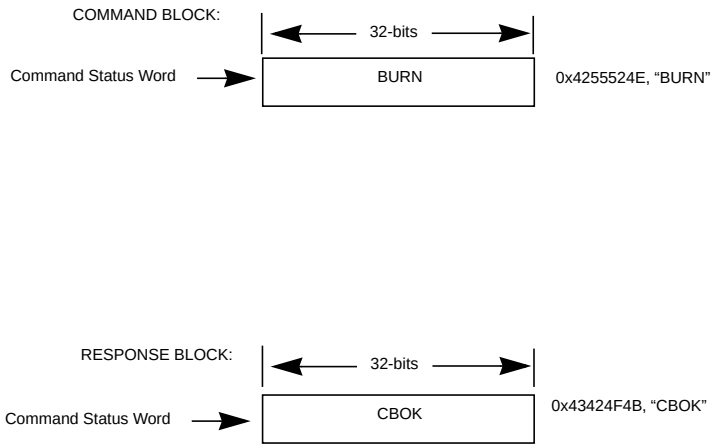


Figure 4-16. BURN Command and Response Format

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

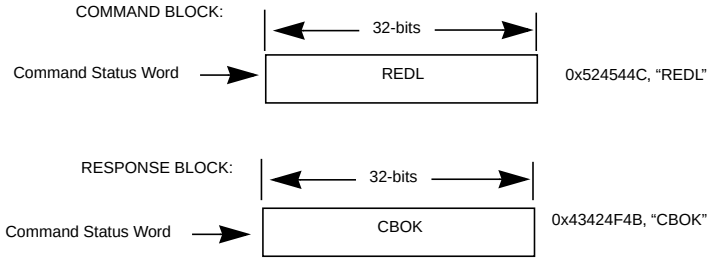


Figure 4-17. REDL Command and Response Format

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.

NEVER FAILS

STUF

Group: Common Boot

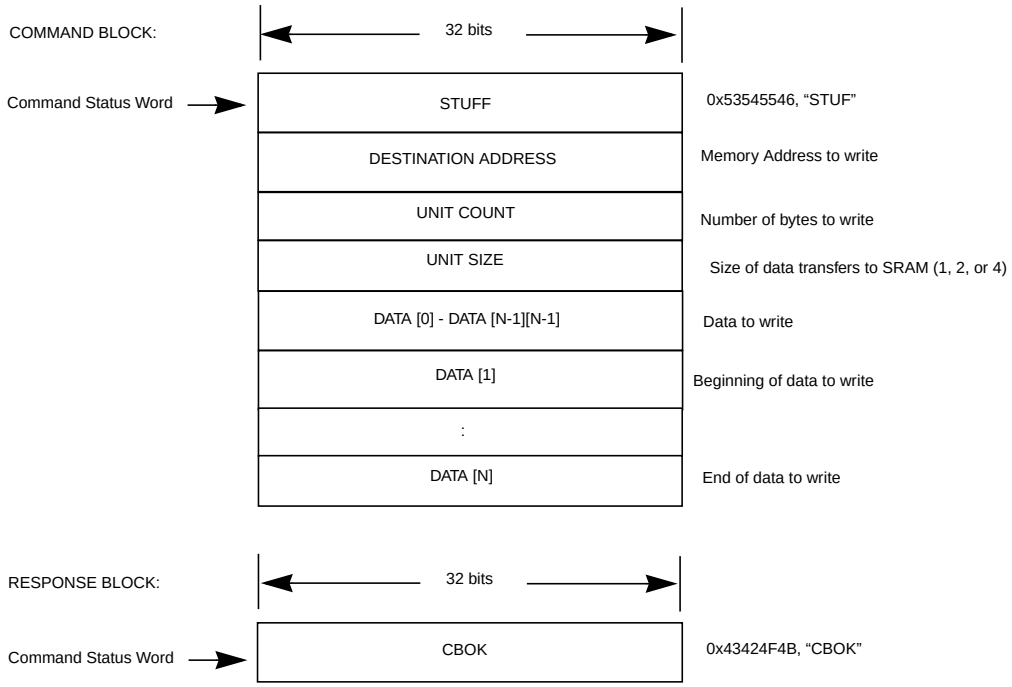


Figure 4-18. STUF Command and Response Format

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.

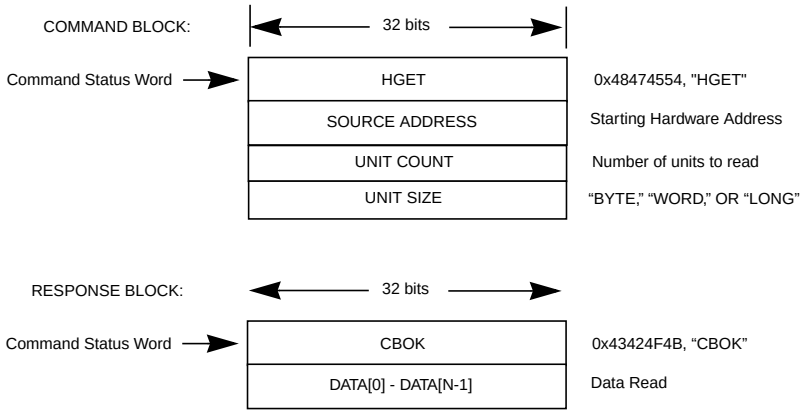


Figure 4-19. HGET Command and Response Format

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.

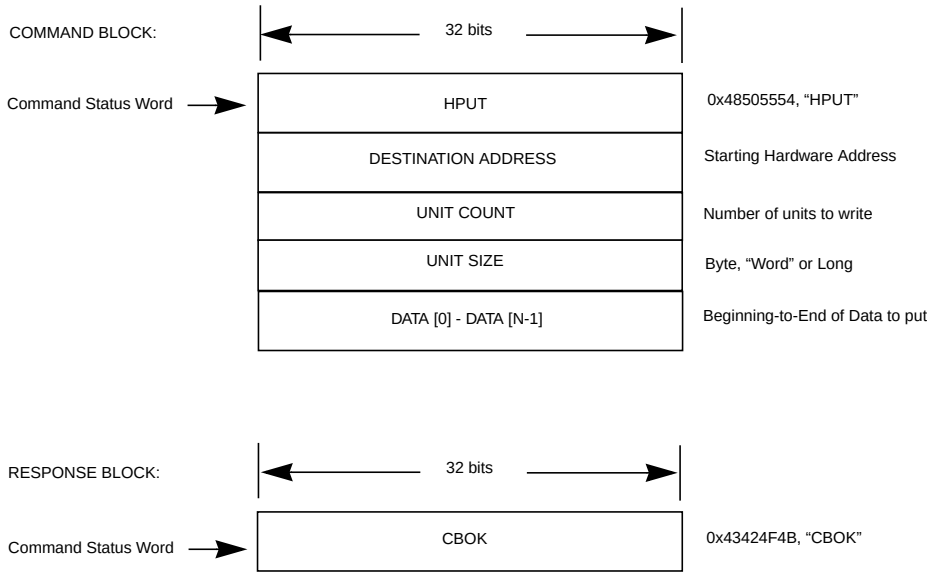


Figure 4-20. HPUT Command and Response Format

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.

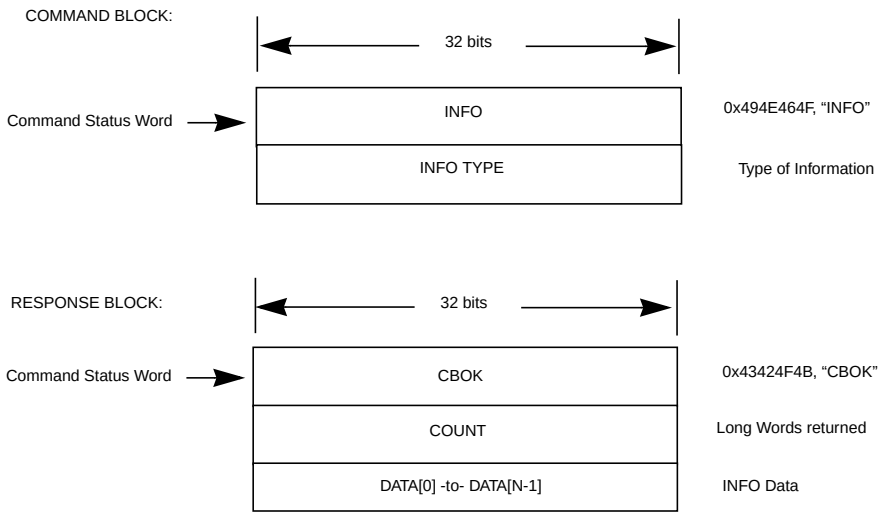


Figure 4-21. INFO Command and Response Format

FUNCTION

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

CODE

0x494E464F (ASCII "INFO")

PARAMETERS

INFO TYPE: Specifies which information the controller is to make 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 the Controller Info structure on page 6-30).

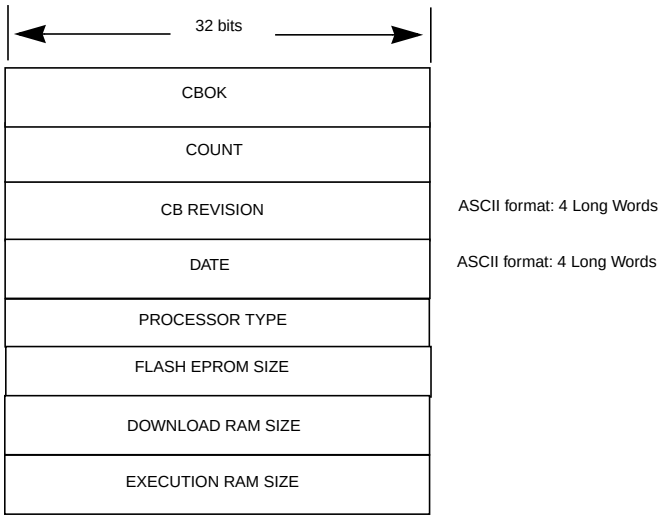


Figure 4-22. Information Structure Type 0 and Type 1

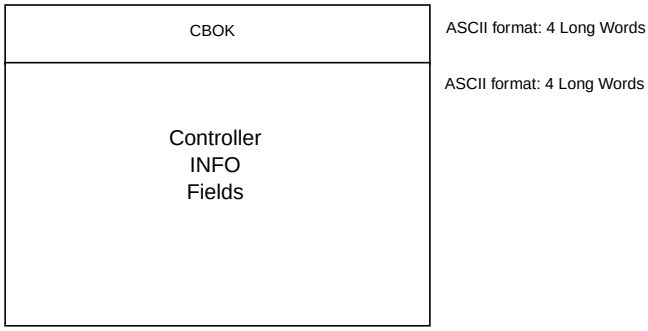


Figure 4-23. Information Structure Type 2

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)

Notes:

CHAPTER 5
5215 DIAGNOSTICS

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 (i.e., 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 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.
- 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 Test

Major Test 0x2 - Front End ATM Test

Major Test 0x3 - DMA Diagnostic Test

MEMORY MAP

Table 5-1 shows a memory map of the 5215 hardware:

Table 5-1. 5215 Hardware Memory Map

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 on-board 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 DIAGNOSTICS

Table 5-2 provides a quick-reference listing of the 5215 Common Boot major and minor diagnostic tests.

Table 5-2. Quick Reference to Common Boot Diagnostics

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, 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 Table 5-12 on page 5-29. 0x2E DIAG_ERROR is written to the CB Last Command Status, FAIL is written to the Command Status Word, followed by the appropriate error data

DIAG 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”.

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 5-1 shows the format of the DIAG command.

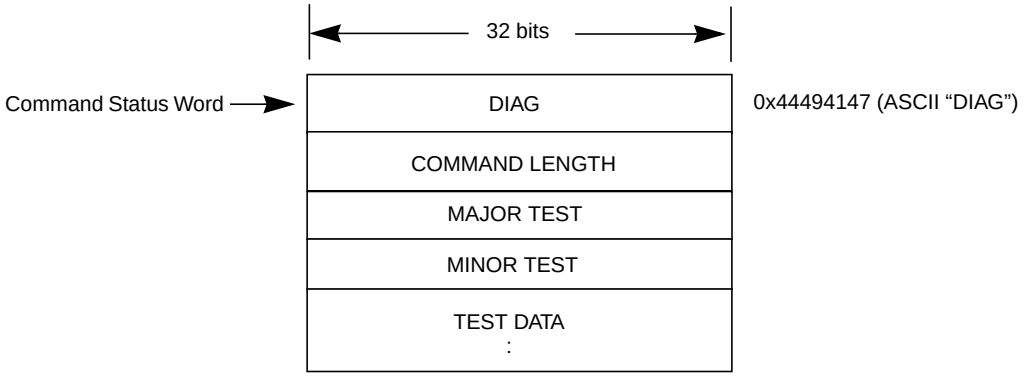


Figure 5-1. Format of the DIAG command

COMMAND FIELDS

The fields in the DIAG command are as follows:

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 Table 5-3 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.

NOTES

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

2. 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.

TEST DATA

Only VMEbus DMA tests require test-specific data. These fields are provided in the VMEbus DMA test description.

Table 5-3. Major Test Numbers

Code	Test Type
0x1	Memory Test
0x2	ATM Front-end
0x3	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. If the test fails, the Command Status Word will contain the ASCII value “FAIL” and the 5215 returns the structure shown in Figure 5-2.

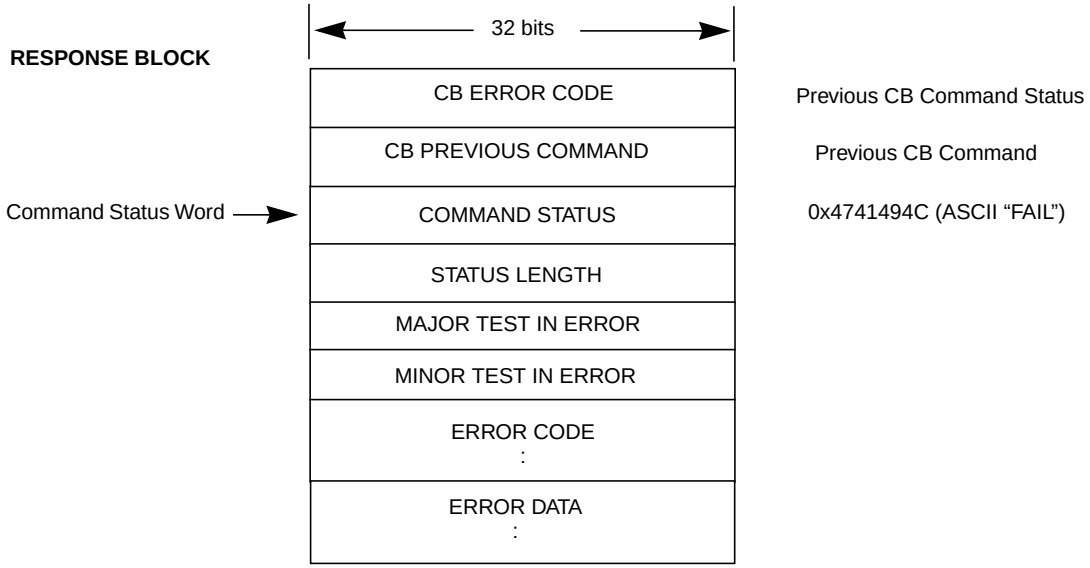


Figure 5-2. “FAIL” Returned Status Block

NOTE

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

The meaning of the fields in the resulting structure shown in Figure 5-2 is as follows:

CB Error Code

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

CB Previous Command

This reports the previously issued CB command which 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. Table 5-12 lists the diagnostic error codes. See Appendix B for listing 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).

Notes:

MEMORY DIAGNOSTICS

MEMORY TESTS — 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. There is another set of static RAMs used by the board referred to as SRAM. The size of this bank of memory is variable up to 1 Mbyte. On-board buffer space and on-board 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.

Memory test command example:

Name	value in hex
Length	0x10
Major	0x1
Minor	0x2
Options	0x0

Figure 5-3. Memory Test Command Format

MEMORY TEST OPTIONS

The Options field is shown in Figure 5-4:

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

Figure 5-4. Memory Test Command 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

Table 5-4 lists the minor tests supported by the memory diagnostic test.

Table 5-4. Minor Test Codes for Memory Tests

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 5-2 on page 5-4.

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 given below.

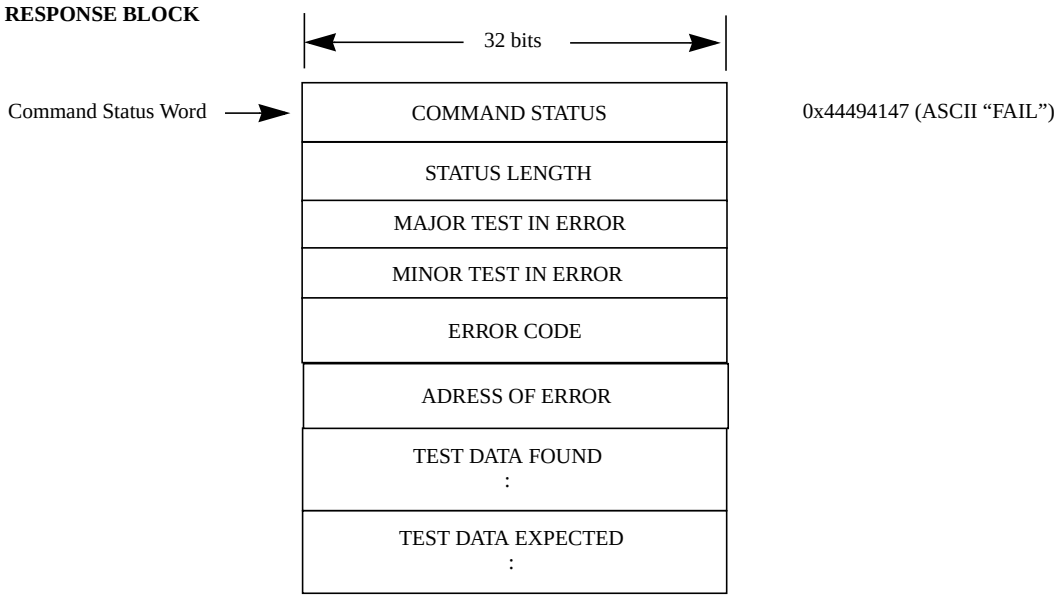


Figure 5-5. CB Diagnostic Error Data Structure

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 which 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, then return and check that each is correct. If no problem is discovered, a walking 1’s test is performed.

ATM FRONT-END DIAGNOSTICS

ATM FRONT-END DIAGNOSTICS — MAJOR TEST CODE 0x2

The format of the command block for the front-end diagnostics is shown in Figure 5-1 on page 5-3. The format of the response block is shown in Figure 5-2 on page 5-4.

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

There are six minor test codes that can be issued in conjunction with the front-end major test code 0x2. These codes are listed in Table 5-5.

Table 5-5. Minor Test Codes for Front-End Tests

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 an 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 which is fragmented by the F-FRED 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, FAIL is written to the Command Status Word. (See TEST RESULTS below).

Command example:

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, 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, FAIL is written to the Command Status Word. (See TEST RESULTS on page 5-14).

Command example:

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, 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, FAIL is written to the Command Status Word. (See TEST RESULTS on page 5-14).

Command example:

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, FAIL is written to the Command Status Word. (See TEST RESULTS on page 5-14).

Command example:

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, FAIL is written to the Command Status Word. (See TEST RESULTS on page 5-14).

Command example:

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 5-2 on page 5-4). 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, FAIL is written to the Command Status Word, followed by error data appropriate to the test failure.

VME DMA DIAGNOSTICS

VME DMA DIAGNOSTIC TEST — MAJOR TEST CODE 0x3

The following sections describe 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 Figure 5-6. The format of the response block is shown in Figure 5-8 on page 5-17.

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

Figure 5-6. Example DMA Diagnostic Command Format

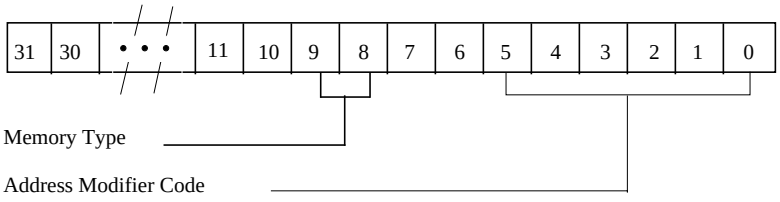
NOTE: Not all DMA diagnostic commands use all of the fields shown above.

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.

The format of the Transfer Options Word is shown below.



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

Figure 5-7. 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 Table 4-3:

Table 5-6. Memory Type for Data Transfers

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

Figure 5-8. Example DMA Diagnostic Response Block Format

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 are shown below. The addgen status registers provide information as part of the DMA diagnostic response block.

Table 5-7. Addgen General Status Register - High True Values

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

Table 5-8. Addgen IOP Status Register - High True Values

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

Table 5-9. MBC Interrupt Port - Low True Values

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

9	8	7	6	5	4	3	2	1	0
dboard hi-pri	rsvd	dboard lo-pri	rsvd	rsvd	rsvd	addgen iop	mmux mbox	addgen timer	vme iack

MINOR TEST CODES

Table 5-10 lists the minor test codes for the DMA Diagnostics.

Table 5-10. Minor Test Codes for DMA Diagnostic Tests

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 on-board buffer. Builds data according to user parameters.
0x6	WRITE BUFFER	Executes a VME read into the specified on-board 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.

Command example:

Name	value in hex
Length	0xC
Major	0x3
Minor	0x0

Figure 5-9. Power-up Test Command Format

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 5-16).

Command example:

Name	value in hex
Length	0x18
Major	0x3
Minor	0x2
Sys address	0x100000
DMA count	0x200
Transfer options	023D

Figure 5-10. Standard DMA Test Command Format

Response - see CB Error Codes in APPENDIX B. DMA Error Codes are shown in Table 5-12 on page 5-29**.**

0x3 - THIS TEST IS RESERVED

Test 0x3 is not supported by the 5215.

Response - 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)

Command example:

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

Figure 5-11. RAW DMA Test Command Format

Response - see CB Error Codes in APPENDIX B. DMA Error Codes are shown in Table 5-12 on page 5-29**.**

0x5 - READ BUFFER

Builds an on-board 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 5-16).

Data seed

Seed value for the data pattern to build.

Data options:

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

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

Figure 5-12. Bit Structure of the READ BUFFER Data Options

Buffer

This value is for the on-board buffer offset in 512 byte increments (i.e. if the value was 0x8, then the on-board 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 on-board 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 (e.g. burst control = 2 will cause a burst value of 32 transfers to be selected).

Reserved

This value should be set to 0x0.

Table 5-11. Data Pattern Values

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

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

Figure 5-13. Read Buffer Command Format

Response - See CB Error Codes in APPENDIX B. DMA Error Codes are shown in Table 5-12 on page 5-29**.**

0x6 - WRITE BUFFER

Executes a VME read DMA to the on-board 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 CB Error Codes in APPENDIX B. DMA Error Codes are shown in Table 5-12 on page 5-29**.**

0x7 - VIRQ

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

Parameters

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.

Command example:

Name	value in hex
Length	0x10
Major	0x3
Minor	0x7
IRQ options	0x0155

Figure 5-14. VIRQ Command Format

Response - See CB Error Codes in APPENDIX B. DMA Error Codes are shown in Table 5-12 on page 5-29**.**

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 (i.e. 10 = 1234).

Parameters

Selected ASIC

1 = Addgen
2 = Megamux
3 = MBC

Command example:

Name	value in hex
Length	0x10
Major	0x3
Minor	0x8
ASIC select	0x1

Figure 5-15. Dump ASIC Registers Command Format

Response - See CB Error Codes in APPENDIX B. DMA Error Codes are shown in Table 5-12 on page 5-29.

DMA DIAGNOSTIC ERROR CODES

The following table lists the VME DMA diagnostic error codes. See **APPENDIX B** for more information on Common Boot error codes:

Table 5-12. DMA Diagnostic 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

CHAPTER 6
HOST ADAPTER RING INTERFACE (HARI)

Note: The functions described herein require firmware revision A02 or later.

HOST ADAPTER RING INTERFACE FUNCTIONS

The VME ATM software interface is a series of channels which facilitate communication between the host system and the 5215 controller. 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 which:
    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. 1.) A **Short I/O Control Block**, which is a shared memory data structure between the host and controller, 2.) **Rings**, which contain a set of pointers to system memory locations, and 3.) **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.

CHANNEL TYPES

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

- 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 (see Figure 6-1):

- **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 is 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 is specified by the host at configuration and/or run-time.

Figure 6-1 illustrates the three basic blocks of the host adapter ring interface.

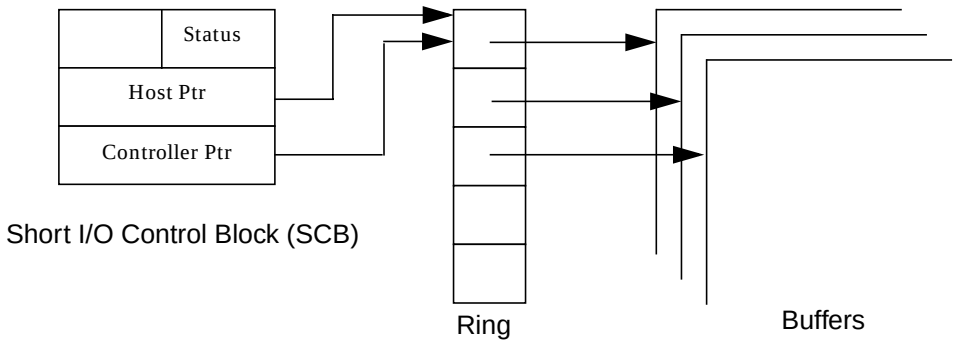


Figure 6-1. The Three Basic Blocks of the Host Adapter Ring Interface

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 6-2a.

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

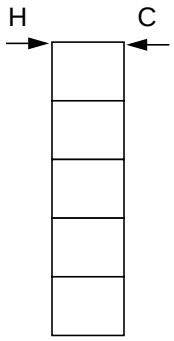


Figure 6-2a. Empty Ring

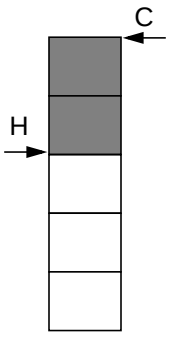


Figure 6-2b. Ring Partially Full

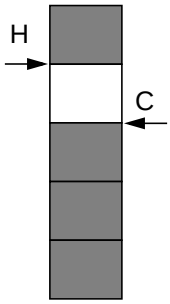


Figure 6-2c. Ring full and Wrapped

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

= Empty Ring Element

= Full Ring Element

Figure 6-2. The Three Basic Ring States

RING EMPTY STATE

When the ring is empty (Figure 6-2a) this 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 (See Figure 6-2b) this 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, then advance its pointer to indicate that the transfer is complete.

Figure 6-2b illustrates the condition where the host has added two items to be processed by the controller. Figure 6-2c 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 6-2c) and 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 in order 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 Short I/O. This field is initialized by the controller when the channel is configured by the host. Afterwards, both the host and controller maintain respective pointers which indicate ownership of the ring elements. Interrupt status and control is also interchanged here.

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

Figure 6-3. Channel Short I/O Control Block (SCB) Structure

FIELDS

The fields in the channel short I/O control structure are:

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 (i.e., all bits are cleared to zero), the controller can post an interrupt. When the interrupt timer field is set to ones (i.e., 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 which 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 in order 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 6-4 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 6-4. 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 which 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 (i.e. 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.

The control ring element structure is shown in Figure 6-5.

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

Figure 6-5. Control Ring Element Structure

FIELDS

The fields in the Control Ring Element Structure are:

EXEC/DMAC

These fields contain the Execution and DMA Control Words. See Figure 6-6 below and Figure 6-7 on page 6-10 for definition of the bits in these fields.

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 6-6. 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”, 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, 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 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 which does not have the EP bit set.

The DMA Control Word (DMAC) contains bit options which control VME transfers.

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

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

The DMAC Word 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 6-7. DMAC Word Definition

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 (i.e., 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 through 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 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. Eachdtat 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.

The structure is shown in Figure 6-8.

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

Figure 6-8. Data Send Ring Element Structure

FIELDS

The fields in the data send ring element structure are:

EXEC/DMAC

These fields contain the Execution and DMA Control Words. See Figure 6-9 below and Figure 6-10 on page 6-15 for definition of the bits in these fields.

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 6-9. 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”, 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, 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 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 which does not have the EP bit set.

The DMA Control Word (DMAC) contains bit options which control VME transfers.

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

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

The DMAC Word 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 6-10. DMAC Word Definition

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 (i.e., 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 through 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 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. See Figure 6-11 for a definition of the bits in this field.

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 6-11. 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 =11. When the packet type is 10, the entire ATM cell header is derived from the Virtual Circuit (VC) Table.

Bits 12, 7-0: Reserved

These bits are reserved and must be cleared to 0

Bit 11: EOM - End of Message

When bit 11 is set to 1, this 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

VCT Index

This field is a reference to the Virtual Circuit (VC) Table entry which 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 prior to reception of 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.

The structure is shown in Figure 6-8.

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 6-12. Data Receive Ring Element Structure

FIELDS

The fields in the data receive ring element structure are:

EXEC/DMAC

Execution and DMA Control Words (DMAC). See Figure 6-13 on page 6-19 and Figure 6-14 on page 6-21 for a definition of the bits in these fields.

The Execution Control Word (EXEC) 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 6-13. 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”, 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, 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 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 which does not have the EP bit set.

The DMA Control Word (DMAC) contains bit options which control VME transfers.

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

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

The DMAC Word 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 6-14. DMAC Word Definition

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 (i.e., 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 through 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 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 which 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 6-15. 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 that 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, i.e. 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 following section defines the command codes and structures issued via the control send channel. All of the command codes share a common “generic” header (see Figure 6-16 on page 6-27). 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), 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 the command status. The data to be sent is appended to this generic header. Figure 6-16 on page 6-27 shows the structure of the Control Send Command structure.

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

Figure 6-16. Control Send Generic Header Structure

FIELDS

The fields in the Control Send command structure are:

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 Figure 6-17 on page 6-28.

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 Figure 6-18 on page 6-29.

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

Figure 6-17. Control Send Command Codes

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
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

Figure 6-18. Control Send Status Codes

CONTROLLER INFO (0x00)

The Controller Info command is diagnostic in nature, and returns some basic operating parameters. After issuing the Controller Info command (0x00), the following information will be returned:

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

Figure 6-19. Controller Info Structure

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

version[16]	“5215 V/ATM X06 ” (reflects your current firmware version)
date[16]	“ 03/04/94 15:00 ” (date and time of the current firmware version)
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 the command status. See Figure 6-16 on page 6-27.

Version/Date

This field is a 32 byte ASCII string which 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

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 disable	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 disable	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.

Offset	Description
0x00	Generic Header

FIELDS

Generic Header

This field contains the command length, command code, and the command status. See Figure 6-16 on page 6-27.

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. Figure 6-20 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 6-20. Configure Control Send/Receive Channel Command Structure

FIELDS

The fields in the Control Send/Receive Channel command are:

Generic Header

This field contains the command length, command code, and the command status. See Figure 6-16 on page 6-27.

Send Channel Configuration Block

This block contains configuration information for the control send channel. See Figure 6-34 on page 6-52.

Receive Channel Configuration Block

This block contains configuration information for the control receive channel. See Figure 6-34 on page 6-52.

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. Figure 6-21 shows the configuration structure for the Configure VATM Data Channels command.

NOTE: This command is archaic, and although still supported, it is recommended to use the Configure Data Channels command (0x12) instead.

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 6-21. VATM Data Channel Command Structure

FIELDS

The fields in the VATM Data Channel command are:

Generic Header

This field contains the command length, command code, and the command status. See Figure 6-16 on page 6-27.

Fixed-Rate Send Channel Configuration Block

This field contains information about the configuration of the fixed-rate send channel. See Figure 6-34 on page 6-52.

Variable-Rate Send Channel Configuration Block

This field contains information about the configuration of the variable-rate send channel. See Figure 6-34 on page 6-52.

Small-Buffer Receive Channel Configuration Block

This field contains information about the configuration of the small-buffer receive channel. See Figure 6-34 on page 6-52.

Large-Buffer Receive Channel Configuration Block

This field contains information about the configuration of the large-buffer receive channel. See Figure 6-34 on page 6-52.

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 naximum transfer units (MTUs), one small (i.e. 256 bytes) and one large (i.e. 9180 bytes). The controller will return incoming packets to the appropriate channel based upon the size of the packet. A small packet will be returned in the large ring if no small buffer elements are available.

Offset	Description
0x00	Generic Header
0x0C	Send Channel Count
0x10	Receive Channel Count
0x14	Send Configure Block 1
	:
	Send Configure Block n
	Receive Configure Block 1
	Receive Configure Block 2

Figure 6-22. Configure Data Channels Command Structure

FIELDS

The fields in the Configure Data Channels command are:

Generic Header

This field contains the command length, command code, and the command status. See Figure 6-16 on page 6-27

Send Count

This field indicates the number of data send channels to configure. The maximum number is 8.

Receive Count

This field indicates the number of data receive channels to configure. The maximum number is 2.

Send Configure

This field specifies the configuration blocks for the send channel. See Figure 6-34 on page 6-52. This field is repeated SEND COUNT times.

Receive Configure

This field specifies the configuration blocks for the receive channel. See Figure 6-34 on page 6-52. 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, i.e., 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.

Offset	Description
0x0	Generic Header
0xC	Burst Count

Figure 6-23. Set VME Burst Count Command Structure

FIELDS

The fields in the Set VME Burst Count command are:

Generic Header

This field contains the command length, command code, and the command status. See Figure 6-16 on page 6-27.

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 EXEC/DMAC on page 6-8).

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.

In order 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.

Offset	Description
0x0	Generic Header
0xC	Channel Configuration Block

Figure 6-24. Configure Raw Receive Channel Command Structure

FIELDS

The fields in the Configure Raw Receive Channel command are:

Generic Header

This field contains the command length, command code, and the command status. See Figure 6-16 on page 6-27.

Channel Configuration Block

This field contains the Channel Configuration Block for this channel. See Figure 6-34 on page 6-52.

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 may wish to configure all table entries at initialization, or dynamically allocate and reuse on an as needed bases.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 A for further information on Circuit Management.

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 6-25. Configure Virtual Circuit Table Index Command Structure

FIELDS

The fields in the Virtual Circuit (VC) Table Index are:

Generic Header

This field contains the command length, command code, and the command status. See Figure 6-16 on page 6-27.

Index

This field specifies the starting index number.

Count

This field specifies the number of sequential indexes. Fields 0x08 through 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 6-26. 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 =11. When the packet type is 10, the entire ATM cell header is derived from the VC Table.

Bits 12, 7-0: Reserved

These bits are reserved and must be cleared to 0.

Bit 11: EOM - End of Message

When set to 1, 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

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:

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
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

Figure 6-27. Congestion Recovery Methods

Bits 11-9: Rate Queue.

This 3-bit field links the virtual circuit to one of eight different rate queues. These queues are:

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”

Figure 6-28. Rate Queue Selections

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.

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 6-29. Configure/Report Rate Queue Command Structure

FIELDS

The fields in the Configure/Report Rate Queue Command structure are:

Generic Header

This field contains the command length, command code, and the command status. See Figure 6-16 on page 6-27.

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 6-30. 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

Example of Preload Values:

The following table shows the preload values, based upon 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

Figure 6-31. Rate Queue Preload Values

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.

Offset	Description
0x0	Generic Header
0xC	Data

Figure 6-32. Write/Read Flash Sector Command Structure

FIELDS

The fields in the Write/Read Flash Sector are:

Generic Header

This field contains the command length, command code, and the command status. See Figure 6-16 on page 6-27.

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 to 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.

Offset	31.....24	23.....16	15.....8	7.....0
0x00	Generic Header			
0x0C	Flags			
0x10	rsvd 0	rsvd 1	oam_mask	oam_compare

Figure 6-33. Configure Congestion Control Structure

FIELDS

The fields in the Configure Congestion Control command are:

Generic Header

This field contains the command length, command code, and the command status. See Figure 6-16 on page 6-27.

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 2 resource allocations:

- where, in Short IO, to locate the channel's SCB

- where, in host memory, to locate the channel's ring

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 6-34. Channel Configuration Block (CCB) Structure

FIELDS

The fields in the Channel Configuration Block are:

SCB Pointer

This field specifies the Short IO Control Block (SCB) offset into Short IO. The controller will initialize the SCB fields upon acceptance of this structure. This field is masked by the firmware with the Short IO size, so the actual physical address may be supplied here if it accurately reflects the "offset" into Short IO.

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 EXEC/DMAC on page 6-8

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 channel configuration block flag bits.

7	6	5	4	3	2	1	0
0	0	0	0	0	RSE	TCM	RSP

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, this bit 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 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.

APPENDIX A
BOOT

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). Refer to **CHAPTER 4** for details on Common Boot.

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 then back to 0. The Hardware Control field is a 16-bit field located at the start of short I/O (i.e., 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, See Figure 4-3) 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 (8) transmit channels, two (2) receive channels, and one (1) raw/OAM cell receive channel; however, a minimum of one (1) send and one (1) 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.

NOTES

A. 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.

B. For host systems which can only perform D16 transfers to controller Short I/O space, rings must be allocated so as NOT to cross a 64K page boundary. This avoids 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 (8) programmable transmission rate queues, of which one (1) 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 "Configure Virtual Circuit Table Index(es)").

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, etc.

The host must provide the following parameters (using the Configure VC Table Index command) prior to sending and receiving packets on a virtual circuit:

1. Packet type (AAL3/4, AAL5, raw, OAM)
2. The 4 byte ATM cell header
3. FCS (CRC-32) enable/disable
4. Transmission Rate
5. 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 insure 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. The desired mode is specified at channel configuration time. All elements within the ring report completions in the same mode.

1. **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.

2. **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.

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 is set to one (1) is encountered.

NOTES

A. 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).

B. 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 insure 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 (2) receive channels, specifying a different MTU for each (i.e. 256 and 9180 bytes). The controller will return reassembled packets, based upon size, to the appropriate channel.

NOTES:

- 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 which 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, i.e., 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)
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. These three methods are mutually exclusive, and are enabled by setting their respective flag bit (bits 3-0).

1. 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.

2. **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.

3. OAM F5 Congestion Cell

The host may *define* what the first byte of an OAM F5 cell is to be recognized as. 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.

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 approbated conform to 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:

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

Figure A-1. Rate Queue Priorities

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:

7	6	5	4	3	2	1	0
GFC				VPI			
VPI				VCI			
VCI							
VCI				PTI			CLP
HEC							
48 byte payload							

GFC - Generic Flow Control
VPI - Virtual Path Identifier
VCI - Virtual Circuit Identifier
PTI - Payload Type Identifier
CLP - Cell Loss Priority
HEC - Header Error Check

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

Notes:

APPENDIX B
COMMON BOOT ERROR CODES

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 is a table that lists the Common Boot error codes.

Table B-1. Common Boot Error Status Codes I

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 is a table of CB Error Codes generated during the DNLD, EXEC and BURN commands.

Table B-2. Common Boot Error Status Codes II.

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.

FAIL STATUS BLOCK

	32-bit Entries	
Command Status Word (CB_START)	FAIL	0x4641494C, “FAIL”
	LENGTH	Byte length of valid information, including this longword
	MAJOR TEST	Defines main area being tested when test occurred
	MINOR TEST	Sub-defines the test program and area failure
	PARAMETER 0	Optional Information on failure
	:	
	PARAMETER N	

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 Test Numbers are shown in the table below.

Table B-3. 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 Test Numbers are shown in the table below.

Table B-4. 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 test numbers are shown in the table below.

Table B-5. 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 test numbers are shown in the table below.

Table B-6. 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

Table B-7. 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

Notes:

APPENDIX C
BASE ADDRESS JUMPER SETTINGS

Base Address Jumper Settings

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 Chapter 2.

Table C-1. Base Address Jumper Settings

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

Address	JA24	JA25	JA26	JA27	JA28	JA29	JA30
0x3200	In	In	Out	Out	In	In	Out
0x3400	In	In	Out	Out	In	Out	In
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

Address	JA24	JA25	JA26	JA27	JA28	JA29	JA30
0x7600	In	Out	Out	Out	In	Out	Out
0x7800	In	Out	Out	Out	Out	In	In
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

Address	JA24	JA25	JA26	JA27	JA28	JA29	JA30
0xBC00	Out	In	Out	Out	Out	Out	In
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

APPENDIX D
SAMPLE DATA STRUCTURES

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       */
};
```

INDEX

Numerics

A

B

C

D

E

F

N

O

P

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