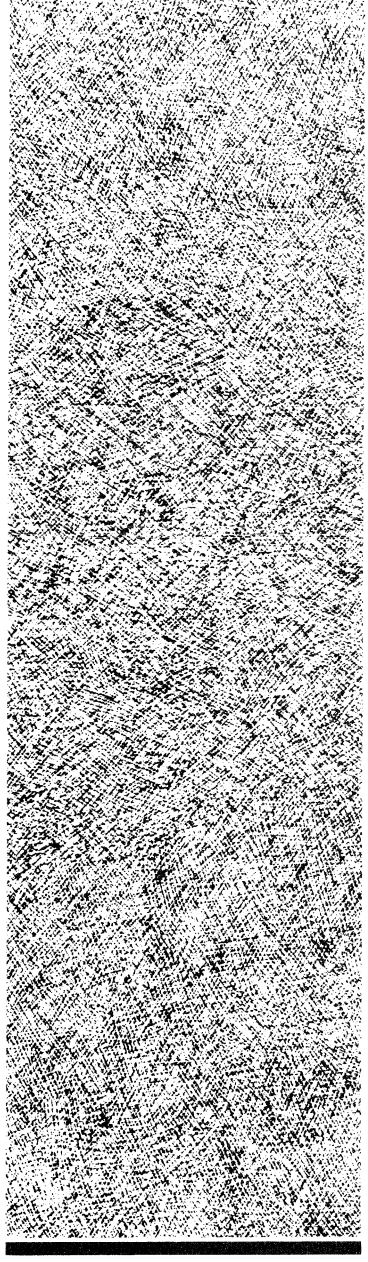
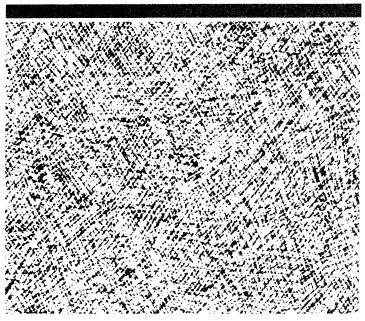
MVME327AFW/D1

MVME327A Firmware User's Manual



MOTOROLA



MVME327A FIRMWARE

USER'S MANUAL

(MVME327AFW/D1)

PREFACE

This manual describes the software interface between a host and the MVME327A firmware. This interface is accomplished over the VMEbus and allows communication between a process running on a host processor and the MVME327A firmware running on the onboard processor (MC68010). The firmware, in turn, allows communication between the host and the Small Computer System Interface (SCSI) bus, local floppy drives, or other onboard facilities.

The mechanism used to produce this interface is the Buffered Pipe Protocol (BPP). This manual defines channel headers, envelopes, and packets. It defines the fields in the packets, the format of device descriptor tables, the format of specific commands, the returned statuses, and the protocol necessary for the interface.

This manual is intended for anyone who wants to design OEM systems, supply additional capability to an existing compatible system, or in a lab environment for experimental purposes.

A basic knowledge of computers and digital logic is assumed.

To use this manual, you should be familiar with the publications listed in the *related documentation* paragraph in Chapter 1 of this manual.

First Edition August 1988

TABLE OF CONTENTS

Page

CHAPTER 1 - GENERAL INFORMATION

1.1 INTRODUCTION 1-1
1.2 FEATURES 1-1
1.3 GLOSSARY 1-2
1.4 SCSI BUS BACKGROUND 1-3
1.5 MVME327A FIRMWARE ARCHITECTURE 1-4
1.6 BUFFERED PIPE PROTOCOL 1-6
1.7 RELATED DOCUMENTATION 1-8
1.8 MANUAL TERMINOLOGY 1-8

CHAPTER 2 - COMMAND/STATUS REGISTER INTERFACE

2.1 INTRODUCTION 2-1
2.2 CSR COMMAND PROTOCOL 2-1
2.3 CSR COMMANDS 2-3
2.3.1 Create Channel CSR Command 2-3
2.3.2 Delete Channel Command 2-4
2.4 RETURN STATUS 2-4
2.4.1 CSR Status Register 2-4
2.4.2 SYSFAIL Handling 2-5
2.4.3 Diagnostic Register 2-5

CHAPTER 3 - BPP DATA STRUCTURES

3.1 INTRODUCTION 3-1
3.2 COMMAND CHANNEL HEADER STRUCTURE 3-1
3.3 ENVELOPE 3-3
3.4 PACKET 3-4

CHAPTER 4 - BUFFERED PIPE PROTOCOL

4.1 INTRODUCTION 4-1
4.2 ESTABLISHING DRIVER/MVME327A CHANNEL COMMUNICATIONS 4-1
4.2.1 Envelope/Packet Enqueueing 4-2
4.2.2 Envelope/Packet Dequeueing 4-3
4.2.3 Buffered Pipe Protocol Summary 4-3

TABLE OF CONTENTS (cont'd)

Page

CHAPTER 5 - OTHER DATA STRUCTURES

5.1 INTRODUCTION 5-1
5.2 DEVICE DESCRIPTORS 5-1
5.2.1 Disk Descriptor Table 5-1
5.2.2 Streaming Tape Descriptor Table 5-5
5.2.3 Start/Stop Tape Descriptor Table 5-7
5.3 SCATTER/GATHER LIST 5-9

CHAPTER 6 - MVME327A COMMANDS

6.1 INTRODUCTION 6-1
6.2 LOCAL FLOPPY COMMANDS 6-1
6.3 SCSI BUS COMMANDS 6-1
6.3.1 High Level Command Translation 6-1
6.3.2 SCSI Level Commands 6-2
6.3.3 SCSI Specific Packet 6-3
6.4 MVME327A COMMANDS 6-6

CHAPTER 7 - HIGH LEVEL COMMANDS

7.1 INTRODUCTION 7-1
7.1.1 BPP Test Command ($00) 7-1
7.1.1.1 BPP Test Command Packet 7-1
7.1.1.2 BPP Test Command Returned Status 7-2
7.1.2 Read Command ($01) 7-2
7.1.2.1 Read Command Packet 7-3
7.1.2.2 Read Command Returned Status 7-5
7.1.3 Write Command ($02) 7-6
7.1.3.1 Write Command Packet 7-6
7.1.3.2 Write Command Returned Status 7-7
7.1.4 Read Descriptor Command ($03) 7-8
7.1.4.1 Read Descriptor Command Packet 7-8
7.1.4.2 Read Descriptor Command Returned Status 7-9
7.1.5 Write Descriptor Command ($04) 7-10
7.1.5.1 Write Descriptor Command Packet 7-10
7.1.5.2 Write Descriptor Command Returned Status 7-12
7.1.6 Format Command ($05) 7-12
7.1.6.1 Format Command Packet 7-13
7.1.6.2 Format Command Returned Status 7-15
7.1.6.3 Defect List Formats 7-16

TABLE OF CONTENTS (cont'd)

	Page
7.1.7 Fix Bad Spot Command ($06)	7-17
7.1.7.1 Fix Bad Spot Command Packet	7-18
7.1.7.2 Fix Bad Spot Command Returned Status	7-19
7.1.8 Read Status Command ($10)	7-20
7.1.8.1 Read Status Command Packet	7-20
7.1.8.2 Read Status Command Returned Status	7-21
7.1.9 Load/Unload/Re-tension Command ($11)	7-22
7.1.9.1 Load/Unload/Re-tension Command Packet	7-22
7.1.9.2 Load/Unload/Re-tension Command Returned Status	7-23
7.1.10 Write Filemark ($12)	7-24
7.1.10.1 Write Filemark Command Packet	7-24
7.1.10.2 Write Filemark Command Returned Status	7-25
7.1.11 Rewind Command ($13)	7-26
7.1.11.1 Rewind Command Packet	7-26
7.1.11.2 Rewind Command Returned Status	7-27
7.1.12 Erase Command ($14)	7-28
7.1.12.1 Erase Command Packet	7-28
7.1.12.2 Erase Command Returned Status	7-29
7.1.13 Space Command ($15)	7-30
7.1.13.1 Space Command Packet	7-30
7.1.13.2 Space Command Returned Status	7-32
7.1.14 Enable Target Command ($20)	7-32
7.1.14.1 Enable Target Command Packet	7-33
7.1.14.2 Enable Target Command Returned Status	7-33
7.1.15 Disable Target Command ($21)	7-34
7.1.15.1 Disable Target Command Packet	7-34
7.1.15.2 Disable Target Command Returned Status	7-35
7.1.16 Reserve Unit Command ($22)	7-36
7.1.16.1 Reserve Unit Command Packet	7-36
7.1.16.2 Reserve Unit Command Returned Status	7-38
7.1.17 Release Unit Command ($23)	7-38
7.1.17.1 Release Unit Command Packet	7-39
7.1.17.2 Release Unit Command Returned Status	7-41
7.1.18 Reset SCSI Command ($25)	7-41
7.1.18.1 Reset SCSI Command Packet	7-42
7.1.18.2 Reset SCSI Command Returned Status	7-43
7.1.19 Custom SCSI Command ($26)	7-43
7.1.19.1 Custom SCSI Command Packet	7-44
7.1.19.2 Custom SCSI Command Returned Status	7-45
7.1.20 Self Test Command ($27)	7-46
7.1.20.1 Self Test Command Packet	7-46
7.1.20.2 Self Test Command Returned Status	7-47

TABLE OF CONTENTS (cont'd)

Page

7.1.21 Target Wait Command ($28) 7-48
7.1.21.1 Target Wait Command Packet 7-48
7.1.21.2 Target Wait Command Returned Status 7-49
7.1.22 Target Execute Command ($29) 7-50
7.1.22.1 Target Execute Command Packet 7-50
7.1.22.2 Target Execute Command Returned Status 7-51
7.1.23 Set SCSI Address Command ($2B) 7-52
7.1.23.1 Set SCSI Address Command Packet 7-52
7.1.23.2 Set SCSI Address Command Returned Status 7-53
7.1.24 Open Command ($2D) 7-53
7.1.24.1 Open Command Packet 7-54
7.1.24.2 Open Command Returned Status 7-55

APPENDIX A - MVME327A SUPPORTED SCSI CONTROLLERS/DEVICES A-1

APPENDIX B - CUSTOM SCSI COMMAND EXAMPLE B-1

APPENDIX C - TARGET ROLE C-1

APPENDIX D - FATAL ERROR CODES D-1

APPENDIX E - ADDITIONAL ERROR CODES E-1

APPENDIX F - C FUNCTION EXAMPLES OF BPP PROTOCOL F-1

INDEX IN-1

CHAPTER 1 - GENERAL INFORMATION

1.1 INTRODUCTION

This manual describes the software interface between a host and the MVME327A firmware. This interface is accomplished over the VMEbus and allows communication between a process running on a host processor and the MVME327A firmware running on the onboard processor (MC68010). The firmware, in turn, allows communication between the host and the Small Computer System Interface (SCSI) bus, local floppy drives, or other onboard facilities.

The mechanism used to produce this interface is the Buffered Pipe Protocol (BPP). This manual defines channel headers, envelopes, and packets. It defines the fields in the packets, the format of device descriptor tables, the format of specific commands, the returned statuses, and the protocol necessary for the interface.

1.2 FEATURES

The features of the MVME327A firmware interface include:

- BPP provides non-busy interface for real time and time critical applications.
- Interrupt driven or polled interfaces are provided by BPP.
- Virtual channels of the BPP support multiple host interface.
- Firmware preserves BPP channel priority when polling channels.
- Contains both a power up diagnostic self test and a self test upon command.
- Multiprocessing supported through RESERVE and RELEASE commands.
- SCSI multi-threading with full disconnect/reselect feature implementation is supported by firmware.
- Firmware provides unlimited command queuing for multiple SCSI devices.
- Incorporates intelligence to interface with some of the more popular SCSI drives.
- High level command set (for SCSI disks and streaming tapes) provides device independence for operating system drivers.
- SCSI firmware conforms to ANSI specification X3T9.2/82-2, Rev. 17B.

. SCSI Common Command Set (Revision 4B) is supported by high level command set.

. Custom SCSI command allows device dependent interface to any SCSI device.

. Target role supported through normal BPP channels for all eight peripheral logical units on the SCSI bus.

. Local floppy interface supports high density 1.2Mb formats with dual speed floppy drives.

1.3 GLOSSARY

The following list is a glossary of terms used throughout this manual.

Host - a computer system which dispatches and/or accepts commands across the VMEbus.

Initiator - a SCSI device that requests an operation to be performed by another SCSI device.

Logical Unit - a physical or virtual device addressable through a target.

Logical Unit Number - An encoded 3-bit identifier for the logical unit.

LUN - Logical unit number.

LSW - Least significant word.

MSW - Most significant word.

SCSI Address - The binary representation of the unique address assigned to a SCSI device.

SCSI Device - A host computer adapter, peripheral controller, or an intelligent peripheral (embedded controller) that can be attached to a SCSI bus.

SCSI ID - The bit significant representation of the SCSI address referring to one of the signal lines DB 7-0.

Target - A SCSI device that performs an operation requested by an initiator. Each target can have up to eight peripheral (LUN) devices associated with it.

Thread - The path that exists between an initiator memory and a SCSI device LUN.

User - The operator of a host system that accesses the MVME327A.

1.4 SCSI BUS BACKGROUND

The SCSI bus is a general purpose 8-bit parallel bus with standardized transfer protocol and a wide vendor support base. SCSI devices (any device on the SCSI bus) may gain access to the bus during the arbitration phase and request services to be performed through a "standardized" command set. (The arbitration phase allows one SCSI device to gain control of the SCSI bus so that it can assume the role of initiator or target.) A SCSI device that requests services from another SCSI device is defined as the initiator. A SCSI device that performs the services requested by an initiator is defined as the target. The MVME327A is a SCSI device that is capable of both the initiator and target roles on the SCSI bus. When accessing SCSI disk and tape formatters for mass storage transfers, the MVME327A is the initiator. When buffering commands and messages from SCSI devices for VMEbus MPU modules, the MVME327A is the target.

Up to eight SCSI devices may reside on the SCSI bus at the same time. Each of these devices is assigned a unique SCSI address of 0 through 7. During the arbitration phase, the SCSI device with the highest address has the highest priority on the bus (if device 7, device 6, and device 0 arbitrate for the bus at the same time, device 7 wins the arbitration).

A logical thread is a standardized communication between one initiator and one target. Only one logical thread may be physically threaded on the bus at any given time. Logical threads become physically threaded during the selection phase on the SCSI bus when the initiator selects the desired target. As soon as the target responds to selection and the initiator terminates the selection phase, information transfer phases are performed under the control of the target device. After the information transfer phases, logical threads are either completed or are broken by disconnection. Threads that are disconnected later resume when the target device reselects the initiator and identifies itself with the established SCSI protocol. If enabled by the initiator, the target device on the SCSI bus determines how many disconnections and reselections are to occur for any logical thread. There exists no limit on the number of disconnections and reselections that may occur during the processing of an initiator command (request).

A single initiator may have more than one logical thread outstanding on the SCSI bus (one physically threaded and others disconnected pending reselection). This ability of an initiator to dispatch more than one logical thread on the SCSI bus is referred to as multi-threading. Multi-threading is only possible on the SCSI bus when the disconnect/reselect feature is enabled. The initiator controls whether the disconnect/reselect feature is enabled during the message-out phase immediately after the selection phase. (Bit 6 of the identify message, if set, enables the target to disconnect and later reselect the initiator.)

1.5 MVME327A FIRMWARE ARCHITECTURE

The MVME327A VMEbus to SCSI bus adapter architecture and design are optimized for the VMEbus BPP interface to the SCSI bus and local floppy interface. Hardware and firmware features are designed to achieve low overhead in accessing SCSI devices from the VMEbus. The local floppy interface is a secondary, low performance-oriented feature and does not require fast access. The SCSI interface is allowed to operate unimpeded by local floppy activity.

The SCSI bus, as seen from the initiator's viewpoint, is really sequential in nature--only one logical thread may be initiated at a time. Because multiple SCSI threads can only be initiated one at a time, commands from the BPP interface directed toward SCSI devices are processed sequentially on the MVME327A. Sequential processing relieves the need for a multitasking manager that would be required if concurrent processing was chosen for this application. The concept of first-in-first-out is preserved when translating a BPP request into SCSI requests. There is no guarantee, however, that the SCSI target devices will complete their SCSI requests in such a manner that order is preserved going back to the BPP interface. In other words, if a single BPP channel contains three ordered requests for SCSI devices A, B, and C (A before B before C), the MVME327A dispatches requests first to device A, then to device B, then to device C. Because multi-threading is supported on the MVME327A, there is no known order in which the requests are completed and returned (in the form of completion status) to the MVME327A. Status is also returned to the BPP interface in a first-back-first-out manner. No effort is made to order requests to and from the BPP interface.

Channel priority is preserved for multiple BPP channels to the MVME327A. Higher priority channels are emptied of requests before lower priority channels. The concept of sequentiality applies only on a channel-by-channel basis. That is, each BPP channel is serviced on a first-in-first-out basis.

Without concurrent processing requirements, the firmware for the MVME327A is structured in a manner to lower the overhead associated with process switches. The MC68010 microprocessor supports priority interrupt processing. This hardware feature is a main influence in the structuring of the firmware.

The SCSI interface has the highest priority on the MVME327A. The SCSI interface is treated as two processes for the MVME327A firmware. Process number 1 is the DMA interface between the MVME327A SCSI FIFO buffers and the VMEbus. Process number 2 is the SCSI bus interface to the MVME327A SCSI FIFO buffers. Process number 1 has a higher priority than process number 2 because DMA block switches must be capable of interrupting SCSI bus transfers to implement scatter/gather (future feature). Consequently, the DMA interface interrupt is assigned to level 5 and the SCSI Bus Interface Chip (SBIC) interrupt is assigned to level 4.

Because all inputs to the MVME327A come through the BPP interface to the VMEbus, and because the SCSI bus interface is not to be pre-empted by floppy activity, the BPP interface (process number 3) has the next higher priority on

the MVME327A. Completing SCSI activity is considered more important than initiating new SCSI activity, so the BPP interface interrupt (refer to attention interrupt in the CSR space control register) has priority level 3, which is lower than the SBIC interrupt priority.

The local floppy drive interface (process number 4) has lower priority than the SCSI bus interface. Because floppy activity is not to interfere with SCSI activity, the floppy interface is made lower in priority than the BPP interface, otherwise the floppy interrupts would interfere with SCSI traffic. Therefore, the floppy interface interrupts are assigned to priority level 2. The floppy interface consists of two 1K FIFO buffers, control logic, an isolation buffer, and the floppy disk subsystem chip (WD37C65).

Floppy operations are supervised with operation time out interrupts. The WD37C65 does not dedicate any of its silicon for the supervisory role necessary for proper floppy operations. In other words, if a command is issued to the WD37C65 to perform some function on the floppy drives, no interrupt is generated if the floppy interface contains some exceptional condition. To accommodate this shortcoming in the chip, the MVME327A provides a timer to supervise floppy activity. Whenever a floppy command is issued to the chip, the timer is programmed to interrupt the MC68010 in a predetermined time out period. If this time out occurs and there is no outstanding floppy interrupt pending, then the floppy operation is aborted and an error results.

In order to keep a reasonably accurate time, the timer interrupt must supercede any other processing on the MVME327A. This is achieved by placing a high priority on the timer interrupt (level 6). It is important to note that the timer interrupt occurs only under three conditions that do not interfere with SCSI traffic (because the amount of time spent in the level 6 interrupt handler is less than 50 microseconds and this interrupt handler is not entered very often). First, if the event programmed in the WD37C65 does not take place before the time out period, the timer interrupt sets an error flag and semaphore for the floppy firmware and then stops the timer; the error flag causes the floppy firmware (running at a lower priority than the SCSI interface and the BPP interface) to detect the error condition and act accordingly. Second, if the traffic on the SCSI bus and BPP interface does not allow the processor interrupt mask to be lowered below level 2, then the floppy interface interrupt handler does not have a chance to cancel the timer interrupt; this condition sets the floppy semaphore, but does not set the error flag because a floppy interrupt is detected but not yet processed. Third, if the floppy firmware does not get any new commands for the floppy interface within a predetermined amount of time, then a MOTOR ON watchdog timer interrupt causes the floppy motors to be turned off.

In summary, the MVME327A firmware is structured around the priority interrupt hardware feature of the MC68010 to lower process switch overhead. No supervisory process management is required because a sequential nature of the SCSI bus allowed the MVME327A to process incoming requests for the SCSI bus sequentially and to process incoming status from the SCSI bus sequentially.

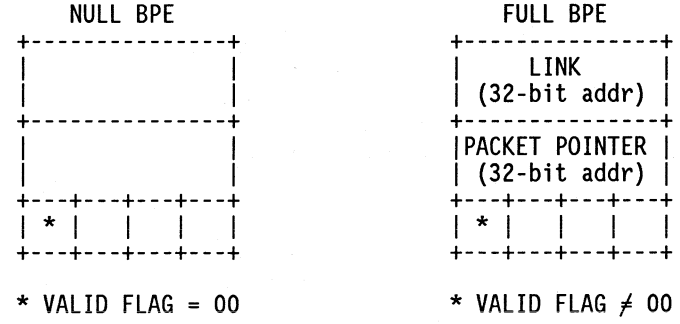
Furthermore, the local floppy interface is designed not to interfere with SCSI traffic by assigning its interrupts to a lower priority than the BPP interface and the SCSI bus interface interrupts. Although a high priority timer interrupt is dedicated for floppy timing functions, minimal interference on the SCSI bus traffic is introduced by the timer interrupts.

1.6 BUFFERED PIPE PROTOCOL

The BPP is a Motorola standard for a communications mechanism between two processors which can access a common address space in system memory. Conceptually, information is passed as fixed length messages through pipes shared by two processors. Each pipe serves as a one-way FIFO communication path and is able to hold any number of messages.

The pipe is implemented as a queue, in which entries are connected as a forward linked list. The queue is constructed such that it never becomes empty, and therefore requires no observation by the communicating processors prior to accessing the entries. The technique involves keeping a dummy entry at the end of the queue at all times. The receiver never dequeues the dummy entry. The sender adds to the queue by first adding a new dummy entry and then using the old dummy entry to hold useful information. Because the dummy entry buffers the queue from going empty, the data structure which contains the messages is called a buffered pipe.

Command packets are not queued directly in the queue. Instead, a data structure called a Buffered Pipe Envelope (BPE) is enqueued, and this envelope is either flagged as NULL, or contains a valid packet pointer, a forward link pointer, and a flag, as shown below.

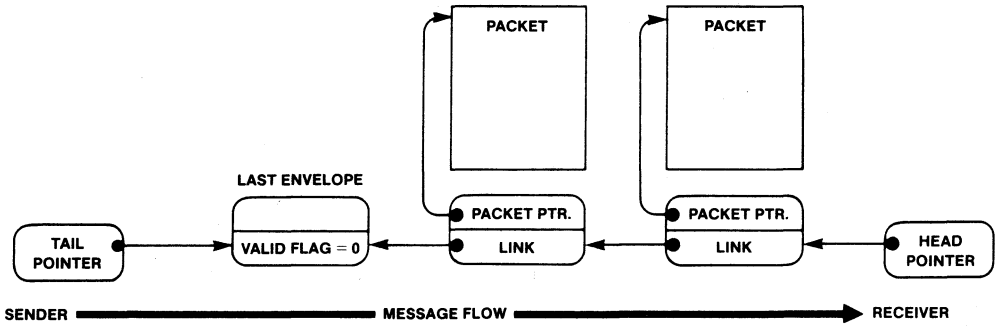


If the envelope is full, the packet pointer field contains a valid pointer. The length is fixed at 32 bits and is a pointer to a command packet data structure.

If the valid flag is nonzero, then the envelope is full and may be dequeued, and LINK is the 32-bit start address of the next BPE in the queue.

If the valid flag is zero, then the envelope is NULL (i.e., it is a dummy entry) and must not be dequeued; this entry marks the end of the queue.

It is important to keep the concept seen by the user of the buffered pipe distinct from its physical implementation as a queue. What the user of the interface sees is a virtual data structure called a buffered pipe into which can be put a pointer; the pipe is sometimes empty. The physical implementation, as shown below, is a queue of BPEs; there is always at least one BPP in the queue, and the last BPE on the queue is always flagged as NULL.



The BPP does not define how the pipes are physically implemented, except that they must reside in a region of RAM shared between the communicating processors. Also, the system design must provide some means for the host processor to transmit the initial TAIL and HEAD pointers to the slave processor when a new queue is established.

The BPP specification allows both interrupt driven and polled modes of operation for the receiver to get messages from the pipe. Also, even though messages are received in the order that are sent, the receiving processor may process the messages in any useful order.

The protocol in no way restricts the content of the information sent. The messages in the MVME327A firmware are pointers to command packet data structures.

1.7 RELATED DOCUMENTATION

The following publications may provide additional helpful information. If not shipped with this product, they may be purchased from Motorola Literature Distribution Center, 616 West 24th Street, Tempe, AZ 85282; telephone (602) 994-6561.

DOCUMENT TITLE	MOTOROLA PUBLICATION NUMBER
MVME327A VMEbus to SCSI Bus Adapter with Floppy Interface User's Manual	MVME327A

NOTE: Although not shown in the above list, each Motorola MCD manual publication number is suffixed with characters which represent the revision level of the document, such as "/D2" (the second revision of a manual); supplement bears the same number as the manual but has a suffix such as "/A1" (the first supplement to the manual).

The following publications are available from the sources indicated.

SCSI Small Computer Systems Interface; draft X3T9.2/82-2 - Revision 17B; Computer and Business Equipment Manufacturers Association, 311 First Street, N.W., Suite 500, Washington, D.C. 20001

Common Command Set (CCS) of the Small Computer System Interface (SCSI) X3T9.2/85-52 - Revision 4B; Computer and Business Manufacturer's Association, 311 First Street, N.W., Suite 500, Washington, D.C. 20001

1.8 MANUAL TERMINOLOGY

Throughout this manual, a convention has been maintained whereby data and address parameters are preceded by a character which specifies the numeric format as follows:

$	dollar	specifies a hexadecimal number
%	percent	specifies a binary number
&	ampersand	specifies a decimal number

Unless otherwise specified, all address references are in hexadecimal throughout this manual.

An asterisk (*) following the signal name for signals which are level significant denotes that the signal is true or valid when the signal is low.

An asterisk (*) following the signal name for signals which are edge significant denotes that the actions initiated by that signal occur on high to low transition.

In this manual, assertion and negation are used to specify forcing a signal to a particular state. In particular, assertion and assert refer to a signal that is active or true; negation and negate indicate a signal that is inactive or false. These terms are used independently of the voltage level (high or low) that they represent.

CHAPTER 2 - COMMAND/STATUS REGISTER INTERFACE

2.1 INTRODUCTION

This chapter describes the interface between the host CPU and the MVME327A. The interface consists of a set of basic commands sent through the Command/Status Register (CSR) interface and a set of commands sent in packets via virtual channels. The CSR commands establish and maintain the virtual channels for host/MVME327A communications. After one or more virtual channels are created, all other commands are transmitted between the host CPU and the MVME327A on these channels. A virtual channel consists of a channel header structure which defines the attributes of the channel, and two pipes. On one pipe, the command pipe, the host CPU enqueues command packets for the MVME327A. On the other pipe, the status pipe, the MVME327A enqueues status packets for the host. A pipe is simply a queue or singly linked list with one CPU manipulating the head pointer and another CPU manipulating the tail pointer. This interface is called a non-busy interface because, once the channels are established, the host CPU never finds the interface to be in a busy state. This means that the host never has to wait to send a command.

2.2 CSR COMMAND PROTOCOL

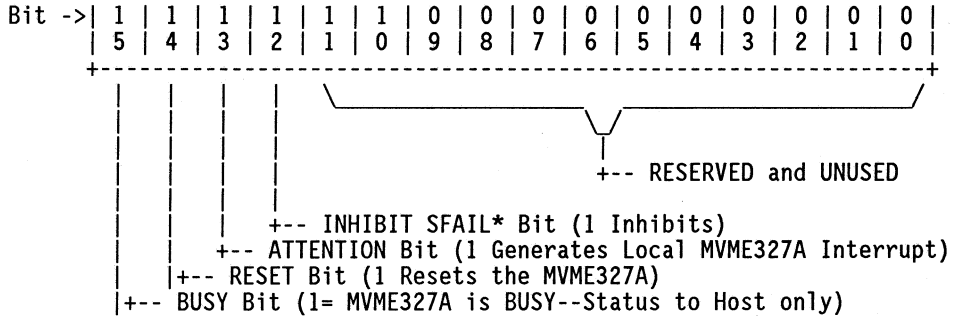
This paragraph contains a discussion of the protocol followed by a driver when issuing a CSR command to the MVME327A.

The CSR space is accessible from the VMEbus as well as from local RAM space. This permits both the host CPU and the MVME327A CPU to read and write any location in this CSR space. The table below shows the MVME327A CSR space as viewed by the VME system bus driver software. The MVME327A control register and the MVME327A TAS register are registers in this space which are used when CSR commands are issued to the MVME327A.

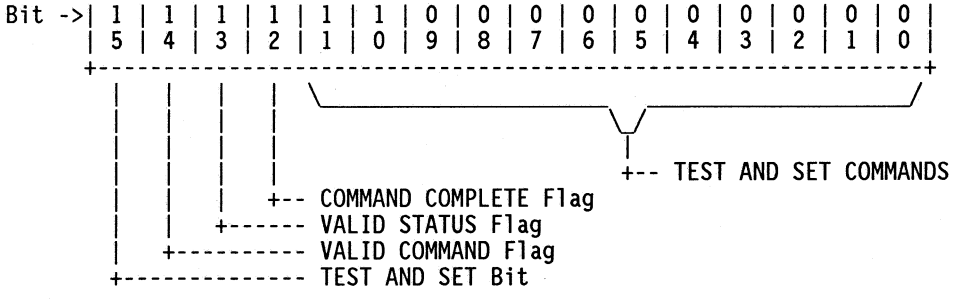
BYTE OFFSET	PARAMETER DESCRIPTION
$00	MVME327A address register (MSW)
$02	MVME327A address register (LSW)
$04	MVME327A address modifier / MVME327A data bus width
$06	MVME327A control register
$08	MVME327A status register / MVME327A diagnostic register
$0A	not used
$0C	not used
$0E	TAS register

NOTE: Byte offset from MVME327A VME system base address.

The bit definitions for the MVME327A control register and the MVME327A TAS register are shown in the following figures:



MVME327A Control Register at Offset $06



MVME327A TAS Register at Offset $0E

The protocol to be followed when issuing a CSR command to the MVME327A is as follows:

a. Driver checks that the BUSY bit in the CSR control register is clear. If the BUSY bit is set, the driver must wait until it is cleared to continue this handshaking with the MVME327A.

b. Driver gains access to the MVME327A CSR space by using the test and set instruction on the MVME327A TAS register TAS bit (bit 15). When TASing of this bit indicates that the bit was clear prior to the TAS the driver has "possession" of the CSR space.

c. Driver writes the CSR command op-code in the MVME327A TAS register (bits 0 through 11), MVME327A address register, MVME327A address modifier register, and MVME327A data bus width register.

d. Driver sets the VALID COMMAND flag in the MVME327A TAS register (bit 14).

e. Driver generates an attention bit interrupt on the MVME327A by setting bit 13 of the MVME327A control register.

f. Driver polls the VALID STATUS flag until set.

g. Driver reads the MVME327A status register.

h. Driver sets the COMMAND COMPLETE flag (bit 12 of the MVME327A TAS register).

i. Driver generates another attention bit interrupt on the MVME327A.

2.3 CSR COMMANDS

Initial communications between a driver and the MVME327A firmware are performed using the MVME327A CSR space. When channels are created there is generally no need for the host and MVME327A to communicate using the CSR commands.

CSR Test And Set Commands

COMMAND FIELD	TEST AND SET COMMAND
$000	Not Used
$001	Create Channel
$002	Delete Channel
$003-$FFF	Reserved

The table above lists the commands that are issued through the CSR space.

2.3.1 Create Channel CSR Command

Channels consist of a command pipe, a status pipe, and a channel header. Channel command packets are shipped by a driver to the MVME327A via command pipes. Channel status packets are returned to drivers via status pipes.

To create a channel a driver builds a channel structure and informs the MVME327A of the existence of the structure. It does this by issuing a "create channel" CSR command to the MVME327A.

The parameters passed with this CSR command are the address of the channel header structure, the address modifier associated with the channel structure, and the data bus width. These are placed in the MVME327A address register, the MVME327A address modifier register, and the MVME327A data bus width register, respectively.

Upon successful execution of the CSR command, the MVME327A firmware adds the channel to its internally kept list of existing channels and assigns a unique channel number which is used if the channel is later deleted. This channel number is written in the channel header by the MVME327A firmware. Refer to the table in the *CSR Status Register* paragraph in this chapter for the possible returned command status values in response to a "create channel" command. Up to 255 channels may be created.

2.3.2 Delete Channel Command

The delete channel command is used to free up resources when a channel is not needed for further communications. It MUST be used to delete a channel if the memory, pipes, etc., associated with the previously active driver are deallocated by the driver memory manager. Failure to do this may result in unrecoverable errors because the MVME327A firmware continues to poll the command pipe of a previously existing channel. The command pipe head pointer and status pipe tail pointer could be pointing to memory which may now be reused for another task.

To delete a channel a driver issues a "delete channel" CSR command to the MVME327A.

The parameters passed with this CSR command are the address of the channel header structure, the address modifier associated with the channel structure, and the data bus width. These are placed in the MVME327A address register, the MVME327A address modifier register, and the MVME327A data bus width register, respectively.

Refer to the table in the *CSR Status Register* paragraph in this chapter for the possible returned command status values in response to a "delete channel" command.

2.4 RETURN STATUS

The following paragraphs discuss the return status.

2.4.1 CSR Status Register

The CSR status register is used to return the status of the a CSR command. The status register is valid when the VALID STATUS bit is set in the CSR TAS register. A status of zero signifies correct command completion and nonzero status indicates an error. If a catastrophic failure occurs, the MVME327A asserts SYSFAIL and inserts an MVME327A error code into the CSR status register. The table below lists the possible status values issued in response to a CSR command.

CSR Status Register Values

VALUE	ERROR
$00	Correct Command Completion
$01	Invalid Command
$02	Could Not Read Channel Header
$03	Could Not Write Channel Header
$04	VME DMA Read failed
$05	VME DMA Write failed
$06	No more free channels
$07	Channel does not exist
$AA	Local Address Error
$AC	AC Fail Condition
$BB	Local Bus Error
$CC	Confidence Test Error
$EE	Unexpected Exception Error

NOTE $AA through $EE are catastrophic errors.

2.4.2 SYSFAIL Handling

SYSFAIL is asserted on MVME327A power up and remains set until the MVME327A has completed its internal diagnostics and initialization sequence. If the MVME327A suffers a non-recoverable internal error during operation, SYSFAIL is asserted and the cause of the failure is posted in the CSR status register. The MVME327A must be reset and reinitialized before operation can be resumed.

2.4.3 Diagnostic Register

The diagnostic register is used to return test completion and error codes for the power up confidence tests and for host invoked diagnostics. The bit definitions for the status byte in the diagnostic register are listed below.

Bit 0 - MPU Failure

If bit 0 is set, then the MPU has failed to execute instructions properly.

The MPU test loads the registers with test values, performs various operations, and checks that the results are correct.

Bit 1 - RAM Failure

If bit 1 is set, then the onboard RAM has failed.

The power up test uses a limited set of test patterns for its checks. The self test command does a more extensive set of tests. If the test passes, then the contents of the RAM is the same as before the self test was initiated.

Bit 2 - ROM Failure

If bit 2 is set, then the onboard ROM has failed.

The ROM test does an exclusive-OR checksum of the entire ROM and compares the result to 0. The actual checksum of the ROM is stored in the last 2 bytes of ROM.

Bit 3 - MAP Decoder Failure

If bit 3 is set, then the self test has not been able to access the PI/T, module status register, SCSI controller, the bus interface module, or the floppy disk controller.

Bit 4 - IRQ Failure

If bit 4 is set, then the IRQ has failed the test.

The IRQ test checks for the correct level and vector from the PI/T timer, the data channel controller, the SCSI interface, the CSR attention bit, the SWI3 bit, the floppy disk controller, and the SWI1 bit.

Bit 5 - Timer Failure

If bit 5 is set, the timer has failed the test.

The power up self test does not check the timer prescaler, but the IPC self test does.

Bit 6 - SCSI Failure

If bit 6 is set, the SCSI interface has failed.

The SCSI interface chip (but not the SCSI bus) is reset and the proper status and interrupts are checked.

Bit 7 - Reserved.

CHAPTER 3 - BPP DATA STRUCTURES

3.1 INTRODUCTION

This chapter describes and illustrates the data structures associated with the Buffered Pipe Protocol (BPP). They are the channel header, envelope, and packet.

3.2 COMMAND CHANNEL HEADER STRUCTURE

A single channel is composed of a command pipe and a status pipe. Each of these pipes is composed of a head pointer and a tail pointer that point to envelopes. Pipes must always contain at least one envelope (called the NULL envelope) in which case both the head and tail pointer point to the same envelope. A NULL envelope is an envelope that has a NULL link pointer. The following represents the channel structure expected by the MVME327A:

Command Channel Header Structure

BYTE OFFSET	PARAMETER DESCRIPTION	
$00	command pipe head pointer (MSW)	
$02	command pipe head pointer (LSW)	
$04	command pipe tail pointer (MSW)	
$06	command pipe tail pointer (LSW)	
$08	status pipe head pointer (MSW)	
$0A	status pipe head pointer (LSW)	
$0C	status pipe tail pointer (MSW)	
$0E	status pipe tail pointer (LSW)	
$10	interrupt level	interrupt vector number
$12	channel priority	address modifier
$14	channel number	valid flag
$16	data bus width	reserved

OFFSET	DESCRIPTION
$00	Command Pipe

The command pipe structure consists of two 32-bit pointers. The first pointer points to the envelope at the head of the command pipe. The second pointer points to the envelope at the tail of the command pipe which is always a NULL envelope.

$08 Status Pipe

The status pipe structure consists of two 32-bit pointers. The first pointer points to the envelope at the head of the status pipe. The second pointer points to the envelope at the tail of the status pipe which is always a NULL envelope.

$10 Interrupt Level

This interrupt level is the level at which the MVME327A interrupts the host CPU when execution of a command packet is complete.

If the interrupt level is zero, no interrupts are issued by the MVME327A for this channel and the host CPU must poll the status pipe to determine when a status packet has been returned.

$11 Interrupt Vector

This is the interrupt vector used by the MVME327A for all interrupts issued by the MVME327A to the host CPU for this channel.

$12 Channel Priority

This priority is used by the MVME327A when it polls all active channels looking for valid command packets. The highest priority is $00 and the lowest priority is $FF.

$13 Address Modifier

This address modifier is used by the MVME327A when accessing all envelopes and packets on the VMEbus for this channel.

$14 Channel Number

This is a unique channel number assigned by the MVME327A at the time a channel is created.

$15 Valid Flag

This flag is set by the MVME327A to a nonzero value when the channel is created to indicate to the host that the MVME327A now recognizes this channel and is ready to accept command packets on it.

$16 Data Bus Width

This field is not used by the MVME327A.

$17 Reserved -

This field is reserved.

3.3 ENVELOPE

The envelope format is as shown in the table below.

Envelope Format

BYTE OFFSET	PARAMETER DESCRIPTION
$00	link pointer (MSW)
$02	link pointer (LSW)
$04	MVME327A command/status packet pointer (MSW)
$06	MVME327A command/status packet pointer (LSW)
$08	valid flag / reserved
$0A	reserved

OFFSET	DESCRIPTION
$00	Link The link field points to the next envelope in the pipe.
$04	Packet Pointer This 32-bit field points to the packet which contains the information specific to the command being issued.
$08	Valid Flag This 8-bit field is the last field in an envelope to be modified. A zero value indicates that the envelope is INVALID, a nonzero value indicates that the envelope is VALID. The valid flag should only be set by a driver when a packet contains a valid command to be processed by an MVME327A. At all other times the valid flag is cleared. The host CPU should never enqueue an envelope on a command pipe with the valid flag set to a nonzero value unless all other data in the envelope and packet is valid.
$09-$0B	Reserved.

3.4 PACKET

The packet format is as shown in the table below.

MVME327A Packet Format

BYTE OFFSET	PARAMETER DESCRIPTION	
	Command Parameters	
$00	command	command control
$02	device type	unit number
$04	reserved	
$06	address modifier	data bus width
$08	primary address (MSW)	
$0A	primary address (LSW)	
$0C	secondary address (MSW)	
$0E	secondary address (LSW)	
$10	transfer count (MSW)	
$12	transfer count (LSW)	
$14	scatter/gather count	
$16	command parameter 1	
$18	command parameter 2	
$1A	command parameter 3	
	Status Parameters	
$1C	fatal error code	recovered error status
$1E	additional error code/status	
$20	retry count	reserved
$22	error status address (MSW)	
$24	error status address (LSW)	
$26	termination transfer count (MSW)	
$28	termination transfer count (LSW)	
$2A	status parameter 1	
$2C	status parameter 2	
$2E	status parameter 3	

The following are descriptions of the command channel packet parameters:

OFFSET — DESCRIPTION

$00 — Command

The command field specifies the command the MVME327A is to execute. The table below is a list of commands supported by the MVME327A.

MVME327A Command Codes

COMMAND CODE	OPERATION	VALID FOR DEVICE TYPE: LOCAL FLOPPY ($01)	SCSI DISK ($05)	SCSI TAPE ($05)	SCSI 327A ($05)	LOCAL 327A ($0F)
$00	BPP Test	X	X	X	X	X
$01	Read	X	X	X		
$02	Write	X	X	X		
$03	Read Descriptor	X	X	X		
$04	Write Descriptor	X	X	X		
$05	Format	X	X			
$06	Fix Bad Spot		X			
$10	Read Status			X		
$11	Load/Unload/Re-tension			X		
$12	Write Filemark			X		
$13	Rewind			X		
$14	Erase			X		
$15	Space			X		
$20	Enable Target				X	
$21	Disable Target				X	
$22	Reserve Unit		X	X	X	
$23	Release Unit		X	X	X	
$25	Reset SCSI		X	X	X	
$26	Custom SCSI Command		X	X	X	
$27	Self Test					X
$28	Target Wait				X	
$29	Target Execute				X	
$2B	Set SCSI Address				X	
$2D	Open		X	X		

$01 Command Control

This field contains flags that are used by the command. the use of this field is command specific.

$02 Device Type

This field specifies a unique type of device for the primary device. The MVME327A valid device types are listed below.

Device Type Assignments

DEVICE TYPE	DEVICE DESCRIPTION
$01	Local Floppy Disk
$05	SCSI Bus
$0F	MVME327A

$03 Unit Number

This is the unit number for the peripheral device to be accessed in a transfer between a device and host memory. SCSI bus device types break this field into two nibbles; the high order nibble represents the SCSI bus address (SCSI controller), and the low order nibble represents the SCSI peripheral device address or often referred to as the Logical Unit Number (LUN) for that controller's device.

$06 Address Modifier

The address modifier is used by the MVME327A when accessing all data structures pointed to by the packet (e.g., user data buffers and disk descriptors).

Command packets are transferred using the address modifier written into the address modifier register when the channel is created by the host CPU. This approach gives the host software the freedom to put packets in one address space and data in another address space, if desired.

$07 Data Bus Width -

This field tells the MVME327A the width of the VMEbus data path. The MVME327A attempts to do 32-bit data transfers if told to in this field. If data is long word aligned, transfers are more efficient; data that is odd word aligned is transferred 16 bits at a time.

Data Bus Width Codes

BUS WIDTH CODE	BUS WIDTH
1	16-Bit
2	32-Bit

$08 Primary Address

Command dependent. Refer to individual command description for usage.

$0C Secondary Address

Command dependent. Refer to individual command description for usage.

$10 Transfer Count

Command dependent. Refer to individual command description for usage.

$14 Scatter/Gather Count

This field is the scatter/gather count and flag. If it is zero, the command is not using the scatter/gather feature.

$16 Command Parameters 1, 2, and 3

These fields may be used by a device for specific commands as needed. If these fields are required, they are described in the paragraphs for each command.

$1C Fatal Error Code

If this field is nonzero, a fatal error occurred, and the value is the error code (refer to Appendix D).

$1D Recovered Error Status

If a recovered error occurs, the MVME327A can return status indicating what the nature of the problem was, and perhaps what action was taken to correct it. The recovered error status field is set to a nonzero value to indicate that error recovery was performed by the MVME327A.

$1E Additional Error Code/Status

The status word is returned in one of two formats. In format 1, the upper byte is nonzero because a request sense (SCSI) command was issued at the request of the target. (A request sense (SCSI) command returns formatted error information about the last SCSI command to the target.) The upper byte contains the FM bit, EOM bit, ILI bit, and sense key (byte 02 of the sense data). The lower byte contains the additional sense code (byte 12 of the sense data), if available, from the sense data. The additional sense code byte is device dependent. Refer to Appendix E for a partial list of these error code definitions.

In format 2, the upper byte is clear and the lower byte contains a specific MVME327A fatal error code that is relative to the current fatal error. Refer to Appendix D for a list of these error code definitions.

Additional Error Code/Status
Format 1

15	14	13	12	Bits 11-8	Bits 7-0
FM	EOM	ILI	Res	Sense Code	Additional Sense Code

Format 2

Bits 15-8	Bits 7-0
0	Error Code

$20 Retry Count

If retries were required during this operation, this field will indicate the number of retries attempted.

$22 Error Status Address

If a transfer terminates because of an error, this field has the logical block (or sector) address at which the error occurred. For example, when a disk error occurs this contains the logical block where the error occurred.

$26 Termination Transfer Count

This field contains the number of bytes successfully transferred, regardless of the completion status of the command.

$2A,2C Status Parameters 1, 2 are not used.

$2E Status Parameter 3

If fatal error code $01 (bad descriptor) or $02 (bad command) is returned, then this status word contains the byte offset where the bad field is located. If the offset is -1 ($FFFF), then the bad field is not indicated.

It should be noted that only the "command" portion of the packet (offsets $00 through $1B) are read by the firmware, and that the firmware returns values in (or alters) only the "status" portion of the packet (offsets $1C through $2F).

CHAPTER 4 - BUFFERED PIPE PROTOCOL

4.1 INTRODUCTION

This chapter describes and illustrates how a host and MVME327A communicate using the Buffered Pipe Protocol (BPP).

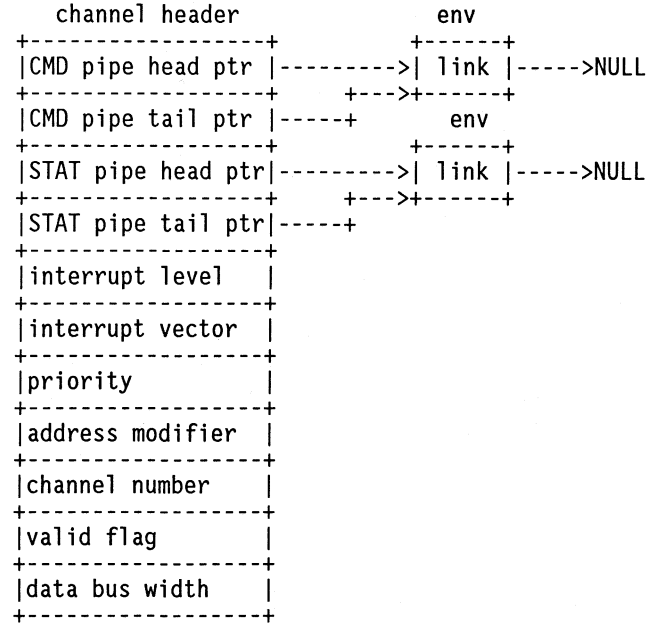
4.2 ESTABLISHING DRIVER/MVME327A CHANNEL COMMUNICATIONS

The following paragraphs deal with setting up a channel that is used to send packets between a host and an MVME327A. Communications between a driver and an MVME327A are handled almost exclusively via command/status packets which are exchanged along these channels. The channel header contains a command pipe and a status pipe. Host drivers pass packets to the MVME327A via the command pipe; the MVME327A returns processed packets via the status pipe.

Before packets can be sent to an MVME327A, a create channel command has to be issued by the host driver.

A channel header structure is built in memory with a NULL envelope on the command pipe, a NULL envelope on the status pipe, and additional parameters. A NULL envelope is an envelope that does not have the valid flag set and has a NULL link pointer. It's packet pointer can either be NULL or be pointing to a packet. The figure below illustrates what the channel header structure with empty channels looks like.

As shown below, the channel header has one envelope on the command pipe and one envelope on the status pipe. The link fields of the envelopes are NULL. The valid flag in the envelopes are zero, meaning that these are the last envelopes in the pipes. Having NULL link pointers and the valid flags set to zero defines them as NULL envelopes.



The rest of the fields in the channel header are initialized with the appropriate values, except for the valid flag and the channel number. The channel number and valid flag are set by the MVME327A at the successful completion of the create channel command.

After the channel header has been set up the host executes the create channel command. This is a CSR command, therefore, the driver must set the TAS bit in the MVME327A TAS Register. When the TAS bit has been set, the driver can proceed with the CSR command protocol. This protocol is described in detail in the Chapter 2, CSR Interface.

After checking the CSR status register to ensure that no problem arose in creating the channel, the driver can begin sending commands to the MVME327A.

4.2.1 Envelope/Packet Enqueueing

The protocol to be followed by the host when enqueueing an envelope onto the command pipe is as follows:

a. Fill in all necessary fields in a packet, and initialize the envelope pointed to by the command tail pointer with the packet address in the packet pointer field. This envelope is referred to as the command envelope and is the old NULL envelope. The command envelope should have a NULL link field and the valid flag should be zero.

b. Set the link field of the envelope pointed to by the command pipe tail pointer field (the command envelope) to the address of a new NULL envelope. This new NULL envelope must have a link field of zero and valid flag cleared to zero.

c. Set the valid flag in the command envelope. This should be the last field that is accessed in the envelope.

d. Set the command pipe tail pointer field to the new NULL envelope.

e. Interrupt the MVME327A by setting the attention bit in the MVME327A control register. Note that the TAS bit and register are not needed for BPP communications.

4.2.2 Envelope/Packet Dequeueing

The protocol to be followed by the host when dequeueing an envelope from a pipe is as follows:

a. Access the envelope pointed to by the pipe head pointer field.

b. If the valid flag of this envelope is set, then the packet associated with this envelope has been processed by the MVME327A at the other end of the pipe, and is ready to be dequeued. To dequeue it, set the pipe head pointer field to the value in the envelope link pointer field.

c. If the valid flag is not set in the envelope, then the MVME327A at the other end of the pipe has not completed the processing of a packet and the envelope is not ready for dequeueing.

It is possible for more than one processed envelope to be on a pipe waiting to be dequeued. A driver can dequeue envelopes and process their associated packets until it encounters an envelope with the valid flag clear.

4.2.3 Buffered Pipe Protocol Summary

Packets are always enqueued on the tail of a pipe and dequeued from the head of a pipe.

The host driver only updates the command pipe tail pointer and the status pipe head pointer; it never updates the command pipe head pointer or the status pipe tail pointer. The host driver always enqueues envelopes at the tail of the command pipe and dequeues envelopes from the head of the status pipe.

The MVME327A only updates the command pipe head pointer and the status pipe tail pointer; it never updates the command pipe tail pointer or the status pipe head pointer. The MVME327A always dequeues envelopes from the head of the command pipe and enqueues envelopes at the tail of the status pipe.

The original command pipe head pointer and status pipe tail pointer in the host channel header are never updated by the MVME327A firmware; only a copy of them in the MVME327A local memory are kept current.

CHAPTER 5 - OTHER DATA STRUCTURES

5.1 INTRODUCTION

This chapter sets forth several other data structures or tables which are used by the firmware.

5.2 DEVICE DESCRIPTORS

Device descriptors define the characteristics of a device. There is a single descriptor defined to handle all disk drive types and two descriptors defined to handle tape drive types. A particular device may not use all of the fields in a descriptor. If a field is not needed, it should be ignored.

5.2.1 Disk Descriptor Table

This descriptor is specific to disk devices. Disk descriptors are used for SCSI disks and the local floppy disk interface of the MVME327A.

Disk Descriptor Table

BYTE OFFSET	PARAMETER DESCRIPTION	
$00	controller type	peripheral type
$02	number of heads	fixed/removable media
$04	number of cylinders (MSW)	
$06	number of cylinders (LSW)	
$08	bytes per sector	
$0A	logical block size	
$0C	logical sectors per track	reserved sectors per zone
$0E	hard/soft sector flag	interleave factor
$10	format init character	retry count
$12	step rate	floppy format
$14	pre-compensation cylinder	
$16	reduced write current cylinder	
$18	zone type / alternate unit	number of alternates
$1A	ECC flag	spiral offset
$1C	reserved	
$1E	reserved	
$20	reserved	
$22	cylinder skew	cache entry size

OFFSET	DESCRIPTION
$00	Controller type A predefined (SCSI) controller type code. Refer to Appendix A.
$01	Peripheral type A predefined (SCSI) peripheral type code. Acceptable codes are: $01 - Floppy disk $02 - Rigid disk
$02	Number of heads This is the number of heads a disk has.
$03	Fixed/removable media (NOTE 1) This is a flag indicating that the media is fixed or removable (0 = fixed, 1 = removable).
$04	Number of cylinders The number of physical cylinders that the disk contains.
$08	Bytes per sector The number of bytes per sector.
$0A	Logical block size The size of data transfers to and from the device. The logical block size must be equal to, or an integral multiple of, the sector size.
$0C	Logical sectors per track The number of logical sectors per track. The logical sectors per track plus the reserved sectors per track equal the number of physical sectors per track.
$0D	Reserved sectors per zone The number of alternate sectors (slipped sectors) per zone. Refer to offset $18 for zone type.
$0E	Hard/soft sector flag (NOTE 1) A flag which indicates whether the disk drive is soft or hard sector format (0 = soft, 1 = hard).

$0F Interleave factor

The interleave factor determines the physical separation of logical sectors. For example, a disk with interleave factor of 2 has logical sectors 1 and 2 one physical sector apart rather than being physically contiguous as they would be with interleave of 1. A value of zero specifies the drive default interleave factor.

$10 Format init character

The format init character is the character used to fill in the data field when a format command is performed. Many SCSI drives do not offer a choice, in which case this field is ignored.

$11 Retry count

The number of times a command which accesses the disk can retry.

$12 Step rate (NOTE 1)

Specifies the stepping rate, in units of time, for the drive (floppy unit = 200 microseconds, Winchester unit = 10 microseconds). The MVME327A, if the step rate is not exactly what the drive expects, rounds up to the next appropriate step rate that the drive expects.

$13 Floppy format (NOTE 2)

Specifies the floppy disk characteristics. This field is used along with the bytes per sector field. See the figure below.

Floppy Format Field

7	6	5	4	3	2	1	0
res	res	res	trk 0 dens	media tpi	drv tpi	data rate	encoding

BIT	DESCRIPTION
7	res - reserved.
6	res - reserved.
5	res - reserved.
4	track 0 dens - specifies the density of track 0 relative to the rest of the floppy (0 = one half of the rest of floppy, 1 = same as the rest of floppy).

3 media tpi - specifies the density (tracks per inch) of the media (0 = 48 tpi, 1 = 96 tpi).

2 drv tpi - specifies the density (tracks per inch) of the drive (0 = 48 tpi, 1 = 96 tpi).

1 data rate - specifies what the data rate is (0 = 5-1/4 inch, 1 = 8-inch).

0 encoding - specifies what the encoding is (0 = FM, 1 = MFM).

$14 Pre-compensation cylinder number (NOTE 1)

$16 Reduced write current cylinder number (NOTE 1)

$18 Zone type (upper nibble)

0 = zone is track.
1 = zone is cylinder.

Alternate unit (lower nibble)

Indicates what unit the number of alternates field is in (0 = tracks, 1 = cylinders).

$19 Number of alternates

The number of alternates reserved for bad spot mapping.

$1A ECC correction

This flag enables the automatic correction of errors with ECC. If ECC is not enabled then ECC is still used to detect errors but does not correct the errors (0 = disable automatic error correction, 1 = enable automatic error correction).

$1B Spiral offset (NOTE 1)

The spiral offset is the number of sectors that sector 0 for each track is offset from the previous track.

$22 Cylinder skew (NOTE 1)

The cylinder skew is the number of sectors to offset the logical sector 0 of the next cylinder from logical sector 0 of the current cylinder.

$23 Cache entry size (NOTE 3)

The size of the cache entries, in units of logical blocks, for this drive.

NOTES: 1. Ignored by embedded SCSI controllers.
2. Used for floppy disk only. Ignored for rigid disk.
3. Ignored unless cache feature is present on drive.

5.2.2 Streaming Tape Descriptor Table

This descriptor is specific to streaming tape devices.

Streaming Tape Descriptor Table

BYTE OFFSET	PARAMETER DESCRIPTION	
$00	controller type	peripheral type = $05
$02	number of drive tracks	number of media tracks
$04	extend on write flag	byte swap flag
$06	buffered mode flag	reserved
$08	physical bytes per block	
$0A	logical block size	
$0C	QIC format	
$0E	streaming count	
$10	reserved	retry count
$12	minimum read transfer size	
$14	minimum write transfer size	
$16	reserved	
$18	reserved	
$1A	reserved	
$1C	reserved	
$1E	reserved	
$20	reserved	
$22	reserved	

OFFSET	DESCRIPTION
$00	Controller type
	A predefined controller type code. Refer to Appendix A.
$01	Peripheral type
	A predefined device type code of $05 for streaming tape.
$02	Number of drive tracks
	This field specifies the number of tracks on a drive.
$03	Number of media tracks
	This field specifies the number of tracks on a tape.

$04 Extend on write flag

Indicates the use of extend on write mode. In the extend mode, the tape controller continues streaming, even if the data buffer has been exhausted (0=non-extend, 1=extend).

$05 Byte swap flag

Controls the byte swap mode (0=no byte swap, 1=byte swap).

$06 Buffered mode flag

Controls the buffered mode of write operation. Non-buffered mode causes the tape controller to report status when all data blocks have been written to tape. Buffered mode causes the tape controller to report status as soon as the data has been transferred to the tape controller's buffer (0=non-buffered, 1=buffered).

$08 Physical bytes per block

The number of physical bytes per block.

$0A Logical block size

The size of data transfers to and from the tape device. The logical block size has to be a multiple of the physical block size.

$0C QIC format

Specifies the standardized recording format on the media. For QIC-11, the field contains an eleven, for QIC-24, the field contains twenty-four, for QIC-120, the field contains one-hundred-twenty, etc.

$0E Streaming count

Specifies the number of times the tape controller re-writes the last block during a write operation in the event the data buffer is exhausted. Use of this field is controller dependent.

$11 Retry count

The number of times a command which accesses the tape can retry.

$12 Minimum read transfer size

Use of this field is controller specific.

$14 Minimum write transfer size

Use of this field is controller specific.

5.2.3 Start/Stop Tape Descriptor Table

This description is specific to start/stop tape devices.

Start/Stop Tape Descriptor Table

BYTE OFFSET	PARAMETER DESCRIPTION	
$00	controller type	peripheral type = $06
$02	speed select code	density select code
$04	flag 1	flag 2
$06	reserved	retry count
$08	physical block size	
$0A		
$0C	logical block size	
$0E		
$10	reserved	
$12	reserved	
$14	reserved	
$16	reserved	
$18	reserved	
$1A	reserved	
$1C	reserved	
$1E	reserved	
$20	reserved	
$22	reserved	

OFFSET	DESCRIPTION
$00	Controller type
	A predefined controller type code. Example: $16 - Kennedy 8124 (1/2-inch)
$01	Peripheral type = $06 (start/stop tape)
	A predefined device type code of $05 for streaming tape.
$02	Speed select code
	On controllers/drivers which support more than one tape speed, the following codes are used to select a particular speed:

$00 - use default speed
$01 - use lowest speed
$02 - use next higher speed
.
.
(codes in ascending order of speeds, as many as necessary)
.
$0F - use highest speed

$03 Density select code

This code value defines the tape width, reel-to-reel or cartridge, number of tracks, BPI, recording method, and if the tracks are recorded in parallel or in series. Codes from X3T9.2 are:

$00 - use default
$01 - 1/2-inch reel-to-reel, 9-track parallel, 800 BPI NRZI
$02 - 1/2-inch reel-to-reel, 9-track parallel, 1600 BPI PE
$03 - 1/2-inch reel-to-reel, 9-track parallel, 6250 BPI GCR
$06 - 1/2-inch reel-to-reel, 9-track parallel, 3200 BPI PE

5

$04 Flag 1

This is bit encoded as follows:

Bit	Definition
7-1	reserved
0	1 = buffered write

record is not written until internal tape controller buffer is full. Status returns on completion of data transfer across the SCSI bus.

$05 Flag 2

This is bit encoded as follows:

Bit	Definition
7-1	reserved
0	1 = byte swap

perform a byte swap on data.

$07 Retry count

The number of times a command which accesses the tape can retry. A zero value means use the default.

$08 Physical block size

This value is the number of bytes per physical block. Zero specifies variable length blocks.

$0C Logical block size

This value specifies the size of block data transfers to/from the tape device in bytes. It must be an integer multiple of the physical block size. Zero specifies variable length records.

5.3 SCATTER/GATHER LIST

A scatter/gather list may be used in several high level or SCSI level commands. Basically, this list of one or more entries substitutes for the single source or destination address for data movement. Each entry in the scatter/gather list consists of a long word (32-bit) address followed by a long word byte count. The total number of these 8-byte entries in the list must be shown in the command packet "scatter/gather count" parameter.

No ending entry or flag is necessary in the scatter/gather list. A pointer to the scatter/gather list is put in place of the VME buffer address in the command packet. If a scatter/gather list is not being used, the scatter/gather count must be zero.

The total byte count of all scatter/gather list entries must equal the packet "transfer block count" multiplied by the device logical block size (as described in the descriptor table).

CHAPTER 6 - MVME327A COMMANDS

6.1 INTRODUCTION

This chapter discusses the two types of commands used with the BPP interface to the MVME327A: high level and SCSI level. These commands may be sent to one of three possible destinations as defined in the command packet "device type" field:

$01 - local floppy disk drive(s)
$05 - SCSI bus
$0F - MVME327A

6.2 LOCAL FLOPPY COMMANDS

All commands to the local floppy disk drive(s) are of the high level command type. These commands are summarized in the *packet* paragraph in Chapter 3 and fully documented in Chapter 7. These high level commands to floppy directly control the operation of the one or more local floppy drives.

6.3 SCSI BUS COMMANDS

There are two ways to communicate a command to a SCSI bus peripheral with the MVME327A: high level commands and SCSI level commands. Each method is discussed in the following paragraphs.

6.3.1 High Level Command Translation

There are two ways to communicate a command to a SCSI bus or local floppy peripheral via the MVME327A: high level commands and SCSI level commands. These paragraphs cover briefly the high level command method.

In previous chapters, the BPP envelope was shown to contain a pointer to a command/status packet. In this packet is the particular high level command along with necessary parameters and pointers. This packet is built by the process issuing the command and resides in memory under control of that process.

When this packet is sent to the MVME327A via BPP (actually only a pointer to this packet is sent), the firmware copies the command portion of the packet over the VMEbus into MVME327A local RAM memory. The firmware then decides the device that the command is meant for and queues the packet appropriately.

When the command packet's turn comes up to be processed, the high level command is translated into zero to many SCSI packets, sending each to the SCSI bus. In the case of the local floppy, the high level command is passed onto the portion of firmware dedicated to this operation, and no SCSI packets or SCSI bus utilization is done.

As an example, with a READ command for a SCSI controller having a Winchester disk drive, normally only one SCSI packet would be built. The exception would be if the amount of data requested exceeded either the SCSI CDB parameter size or the number of bytes could not be expressed by a 24-bit number (limitation of SCSI controller chip), then the firmware builds two or more SCSI packets to perform the one high level READ command.

Some high level commands may result in several different SCSI packets being built and issued. For instance, on certain supported controllers, the WRITE DESCRIPTOR high level command could result in SCSI commands "test unit ready", "mode sense", "mode sense", and "mode select" being issued over the SCSI bus. Other high level commands, such as READ DESCRIPTOR, may not even access the SCSI bus.

Because of the wide latitude given in using SCSI controllers, various controllers can comply with the SCSI standard and yet differ in parameters, format of parameters, etc. The Common Command Set (CCS) narrows these differences down, but still (in its present form) leaves chance for implementing things differently. The MVME327A firmware supports, with its high level commands, certain SCSI controllers. The controller/device type code is given to the firmware in the initial WRITE DESCRIPTOR (or READ DESCRIPTOR) command. The firmware keeps track of this information for succeeding high level commands to the associated SCSI address/logical unit, and can compensate for the different implementations of SCSI based upon this information.

One controller type code (hexadecimal OF) has been assigned for CCS disk controllers. When this code is specified in a WRITE (or READ) DESCRIPTOR, the firmware expects the SCSI controller to comply with the CCS specifications (Common Command Set, Rev. 4B), and builds SCSI packets for further high level commands based on the standard and information it can obtain from the controller by the way of "mode sense" command return data.

Refer to Appendix A for controller type codes and other information on supported controllers.

6.3.2 SCSI Level Commands

The second type of command to the MVME327A is what might be termed a low level command. This is implemented in the MVME327A firmware as high level commands CUSTOM SCSI and TARGET EXECUTE which allow the user to specify on a low level exactly the command and data being sent via the SCSI bus.

Both commands contain a pointer in its command packet which points to a SCSI packet. The SCSI packet contains the low level SCSI bus command and other pertinent data.

6.3.3 SCSI Specific Packet

The SCSI specific packet supplied by the user is in the same form as the packet built by the firmware. In this packet are such necessary data as the SCSI Command Descriptor Block (CDB), data length, the data pointer, the message(s) to be sent to the SCSI device, the buffer addresses to store message(s) and the status from the SCSI device, and a script. Various other SCSI and firmware specific data are also passed through the SCSI specific packet. For more information regarding the use of SCSI specific packets refer to Appendix B (Custom SCSI Command Example), and Appendix C (Target Role).

The table below defines the SCSI specific packet parameters.

SCSI Specific Packet Description

BYTE OFFSET	PARAMETER DESCRIPTION	
$00	link pointer (MSW)	
$02	link pointer (LSW)	
$04	control word	
$06	command length	reserved
$08	CDB 0	CDB 1
$0A	CDB 2	CDB 3
$0C	CDB 4	CDB 5
$0E	CDB 6	CDB 7
$10	CDB 8	CDB 9
$12	CDB A	CDB B
$14	data length (MSW)	
$16	data length (LSW)	
$18	data pointer (MSW)	
$1A	data pointer (LSW)	
$1C	SCSI status	initiator ID
$1E	message in flag	message out flag
$20	message in length	
$22	message in pointer (MSW)	
$24	message in pointer (LSW)	
$26	message in byte 1	message in byte 2
$28	message in byte 3	message in byte 4
$2A	message in byte 5	message in byte 6
$2C	message out length	
$2E	message out pointer (MSW)	
$30	message out pointer (LSW)	
$32	message out byte 1	message out byte 2
$34	message out byte 3	message out byte 4
$36	message out byte 5	message out byte 6
$38	script byte 1	script byte 2
$3A	script byte 3	script byte 4
$3C	script byte 5	script byte 6
$3E	script byte 7	script byte 8

OFFSET	DESCRIPTION
$00	Link pointer is for users wanting to perform linked SCSI commands. This pointer is used to forward link another SCSI specific packet. This linking process may go on indefinitely, depending on the support provided by the SCSI target. The link bit in the CDB control byte must be properly set for the proper linkage by the SCSI target and the MVME327A firmware. Refer to the SCSI specification X3T9.2/82-2 Rev. 17B, paragraph 6.2.6 for a description of the control byte and linked commands. Currently not supported. Must be zero.
$04	The control word contains the control information needed to transfer data across the MVME327A.

Custom SCSI Command Control Word Bit Definitions

15	14	13	12	11	10	9	8	7	6-0
DMA	SYNC	PAR	SCHK	LWDATA	B SWAP	SG	LINK	NO ATN	res

BIT	DESCRIPTION
15	DMA - if set, SCSI data phase uses the DMA controller to transfer data from the SCSI bus to VME memory. This bit should only be set for reads and writes.
14	SYNC - if set, the SCSI data phase is synchronous.
13	PAR - if set, parity checking is enabled.
12	SCHK - if set, status checking is done by the user.
11	LWDATA - if set, data phase can use 32-bit (long word) transfers.
10	B SWAP - if set, byte swap is enabled for data phase. **Only use 16-bit transfers when byte swap is turned on (the MVME327A does not perform word swapping).**
9	SG - if set, the data transfer uses scatter/gather and a scatter/gather list is used for the transfers.
8	LINK - if set, a linked SCSI command is to be performed. Linked commands are only used in custom SCSI applications. Currently not supported.
7	NO ATN - if set, do not assert ATN during selection. Message protocol will not be instituted.
6-0	res - reserved.

$06	Command length is the length of the command descriptor block. This value is usually 6, 10, or 12 (decimal).
$07	Reserved for future use.
$08-13	CDB is the command descriptor block. If the MVME327A is operating in initiator role, this CDB is passed to the threaded SCSI target. If the MVME327A is operating in target role, the CDB received from the initiator is loaded in this area.
$14	Data length is the length of the data transfer (in bytes) which is located in this field.
$18	Data pointer points to the buffer where the data is to be read/written. If scatter/gather is used, this field points to the scatter/gather list.
$1C	SCSI status byte is either read or written from/to this location. In initiator role, the status byte that is received from the target is stored in this location. In the target role, the contents of this location are passed to the initiator.
$1D	Initiator ID byte is used for target role only. When a WAIT target command is given to the MVME327A, a SCSI specific packet is given to the MVME327A to hold the received CDBs. When a selection of the MVME327A takes place, a CDB is written to this SCSI specific packet and returned to the user for processing. The initiator on the SCSI bus that selected the MVME327A may need to be identified for implementing the RESERVE and RELEASE commands. The SCSI address of the selecting initiator is passed to the user in this byte.
$1E	Message in flag signals that the message is in this SCSI specific packet ($00) or that there is a pointer to a message buffer ($FF).
$1F	Message out flag signals that the message is in this SCSI specific packet ($00) or that there is a pointer to a message buffer ($FF).
$20	Message in length is the message length in bytes.
$22	If message in pointer flag is set ($FF), then this location contains a pointer to a message buffer in VME memory.
$26-2B	If message in bytes flag is not set ($00), then these locations contain the message.
$2C	Message out length is the message length in bytes.
$2E	If message out pointer flag is set ($FF), then this location contains a pointer to a message buffer in VME memory.

$32-37	If message out bytes flag is not set ($00), then these locations contain the message.
$38-3F	Script byte locations contain the script for the indicated SCSI command. Scripts and script codes are defined in Appendix B and C.

6.4 **MVME327A COMMANDS**

There are several high level commands (including "custom SCSI command" which then allows a SCSI level command) which are either exclusively for the MVME327A adapter board (such as self test) or shared with the SCSI bus (such as "reserve unit"). These commands are summarized in the *packet* paragraph in Chapter 3 and detailed in Chapter 7.

CHAPTER 7 - HIGH LEVEL COMMANDS

7.1 INTRODUCTION

The MVME327A high level commands are described in this chapter.

In the following description of the commands, any field that is marked XX or XXXX is a don't care field for that particular command. To assure upward compatibility with future revisions of MVME327A firmware, it is recommended that all don't care (XX and XXXX) areas of the command packet be cleared to zero, particularly the "command control" byte at offset $01. For the status section of the packet, which is returned to the user after completion of a command, the don't care areas denote invalid data and should be ignored by the user.

7.1.1 BPP Test Command ($00)

The BPP test command is used to assure the functionality of a channel and BPP protocol. It is essentially a No Operation (NOP) and is useful in driver software development.

7.1.1.1 BPP Test Command Packet

The table below defines the BPP test command packet parameters.

BPP Test Command Packet

BYTE OFFSET	PARAMETER DESCRIPTION		
$00	command = $00		XX
$02	device type		XX
$04		XXXX	
$06		XXXX	
$08		XXXX	
$0A		XXXX	
$0C		XXXX	
$0E		XXXX	
$10		XXXX	
$12		XXXX	
$14		XXXX	
$16		XXXX	
$18		XXXX	
$1A		XXXX	

OFFSET	DESCRIPTION
$00	Command value for BPP test ($00).
$02	Device type. Valid types are $01, $05, and $0F.

7.1.1.2 BPP Test Command Returned Status

The table defines the BPP test command status packet parameters.

BPP Test Command Status Packet

BYTE OFFSET	PARAMETER DESCRIPTION	
$1C	fatal error code	XX
$1E	XXXX	
$20	XX	XX
$22	XXXX	
$24	XXXX	
$26	XXXX	
$28	XXXX	
$2A	XXXX	
$2C	XXXX	
$2E	XXXX	

OFFSET	DESCRIPTION
$1C	A nonzero fatal error code if the command did not complete successfully.

7.1.2 Read Command ($01)

A read command is issued to read logical blocks of data from the media.

7.1.2.1 Read Command Packet

The table defines the read command packet parameters.

Read Command Packet

BYTE OFFSET	PARAMETER DESCRIPTION	
$00	command = $01	command control
$02	device type	unit number
$04	XX	XX
$06	address modifier	data bus width
$08	logical block number (MSW)	
$0A	logical block number (LSW)	
$0C	VME buffer address (MSW)	
$0E	VME buffer address (LSW)	
$10	transfer block count (MSW)	
$12	transfer block count (LSW)	
$14	scatter/gather count	
$16	XXXX	
$18	XXXX	
$1A	XXXX	

OFFSET	DESCRIPTION
$00	Command value for read ($01).
$01	Command control specifies the options for the read command.

7	6	5	4	3	2	1	0
NC	RR	SIL	res	res	res	res	res

BIT	DESCRIPTION
7	NC - If set, bypass the cache.
6	RR - Read reverse. If set, read in reverse direction on tape. Ignored by disk devices and tape devices not supporting reverse reading.
5	SIL - Suppress illegal length indicator. If set, suppress the illegal length indication on tape drives supporting it. Ignored otherwise.
4	res - reserved.
3	res - reserved.
2	res - reserved.
1	res - reserved.
0	res - reserved.

$02	Device type.
$03	Unit number to which the command applies.
$06	Address modifier of the memory space where the VME read buffer is located.
$07	Data bus width is a code to indicate the width of the data bus. $01 indicates a 16-bit data bus, $02 indicates a 32-bit data bus.
$08	Logical block number is the source of the read data for direct access devices. This field indicates the starting block number. For sequential devices this field is ignored. The block size is defined in the descriptor.
$0C	VME buffer address is a pointer to the destination VME memory where a read buffer is located. However, if the scatter/gather count (offset $14) is nonzero, this field is a pointer to a scatter/gather table that contains a number of entries specified by the scatter/gather count.
$10	Transfer block count is the number of blocks of data to be transferred from device media to VME memory. With variable length blocks (available on some tape drives), this count effectively becomes the length of the block in bytes.
$14	Scatter/gather count is the number of entries in the scatter/gather table. If the count is zero, the command is not using the scatter/gather feature.

7.1.2.2 Read Command Returned Status

The table defines the read command status packet parameters.

Read Command Status Packet

BYTE OFFSET	PARAMETER DESCRIPTION
$1C	fatal error code / recovered error status
$1E	additional error code/status
$20	retry count / XX
$22	error status address (MSW)
$24	error status address (LSW)
$26	termination transfer count (MSW)
$28	termination transfer count (LSW)
$2A	filemark position (MSW)
$2C	filemark position (LSW)
$2E	status parameter 3

OFFSET	DESCRIPTION
$1C	A nonzero fatal error code if the command did not complete successfully.
$1D	A nonzero recovered error status if the command completed successfully after retries.
$1E	Additional status error information if the fatal error code is nonzero.
$20	The number of retries that were performed by the MVME327A.
$22	This is error status address. If a transfer terminates because of an error, this field has the logical block address at which the error occurred.
$26	The number of bytes successfully transferred.
$2A	For sequential devices (e.g., tape), filemark position count. If value returned is $FFFFFFFF, position is indeterminate. This field is unused for random access devices (e.g., disk).
$2E	If fatal error code is $02 (bad command), then this field contains an offset into the command packet of the offending parameter.

7.1.3 Write Command ($02)

A write command is issued to write logical blocks of data to the media.

7.1.3.1 Write Command Packet

The table below defines the write command packet parameters.

Write Command Packet

BYTE OFFSET	PARAMETER DESCRIPTION	
$00	command = $02	XX
$02	device type	unit number
$04	XX	XX
$06	address modifier	data bus width
$08	logical block number (MSW)	
$0A	logical block number (LSW)	
$0C	VME buffer address (MSW)	
$0E	VME buffer address (LSW)	
$10	transfer block count (MSW)	
$12	transfer block count (LSW)	
$14	scatter/gather count	
$16	XXXX	
$18	XXXX	
$1A	XXXX	

7

OFFSET	DESCRIPTION
$00	Command value for write ($02).
$02	Device type.
$03	Unit number to which the command applies.
$06	Address modifier of the memory space where the data buffer is located.
$07	Data bus width is a code to indicate the width of the data bus. $01 indicates a 16 bit data bus, $02 indicates a 32 bit data bus.
$08	Logical block number is the destination of the write data for direct access devices. This field indicates the starting block number. For sequential devices this field is ignored. The block size is defined in the descriptor.

$0C — VME buffer address is a pointer to the source VME memory where the write data is located. However, if the scatter/gather count (offset $14) is nonzero, this field is a pointer to a scatter/gather table that contains a number of entries specified by the scatter/gather count.

$10 — Transfer block count is the number of blocks of data to be transferred from VME memory to device media. With variable length blocks (available on some tape drives), this count effectively becomes the length of the block in bytes.

$14 — The scatter/gather count is the number of entries in the scatter/gather table. If the count is zero, the command is not using the scatter/gather feature.

7.1.3.2 Write Command Returned Status

The table defines the write command status packet parameters.

Write Command Status Packet

BYTE OFFSET	PARAMETER DESCRIPTION
$1C	fatal error code — recovered error status
$1E	additional error code/status
$20	retry count — XX
$22	error status address (MSW)
$24	error status address (LSW)
$26	termination transfer count (MSW)
$28	termination transfer count (LSW)
$2A	filemark position (MSW)
$2C	filemark position (LSW)
$2E	status parameter 3

OFFSET — DESCRIPTION

$1C — A nonzero fatal error code if the command did not complete successfully.

$1D — A nonzero recovered error status if the command completed successfully after retries.

$1E — Additional status error information if the fatal error code is nonzero.

$20 — The number of retries that were performed by the MVME327A.

$22	This is error status address. If a transfer terminates because of an error, this field has the logical block address at which the error occurred.
$26	The number of bytes successfully transferred.
$2A	For sequential devices (e.g., tape), filemark position count. If value returned is $FFFFFFFF, position is indeterminate. This field is unused for random access devices (e.g., disk).
$2E	If fatal error code is $02 (bad command), then this field contains an offset into the command packet of the offending parameter.

7.1.4 Read Descriptor Command ($03)

If the read descriptor command is sent before the first write descriptor command for a device type/unit number combination, the device descriptor table must contain the controller type code and peripheral type code prior to issuing the read descriptor command. In this case, this command queries the device/unit and returns available information in the descriptor table.

Otherwise, if a write descriptor command has been previously issued for a device type/unit number combination, the read descriptor command returns a copy of the information existing in the internally kept descriptor table, put there by the write descriptor command.

7.1.4.1 Read Descriptor Command Packet

The table below defines the read descriptor command packet parameters.

Read Descriptor Command Packet

BYTE OFFSET	PARAMETER DESCRIPTION	
$00	command = $03	XX
$02	device type	unit number
$04	XX	XX
$06	address modifier	data bus width
$08	XXXX	
$0A	XXXX	
$0C	pointer to device descriptor table (MSW)	
$0E	pointer to device descriptor table (LSW)	
$10	XXXX	
$12	XXXX	
$14	XXXX	
$16	XXXX	
$18	XXXX	
$1A	XXXX	

OFFSET	DESCRIPTION
$00	Command value for read descriptor ($03).
$02	Device type.
$03	Unit number to which the command applies.
$06	Address modifier of the memory space where the device descriptor table is located.
$07	Data bus width is a code to indicate the width of the data bus. $01 indicates a 16 bit data bus, $02 indicates a 32 bit data bus.
$0C	Pointer to the device descriptor table is a pointer into VME memory where the device descriptor table begins.

7.1.4.2 Read Descriptor Command Returned Status

The table defines the read descriptor command status packet parameters.

Read Descriptor Command Status Packet

BYTE OFFSET	PARAMETER DESCRIPTION
$1C	fatal error code / recovered error status
$1E	additional error code/status
$20	retry count / XX
$22	XXXX
$24	XXXX
$26	XXXX
$28	XXXX
$2A	XXXX
$2C	XXXX
$2E	status parameter 3

OFFSET	DESCRIPTION
$1C	A nonzero fatal error code if the command did not complete successfully.
$1D	A nonzero recovered error status if the command completed successfully after retries.
$1E	Additional status error information if the fatal error code is nonzero.

$20	The number of retries that were performed by the MVME327A.
$2E	If fatal error code is $01 (bad descriptor) or $02 (bad command), then this field contains an offset into the descriptor table or command packet of the offending parameter.

7.1.5 Write Descriptor Command ($04)

The write descriptor command defines the characteristics of a device. The host builds a device descriptor table and passes the table to the MVME327A via the write descriptor command. Usually, the first command issued by the host to a device is the write descriptor command. In fact, a successful write descriptor command is required before most other high level commands can be sent to the specific device. (Exceptions are Read Descriptor, BPP Test, and Open commands.)

7.1.5.1 Write Descriptor Command Packet

The table below defines the write descriptor command packet parameters.

Write Descriptor Command Packet

BYTE OFFSET	PARAMETER DESCRIPTION	
$00	command = $04	XX
$02	device type	unit number
$04	XX	XX
$06	address modifier	data bus width
$08	XXXX	
$0A	XXXX	
$0C	pointer to device descriptor table (MSW)	
$0E	pointer to device descriptor table (LSW)	
$10	XXXX	
$12	XXXX	
$14	XXXX	
$16	XXXX	
$18	XXXX	
$1A	XXXX	

OFFSET	DESCRIPTION
$00	Command value for write descriptor ($04).
$02	Device type.
$03	Unit number to which the command applies.

$06 Address modifier of the memory space where the device descriptor table is located.

$07 Data bus width is a code to indicate the width of the data bus. $01 indicates a 16-bit data bus, $02 indicates a 32-bit data bus.

$0C Pointer to the device descriptor table is a pointer into VME memory where the device descriptor table begins.

7.1.5.2 Write Descriptor Command Returned Status

The table defines the write descriptor command status packet parameters.

Write Descriptor Command Status Packet

BYTE OFFSET	PARAMETER DESCRIPTION	
$1C	fatal error code	recovered error status
$1E	additional error code/status	
$20	retry count	XX
$22	XXXX	
$24	XXXX	
$26	XXXX	
$28	XXXX	
$2A	XXXX	
$2C	XXXX	
$2E	status parameter 3	

OFFSET	DESCRIPTION
$1C	A nonzero fatal error code if the command did not complete successfully.
$1D	A nonzero recovered error status if the command completed successfully after retries.
$1E	Additional status error information if the fatal error code is nonzero.
$20	The number of retries that were performed by the MVME327A.
$2E	If fatal error code is $01 (bad descriptor) or $02 (bad command), then this field contains an offset into the descriptor table or command packet of the offending parameter.

7

7.1.6 Format Command ($05)

The format command ensures that the medium is formatted as specified by the write descriptor parameters. A list of defects may be supplied by the user so they can be locked out. This list, if supplied, is in one of two formats: defect list type 1 is a physical sector format, defect list type 2 is a bytes from index format. All data is lost on the area of the disk that is formatted. Many SCSI drives permit formatting the entire disk only.

Primary defect list:

Refers to the list of defects recorded on the medium (if any) by the manufacturer or written later by a specific command.

Grown defect list:

Includes defects identified to or by the controller. This list does not include the primary list of defects. These defects are classified as flaws appearing when the medium has been formatted and used to store and retrieve data.

Certification:

During format, after the format pattern is written, the disk is verified.

7.1.6.1 Format Command Packet

The table below defines the format command packet parameters.

Format Command Packet

BYTE OFFSET	PARAMETER DESCRIPTION	
$00	command = $05	command control
$02	device type	unit number
$04	XX	XX
$06	address modifier	data bus width
$08	starting cylinder	
$0A	starting head	reserved
$0C	pointer to defect list (MSW)	
$0E	pointer to defect list (LSW)	
$10	number of tracks (MSW)	
$12	number of tracks (LSW)	
$14	XXXX	
$16	defect list type	
$18	defect list count	
$1A	XXXX	

OFFSET	DESCRIPTION
$00	Command value for format ($05).
$01	Command control specifies the options for the format command.

7	6	5	4	3	2	1	0
CL	DP	DC	SF	res	res	res	res

BIT		DESCRIPTION
7	CL	Complete list clear and no defect list is supplied: use existing grown defect list, no user list is supplied. clear and defect list is supplied: use existing grown defect list, add user list to the grown list. set and no defect list is supplied: do not use existing grown defect list, no user list is supplied. set and defect list is supplied: do not use existing grown defect list, use user list as the complete defect list.
6	DP	Disable primary list of defects clear: use manufacturer's primary list if the controller/drive supports it. set: do not use manufacturer's primary list.
5	DC	Disable certification process clear: use certification if the controller/drive supports it. (Drive is certified at format time.) set: do not use certification.
4	SF	Stop format clear: stop formatting when any of the controller/drive defect lists cannot be accessed. set: format even if a controller/drive defect list cannot be accessed.
3	res - reserved.	
2	res - reserved.	
1	res - reserved.	
0	res - reserved.	

$02 Device type.

$03 Unit number to which the command applies.

$06 Address modifier of the memory space where the defect list is located.

$07 Data bus width is a code to indicate the width of the data bus. $01 indicates a 16-bit data bus, $02 indicates a 32-bit data bus.

$08 Cylinder number where formatting is to start. This field is ignored if the entire disk is to be formatted.

$0A Starting head number where formatting is to start. This field is ignored if the entire disk is to be formatted.

$0C Pointer to defect list is a pointer to the defect list in VME memory. This field is ignored if the defect list count is zero.

$10 Number of tracks to format. If the value is $FFFFFFFF, the entire disk is to be formatted, and the starting cylinder and head numbers are ignored.

$16 Format of the defect list: $01 - physical sector format (defect list type 1), $02 - cylinder, head, and bytes from index format (defect list type 2). Refer to the *Fixed Bad Spot Command* paragraph in this chapter. This field is ignored if the defect list count is zero.

$18 Defect list count is the number of entries in the defect list. If no defect list is supplied, this value must be zero.

7.1.6.2 Format Command Returned Status

The table defines the format command status packet parameters.

Format Command Status Packet

BYTE OFFSET	PARAMETER DESCRIPTION	
$1C	fatal error code	recovered error status
$1E	additional error code/status	
$20	retry count	XX
$22	XXXX	
$24	XXXX	
$26	XXXX	
$28	XXXX	
$2A	XXXX	
$2C	XXXX	
$2E	status parameter 3	

OFFSET	DESCRIPTION
$1C	A nonzero fatal error code if the command did not complete successfully.
$1D	A nonzero recovered error status if the command completed successfully after retries.
$1E	Additional status error information if the fatal error code is nonzero.
$20	The number of retries that were performed by the MVME327A.
$2E	If fatal error code is $02 (bad command), then this field contains an offset into the command packet of the offending parameter.

7.1.6.3 Defect List Formats

The defect list formats used by the format, read manufacturer's defect list and write manufacturer's defect list command, are defined in the following paragraphs.

Physical Sector Defect List (Type 1)

Each defect descriptor for the physical sector format specifies the beginning of a defect location on the medium. Each defect descriptor is comprised of the cylinder number, head number, and sector (within the track) of the defect. The defect descriptors MUST be in ascending order. For determining ascending order, the cylinder number is considered the most significant part of the address and the sector is considered the least significant. A defect sector of $FFFFFFFF indicates that the entire track shall be reassigned.

Physical Sector Format

BYTE NUMBER	DEFECT DESCRIPTOR
$00	cylinder number of defect (MSB)
$01	cylinder number of defect
$02	cylinder number of defect
$03	cylinder number of defect (LSB)
$04	head number of defect (MSB)
$05	head number of defect (LSB)
$06	defect sector on track(MSB)
$07	defect sector on track
$08	defect sector on track
$09	defect sector on track (LSB)

Bytes from Index Defect List (Type 2)

Each defect descriptor for the bytes from index format specifies the beginning of a defect location on the medium. Each defect descriptor is comprised of the cylinder number, head number, and bytes from index of the defect. The defect descriptors MUST be in ascending order. For determining ascending order, the cylinder number is considered the most significant part of the address and the bytes from index is considered the least significant. A defect bytes from index of $FFFFFFFF indicates that the entire track shall be reassigned.

Bytes From Index Format

BYTE NUMBER	DEFECT DESCRIPTOR
$00	cylinder number of defect (MSB)
$01	cylinder number of defect
$02	cylinder number of defect
$03	cylinder number of defect (LSB)
$04	head number of defect (MSB)
$05	head number of defect (LSB)
$06	defect bytes from index (MSB)
$07	defect bytes from index
$08	defect bytes from index
$09	defect bytes from index (LSB)

7.1.7 Fix Bad Spot Command ($06)

The fix bad spot command fixes a defective sector or track on the disk by whatever means the controller uses. It could map a sector, a track, or use any other method it needs. The location of the bad spot must be supplied. The fix bad spot command allows the host to fix bad spots on a disk without reformatting the disk.

7.1.7.1 Fix Bad Spot Command Packet

The table below defines the fix bad spot command packet parameters.

Fix Bad Spot Command Packet

BYTE OFFSET	PARAMETER DESCRIPTION	
$00	command = $06	command control
$02	device type	unit number
$04	XX	XX
$06	address modifier	data bus width
$08	XXXX	
$0A	XXXX	
$0C	pointer to defect list (MSW)	
$0E	pointer to defect list (LSW)	
$10	XXXX	
$12	XXXX	
$14	XXXX	
$16	XXXX	
$18	defect list count	
$1A	XXXX	

OFFSET	DESCRIPTION
$00	Command value for fix bad spot ($06).
$01	Command control specifies the options for the fix bad spot command.

7	6	5	4	3	2	1	0
res	res	res	res	res	res	res	alt type

BIT	DESCRIPTION
7	res - reserved.
6	res - reserved.
5	res - reserved.
4	res - reserved.
3	res - reserved.
2	res - reserved.
1	res - reserved.
0	alt type - defines the type of alteration to use. A zero indicates that MVME327A should attempt to do sector mapping, a one indicates that the MVME327A should map the entire track to an alternate track.

$02 Device type.

$03 Unit number to which the command applies.

$06 Address modifier of the memory space where the defect list is located.

$07 Data bus width is a code to indicate the width of the data bus. $01 indicates a 16-bit data bus, $02 indicates a 32-bit data bus.

$0C Pointer to defect list in the VME memory. The defect list is in the form of a list of logical sectors sorted in ascending order. Note that the MVME327A returns a logical sector number if an error occurs on data transfer commands. This returned value may be placed in the (sorted) defect list to fix the bad spot. Each entry in the defect list is a long word (32 bits).

$18 Defect list count is the number of entries in the defect list.

7.1.7.2 Fix Bad Spot Command Returned Status

The table defines the fix bad spot command status packet parameters.

Fix Bad Spot Command Status Packet

BYTE OFFSET	PARAMETER DESCRIPTION	
$1C	fatal error code	recovered error status
$1E	additional error code/status	
$20	retry count	XX
$22	XXXX	
$24	XXXX	
$26	XXXX	
$28	XXXX	
$2A	XXXX	
$2C	XXXX	
$2E	status parameter 3	

OFFSET DESCRIPTION

$1C A nonzero fatal error code if the command did not complete successfully.

$1D A nonzero recovered error status if the command completed successfully after retries.

$1E	Additional status error information if the fatal error code is nonzero.
$20	The number of retries that were performed by the MVME327A.
$2E	If fatal error code is $02 (bad command), then this field contains an offset into the command packet of the offending parameter.

7.1.8 Read Status Command ($10)

The read status command applies to tape media only and returns "drive not ready", "tape write protected", and filemark position information.

7.1.8.1 Read Status Command Packet

The table below defines the read status command packet parameters.

Read Status Command Packet

BYTE OFFSET	PARAMETER DESCRIPTION		
$00	command = $10		XX
$02	device type = $05		unit number
$04	XX		XX
$06	XX		XX
$08		XXXX	
$0A		XXXX	
$0C		XXXX	
$0E		XXXX	
$10		XXXX	
$12		XXXX	
$14		XXXX	
$16		XXXX	
$18		XXXX	
$1A		XXXX	

OFFSET	DESCRIPTION
$00	Command value for read status ($10).
$02	Device type must be SCSI ($05).
$03	Unit number to which the command applies.

7.1.8.2 Read Status Command Returned Status

The table defines the read status command status packet parameters.

Read Status Command Status Packet

BYTE OFFSET	PARAMETER DESCRIPTION
$1C	fatal error code / recovered error status
$1E	additional error code/status
$20	retry count / XX
$22	XXXX
$24	XXXX
$26	XXXX
$28	XXXX
$2A	filemark position (MSW)
$2C	filemark position (LSW)
$2E	status parameter 3

OFFSET	DESCRIPTION
$1C	A nonzero fatal error code if the command did not complete successfully. The minimum status is drive not ready, write protected, or no error.
$1D	A nonzero recovered error status if the command completed successfully after retries.
$1E	Additional status error information if the fatal error code is nonzero.
$20	The number of retries that were performed by the MVME327A.
$2A	For sequential devices (e.g., tape), filemark position count. If value returned is $FFFFFFFF, position is indeterminate. This field is unused for random access devices (e.g., disk).
$2E	If fatal error code is $02 (bad command), then this field contains an offset into the command packet of the offending parameter.

7.1.9 Load/Unload/Re-tension Command ($11)

This command performs a tape load, unload, or a re-tension.

7.1.9.1 Load/Unload/Re-tension Command Packet

The table below defines the load/unload/re-tension command packet parameters.

Load/Unload/Re-tension Command Packet

BYTE OFFSET	PARAMETER DESCRIPTION		
$00	command = $11		command control
$02	device type = $05		unit number
$04	XX		XX
$06	XX		XX
$08		XXXX	
$0A		XXXX	
$0C		XXXX	
$0E		XXXX	
$10		XXXX	
$12		XXXX	
$14		XXXX	
$16		XXXX	
$18		XXXX	
$1A		XXXX	

7

OFFSET	DESCRIPTION
$00	Command value for load/unload/re-tension ($11)
$01	This command control field contains flags that are used by the load/unload/re-tension command.

7	6	5	4	3	2	1	0
I	res	res	res	res	res	RET	LOAD

BIT	DESCRIPTION
7	I (immediate) - If this bit is set, status is returned after command is accepted (before operation is complete).
6	res - reserved.
5	res - reserved.
4	res - reserved.

	3	res - reserved.
	2	res - reserved.
	1	RET - If this bit is set, tape re-tension is performed on the tape media before the completion of this command. Re-tension usually forces the tape to move from BOT to EOT and back to BOT. This command option may not be available on all tape devices. On those not supporting a re-tension, this bit is ignored.
	0	LOAD - If this bit is set, the tape is positioned at BOT after the execution of the command. If this bit is not set, the tape may be positioned either at the BOT or the EOT, depending on the particular tape controller.
$02		Device type must be SCSI ($05).
$03		Unit number to which the command applies.

7.1.9.2 Load/Unload/Re-tension Command Returned Status

The table defines the load/unload/re-tension command status packet parameters.

Load/Unload/Re-tension Command Status Packet

BYTE OFFSET	PARAMETER DESCRIPTION
$1C	fatal error code / recovered error status
$1E	additional error code/status
$20	retry count / XX
$22	XXXX
$24	XXXX
$26	XXXX
$28	XXXX
$2A	filemark position (MSW)
$2C	filemark position (LSW)
$2E	status parameter 3

OFFSET	DESCRIPTION
$1C	A nonzero fatal error code if the command did not complete successfully.
$1D	A nonzero recovered error status if the command completed successfully after retries.

$1E	Additional status error information if the fatal error code is nonzero.
$20	The number of retries that were performed by the MVME327A.
$2A	For sequential devices (e.g., tape), filemark position count. If value returned is $FFFFFFFF, position is indeterminate. This field is unused for random access devices (e.g., disk).
$2E	If fatal error code is $02 (bad command), then this field contains an offset into the command packet of the offending parameter.

7.1.10 Write Filemark ($12)

The write filemark command is used to write one or more filemarks to the tape.

7.1.10.1 Write Filemark Command Packet

The table below defines the write filemark command packet parameters.

Write Filemark Command Packet

BYTE OFFSET	PARAMETER DESCRIPTION		
$00	command = $12		XX
$02	device type = $05		unit number
$04	XX		XX
$06	XX		XX
$08		XXXX	
$0A		XXXX	
$0C		XXXX	
$0E		XXXX	
$10		filemark count (MSW)	
$12		filemark count (LSW)	
$14		XXXX	
$16		XXXX	
$18		XXXX	
$1A		XXXX	

OFFSET	DESCRIPTION
$00	Command value for write filemark ($12).
$02	Device type must be SCSI ($05).
$03	Unit number to which the command applies.
$10	Number of sequential filemarks to be written on the tape.

7.1.10.2 Write Filemark Command Returned Status

The table defines the write filemark command status packet parameters.

Write Filemark Command Status Packet

BYTE OFFSET	PARAMETER DESCRIPTION	
$1C	fatal error code	recovered error status
$1E	additional error code/status	
$20	retry count	XX
$22	XXXX	
$24	XXXX	
$26	XXXX	
$28	XXXX	
$2A	filemark position (MSW)	
$2C	filemark position (LSW)	
$2E	status parameter 3	

OFFSET	DESCRIPTION
$1C	A nonzero fatal error code if the command did not complete successfully.
$1D	A nonzero recovered error status if the command completed successfully after retries.
$1E	Additional status error information if the fatal error code is nonzero.
$20	The number of retries that were performed by the MVME327A.
$2A	For sequential devices (e.g., tape), filemark position count. If value returned is $FFFFFFFF, position is indeterminate. This field is unused for random access devices (e.g., disk).
$2E	If fatal error code is $02 (bad command), then this field contains an offset into the command packet of the offending parameter.

7.1.11 Rewind Command ($13)

The rewind command causes the tape to be rewound to beginning-of-tape.

7.1.11.1 Rewind Command Packet

The table below defines the rewind command packet parameters.

Rewind Command Packet

BYTE OFFSET	PARAMETER DESCRIPTION		
$00	command = $13		command control
$02	device type = $05		unit number
$04	XX		XX
$06	XX		XX
$08		XXXX	
$0A		XXXX	
$0C		XXXX	
$0E		XXXX	
$10		XXXX	
$12		XXXX	
$14		XXXX	
$16		XXXX	
$18		XXXX	
$1A		XXXX	

7

OFFSET	DESCRIPTION
$00	Command value for rewind ($13)
$01	Command control specifies the options for the rewind command.

7	6	5	4	3	2	1	0
I	res	res	res	res	res	res	res

BIT	DESCRIPTION
6-0	res - reserved.
7	I (immediate) - if set, completion status is returned immediately upon acceptance of command by the controller (before rewind is complete).

$02	Device type must be SCSI ($05).
$03	Unit number to which the command applies.

7.1.11.2 Rewind Command Returned Status

The table defines the rewind command status packet parameters.

Rewind Command Status Packet

BYTE OFFSET	PARAMETER DESCRIPTION	
$1C	fatal error code	recovered error status
$1E	additional error code/status	
$20	retry count	XX
$22	XXXX	
$24	XXXX	
$26	XXXX	
$28	XXXX	
$2A	filemark position (MSW)	
$2C	filemark position (LSW)	
$2E	status parameter 3	

OFFSET	DESCRIPTION
$1C	A nonzero fatal error code if the command did not complete successfully.
$1D	A nonzero recovered error status if the command completed successfully after retries.
$1E	Additional status error information if the fatal error code is nonzero.
$20	The number of retries that were performed by the MVME327A.
$2A	For sequential devices (e.g., tape), filemark position count. If value returned is $FFFFFFFF, position is indeterminate. This field is unused for random access devices (e.g., disk).
$2E	If fatal error code is $02 (bad command), then this field contains an offset into the command packet of the offending parameter.

7.1.12 Erase Command ($14)

The erase command is used to erase a complete tape or if the controller is capable, a part of the tape.

7.1.12.1 Erase Command Packet

The table below defines the erase command packet parameters.

Erase Command Packet

BYTE OFFSET	PARAMETER DESCRIPTION		
$00	command = $14		command control
$02	device type = $05		unit number
$04	XX		XX
$06	XX		XX
$08		XXXX	
$0A		XXXX	
$0C		XXXX	
$0E		XXXX	
$10		XXXX	
$12		XXXX	
$14		XXXX	
$16		XXXX	
$18		XXXX	
$1A		XXXX	

OFFSET	DESCRIPTION
$00	Command value for erase ($14).
$01	Command control specifies the options for the rewind command.

7	6	5	4	3	2	1	0
res	res	res	res	res	res	res	S

BIT	DESCRIPTION
7-1	res - reserved.
0	S (short erase) - if set, controller attempts to perform an erase of several inches of tape at the current position. The amount is controller dependent.

$02	Device type must be SCSI ($05).
$03	Unit number to which the command applies.

7.1.12.2 Erase Command Returned Status

The table defines the erase command status packet parameters.

Erase Command Status Packet

BYTE OFFSET	PARAMETER DESCRIPTION
$1C	fatal error code / recovered error status
$1E	additional error code/status
$20	retry count / XX
$22	XXXX
$24	XXXX
$26	XXXX
$28	XXXX
$2A	filemark position (MSW)
$2C	filemark position (LSW)
$2E	status parameter 3

OFFSET	DESCRIPTION
$1C	A nonzero fatal error code if the command did not complete successfully.
$1D	A nonzero recovered error status if the command completed successfully after retries.
$1E	Additional status error information if the fatal error code is nonzero.
$20	The number of retries that were performed by the MVME327A.
$2A	For sequential devices (e.g., tape), filemark position count. If value returned is $FFFFFFFF, position is indeterminate. This field is unused for random access devices (e.g., disk).
$2E	If fatal error code is $02 (bad command), then this field contains an offset into the command packet of the offending parameter.

7.1.13 Space Command ($15)

The space command is used to position the tape. The tape can be positioned forward or backward. The positioning can be relative to the current position or to the beginning-of-tape.

7.1.13.1 Space Command Packet

The table below defines the space command packet parameters.

Space Command Packet

BYTE OFFSET	PARAMETER DESCRIPTION	
$00	command = $15	command control
$02	device type = $05	unit number
$04	XX	XX
$06	XX	XX
$08	XXXX	
$0A	XXXX	
$0C	XXXX	
$0E	XXXX	
$10	number of filemarks/blocks (MSW)	
$12	number of filemarks/blocks (LSW)	
$14	XXXX	
$16	XXXX	
$18	XXXX	
$1A	XXXX	

OFFSET	DESCRIPTION
$00	Command value for space ($15)
$01	This command control field contains flags that are used by the space command.

Command Control Field

7	6	5	4	3	2	1	0
res	res	res	res	res	mode	space type	

BIT	DESCRIPTION
7	res - reserved.
6	res - reserved.
5	res - reserved.
4	res - reserved.
3	res - reserved.

2 mode - 0 = space relative to current position, 1 = space relative to beginning-of-tape.

For mode = 0, the tape is positioned on the EOT side of the last block or filemark upon completion of the command for spacing in the forward direction (a positive value in the number of filemarks/blocks field). It is positioned on the BOT side of the last block or filemark upon the completion of the command for spacing in the reverse direction (a negative value in the number of filemarks/blocks field).

For mode = 1, the tape is positioned on the EOT side of the last block or filemark upon the completion of the command for spacing in the forward or reverse direction.

0 and 1 space type - Defines how the number of filemarks/blocks field is interpreted.

00 - the field is in units of logical blocks.

01 - the field is in units of filemarks.

10 - the field is in units of sequential filemarks. The tape is scanned for the number of sequential filemarks specified. This option is not supported by all tape controllers. Tape movement is in forward direction only.

11 - the tape is to be spaced to the physical end-of-data. The number of filemarks/blocks field is not used when this option is chosen. Tape movement is in forward direction only.

\$02 Device type must be SCSI (\$05).

\$03 Unit number to which the command applies.

\$10 The number of filemarks/blocks to space the tape. Refer to space type in the command control field of the packet for the interpretation of this field. The number of filemarks/blocks field is interpreted as a signed (2's complement) integer, necessary for backward tape movement.

7.1.13.2 Space Command Returned Status

The table defines the space command status packet parameters.

Space Command Status Packet

BYTE OFFSET	PARAMETER DESCRIPTION
$1C	fatal error code / recovered error status
$1E	additional error code/status
$20	retry count / XX
$22	XXXX
$24	XXXX
$26	XXXX
$28	XXXX
$2A	filemark position (MSW)
$2C	filemark position (LSW)
$2E	status parameter 3

OFFSET	DESCRIPTION
$1C	A nonzero fatal error code if the command did not complete successfully.
$1D	A nonzero recovered error status if the command completed successfully after retries.
$1E	Additional status error information if the fatal error code is nonzero.
$20	The number of retries that were performed by the MVME327A.
$2A	For sequential devices (e.g., tape), filemark position count. If value returned is $FFFFFFFF, position is indeterminate. This field is unused for random access devices (e.g., disk).
$2E	If fatal error code is $02 (bad command), then this field contains an offset into the command packet of the offending parameter.

7.1.14 Enable Target Command ($20)

Enable target command initializes a LUN on the MVME327A for target mode operation. This command executes an implicit channel reservation command. Only the host that issued the command is able to access and service this LUN.

7.1.14.1 Enable Target Command Packet

The table below defines the enable target command.

Enable Target Command Packet

BYTE OFFSET	PARAMETER DESCRIPTION		
$00	command = $20		XX
$02	device type = $05		unit number
$04	XX		XX
$06	XX		XX
$08		XXXX	
$0A		XXXX	
$0C		XXXX	
$0E		XXXX	
$10		XXXX	
$12		XXXX	
$14		XXXX	
$16		XXXX	
$18		XXXX	
$1A		XXXX	

OFFSET	DESCRIPTION
$00	Command value for enable target ($20).
$02	Device type must be SCSI ($05).
$03	Unit number: the most significant nibble corresponds to the SCSI address (0-7) of the MVME327A on the SCSI bus (refer to Set SCSI address command). The least significant nibble corresponds to the target LUN that is being accessed (0-7). Any nibble with a value greater than 7 results in an error.

7.1.14.2 Enable Target Command Returned Status

The table defines the enable target command status packet parameters.

Enable Target Command Status Packet

BYTE OFFSET	PARAMETER DESCRIPTION		
$1C	fatal error code		XX
$1E	additional error code/status		
$20	XX		XX
$22		XXXX	
$24		XXXX	
$26		XXXX	
$28		XXXX	

Enable Target Command Status Packet (cont'd)

BYTE OFFSET	PARAMETER DESCRIPTION
$2A	XXXX
$2C	XXXX
$2E	status parameter 3

OFFSET	DESCRIPTION
$1C	A nonzero fatal error code if the command did not complete successfully.
$1E	Additional status error information if the fatal error code is nonzero.
$2E	If fatal error code is $02 (bad command), then this field contains an offset into the command packet of the offending parameter.

7.1.15 **Disable Target Command ($21)**

Disable target command releases an MVME327A target LUN. After this command is executed, the LUN no longer exists until another target enable ($20) command is executed for that LUN. This command executes an implicit channel release command to free this LUN.

7.1.15.1 **Disable Target Command Packet**

The table below defines the disable target command.

Disable Target Command Packet

BYTE OFFSET	PARAMETER DESCRIPTION		
$00	command = $21		XX
$02	device type = $05		unit number
$04	XX		XX
$06	XX		XX
$08		XXXX	
$0A		XXXX	
$0C		XXXX	
$0E		XXXX	
$10		XXXX	
$12		XXXX	
$14		XXXX	
$16		XXXX	
$18		XXXX	
$1A		XXXX	

OFFSET	DESCRIPTION
$00	Command value for disable target ($21)
$02	Device type must be SCSI ($05).
$03	Unit number: the most significant nibble corresponds to the SCSI address (0-7) of the MVME327A on the SCSI bus (refer to Set SCSI Level Command). The least significant nibble corresponds to the target LUN that is being accessed (0-7). Any nibble with a value greater than 7 results in an error.

7.1.15.2 Disable Target Command Returned Status

The table defines the disable target command status packet parameters.

Disable Target Command Status Packet

BYTE OFFSET	PARAMETER DESCRIPTION	
$1C	fatal error code	XX
$1E	additional error code/status	
$20	XX	XX
$22	XXXX	
$24	XXXX	
$26	XXXX	
$28	XXXX	
$2A	XXXX	
$2C	XXXX	
$2E	status parameter 3	

OFFSET	DESCRIPTION
$1C	A nonzero fatal error code if the command did not complete successfully.
$1E	Additional status error information if the fatal error code is nonzero.
$2E	If fatal error code is $02 (bad command), then this field contains an offset into the command packet of the offending parameter.

7.1.16 Reserve Unit Command ($22)

Reserve unit command allows a host exclusive access to a target LUN.

7.1.16.1 Reserve Unit Command Packet

The table below defines the reserve unit command.

Reserve Unit Command Packet

BYTE OFFSET	PARAMETER DESCRIPTION		
$00	command = $22		command control
$02	device type		unit number
$04	XX		XX
$06	XX		XX
$08		XXXX	
$0A		XXXX	
$0C		XXXX	
$0E		XXXX	
$10		XXXX	
$12		XXXX	
$14		XXXX	
$16		XXXX	
$18		XXXX	
$1A		XXXX	

7

OFFSET — DESCRIPTION

$00 — Command value for reserve unit ($22).

$01 — Command control: the following bits are used by the reserve unit command:

Command Control Field

7	6	5	4	3	2	1	0
res	res	res	res	res	res	level1	level0

BIT — DESCRIPTION

7 — res - reserved.
6 — res - reserved.
5 — res - reserved.
4 — res - reserved.
3 — res - reserved.
2 — res - reserved.

1 Level 1 - If set, this bit indicates reservation level 1. This is used on the MVME327A to reserve on the SCSI bus. The MVME327A reserves the SCSI unit specified in the packet for the MVME327A by sending the RESERVE command to that unit.

0 Level 0 - If set, this bit indicates reservation level 0 (VMEbus reservation) and subsequent commands from the BPP channels other than the one reserved are rejected with a reservation error. This bit is used to lock out other VMEbus channels and to prevent them from using a reserved unit.

NOTE

At least one of the levels must be set. If neither is set, an error is returned.

Relevant bit combinations for SCSI devices:

Reservations for SCSI Devices

0	0	0	0	0	0	0	0	not allowed
0	0	0	0	0	0	0	1	VMEbus reservation only
0	0	0	0	0	0	1	0	SCSI reservation only
0	0	0	0	0	0	1	1	VMEbus and SCSI reservation

Relevant bit combinations for local devices:

Reservations for Local Devices

0	0	0	0	0	0	0	0	not allowed
0	0	0	0	0	0	0	1	VMEbus reservation only

$02 Device type.

$03 Unit number.

7.1.16.2 Reserve Unit Command Returned Status

The table defines the reserve unit command status packet parameters.

Reserve Unit Command Status Packet

BYTE OFFSET	PARAMETER DESCRIPTION	
$1C	fatal error code	recovered error status
$1E	additional error code/status	
$20	retry count	XX
$22	XXXX	
$24	XXXX	
$26	XXXX	
$28	XXXX	
$2A	XXXX	
$2C	XXXX	
$2E	status parameter 3	

OFFSET	DESCRIPTION
$1C	A nonzero fatal error code if the command did not complete successfully.
$1D	A nonzero recovered error status if the command completed successfully after retries.
$1E	Additional status error information if the fatal error code is nonzero.
$20	The number of retries that were performed by the MVME327A.
$2E	If fatal error code is $02 (bad command), then this field contains an offset into the command packet of the offending parameter.

7.1.17 Release Unit Command ($23)

Release unit command relinquishes a host from exclusive access to a target LUN.

7.1.17.1 Release Unit Command Packet

The table below defines the release unit command.

Release Unit Command Packet

BYTE OFFSET	PARAMETER DESCRIPTION		
$00	command = $23		command control
$02	device type		unit number
$04	XX		XX
$06	XX		XX
$08		XXXX	
$0A		XXXX	
$0C		XXXX	
$0E		XXXX	
$10		XXXX	
$12		XXXX	
$14		XXXX	
$16		XXXX	
$18		XXXX	
$1A		XXXX	

OFFSET	DESCRIPTION
$00	Command value for release unit ($23)
$01	Command control: the following bits are used by the release unit command:

Command Control Field

7	6	5	4	3	2	1	0
res	res	res	res	res	res	level1	level0

BIT	DESCRIPTION
7	res - reserved.
6	res - reserved.
5	res - reserved.
4	res - reserved.
3	res - reserved.
2	res - reserved.
1	Level 1 - If set, this bit indicates release level 1. This is used on the MVME327A to release the SCSI bus. The MVME327A releases the SCSI unit specified in the packet for the MVME327A by sending the RELEASE command to that unit.

0 Level 0 - If set, this bit indicates release level 0 (VMEbus release) and subsequent commands from the BPP channels other than the one released are accepted. This bit is used to release and allow other VMEbus channels to use a released unit.

NOTE

At least one of the levels must be set. If neither is set, an error is returned.

Relevant bit combinations for SCSI devices:

Release for SCSI Devices

0	0	0	0	0	0	0	0	not allowed
0	0	0	0	0	0	0	1	VMEbus release only
0	0	0	0	0	0	1	0	SCSI release only
0	0	0	0	0	0	1	1	VMEbus and SCSI release

Relevant bit combinations for local devices:

Release for Local Devices

0	0	0	0	0	0	0	0	not allowed
0	0	0	0	0	0	0	1	VMEbus release only

7

$02 Device type.

$03 Unit number.

7.1.17.2 Release Unit Command Returned Status

The table defines the release unit command status packet parameters.

Release Unit Command Status Packet

BYTE OFFSET	PARAMETER DESCRIPTION	
$1C	fatal error code	XX
$1E	additional error code/status	
$20	XX	XX
$22	XXXX	
$24	XXXX	
$26	XXXX	
$28	XXXX	
$2A	XXXX	
$2C	XXXX	
$2E	status parameter 3	

OFFSET	DESCRIPTION
$1C	A nonzero fatal error code if the command did not complete successfully.
$1E	Additional status error information if the fatal error code is nonzero.
$2E	If fatal error code is $02 (bad command), then this field contains an offset into the command packet of the offending parameter.

7.1.18 Reset SCSI Command ($25)

Reset SCSI command either resets all devices on the SCSI bus or attempts to reset a specified device.

7.1.18.1 Reset SCSI Command Packet

The table below defines the reset SCSI command.

Reset SCSI Command Packet

BYTE OFFSET	PARAMETER DESCRIPTION		
$00	command = $25		XX
$02	device type = $05		unit number
$04	XX		XX
$06	XX		XX
$08		XXXX	
$0A		XXXX	
$0C		XXXX	
$0E		XXXX	
$10		XXXX	
$12		XXXX	
$14		XXXX	
$16		XXXX	
$18		XXXX	
$1A		XXXX	

OFFSET	DESCRIPTION
$00	Command value for reset SCSI ($25)
$02	Device type must be SCSI ($05).
$03	Unit number: the most significant nibble corresponds to the SCSI address (0-7) of the device to be reset on the SCSI bus. The least significant nibble is not used. If the most significant nibble is 8, the entire SCSI bus is reset. If a bus reset is performed, any outstanding SCSI commands are returned to the callers with a reset status. If only a SCSI device is reset, then all pending commands to that SCSI device are aborted by the target. Resetting the SCSI device corresponding to the MVME327A (its own address, that is) results in an error.

CAUTION

RESETTING THE SCSI BUS CAUSES CURRENT COMMANDS BY OTHER INITIATORS OR OTHER HOSTS (IN A MULTI-PROCESSING ENVIRONMENT) TO BE TERMINATED.

7.1.18.2 Reset SCSI Command Returned Status

The table defines the reset SCSI command status packet parameters.

Reset SCSI Command Status Packet

BYTE OFFSET	PARAMETER DESCRIPTION
$1C	fatal error code XX
$1E	additional error code/status
$20	XX XX
$22	XXXX
$24	XXXX
$26	XXXX
$28	XXXX
$2A	XXXX
$2C	XXXX
$2E	status parameter 3

OFFSET	DESCRIPTION
$1C	A nonzero fatal error code if the command did not complete successfully.
$1E	Additional status error information if the fatal error code is nonzero.
$2E	If fatal error code is $02 (bad command), then this field contains an offset into the command packet of the offending parameter.

7.1.19 Custom SCSI Command ($26)

Custom SCSI command allows the user absolute control of commands dispatched over the SCSI bus. For more information regarding this command, refer to Appendix B.

7.1.19.1 **Custom SCSI Command Packet**

The table below defines the custom SCSI command.

Custom SCSI Command Packet

BYTE OFFSET	PARAMETER DESCRIPTION	
$00	command = $26	XX
$02	device type = $05	unit number
$04	XX	XX
$06	address modifier	data bus width
$08	SCSI specific packet pointer (MSW)	
$0A	SCSI specific packet pointer (LSW)	
$0C	pointer to scatter/gather list (MSW)	
$0E	pointer to scatter/gather list (LSW)	
$10	XXXX	
$12	XXXX	
$14	scatter/gather count	
$16	XXXX	
$18	XXXX	
$1A	XXXX	

OFFSET	DESCRIPTION
$00	Command value for custom SCSI ($26)
$02	Device type must be SCSI ($05).
$03	Unit number: the most significant nibble corresponds to the SCSI address (0-7) of the device to be accessed. The least significant nibble corresponds to the peripheral device LUN that is being accessed (0-7). Any nibble with a value greater than 7 results in an error. If the MVME327A SCSI address is used in the primary unit, an error results.
$06	Address modifier: if the particular action requested by the command requires data transfers over the VMEbus, this address modifier is used for the transfers.
$07	Data bus width: (01 = 16-bit, 02 = 32-bit).
$08	This address points to a SCSI specific packet. The SCSI specific packet is defined in the *SCSI specific packet* paragraph in Chapter 6.
$0C	If the scatter/gather count (offset $14) is nonzero, this field is a pointer to a scatter/gather table that contains a number of entries specified by the scatter/gather count.

$14 Scatter/gather count is the number of entries in the scatter/gather table. If the count is zero, the command is not using the scatter/gather feature.

7.1.19.2 Custom SCSI Command Returned Status

The table defines the custom SCSI command status packet parameters.

Custom SCSI Command Status Packet

BYTE OFFSET	PARAMETER DESCRIPTION
$1C	fatal error code — recovered error status
$1E	additional error code/status
$20	retry count — XX
$22	error status address (MSW)
$24	error status address (LSW)
$26	termination transfer count (MSW)
$28	termination transfer count (LSW)
$2A	XXXX
$2C	XXXX
$2E	status parameter 3

OFFSET	DESCRIPTION
$1C	A nonzero fatal error code if the command did not complete successfully.
$1D	A nonzero recovered error status if the command completed successfully after retries.
$1E	Additional status error information if the fatal error code is nonzero.
$20	The number of retries that were performed by the MVME327A.
$22	This is error status address. If a transfer terminates because of an error, this field has the logical block address at which the error occurred.
$26	The number of bytes successfully transferred.
$2E	If fatal error code is $02 (bad command), then this field contains an offset into the command packet of the offending parameter.

7.1.20 Self Test Command ($27)

The extended self test, called as a high level command stores a $00 value in the fatal error code field of the status packet upon a successful completion.

If the self test fails, a $CC value is stored in the fatal error code field of the status packet, and an extended status byte is stored in the diagnostic register and the additional status field of the status packet.

When self test is called as a high level command, it waits for all pending command packet operations to complete before executing, and all following commands are blocked until the self test completes, EXCEPT ANOTHER SELF TEST COMMAND.

If a second self test command is issued before the first command has completed, the results and further operation of the MVME327A is undefined.

7.1.20.1 Self Test Command Packet

The table below defines the self test command.

Self Test Command Packet

BYTE OFFSET	PARAMETER DESCRIPTION		
$00	command = $27		XX
$02	device type = $0F		XX
$04	XX		XX
$06	XX		XX
$08		XXXX	
$0A		XXXX	
$0C		XXXX	
$0E		XXXX	
$10		XXXX	
$12		XXXX	
$14		XXXX	
$16		XXXX	
$18		XXXX	
$1A		XXXX	

OFFSET	DESCRIPTION
$00	Command value for the self test ($27)
$02	Device type must be ($0F).

7.1.20.2 Self Test Command Returned Status

The table defines the self test command status packet parameters.

Self Test Command Status Packet

BYTE OFFSET	PARAMETER DESCRIPTION	
$1C	fatal error code	XX
$1E	additional error code/status	
$20	XX	XX
$22	XXXX	
$24	XXXX	
$26	XXXX	
$28	XXXX	
$2A	XXXX	
$2C	XXXX	
$2E	status parameter 3	

OFFSET — DESCRIPTION

$1C — A nonzero fatal error code if the command did not complete successfully.

$1E — This additional status information is the same as the diagnostic register with bit definition as follows:

Bit 0 - MPU failure

If bit 0 is set, then the MPU has failed to execute instructions properly.

Bit 1 - RAM failure

If bit 1 is set, then the onboard RAM has failed.

Bit 2 - ROM failure

If bit 2 is set, then the onboard ROM has failed.

Bit 3 - Map decoder failure

If bit 3 is set, then the self test has not been able to access the PI/T, module status register, SCSI controller, bus interface module, or the floppy disk controller.

Bit 4 - IRQ Failure

If bit 4 is set, then the IRQ test has failed.

Bit 5 - Timer failure

If bit 5 is set, then the timer test has failed.

Bit 6 - SCSI failure

If bit 6 is set, the SCSI interface has failed.

Bits 7-15 - Reserved.

$2E — If fatal error code is $02 (bad command), then this field contains an offset into the command packet of the offending parameter.

7.1.21 Target Wait Command ($28)

The target wait command is issued in anticipation of being selected as a target. This is the first of two BPP commands needed to execute as a target, a single SCSI command. For more information regarding this command, refer to Appendix C. This command does not return status until the LUN it represents is selected.

7.1.21.1 Target Wait Command Packet

The table below defines the target wait command.

Target Wait Command Packet

BYTE OFFSET	PARAMETER DESCRIPTION	
$00	command = $28	XX
$02	device type = $05	unit number
$04	XX	XX
$06	address modifier	data bus width
$08	SCSI specific packet pointer (MSW)	
$0A	SCSI specific packet pointer (LSW)	
$0C	XXXX	
$0E	XXXX	
$10	XXXX	
$12	XXXX	
$14	XXXX	
$16	XXXX	
$18	XXXX	
$1A	XXXX	

OFFSET	DESCRIPTION
$00	Command value for target wait ($28)
$02	Device type must be SCSI ($05).
$03	Unit number: the most significant nibble corresponds to the SCSI address (0-7) of the MVME327A. The least significant nibble corresponds to the peripheral device LUN (0-7). Any nibble with a value greater than 7 results in an error.
$06	Address modifier is used for accesses to the SCSI specific packet.
$07	Data bus width: (01 = 16-bit, 02 = 32-bit) for accesses to the SCSI specific packet.
$08	This address points to a SCSI specific packet. It is defined in *SCSI specific packet* paragraph in Chapter 6. However, for this command, only the SYNC bit in the control word is used. The rest of the SCSI specific packet should be cleared.

7.1.21.2 Target Wait Command Returned Status

The table defines the target wait command status packet parameters.

Target Wait Command Status Packet

BYTE OFFSET	PARAMETER DESCRIPTION
$1C	fatal error code XX
$1E	additional error code/status
$20	XX XX
$22	XXXX
$24	XXXX
$26	XXXX
$28	XXXX
$2A	XXXX
$2C	XXXX
$2E	status parameter 3

OFFSET	DESCRIPTION
$1C	A nonzero fatal error code if the command did not complete successfully.
$1E	Additional status error information if the fatal error code is nonzero.

$2E	If fatal error code is $02 (bad command), then this field contains an offset into the command packet of the offending parameter.

7.1.22 Target Execute Command ($29)

The target execute command is the second of two BPP commands needed to execute, as a target, a single SCSI command. For more information regarding this command, refer to Appendix C.

7.1.22.1 Target Execute Command Packet

The table below defines the target execute command.

Target Execute Command Packet

BYTE OFFSET	PARAMETER DESCRIPTION	
$00	command = $29	XX
$02	device type = $05	unit number
$04	XX	XX
$06	address modifier	data bus width
$08	SCSI specific packet pointer (MSW)	
$0A	SCSI specific packet pointer (LSW)	
$0C	pointer to scatter/gather list (MSW)	
$0E	pointer to scatter/gather list (LSW)	
$10	XXXX	
$12	XXXX	
$14	scatter/gather count	
$16	XXXX	
$18	XXXX	
$1A	XXXX	

OFFSET	DESCRIPTION
$00	Command value for target execute ($29).
$02	Device type must be SCSI ($05).
$03	Unit number: the most significant nibble corresponds to the SCSI address (0-7) of the SCSI device to be accessed. The least significant nibble corresponds to the peripheral device LUN that is being accessed (0-7). Any nibble with a value greater than 7 results in an error.

$06 Address modifier: the address modifier is used for accesses to the SCSI specific packet.

$07 Data bus width: (01 = 16-bit, 02 = 32-bit) access to the SCSI specific packet may use 32-bit data transfers if this byte is 02, otherwise only 16-bit or 8-bit accesses are used.

$08 This address points to a SCSI specific packet. It is defined in *SCSI specific packet* paragraph in Chapter 6.

$0C If the scatter/gather count (offset $14) is nonzero, this field is a pointer to a scatter/gather table that contains a number of entries specified by the scatter/gather count.

$14 Scatter/gather count is the number of entries in the scatter/gather table. If the count is zero, the command is not using the scatter/gather feature.

7.1.22.2 Target Execute Command Returned Status

The table defines the target execute command status packet parameters.

Target Execute Command Status Packet

BYTE OFFSET	PARAMETER DESCRIPTION
$1C	fatal error code XX
$1E	additional error code/status
$20	XX XX
$22	XXXX
$24	XXXX
$26	termination transfer count (MSW)
$28	termination transfer count (LSW)
$2A	XXXX
$2C	XXXX
$2E	status parameter 3

OFFSET DESCRIPTION

$1C A nonzero fatal error code if the command did not complete successfully.

$1E Additional status error information if the fatal error code is nonzero.

$22 This is error status address. If a transfer terminates because of an error, this field has the logical block address at which the error occurred.

$26	The number of bytes successfully transferred.
$2E	If fatal error code is $02 (bad command), then this field contains an offset into the command packet of the offending parameter.

7.1.23 Set SCSI Address Command ($2B)

The MVME327A defaults to the SCSI address selected by header J8. This command changes the SCSI address of the MVME327A. Care must be taken when issuing this command. For example, if any outstanding commands are disconnected on the SCSI bus with pending reselection, this command would destroy those pending threads. If reservations on the SCSI address are also pending on the SCSI bus, the system may experience disastrous results. This command is intended to be used during system initialization before any activity on the SCSI bus is initiated.

7.1.23.1 Set SCSI Address Command Packet

The table below defines the set SCSI address command.

Set SCSI Address Command Packet

BYTE OFFSET	PARAMETER DESCRIPTION		
$00	command = $2B		XX
$02	device type = $05		unit number = $80
$04	XX		XX
$06	XX		XX
$08		XXXX	
$0A		XXXX	
$0C		XXXX	
$0E		XXXX	
$10		XXXX	
$12		XXXX	
$14		XXXX	
$16		SCSI address	
$18		XXXX	
$1A		XXXX	

OFFSET	DESCRIPTION
$00	Command value for set SCSI address ($2B).
$02	Device type must be SCSI ($05).
$03	Unit number: must be $80.
$16	SCSI address is the new address (0 to 7).

7.1.23.2 Set SCSI Address Command Returned Status

The table defines the set SCSI address command status packet parameters.

Set SCSI Address Command Status Packet

BYTE OFFSET	PARAMETER DESCRIPTION	
$1C	fatal error code	XX
$1E	XXXX	
$20	XX	XX
$22	XXXX	
$24	XXXX	
$26	XXXX	
$28	XXXX	
$2A	XXXX	
$2C	XXXX	
$2E	status parameter 3	

OFFSET	DESCRIPTION
$1C	A nonzero fatal error code if the command did not complete successfully.
$2E	If fatal error code is $02 (bad command), then this field contains an offset into the command packet of the offending parameter.

7.1.24 Open Command ($2D)

The open command is used as a "safe" access to SCSI devices at system initialization time in order to obtain system configuration information which may be on the device.

With this command, the user is able to specify the maximum number of blocks and bytes needed without previous knowledge of the logical or physical block size of the device. No previous write descriptor command is required to the device. The read always starts with the first block on the media.

If the block count specified is too high for the number of bytes requested, only the requested number of bytes are transferred to the caller's buffer. The remaining bytes are still read from the device but not transferred across the VMEbus. No fatal error is returned for this condition unless attempt to read past end of media is made.

If the block count specified is too low for the number of bytes requested, a fatal error is returned after the fewer bytes are transferred.

7.1.24.1 Open Command Packet

The table below defines the open command.

Open Command Packet

BYTE OFFSET	PARAMETER DESCRIPTION	
$00	command = $2D	command control = $00
$02	device type = $05	unit number
$04	XX	XX
$06	address modifier	data bus width
$08	XXXX	
$0A	XXXX	
$0C	VME buffer address (MSW)	
$0E	VME buffer address (LSW)	
$10	transfer block count (MSW)	
$12	transfer block count (LSW)	
$14	scatter/gather count = $0000	
$16	XXXX	
$18	transfer byte count (MSW)	
$1A	transfer byte count (LSW)	

OFFSET	DESCRIPTION
$00	Command value for open ($2D).
$01	Command control must be $00.
$02	Device type must be SCSI ($05).
$03	Unit number to which the command applies.
$06	Address modifier of the memory space where the data buffer is located.
$07	Data bus width is a code to indicate the width of the data bus. $01 indicates a 16-bit data bus, $02 indicates a 32-bit data bus.
$0C	VME buffer address is a pointer to the destination VME memory where the data buffer is located.
$10	Transfer block count is the number of blocks of data to be read (maximum).
$14	Scatter/gather count must be zero. Scatter/gather cannot be used with this command.
$18	Transfer byte count is the requested number of bytes for this command to transfer. The user must assure that the buffer is of adequate size.

7.1.24.2 Open Command Returned Status

The table below defines the open command status packet parameters.

Open Command Status Packet

BYTE OFFSET	PARAMETER DESCRIPTION
$1C	fatal error code — recovered error status
$1E	additional error code/status
$20	retry count — XX
$22	error status address (MSW)
$24	error status address (LSW)
$26	termination transfer count (MSW)
$28	termination transfer count (LSW)
$2A	XXXX
$2C	XXXX
$2E	status parameter 3

OFFSET	DESCRIPTION
$1C	A nonzero fatal error code if the command did not complete successfully.
$1D	A nonzero recovered error status if the command completed successfully after retries.
$1E	Additional status error information if the fatal error code is nonzero.
$20	The number of retries that were performed by the MVME327A.
$22	This is error status address. If a transfer terminates because of an error, this field has the physical block address at which the error occurred.
$26	The number of bytes successfully transferred.
$2E	If fatal error code is $02 (bad command), then this field contains an offset into the command packet of the offending parameter.

APPENDIX A - MVME327A SUPPORTED SCSI CONTROLLERS/DEVICES

The following controllers/devices are supported in release 2.0 of the firmware.

CONTROLLER TYPE CODE	PERIPHERAL TYPE CODE	DESCRIPTION
$0F	$02	CCS Rev. 4B Winchester Disk Drives (Refer to list of tested controllers/ drives below)
$10	$02	CDC 94161 Winchester Disk Drive (Wren III)
$11	$02	Micropolis 1375 Winchester Disk Drive
$12	$05	Archive 2060S QIC24 Tape Drive Archive 2125S QIC24/120 Tape Drive Archive 2150S QIC24/120/150 Tape Drive
$13	$02	CDC 94171 Winchester Disk Drive (Wren IV)
$14	$02	Seagate ST-251N Winchester Disk Drive
$15	$02	CCS Rev. 4A Winchester Disk Drives
$17	$02	CCS Rev. 4B Synchronous Data Transfer Winchester Disk Drives
$18	$05	Exabyte 8200 Tape Drive

Common Command Set (CCS Rev. 4B) Tested Controllers/Drives

Some manufacturers SCSI Winchester drives are CCS compatible, but conform to an earlier revision of CCS (earlier than 4B).

Controller types $10, $11, and $13 may use the CCS controller type code of $0F, but suffer loss of features not included under the CCS specification. As an example, type $13 (Wren IV) is not able to use caching when operated as a CCS controller type (code $0F).

A

The following have been tested and operate satisfactorily using the controller type $0F on the MVME327A firmware, release 2.0. Inclusion in this list does not imply endorsement by Motorola Microcomputer Division, nor does exclusion from the list imply lack of endorsement or that a controller/drive will not operate satisfactorily with the MVME327A.

CDC 94161 (Wren III)
CDC 94171 (Wren IV)
Maxtor XT-4380S
Micropolis 1375
Seagate ST-296N

APPENDIX B - CUSTOM SCSI COMMAND EXAMPLE

INTRODUCTION

Most BPP high level commands that are initiated to a SCSI peripheral device, such as read ($01), write ($02), or write descriptor ($04), are translated by the MVME327A firmware into one or more SCSI bus commands in order to accomplish the function of the defined BPP high level command. This translation incorporates two pieces of intelligence. The first is a knowledge of the SCSI protocol and the second is a knowledge of the physical attributes of the peripheral and its controller. The user is relieved of knowing this information if the high level commands exercise all the functionally that the user requires and a firmware supported device is used. However, not all users interface with supported devices and some users require a broader spectrum of SCSI bus commands. The custom SCSI ($26) command is for the user who needs more flexibility than the BPP high level commands offer.

The custom SCSI command gives the user explicit control of commands initiated on the SCSI bus. This BPP command also bypasses most of the translation firmware and therefore reduces MVME327A firmware overhead time.

The key to the custom SCSI command is the SCSI specific packet (refer to *SCSI specific packet* paragraph in Chapter 6). For other high level commands the SCSI specific packet is generated by the firmware but for this command the SCSI specific packet is supplied by the user. The data contained in this packet controls not only the information exchanged during SCSI bus information transfer phases but also certain hardware and firmware features.

The SCSI specification defines an initiator as a SCSI device that requests an operation to be performed by another SCSI device (the target). However, it is the target that dictates the sequence of information phases to be executed on the bus. To keep track of the phase sequencing, a "script" is necessary. A script is the expected sequence of information phases to be executed on the SCSI bus. The script is used to resume a disconnected thread and to insure proper phase sequencing. Without a script, the firmware would have no way of checking whether the target performed the command that was requested because no attempt is made by the MVME327A firmware to interpret the command.

Script Value Definitions

$00	Data out phase
$01	Data in phase
$02	Command phase
$03	Status phase
$04	Undefined phase - DO NOT USE
$05	Undefined phase - DO NOT USE
$06	Message out phase
$07	Message in phase
$08	End of script

EXAMPLES

For the following examples, the "control word" in the SCSI specific packets reflect the following:

- Synchronous transfers will not be attempted; however, if the target requests synchronous data rate negotiations, the MVME327A establishes an acceptable rate. The user can read the message in the field in the SCSI specific packet of a successfully returned BPP packet to determine what synchronous data rate was established.

- Status checking is done by the firmware. If the user checks the status then the status byte in the SCSI specific packet for a completed custom SCSI command should be evaluated. When the user checks the status, a nonzero fatal error code may be returned in the BPP packet due to the SCSI status.

- The data bus width is 16 bits. The "data bus width" code in the BPP packet is $01.

- Byte swapping is not enabled.

- Scatter/gather is not used.

- Linked commands are not supported.

Additionally, the target device resides at SCSI address 2, LUN 0. The "primary unit number" in the BPP packet is $20.

For initiator mode, the message out bytes are what the user wants to send to the target (usually an "identify" message). The message in bytes, for a returned packet, are what the target sent the initiator.

Test Unit Ready Example

The test unit ready command provides a means of checking if the logical unit is ready. The table below defines the SCSI specific packet parameters.

SCSI Specific Packet Description

BYTE OFFSET	PARAMETER DESCRIPTION		
$00		$0000	
$02		$0000	
$04		$0000	
$06	$06		XX
$08	$00		$00
$0A	$00		$00
$0C	$00		$00
$0E	XX		XX
$10	XX		XX
$12	XX		XX
$14		$0000	
$16		$0000	
$18		$0000	
$1A		$0000	
$1C	XX		XX
$1E	$00		$00
$20		XXXX	
$22		XXXX	
$24		XXXX	
$26	XX		XX
$28	XX		XX
$2A	XX		XX
$2C		$0001	
$2E		XXXX	
$30		XXXX	
$32	$C0		XX
$34	XX		XX
$36	XX		XX
$38	$06		$02
$3A	$03		$07
$3C	$08		XX
$3E	XX		XX

The CDB for this command is six bytes long.
CDB = $00, $00, $00, $00, $00, $00.
No data is transferred.
The $C0 message out is an "identify" message and indicates that the initiator supports disconnects and is addressing LUN 0 of the selected device.
Script = $06, $02, $03, $07, $08.

B

Extended Read Example

The read command requests that the target transfer data to the initiator. Not all controllers support the extended read command. The table below defines the SCSI specific packet parameters.

SCSI Specific Packet Description

BYTE OFFSET	PARAMETER DESCRIPTION		
$00		$0000	
$02		$0000	
$04		$8000	
$06	$0A		XX
$08	$28		$00
$0A	$00		$07
$0C	$00		$00
$0E	$00		$10
$10	$00		$00
$12	XX		XX
$14		$0020	
$16		$0000	
$18		$0003	
$1A		$0000	
$1C	XX		XX
$1E	$00		$00
$20		XXXX	
$22		XXXX	
$24		XXXX	
$26	XX		XX
$28	XX		XX
$2A	XX		XX
$2C		$0001	
$2E		XXXX	
$30		XXXX	
$32	$C0		XX
$34	XX		XX
$36	XX		XX
$38	$06		$02
$3A	$01		$03
$3C	$07		$08
$3E	XX		XX

DMA is enabled.
The CDB for this command is ten ($0A) bytes long.
CDB = $28, $00, $00, $07, $00, $00, $00, $10, $00, $00.
$1000 blocks of data transferred from logical sector $70000 to VME address $30000.
For this device, each block equals $200 bytes.
"Identify with reselection" message is in the SCSI specific packet.
Script = $06, $02, $01, $03, $07, $08.

Request Sense Example

The request sense command requests that the target transfer sense data to the initiator. A request sense command is issued in response to a "check condition" status for the previous command. A check condition status is sent because of an error, exception, or abnormal condition. The sense data is device specific and its length and content vary from controller to controller. The table below defines the SCSI specific packet parameters.

SCSI Specific Packet Description

BYTE OFFSET	PARAMETER DESCRIPTION		
$00		$0000	
$02		$0000	
$04		$0080	
$06	$06		XX
$08	$03		$00
$0A	$00		$00
$0C	$14		$00
$0E	XX		XX
$10	XX		XX
$12	XX		XX
$14		$0000	
$16		$0014	
$18		$0004	
$1A		$0000	
$1C	XX		XX
$1E	$00		$00
$20		XXXX	
$22		XXXX	
$24		XXXX	
$26	XX		XX
$28	XX		XX
$2A	XX		XX
$2C		$0000	
$2E		XXXX	
$30		XXXX	
$32	XX		XX
$34	XX		XX
$36	XX		XX
$38	$02		$01
$3A	$03		$07
$3C	$08		XX
$3E	XX		XX

Message protocol has been turned off.
Transfers of less then $100 bytes are inefficient using DMA, therefore it is turned off.
The CDB for this command is six ($06) bytes long.

CDB = $03, $00, $00, $00, $14, $00.
This controller returns $14 bytes of sense data to VME address $40000.
Script = $02, $01, $03, $07, $08.
(Message in phase is still used for "command complete" message).

APPENDIX C - TARGET ROLE

INTRODUCTION

Target role is implemented on the MVME327A as a means for SCSI devices to communicate with VMEbus MPU modules. BPP is still utilized as the interface between the host VMEbus CPU and the MVME327A. The direction of flow on the buffered pipes is logically reversed for target role. That is, a CDB from a SCSI initiator is sent through the MVME327A to the host CPU on a status pipe and the response to the CDB is returned to the SCSI initiator via a command pipe between the MVME327A and the host CPU.

The MVME327A implementation of target role requires two BPP commands to execute one SCSI command. Through the target wait command ($28), the host CPU provides a free (empty) SCSI specific packet (refer to *SCSI specific packet* paragraph in Chapter 6) to the MVME327A to receive a CDB from the SCSI bus and return the CDB to the host CPU. The host CPU interprets the CDB, determines what action needs to be taken, fills out another SCSI specific packet, and issues a target execute command ($29).

Target Driver Algorithm

Listed below is the sequence of events necessary for a VMEbus module to act like a SCSI target device.

a. Create a BPP channel between the host and the MVME327A. A unique vector may be desired for target role service.

b. Issue an enable target command ($20) on the channel created in step a. The SCSI address of the MVME327A (0-7) is the most significant nibble and the desired LUN (0-7), to be enabled for target role, is the least significant nibble of the primary unit number in the BPP command packet.

c. Wait for the status envelope/packet to be returned for the enable target command of step b. If no error is encountered (fatal error code in BPP packet = 0), the desired LUN on the MVME327A is now enabled for target role. If the fatal error code <> 0, examine the error condition and determine what caused the error before proceeding.

d. Issue a target wait command ($28) on the channel created in step a. This command includes the free SCSI specific packet that is used by the MVME327A to return target service requests back to the host CPU. Because selection of the MVME327A as a target is at the control of the initiator, response to the target wait command depends on the selection of the MVME327A. If no SCSI initiator selects the enabled target LUN on the MVME327A, no status (for the target wait command) is returned to the host CPU. In other words, target service is totally asynchronous and unsolicited.

If and when the SCSI initiator selects this LUN on the MVME327A, the MVME327A reads the identify message (if ATN was activated during selection), reads in the CDB, sends a disconnect message (if the initiator supports disconnects), disconnects from the SCSI bus (if the initiator supports disconnects), and returns via the BPP channel (created in step a) the completion status for the target wait command. Additionally, the SCSI specific packet now contains messages (if used), the CDB, the SCSI address of the initiator, and a partial script.

e. Upon receiving good status (fatal error code in BPP packet = 0) for the target wait command, the host CPU has assumed the target role on the SCSI bus through the MVME327A. The application routine on the host CPU must now interpret the command in the CDB area of the SCSI specific packet and prepare the necessary data structures to perform the desired service.

f. The target service routine must now prepare a target execute command packet to allow the MVME327A to execute the desired services for the SCSI initiator. The SCSI specific packet associated with the target execute command contains all the information required to execute the target services. Information transfer phases are executed according to the script specified in the SCSI specific packet. The MVME327A automatically reselects the disconnected initiator (if disconnected) after receiving the target execute command. (The primary unit number in the target execute command packet must match the primary unit number in the enable target and target wait commands.) The MVME327A also automatically codes the correct identify message for the identification process after reselection, if necessary. The message in phase for the identification message is not to be included in the script. If a data phase is to be performed during the SCSI thread, then it is usually the first phase specified in the script area of the SCSI specific packet associated with the target execute command. If no data phase is to be performed, then the status phase is usually the first phase specified in the script. Scripts must end with an end of script code ($08) to allow the MVME327A to complete its services. Issue the properly prepared target execute command on the channel created in step a to service the selecting initiator on the SCSI bus.

g. Wait for status (fatal error code in the BPP packet = 0) for the target execute command. If no error occurred, the services were successfully performed on the SCSI bus by the MVME327A. If an error occurred, determine what measures to follow to correct the error condition.

h. Go to step d to continue target service or send a disable target command ($21) if target role is to be turned off for that LUN.

Script Value Definitions

$00	Data out phase
$01	Data in phase
$02	Command phase
$03	Status phase
$04	Undefined phase - DO NOT USE
$05	Undefined phase - DO NOT USE
$06	Message out phase
$07	Message in phase
$08	End of script
$0A	Return SCSI specific packet - initiator on SCSI bus
$0B	Return SCSI specific packet - initiator disconnected

The last byte in the script of the SCSI specific packet associated with a successfully completed BPP target wait command is either $0A or $0B.

EXAMPLES

For the following examples, the control word in the SCSI specific packet of the target execute commands reflect the following:

- Synchronous transfers are not attempted but are negotiated during a target wait command only. The SYNC bit is not set for a target wait command.
- Parity checking is not implemented.
- Status is sent by the target to the initiator. No status checking.
- The data bus width is 32 bits. The data bus width code in the BPP packet is $02.
- Byte swapping is not enabled.
- Scatter/gather is not used.
- Linked commands are not supported.
- Targets do not have control over whether message protocol is used or not.

Additionally,

- The MVME327A resides at SCSI address 7.
- A successful target enable command was issued to MVME327A LUN 2. The primary unit number in the BPP packet is $72.
- The target mode driver only supports SCSI processor device commands test unit ready, send, receive, and request sense.

In anticipation of being selected, the host issues a target wait command. The content of the included SCSI specific packet is unimportant but should be zeroed for easier interpretation after target wait status is returned.

Some time later a device on the SCSI bus selects the enabled LUN. After successful completion status for the target wait command (fatal error code in BPP packet = 0) is returned, the SCSI specific packet appears as follows:

SCSI Specific Packet Description

BYTE OFFSET	PARAMETER DESCRIPTION		
$00		XXXX	
$02		XXXX	
$04		XXXX	
$06	$06		XX
$08	$00		$40
$0A	$00		$00
$0C	$00		$00
$0E	XX		XX
$10	XX		XX
$12	XX		XX
$14		XXXX	
$16		XXXX	
$18		XXXX	
$1A		XXXX	
$1C	XX		$04
$1E	XX		XX
$20		XXXX	
$22		XXXX	
$24		XXXX	
$26	XX		XX
$28	XX		XX
$2A	XX		XX
$2C		XXXX	
$2E		XXXX	
$30		XXXX	
$32	XX		XX
$34	XX		XX
$36	XX		XX
$38	$02		$0A
$3A	XX		XX
$3C	XX		XX
$3E	XX		XX

a. The CDB contains a test unit ready command. A test unit ready command is issued to check if the LUN is ready to accept commands.

b. The initiator is at SCSI address 4.

c. The last byte in the script is an $0A (initiator on bus) indicating that the initiator does not support disconnection.

d. There are only two bytes in the script $02 (command phase) and $0A.

From the above information the target driver can conclude:

a. No message phases are indicated in the script. The initiator selected without asserting ATN.

b. The message area is not valid because selecting without ATN asserted means the initiator does not support the SCSI message protocol.

c. The CDB is valid for this driver. The driver executes this command.

It should be noted that the initiator is tying up the bus by not allowing disconnects.

A test unit ready command is issued to check if the LUN is ready to accept commands. Because it is, a good status is returned to the initiator. A test unit ready command does not involve a data transfer.

In response to the test unit ready command, a BPP target execute command is issued to MVME327A LUN 2 with the following associated SCSI specific packet.

C

SCSI Specific Packet Description

BYTE OFFSET	PARAMETER DESCRIPTION			
$00		$0000		
$02		$0000		
$04		$0000		
$06	XX		XX	
$08	XX		XX	
$0A	XX		XX	
$0C	XX		XX	
$0E	XX		XX	
$10	XX		XX	
$12	XX		XX	
$14		$0000		
$16		$0000		
$18		XXXX		
$1A		XXXX		
$1C	$00		XX	
$1E	$00		$00	
$20		$0001		
$22		XXXX		
$24		XXXX		
$26	$00		XX	
$28	XX		XX	
$2A	XX		XX	
$2C		XXXX		
$2E		XXXX		
$30		XXXX		
$32	XX		XX	
$34	XX		XX	
$36	XX		XX	
$38	$03		$07	
$3A	$08		XX	
$3C	XX		XX	
$3E	XX		XX	

Upon receiving successful completion status for the target execute command (fatal error code in the BPP packet = 0) the driver issues another target wait command.

The MVME327A LUN 2 is selected again. After successful completion status for the target wait command (fatal error code in the BPP packet = 0) is returned, the SCSI specific packet appears as follows:

SCSI Specific Packet Description

BYTE OFFSET	PARAMETER DESCRIPTION		
$00		XXXX	
$02		XXXX	
$04		XXXX	
$06	$06		XX
$08	$08		$40
$0A	$10		$00
$0C	$00		$00
$0E	XX		XX
$10	XX		XX
$12	XX		XX
$14		XXXX	
$16		XXXX	
$18		XXXX	
$1A		XXXX	
$1C	XX		$06
$1E	XX		XX
$20		$0001	
$22		XXXX	
$24		XXXX	
$26	$01		$03
$28	$01		$32
$2A	$04		$04
$2C		XXXX	
$2E		XXXX	
$30		XXXX	
$32	$C2		$01
$34	$03		$01
$36	$32		$04
$38	$07		$02
$3A	$07		$0B
$3C	XX		XX
$3E	XX		XX

a. The CDB contains a receive command. A receive command requests that the target transfer bytes of data to the initiator.

b. The initiator is at SCSI address 6.

c. The last byte in the script is $0B.

d. The script contains several bytes including two message-in phase bytes.

From the above information the target driver can conclude:

a. The initiator supports message protocol because it is disconnected.

b. This is the first command to the LUN from that initiator since power up. Because this initiator supports synchronous data transfers, synchronous data transfer rate negotiation was attempted.

c. The negotiated rate can be determined by examining the message out field. If message out byte 2 = $07 (message reject) then data transfers are synchronous. For this example, a synchronous data rate was established. To interpret the negotiated rate, refer to paragraph 5.5.5 Synchronous Data Transfer Request Message in the SCSI specification.

d. The CDB is valid for this driver. The driver executes this command.

The receive command is requesting that the target transfer $100000 bytes of data to the initiator. The driver has $100000 bytes of data at VME address $400000 that the initiator wants. A good status is to be returned to the initiator. DMA is enabled.

In response to the receive command, a BPP target execute command is issued with the following associated SCSI specific packet:

SCSI Specific Packet Description

BYTE OFFSET	PARAMETER DESCRIPTION		
$00		$0000	
$02		$0000	
$04		$8800	
$06	XX		XX
$08	XX		XX
$0A	XX		XX
$0C	XX		XX
$0E	XX		XX
$10	XX		XX
$12	XX		XX
$14		$0010	
$16		$0000	
$18		$0004	
$1A		$0000	
$1C	$00		XX
$1E	$00		$00
$20		$0001	
$22		XXXX	
$24		XXXX	
$26	$00		XX
$28	XX		XX
$2A	XX		XX
$2C		XXXX	
$2E		XXXX	
$30		XXXX	
$32	XX		XX
$34	XX		XX
$36	XX		XX
$38	$01		$03
$3A	$07		$08
$3C	XX		XX
$3E	XX		XX

Upon receiving successful completion status for the target execute command (fatal error code in BPP packet = 0) the driver issues another target wait command.

The MVME327A LUN 2 is selected again. After successful completion status for the target wait command (fatal error code in BPP packet = 0) is returned, the SCSI specific packet appears as follows:

SCSI Specific Packet Description

BYTE OFFSET	PARAMETER DESCRIPTION		
$00		XXXX	
$02		XXXX	
$04		XXXX	
$06	$06		XX
$08	$12		$40
$0A	$00		$00
$0C	$20		$00
$0E	XX		XX
$10	XX		XX
$12	XX		XX
$14		XXXX	
$16		XXXX	
$18		XXXX	
$1A		XXXX	
$1C	XX		$04
$1E	XX		XX
$20		XXXX	
$22		XXXX	
$24		XXXX	
$26	XX		XX
$28	XX		XX
$2A	XX		XX
$2C		XXXX	
$2E		XXXX	
$30		XXXX	
$32	XX		XX
$34	XX		XX
$36	XX		XX
$38	$02		$0A
$3A	XX		XX
$3C	XX		XX
$3E	XX		XX

a. The CDB contains an inquiry command.

b. The initiator is at SCSI address 4.

c. The last byte in the script is $0A.

From this information the target driver can conclude that the CDB is invalid for this driver.

The driver makes no attempt to fulfill the inquiry request. A check status is returned to the initiator indicating that there is something wrong and that the next command that this initiator (4) sends should be a request sense.

In response to the invalid command, a BPP target execute command is issued with the following associated SCSI specific packet:

SCSI Specific Packet Description

BYTE OFFSET	PARAMETER DESCRIPTION		
$00		$0000	
$02		$0000	
$04		$0000	
$06	XX		XX
$08	XX		XX
$0A	XX		XX
$0C	XX		XX
$0E	XX		XX
$10	XX		XX
$12	XX		XX
$14		$0000	
$16		$0000	
$18		XXXX	
$1A		XXXX	
$1C	$02		XX
$1E	$00		$00
$20		$0001	
$22		XXXX	
$24		XXXX	
$26	$00		XX
$28	XX		XX
$2A	XX		XX
$2C		XXXX	
$2E		XXXX	
$30		XXXX	
$32	XX		XX
$34	XX		XX
$36	XX		XX
$38	$03		$07
$3A	$08		XX
$3C	XX		XX
$3E	XX		XX

Upon receiving successful completion status for the target execute command (fatal error code in BPP packet = 0) the driver issues another target wait command.

The MVME327A LUN 2 is selected for the fourth time. After successful completion status for the target wait command (fatal error code in BPP packet = 0) is returned, the SCSI specific packet appears as follows:

C

SCSI Specific Packet Description

BYTE OFFSET	PARAMETER DESCRIPTION		
$00		XXXX	
$02		XXXX	
$04		XXXX	
$06	XX		XX
$08	$0A		$40
$0A	$00		$84
$0C	$00		$00
$0E	XX		XX
$10	XX		XX
$12	XX		XX
$14		XXXX	
$16		XXXX	
$18		XXXX	
$1A		XXXX	
$1C	XX		$06
$1E	XX		XX
$20		XXXX	
$22		XXXX	
$24		XXXX	
$26	$04		XX
$28	XX		XX
$2A	XX		XX
$2C		XXXX	
$2E		XXXX	
$30		XXXX	
$32	XX		XX
$34	XX		XX
$36	XX		XX
$38	$02		$07
$3A	$0B		XX
$3C	XX		XX
$3E	XX		XX

a. The CDB contains a send command. A send command requests that the target accept data bytes transferred by the initiator.

b. The initiator is at SCSI address 6.

c. The last byte in the script is $0B.

d. The script contains only three bytes including $02 (command phase) and $07 (message in phase).

From this information the target driver can conclude:

a. This initiator supports message protocol because it is disconnected.

b. The request sense data for the check status just sent is not for this initiator.

c. The CDB is valid for this driver. The driver executes this command.

The send command is requesting that the target accept $8400 bytes of data from the initiator. The driver has a buffer at VME address $80000 for saving incoming data. A good status is to be returned to the initiator. DMA is enabled.

In response to the send command, a BPP target execute command is issued to the MVME327A LUN 2 with the following SCSI specific packet:

SCSI Specific Packet Description

BYTE OFFSET	PARAMETER DESCRIPTION		
$00		$0000	
$02		$0000	
$04		$8800	
$06	XX		XX
$08	XX		XX
$0A	XX		XX
$0C	XX		XX
$0E	XX		XX
$10	XX		XX
$12	XX		XX
$14		$0000	
$16		$8400	
$18		$0008	
$1A		$0000	
$1C	$00		XX
$1E	$00		$00
$20		$0001	
$22		XXXX	
$24		XXXX	
$26	$00		XX
$28	XX		XX
$2A	XX		XX
$2C		XXXX	
$2E		XXXX	
$30		XXXX	
$32	XX		XX
$34	XX		XX
$36	XX		XX
$38	$00		$03
$3A	$07		$08
$3C	XX		XX
$3E	XX		XX

Upon receiving successful completion status for the target execute command (fatal error code in BPP packet = 0) the driver issues another target wait command.

The MVME327A LUN 2 is selected again. After successful completion for the target wait command (fatal error code in BPP packet = 0) is returned, the SCSI specific packet appears as follows:

SCSI Specific Packet Description

BYTE OFFSET	PARAMETER DESCRIPTION		
$00		XXXX	
$02		XXXX	
$04		XXXX	
$06	XX		XX
$08	$03		$40
$0A	$00		$00
$0C	$20		$00
$0E	XX		XX
$10	XX		XX
$12	XX		XX
$14		XXXX	
$16		XXXX	
$18		XXXX	
$1A		XXXX	
$1C	XX		$04
$1E	XX		XX
$20		XXXX	
$22		XXXX	
$24		XXXX	
$26	XX		XX
$28	XX		XX
$2A	XX		XX
$2C		XXXX	
$2E		XXXX	
$30		XXXX	
$32	XX		XX
$34	XX		XX
$36	XX		XX
$38	$02		$0A
$3A	XX		XX
$3C	XX		XX
$3E	XX		XX

a. The CDB contains a request sense command. A request sense command requests that the target send sense data to the initiator.

b. The initiator is at SCSI address 4.

c. The last byte in the script is $0A.

Form this information the driver can conclude:

a. There is current sense data for this initiator.

b. The CDB is valid for this driver. The driver executes this command.

The request sense command command is requesting that the target transfer $20 bytes of sense data to the initiator. The sense data is located at VME address $60000. For more information regarding the contents of the sense data that should be transferred to the initiator, refer to paragraph 7.1.2 Request Sense Command in the SCSI specification.

In response to the request sense command, a BPP target execute command is issued to the MVME327A LUN 2 with the following SCSI specific packet:

SCSI Specific Packet Description

BYTE OFFSET	PARAMETER DESCRIPTION		
$00		$0000	
$02		$0000	
$04		$0000	
$06	XX		XX
$08	XX		XX
$0A	XX		XX
$0C	XX		XX
$0E	XX		XX
$10	XX		XX
$12	XX		XX
$14		$0000	
$16		$0020	
$18		$0006	
$1A		$0000	
$1C	$00		XX
$1E	$00		$00
$20		$0001	
$22		XXXX	
$24		XXXX	
$26	$00		XX
$28	XX		XX
$2A	XX		XX
$2C		XXXX	
$2E		XXXX	
$30		XXXX	
$32	XX		XX
$34	XX		XX
$36	XX		XX
$38	$00		$03
$3A	$07		$08
$3C	XX		XX
$3E	XX		XX

Upon receiving successful completion for the target execute command (fatal error code in BPP packet = 0) the driver determines that this (MVME327A LUN 2) is no longer needed and issues a BPP disable target command.

APPENDIX D - FATAL ERROR CODES

INTRODUCTION

This appendix lists the fatal error codes and a short description of each error.

The table below defines the fatal error codes.

Fatal Error Codes

CODE (HEX)	ERROR DESCRIPTION	NOTES
$00	Good	1
	$01-0F Command Parameter Errors	
$01	Bad descriptor	2,3
$02	Bad command	2,3
$03	Unimplemented command	
$04	Bad drive	3
$05	Bad logical address	3
$06	Bad scatter/gather table	
$07	Unimplemented device	
$08	Unit not initialized	3
	$10-1F Media Errors	
$10	No ID found on track	3
$11	Seek error	3
$12	Relocated track error	3
$13	Record not found, bad ID	3
$14	Data sync fault	3
$15	ECC error	3
$16	Record not found	3
$17	Media error	3
	$20-2F Drive Errors	
$20	Drive fault	3
$21	Write protected media	3
$22	Motor not on	3
$23	Door open	3
$24	Drive not ready	3
$25	Drive busy	3

Fatal Error Codes (cont'd)

CODE (HEX)	ERROR DESCRIPTION	NOTES
	$30-3F VME DMA Errors	
$30	VMEbus error	3,4
$31	Bad address alignment	3
$32	Bus time-out	3
$33	Invalid DMA transfer count	3
	$40-4F Disk Format Errors	
$40	Not enough alternates	3
$41	Format failed	3
$42	Verify error	3
$43	Bad format parameters	3
$44	Cannot fix bad spot	3
$45	Too many defects	3
	$80-FF MVME327A Specific Errors	
$80	SCSI error, additional status available	3
$81	Indeterminate media error, no additional information	3
$82	Indeterminate hardware error	3
$83	Blank check (EOD or corrupted WORM)	3
$84	Incomplete extended message from target	3,5
$85	Invalid reselection by an unthreaded target	3,5
$86	No status returned from target	3,5
$87	Message out not transferred to target	3,5
$88	Message in not received from target	3,5
$89	Incomplete data read to private buffer	3
$8A	Incomplete data write from private buffer	3
$8B	Incorrect CDB size was given	3,5
$8C	Undefined SCSI phase was requested	3,5
$8D	Time-out occurred during a select phase	3
$8E	Command terminated due to SCSI bus reset	3
$8F	Invalid message received	3,5
$90	Command not received	6
$91	Unexpected status phase	3
$92	SCSI script mismatch	3,5,9
$93	Unexpected disconnect caused command failure	3
$94	Request sense command was not successful	10
$95	No write descriptor for controller drive	7
$96	Incomplete data transfer	3
$97	Out of local resources for command processing	11
$98	Local memory resources lost	
$99	Channel reserved for another VME host	12
$9A	Device reserved for another SCSI device	12

Fatal Error Codes (cont'd)

CODE (HEX)	ERROR DESCRIPTION	NOTES
$9B	Already enabled, expecting target response	6
$9C	Target not enabled	6
$9D	Unsupported controller type	7
$9E	Unsupported peripheral device type	7
$9F	Block size mismatch	8
$A0	Invalid cylinder number in format defect list	7
$A1	Invalid head number in format defect list	7
$A2	Block size mismatch--nonfatal	8
$A3	Our SCSI ID was not changed by command	13
$A4	Our SCSI ID has changed	6,14
$A5	No target enable has been completed	6
$A6	Cannot do longword transfers	7
$A7	Cannot do DMA transfers	7
$A8	Invalid logical block size	7,8
$A9	Sectors per track mismatch	7
$AA	Number of heads mismatch	7
$AB	Number of cylinders mismatch	7
$AC	Invalid floppy parameter(s)	
$AD	Already reserved	12
$AE	Was not reserved	12
$AF	Invalid sector number	7
$CC	Self test failed	

NOTES:
1. The termination transfer count is always valid for a command that transfers data.
2. The bad byte is indicated by its offset value in status parameter 3. If the value is -1 ($FFFF) then the bad byte is not indicated.
3. Additional status information may be available in the additional error code/status field of the BPP packet.
4. VMEbus error address contained in error status address field of BPP packet is currently not valid.
5. SCSI processing may not have finished. A SCSI bus reset command may need to be executed to put the SCSI bus in a known state. Refer to caution regarding SCSI bus resets.
6. Target mode only.
7. Designated parameter is in error.
8. Block size requested does not correspond to block size of device.

Fatal Error Codes (cont'd)

CODE (HEX)	ERROR DESCRIPTION	NOTES

9. Target device did not behave as indicated by the script in the SCSI specific packet.

10. Error condition flagged by target device cannot be reported. Probably due to a hardware problem.

11. Command cannot be executed because local resources required exceed available local resources. Resubmit command when MVME327A is less busy.

12. Valid for the reserve/release commands.

13. Set SCSI address command was unsuccessful because at least one SCSI command was in progress.

14. Set SCSI address command was issued and all pending target wait commands were returned. Set SCSI address command may or may not be successful.

APPENDIX E - ADDITIONAL ERROR CODES

Error code definitions for "additional error code" field of BPP packet.

Hex Value	Message
00	no additional error code
01	no index/sector signal
02	no seek complete
03	write fault
04	drive not ready
05	drive not defined
06	track ZERO not found
07	multiple drives selected
08	logical unit communications failure
09	track following error
0A-0F	are RESERVED
10	ID CRC or ECC error
11	unrecovered READ error
12	no address mark found for ID field
13	no address mark found for data area
14	no record found
15	seek positioning error
16	data synchronization mark error
17	recovered data with target read retries
18	recovered data with ECC correction
19	defect list error
1A	parameter overrun - parameter list too long
1B	synchronous transfer error
1C	primary defect list not found
1D	all bytes did not compare during a VERIFY cmd
1E	recovered ID with ECC correction
1F	is RESERVED
20	invalid command operation code
21	illegal logical block address
22	illegal function for device type
23	is RESERVED
24	illegal field in CDB
25	invalid LUN
26	invalid field in parameter list
27	disk is write protected
28	medium change
29	power on or bus device reset
2A	mode select parameters have changed
2B-2F	are RESERVED
30	incompatible cartridge
31	medium format corrupted
32	no defect spare location available

Hex Value	Message
33-3F	are RESERVED
40	RAM failure
41	data path diagnostic failure
42	power on diagnostic failure
43	message reject error
44	internal controller error
45	select/reselect failed
46	unsuccessful 'soft' reset
47	SCSI interface parity error
48	initiator detected error
49	inappropriate/illegal message
4A-4F	are RESERVED
50-5F	are RESERVED
60-6F	are RESERVED
70-7F	are RESERVED

80 through FF are vendor unique error codes

APPENDIX F - C FUNCTION EXAMPLES OF BPP PROTOCOL

The following is an example file written in C illustrating one way to implement the protocol defined in this manual. Motorola makes no claims for this code as to suitability or fitness for the user's application.

```
/* This file contains example C functions for
 *      crechan -- creating a BPP channel
 *      delchan -- deleting a BPP channel
 *      sendpkt -- enqueuing a packet to the command pipe
 *      rcvpkt  -- dequeuing a packet from the status pipe
 *
 * Also included is creation and management of a pool of BPP
 * envelopes.
 */

#define  BASE       0xffffa600  /* base address of MVME327A board */
#define  NENV       10          /* number of envelopes */
#define  CREATE     0x001       /* create channel command */
#define  DELETE     0x002       /* delete channel command */
#define  TASBIT     0x8000
#define  CMD_VALID  0x4000
#define  STAT_VALID 0x2000
#define  CMD_COMPL  0x1000
#define  ATTENTION  0x20
#define  BUSY       0x80
#define  TIMEOUT    0xff
#define  VALID      1

/* structure definition for a command packet */
struct cmdpkt
   {
   unsigned char   command;        /* command */
   unsigned char   cmdcntrl;       /* command control */
   unsigned char   dev;            /* device */
   unsigned char   unit;           /* unit number */
   unsigned char   secdev;         /* secondary device */
   unsigned char   secunit;        /* secondary unit */
   unsigned char   addmod;         /* address modifier */
   unsigned char   dbuswidth;      /* data bus width */
   unsigned char  *priaddr;        /* primary address */
   unsigned char  *secaddr;        /* secondary address */
   unsigned long   tcount;         /* transfer count in blocks */
   unsigned short  sgcount;        /* number of S/G entries in list */
   unsigned short  cmdpar1;        /* command parameter 1 */
   unsigned short  cmdpar2;        /* command parameter 2 */
   unsigned short  cmdpar3;        /* command parameter 3 */
```

F

```
    unsigned char   fatal;          /* fatal error code */
    unsigned char   recovered;      /* recovered error code */
    unsigned short  addstat;        /* additional error status */
    unsigned char   retrycnt;       /* number of retries */
    unsigned char   reserved;       /* reserved */
    unsigned char  *errstadd;       /* error status address */
    unsigned long   termcnt;        /* termination transfer count */
    unsigned short  statpar1;       /* status parameter 1 */
    unsigned short  statpar2;       /* status parameter 2 */
    unsigned short  statpar3;       /* status parameter 3 */
    };

/* structure definition of a BPP envelope */
struct vmeenv
    {
    struct vmeenv *link;
    struct cmdpkt *vpkt;
    unsigned char valid;
    char          fill1;
    short         fill2;
    };

/* structure definition of the MVME327A CSR register */
struct csrreg
    {
    long          *addreg;      /* address register */
    unsigned char addmod;       /* address modifier */
    unsigned char dbwidth;      /* data bus width */
    unsigned char ctrl;         /* control register */
    unsigned char fill1;
    unsigned char stat;         /* status register */
    unsigned char diagreg;      /* diagnostic register */
    long          fill2;
    unsigned short tasreg;      /* TAS register */
    };

/* structure definition of a BPP channel header in VME memory */
struct vcch
    {
    struct vmeenv *cphp;        /* command pipe head ptr. */
    struct vmeenv *cptp;        /* command pipe tail ptr. */
    struct vmeenv *sphp;        /* status pipe head ptr. */
    struct vmeenv *sptp;        /* status pipe tail ptr. */
    unsigned char level;        /* interrupt level */
    unsigned char vector;       /* interrupt vector number */
    unsigned char prior;        /* channel priority */
    unsigned char addmod;       /* address modifier */
    unsigned char chan;         /* channel number */
    unsigned char valid;        /* valid flag */
    unsigned char dbwidth;      /* data bus width */
    unsigned char fill;
    };
```

```
/* global variables */
struct vcch ch;                 /* channel header */
struct vmeenv env[NENV];        /* envelope pool  */
struct vmeenv *fenvptr;         /* free env ptr.  */

/**********************************/
/* crechan - create a BPP channel */
/**********************************/

crechan()
   {
   int i;
   struct vmeenv *eptr;
   struct vmeenv *tptr;
   struct csrreg *csrptr;
   char docmd();

   csrptr = (struct csrreg *)BASE;
   ch.valid = 0;
   ch.level = 3;            /* interrupt level 3 */
   ch.prior = 1;            /* channel priority 1 */
   ch.vector = 0x60;        /* interrupt vector number 0x60 */
   ch.addmod = 0x3d;        /* address modifier */

   /* initialize the bpp envelopes */
   fenvptr = env;
   ch.cphp = ch.cptp = fenvptr;  /* null envelope in command pipe */
   ch.cphp->link = 0;
   ch.cphp->valid = 0;
   fenvptr += 1;
   ch.sphp = ch.sptp = fenvptr;  /* null envelope in status pipe */
   ch.sphp->link = 0;
   ch.sphp->valid = 0;
   /* link rest of envelopes in free pool */
   fenvptr += 1;
   tptr = fenvptr;

   for (i = 0; i < (NENV-2); i++)
      {
      eptr = tptr;
      tptr += 1;
      eptr->link = tptr;
      eptr->valid = 0;
      }
   eptr->link = 0;
   eptr->valid = 0;

   /* try to open the channel */

   if(docmd(CREATE,&ch,csrptr) != 0)
      {
      printf("Channel NOT opened due to timeout.\n");
      return;
      }
```

```
    if(csrptr->stat != 0)
       {
       printf("Channel NOT opened: status = %x\n",csrptr->stat);
       return;
       }
    }

/****************************************/
/* delchan - delete an existing channel */
/****************************************/

delchan()
    {
    struct csrreg *csrptr;
    char docmd();

    csrptr = (struct csrreg *)BASE;

    if(docmd(DELETE,&ch,csrptr) != 0)
       {
       printf("Channel NOT deleted due to timeout.\n");
       return;
       }

    if(csrptr->stat != 0)
       {
       printf("Channel NOT deleted: status = %x\n",csrptr->stat);
       return;
       }
    }

/*********************************/

char docmd(cmd,cp,csr)
    int cmd;
    unsigned char *cp;
    register struct csrreg *csr;

    {
    register unsigned int timeout;

    timeout = 500000;
    while ((csr->ctrl & BUSY) && timeout)
       timeout--;

    if (timeout == 0)
       {
       printf("Timeout: waiting for BUSY bit in csr ctrl\n");
       return(TIMEOUT);
       }
```

```
    timeout = 500000;
    while (timeout && !tas(&csr->tasreg))
       timeout--;

    if (timeout == 0)
       {
       printf("Timeout: waiting for TAS bit in csr\n");
       return(TIMEOUT);
       }

    csr->addmod = 0x3d;
    csr->dbwidth = 2;
    csr->addreg = (long *)cp;
    csr->tasreg = TASBIT | CMD_VALID | cmd;
    csr->ctrl |= ATTENTION;

    /* poll for results */
    timeout = 5000000;
    while ((((csr->tasreg) & STAT_VALID) == 0) && timeout)
       timeout--;

    if (timeout == 0)
       {
       printf("Timeout: waiting for VALID STATUS bit in csr\n");
       return(TIMEOUT);
       }

    csr->tasreg |= CMD_COMPL;
    csr->ctrl |= ATTENTION;

    return(0);
    }

/*********************************/

/* tas - use the tas instruction on csr TAS bit
 *
 * WARNING: Do NOT use the cc optimizer (-O option) on
 * this function without checking out generated code!
 * (Included here for completeness, but normally
 * compiled separately.)
 */

tas(csr)
    unsigned short *csr;
    {
    asm(" mov.l 8(%fp),%a0");        /* move 1st param. to a0 */
    asm(" clr.l %d0");               /* clear return value    */
    asm(" tas.b (%a0)");             /* do the "tas" instruction */
    asm(" bmi.b tasdone");
    asm(" sub.l &1,%d0");            /* return -1 */
    asm("tasdone: ");
    }
```

```
/*************************************************/
/* sendpkt - enqueues a packet to command pipe */
/*************************************************/

int sendpkt(pktptr)
   struct cmdpkt *pktptr;
   {
   struct csrreg *csr;
   struct vmeenv *envptr;
   struct vmeenv *getenv();

   csr = (struct csrreg *)BASE;
   envptr = ch.cptp;

   /* get a null envelope and put on bpp cmd queue */
   if ((envptr->link = getenv()) == 0)
      {
      printf("\n***** Out of bpp envelopes.\n");
      return(-1);
      }
   envptr->vpkt = pktptr;
   ch.cptp = envptr->link;
   envptr->valid = VALID;
   csr->ctrl |= ATTENTION;
   return(0);
   }

/*********************************/

/* getenv - gets an envelope from the free pool and returns
 * a pointer to it; else if no more envelopes in the free
 * pool, returns a zero.
 */

struct vmeenv *getenv()
   {
   struct vmeenv *envptr;

   envptr = fenvptr;
   if (envptr != 0)
      {
      fenvptr = envptr->link;
      envptr->link = 0;
      envptr->valid = 0;
      return(envptr);
      }
   else
      return(0);
   }
```

```
/*********************************************/
/* rcvpkt - dequeues a packet from status pipe */
/* (returns pointer to packet unless no packet */
/* available, then returns 0)                  */
/*********************************************/

struct cmdpkt *rcvpkt()
   {
   struct vmeenv *envptr;
   struct cmdpkt *pktptr;

   envptr = ch.sphp;
   if (envptr->valid)
      {
      pktptr = envptr->vpkt;
      ch.sphp = envptr->link;

      /* put envelope back into pool */
      envptr->link = fenvptr;
      fenvptr = envptr;
      envptr->valid = 0;

      return(pktptr);
      }
   else
      return(0);
   }
```

INDEX

In this index a page number indicates only where reference to a topic begins; the reference may extend to the following page or pages.

A

Additional Error Code/Status, 3-8
ADDITIONAL ERROR CODES, E-1
assert, 1-9
asterisk, 1-8
attention interrupt, 1-5

B

BPE, 1-6 (see also Buffered Pipe Envelope)
BPP, 1-1, 1-6 (see also Buffered Pipe Protocol)
BPP DATA STRUCTURES, 3-1
BPP interface interrupt, 1-5
BPP Test Command $00, 7-1
BPP Test Command Packet, 7-1
BPP Test Command Returned Status, 7-2
BPP Test Command Status Packet, 7-2
Buffered Pipe Envelope, 1-6 (see also BPE)
Buffered Pipe Protocol, 1-1, 1-6, 4-1 (see also BPP)
Buffered Pipe Protocol Summary, 4-3
Bytes from Index Defect List Type 2, 7-17
Bytes From Index Format, 7-17

C

C FUNCTION EXAMPLES OF BPP PROTOCOL, F-1
CCS, 6-2, A-1 (see also Common Command Set)
Certification, 7-13
channel, 2-3
Channel command packets, 2-3
channel header, 2-3, 4-1
Channel priority, 1-4
Channel status packets, 2-3
Command Channel Header Structure, 3-1
Command Codes, 3-5
command packets, 2-1
Command Parameter Errors, D-1
command pipe, 2-1, 2-3, 4-1
Command/Status Register, 2-1 (see also CSR)
Common Command Set, 6-2, A-1 (see also CCS)

control register, 2-2
controller type code, 7-8
Create Channel CSR Command, 2-3
CSR, 2-1 (see also Command/Status Register)
CSR COMMAND PROTOCOL, 2-1
CSR COMMANDS, 2-3
CSR Status Register, 2-4
CSR Status Register Values, 2-5
CSR Test And Set Commands, 2-3
custom SCSI command, B-1
Custom SCSI Command $26, 7-43
Custom SCSI Command Control Word Bit Definitions, 6-4
CUSTOM SCSI COMMAND EXAMPLE, B-1
Custom SCSI Command Packet, 7-44
Custom SCSI Command Returned Status, 7-45
Custom SCSI Command Status Packet, 7-45

D

Data Bus Width Codes, 3-6
Defect List Formats, 7-16
Delete Channel Command, 2-4
DEVICE DESCRIPTORS, 5-1
Device Type Assignments, 3-6
Diagnostic Register, 2-5
Disable Target Command $21, 7-34
Disable Target Command Packet, 7-34
Disable Target Command Returned Status, 7-35
Disable Target Command Status Packet, 7-35
Disk Descriptor Table, 5-1
Disk Format Errors, D-2
Drive Errors, D-1

E

embedded controller, 1-2
Enable Target Command $20, 7-32
Enable Target Command Packet, 7-33
Enable Target Command Returned Status, 7-33
Enable Target Command Status Packet, 7-33
envelope, 3-1, 3-3, 4-1
Envelope Format, 3-3
Envelope/Packet Dequeueing, 4-3
Envelope/Packet Enqueueing , 4-2
Erase Command $14, 7-28
Erase Command Packet, 7-28
Erase Command Returned Status, 7-29
ESTABLISHING DRIVER/MVME327A CHANNEL COMMUNICATIONS, 4-1
extended read command, B-4
Extended Read Example, B-4

F

fatal error code, 7-46
FATAL ERROR CODES, D-1
FEATURES , 1-1
Fix Bad Spot Command $06, 7-17
Fix Bad Spot Command Packet, 7-18
Fix Bad Spot Command Returned Status, 7-19
Fix Bad Spot Command Status Packet, 7-19
floppy drive interface, 1-5
Floppy Format Field, 5-3
floppy interface, 1-4
Format Command $05, 7-12
Format Command Packet, 7-13
Format Command Returned Status, 7-15
Format Command Status Packet, 7-15

G

GLOSSARY , 1-2
Grown defect list, 7-13

H

head pointer, 3-1
header J8, 7-52
hexadecimal, 1-8
High Level Command Translation, 6-1
high level commands, 5-9, 6-1, 7-1, B-1
Host, 1-2
host computer adapter, 1-2

I

Initiator, 1-2, 1-3
intelligent peripheral, 1-2
invalid command, C-10
IRQ Failure, 2-6

L

Least significant word, 1-2 (see also LSW)
list of defects, 7-12
Load/Unload/Re-tension Command $11, 7-22
Load/Unload/Re-tension Command Packet, 7-22
Load/Unload/Re-tension Command Returned Status , 7-23
Load/Unload/Re-tension Command Status Packet, 7-23

LOCAL FLOPPY COMMANDS, 6-1
Logical Unit, 1-2
Logical Unit Number, 1-2 (see also LUN)
LSW, 1-2 (see also Least significant word)
LUN, 1-2 (see also Logical Unit Number)

M

MANUAL TERMINOLOGY, 1-8
MAP Decoder, 2-6
Media Errors, D-1
modes of operation
  interrupt driven, 1-7
  polled, 1-7
Most significant word, 1-2 (see also MSW)
MPU Failure, 2-5
MSW, 1-2 (see also Most significant word)
multi-threading, 1-3
MVME327A COMMANDS, 6-1, 6-6
MVME327A FIRMWARE ARCHITECTURE, 1-4
MVME327A Specific Errors, D-2

N

negate, 1-9

O

Open Command $2D, 7-53
Open Command Packet, 7-54
Open Command Returned Status, 7-55
Open Command Status Packet, 7-55
other data structures, 5-1

P

PACKET, 3-4
Packet Format, 3-4
peripheral controller, 1-2
peripheral type code, 7-8
Physical Sector Defect List Type 1, 7-16
Physical Sector Format, 7-16
Primary defect list, 7-12

R

RAM Failure, 2-5
READ command, 6-2
Read Command $01, 7-2
Read Command Packet, 7-3
Read Command Returned Status, 7-5
Read Command Status Packet, 7-5
Read Descriptor Command $03 , 7-8
Read Descriptor Command Packet, 7-8
Read Descriptor Command Returned Status, 7-9
Read Descriptor Command Status Packet, 7-9
Read Status Command $10, 7-20
Read Status Command Packet, 7-20
Read Status Command Returned Status, 7-21
Read Status Command Status Packet, 7-21
receive command, C-8
RELATED DOCUMENTATION, 1-8
Release Unit Command $23, 7-38
Release Unit Command Packet, 7-39
Release Unit Command Returned Status, 7-41
Release Unit Command Status Packet, 7-41
request sense command, B-5, C-16
Request Sense Example, B-5
Reserve Unit Command $22, 7-36
Reserve Unit Command Packet, 7-36
Reserve Unit Command Returned Status, 7-38
Reserve Unit Command Status Packet, 7-38
Reset SCSI Command $25, 7-41
Reset SCSI Command Packet, 7-42
Reset SCSI Command Returned Status, 7-43
Reset SCSI Command Status Packet, 7-43
RETURN STATUS, 2-4
Rewind Command $13, 7-26
Rewind Command Packet, 7-26
Rewind Command Returned Status, 7-27
Rewind Command Status Packet, 7-27
ROM Failure, 2-6

S

SCATTER/GATHER LIST, 5-9
Script Value Definitions, C-3
SCSI Address, 1-2, 1-3, 7-52
SCSI bus, 1-4
SCSI BUS BACKGROUND, 1-3
SCSI BUS COMMANDS, 6-1
SCSI command, C-1
SCSI commands, 6-2
SCSI controllers, 6-2
SCSI Device, 1-2

SCSI Failure, 2-6
SCSI ID, 1-2
SCSI interface, 1-4
SCSI level commands, 5-9, 6-1, 6-2
SCSI Specific Packet, 6-3, B-1
SCSI Specific Packet Description, 6-3
Self Test Command $27, 7-46
Self Test Command Packet, 7-46
Self Test Command Returned Status, 7-47
Self Test Command Status Packet, 7-47
send command, C-13
Sequential processing, 1-4
Set SCSI Address Command $2B, 7-52
Set SCSI Address Command Packet, 7-52
Set SCSI Address Command Returned Status, 7-53
Set SCSI Address Command Status Packet, 7-53
Space Command $15, 7-30
Space Command Packet, 7-30
Space Command Returned Status, 7-32
Space Command Status Packet, 7-32
Start/Stop Tape Descriptor Table, 5-7
Status checking, B-2
status packets, 2-1
status pipe, 2-1, 2-3, 4-1
Streaming Tape Descriptor Table, 5-5
SUPPORTED SCSI CONTROLLERS/DEVICES, A-1
Synchronous transfers, B-2, C-3
SYSFAIL, 2-4
SYSFAIL Handling, 2-5

T

tail pointer, 3-1
Target, 1-2, 1-3
Target Driver Algorithm, C-1
Target Execute Command $29, 7-50
Target Execute Command Packet, 7-50
Target Execute Command Returned Status, 7-51
Target Execute Command Status Packet, 7-51
TARGET ROLE, C-1
Target Wait Command $28, 7-48
Target Wait Command Packet, 7-48
Target Wait Command Returned Status, 7-49
Target Wait Command Status Packet, 7-49
TAS register, 2-2
test unit ready command, B-3
Test Unit Ready Example, B-3
Thread, 1-2, 1-3
time out interrupts, 1-5
Timer Failure, 2-6
timer interrupt, 1-5

U

User, 1-2

V

virtual channel, 2-1
VME DMA Errors, D-2
VMEbus MPU modules, C-1

W

Write Command $02, 7-6
Write Command Packet, 7-6
Write Command Returned Status, 7-7
Write Command Status Packet, 7-7
write descriptor command, 7-8
Write Descriptor Command $04, 7-10
Write Descriptor Command Packet, 7-10
Write Descriptor Command Returned Status, 7-12
Write Descriptor Command Status Packet, 7-12
Write Filemark $12, 7-24
Write Filemark Command Packet, 7-24
Write Filemark Command Returned Status, 7-25
Write Filemark Command Status Packet, 7-25

SUGGESTION/PROBLEM REPORT

Motorola welcomes your comments on its products and publications. Please use this form.

To: Motorola Inc.
Microcomputer Division
2900 S. Diablo Way
Tempe, Arizona 85282
Attention: Publications Manager
Maildrop DW164

Product: _____ Manual: _____

COMMENTS: _____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

(For additional comments use other side)

Please Print

Name _____ Title _____

Company _____ Division _____

Street _____ Mail Drop _____

City _____ Phone _____

State _____ Zip _____ Country _____

For Additional Motorola Publications
Literature Distribution Center
616 West 24th Street
Tempe, AZ 85282
(602) 994-6561

Motorola Field Service Division/Customer Support
(800) 528-1908
(602) 438-3100

COMMENTS: _____

Microcomputer Division
2900 South Diablo Way
Tempe, Arizona 85282
P.O. Box 2953
Phoenix, Arizona 85062

10336 PRINTED IN USA (8/88) MPS 1500

68NW9209D55A