Storage System Unibus Port Description
AA-L621A-TK
A Part of UDA50 Programmer's Doc. Kit
QP905-GZ

CHAPTER 1

INTRODUCTION

1.1 Overview of MSCP Subsystem

Mass Storage Control Protocol (MSCP) is the protocol used by a family of mass storage controllers and devices designed and built by Digital Equipment Corporation. In a system that uses an MSCP storage subsystem, the controller contains intelligence to perform the detailed I/O handling tasks. This arrangement allows the host to simply send command messages (requests for reads or writes) to the controller and receive response messages back from the controller. The host does not concern itself with details such as device type, media geometry, media format, error recovery, etc.

The host uses two levels of software to communicate with the mass storage subsystem. They are the class driver and the port driver. The class driver is the higher level and is concerned with tasks being performed. The class driver's concern with details is limited to the general type of device (such as disk) and the capacity. The class driver is not concerned with the communication link (I/O bus), type of controller, or the exact model of device(s) being used.

The port driver is the lower level and is concerned only with communications services such as passing messages on and off of the communications link. The port driver is not concerned with the meaning of the messages, nor is it concerned with the exact type of controller or the exact model of storage unit(s). Thus each driver has its own level of responsibilities and shields the other from unnecessary details.

In the controller architecture, there are also two levels of software. The lower of these two is also a "port driver" and, like the port driver in the host, is concerned only with passing messages on and off of the bus. The higher level of controller software is the "MSCP Server". It constitutes the intelligence of the controller and therefore defines the functionality of the controller.

The MSCP server concerns itself with determining the number of devices, their type, geometry, unit number, availability, status, etc. The MSCP server receives requests from the host and sends responses to the host. It optimizes the requests, performs the operations, transfers the data to/from the host, transfers the data to/from the

device, and buffers the data as necessary. The MSCP server performs error detection and recovery, and reports any significant errors to the host.

Because the MSCP server handles the error detection and recovery by itself, the host sees a "perfect media", an important characteristic of an MSCP subsystem. That is, the host need only report errors to higher level (user) software, as the MSCP server performs all error recovery and media defect (bad block) handling.

The host's class driver and the controller's MSCP server route their messages through the path of the two port drivers and a hardware interconnect. This is their physical connection. However, their logical communication is a direct connection because the port driver details are below their level of concern. Therefore, there are two paths to consider, a physical message path and a logical MSCP connection. This is illustrated in Figure 1.

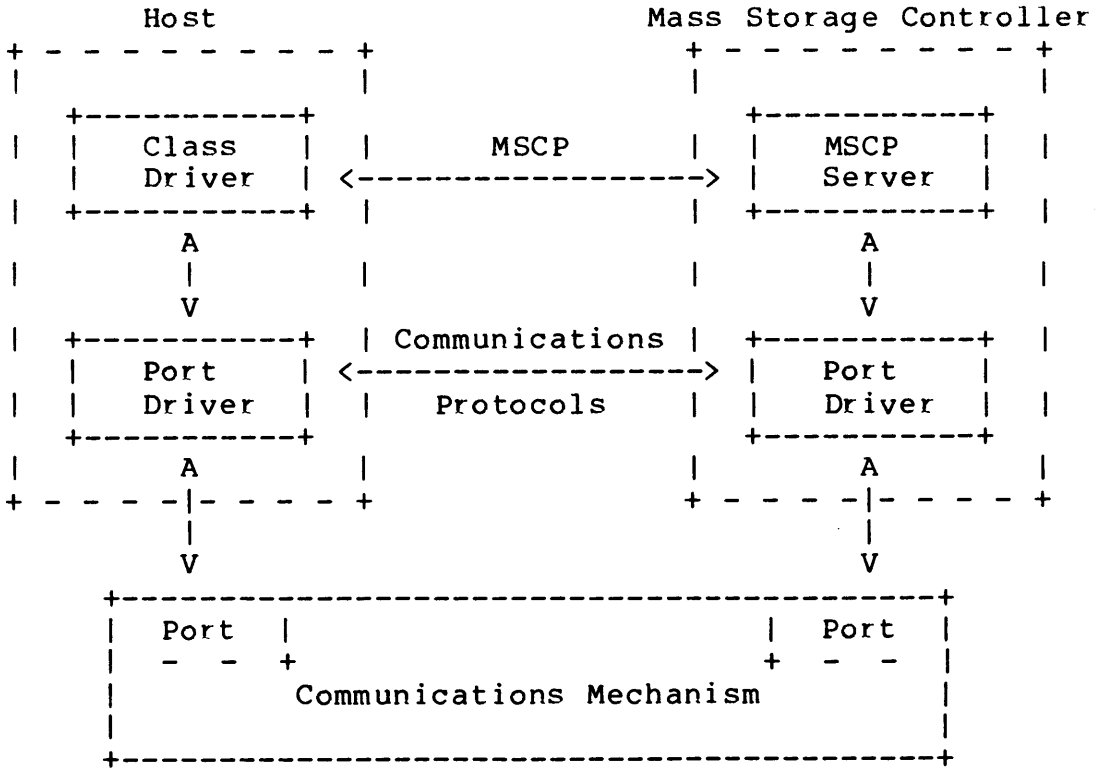


Figure 1 Example System

In summary, an MSCP subsystem is characterized by an intelligent controller that provides the host with the view of perfect media of a fixed size for a given device. It is further characterized by host independence from a specific bus, controller, or device type.

1.2 Purpose

The purpose of this manual is to provide information on the protocol of the port to the detail necessary for writing a port driver.

1.3 Scope

The scope of this manual is limited to the details of the port itself. It does not assume any specific type of host processor or operating system. It does not assume any particular controller or device type.

CHAPTER 2

OVERVIEW OF THE PORT ARCHITECTURE

2.1 OVERVIEW OF THE PORT ARCHITECTURE

The port is a member of a family of related ports and communications mechanisms. This family shares a single System Communications Architecture (SCA) which is described briefly below:

In SCA, communication takes place between pairs of processes resident in separate subsystems. Subsystems include host computers and device controllers. The corresponding processes are host-resident class drivers and controller-resident protocol servers.

The actual transfer of messages uses a set of services called Systems Communications Services (SCS). The Unibus port, in conjunction with a host-resident port driver, implements an instance of SCS.

Communication between a pair of processes takes place over a "connection" which is a soft communication path through the port. A single port typically will implement several connections concurrently. SCS provides primitives for establishing and terminating connections and for determining their states. Once a connection has been established, the following three SCS services are available across that connection.

1. Sequential Message
2. Datagram
3. Block Data Transfer

When a connection is terminated, all outstanding communications on that connection must be discarded; that is, the receiver must "throw away" all unacknowledged messages and the sender must "forget" that such messages have been sent.

The Unibus implementation of SCS/SCA has the following characteristics.

1. Communication is always point-to-point in that exactly two subsystems are connected; one of these is always a host.

2. The port need not be aware of mapping or memory management, since buffers are identified by a Unibus address and are contiguous within the bus address space.

3. The host need never directly initiate a block data transfer.

Because of the point-to-point property, and because the device controller knows the class of device (such as disk) which it controls, the port effectively is integral with the controller and all necessary connections can be established by the port/controller when it is initialized. The host must provide for the necessary connections at system generation or system configuration time. The connections are terminated either by the port entering the fatal error state, or by reinitializing the port. Note that with the Unibus port, all connections are established or terminated as a group.

The Sequential Message service guarantees sequential and duplicate-free delivery for all messages sent over a given connection. Messages are received by the receiving process in the exact order in which the sending process queued them for sending, and no duplicate of a given message will be received. If these guarantees cannot be met, or if a message cannot be delivered for any reason, the port enters the fatal error state and all port connections are terminated.

The Datagram service does not guarantee reception, sequential reception or duplicate-free reception of datagrams, though the probability of failure is required to be "very low". The Unibus port itself can never be the cause of such failure so that, if the using processes do make such guarantees for datagrams, then the Datagram Service over the Unibus port would be equivalent to the Sequential Message service.

The Block Data Transfer service is used to move data between named buffers in host memory and the device controller. In order to allow the port to be unaware of mapping or memory management, the "name" of a buffer is merely the bus address of the first byte of the buffer. Since the host never directly initiates a Block Data Transfer, there is no need for the host to be aware of controller buffering.

A problem common to all communicating asynchronous processes is the need for flow control. This need arises from the possibility of a sending process producing congestion or deadlock in a receiving process by sending messages more quickly than the receiver can cope with them. Flow control simply guarantees that the receiving process has buffers in which to place incoming messages. The sending process is inhibited when all such buffers are full.

The Datagram service does not use flow control. This means that if the receiving process does not have an available buffer in which to store the datagram, then the datagram must be either processed immediately or discarded. Discarding is explicitly permitted by the

rules of Datagram service.

The Sequential Message service does use flow control. Each potential receiving process must reserve or pre-allocate some number of buffers into which messages may be received over its connection. This number is therefore the maximum number of messages which the sender may have outstanding with the receiver, and it is communicated to the sender by the receiver in the form of a "credit" for the connection. The message credits machinery for the Unibus port is described in detail in Section "Message Credits".

2.2 THE UNIBUS PORT ARCHITECTURE

The Unibus Port is the software/hardware interface between port drivers and controllers. It provides the following general capabilities.

1. Provides, at initialization time, information for verifying correct operation of the subsystem controller.

2. Allows for parallel operation of multiple devices attached to the controller, with full duplexing of operation initiation and completion signals, and of data transfers.

3. Mimimizes host interrupts during peak I/O loads.

The overall port consists of two layers.

1. The Transport Layer: This is the machinery for the bi-directional transmission of words and control signals; in the case of the Unibus Port, it is the I/O bus along with any needed bus adapter logic in both the host and the controller.

2. The Logical Layer: This is a set of rules and procedures implemented partly in the host and partly in the controller. The tasks of the logical layer are the exchange of control messages and the verification of correct operation of the transport mechanism and of the overall port.

   The host is assumed to provide a port driver which is the interface between a class driver (e.g. for disk or tape) and the transport mechanism. Thus the port driver implements the host's side of the port proper.

The port's logical layer is implemented as a set of data structures in host memory which are operated on by both the host and the subsystem controller by a set of rules to be discussed later.

The port architecture does use interrupts and Unibus I/O page registers for certain aspects of port operation. These are concerned with the operation of the port itself and not directly with that of the I/O devices attached to the controller.

The port design assumes a command/response relartionship. Command/response transmission uses an asynchronous, packet-oriented protocol. The actual transmission of commands and responses is effected by the port via DMA transfers from/to a communication region in host memory. The port polls this region for commands; the host polls it for responses. From the viewpoint of the host, an I/O operation begins when the host deposits a command descriptor in the command ring. The operation is seen as complete when the corresponding response packet is removed by the host from the response ring.

Interrupts are generated by the port when the command ring makes the transition full to not-full or when the response ring makes the transition empty to not-empty.

CHAPTER 3

TRANSPORT LAYER

3.1 TRANSPORT LAYER

The transport layer is the Unibus and any associated host based or controller based logic for adapting to the bus. This machinery must have the following characteristics.

1. Conform to the Unibus specification (PDP11 BUS HANDBOOK)

2. Allow repeated access (whether reads, writes or any mixture) to the same host memory location.

3.2 Transport Layer Requirements

The host-resident port driver and the controller must provide transport layer control facilities for dealing with the following situations.

1. Transmission of commands and responses.

2. Sequential delivery of commands: Commands must be fetched by the controller in the order in which they were queued to the transport mechanism.

3. Asynchronous communication: It is possible that responses will be sent in an order different from that of the triggering commands.

4. Unsolicited responses: The controller may send unsolicited messages at any time, assuming they have been enabled. The enabling mechanism and the message formats/semantics depend on the higher-level command/response protocol.

5. Full duplex communication: A command, response or unsolicited response may occur at any time.

6. Port failure recovery: The port driver or class driver must be able to place a timer on the controller, and must be able to reinitialize the port in the event of controller timeout.

The port/controller will enter the fatal error state if a command or response is lost and the port/controller cannot notify a higher level of host software.

3.3 UNIBUS I/O PAGE REGISTERS

Two 16-bit registers in the Unibus I/O page are used for control of the port. These registers are always read as words. The register pair begins on a longword boundary. The register names, addresses and functions are:

IP	7xxxx0/4	initialization and polling
SA	7xxxx2/6	status, address and purge

The IP register has two functions as detailed below.

1. When written with any value, it causes a hard initialization of the port and the device controller.

2. When read while the port is operating, it causes the controller to initiate polling as discussed in Section "Polling".

The SA register has four functions as listed below.

1. When read by the host during initialization, it communicates data and error information relating to the initialization process.

2. When written by the host during initialization, it communicates certain host-specific parameters to the port.

3. When read by the host during normal operation, it communicates status information including port- and controller-detected fatal errors.

4. When zeroed by the host during both initialization and normal operation, it signals the port that the host has successfully completed a bus adapter purge in response to a port-initiated purge request.

The detailed operation of these registers is discussed in Section "DETAILS OF INITIALIZATION". Note that only word transfers to/from IP and SA are permissible; the behavior of byte transfers is undefined.

CHAPTER 4

LOGICAL LAYER

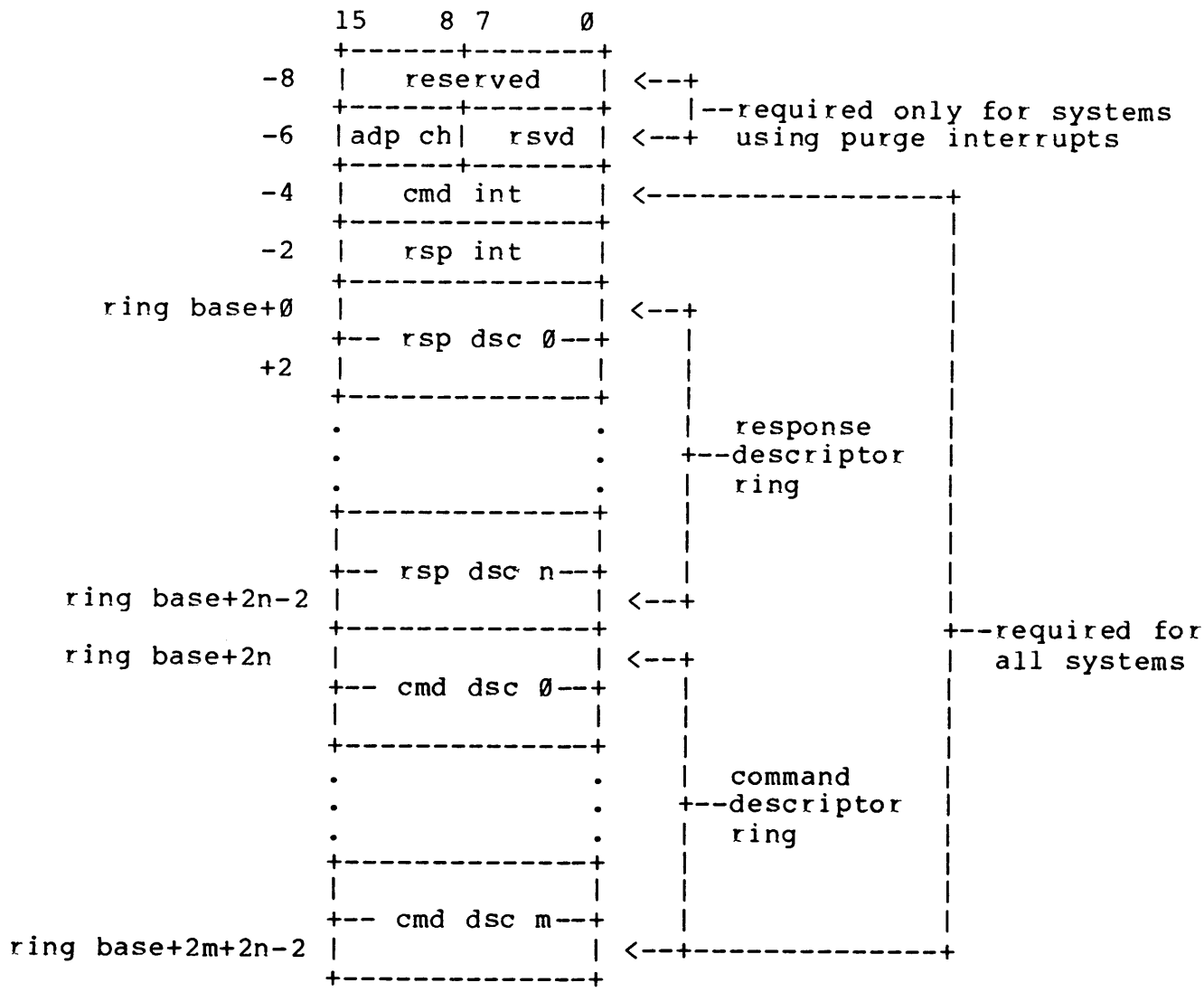
4.1 COMMUNICATIONS AREA

The communications area consists of two sections.

1. A header area containing interrupt identification words. A portion of this area is unique to hosts having bus adapters requiring purges for repeated access to the same location.

2. A variable-length section containing the response and command rings, organized into rings. Note that strictly speaking these should be called "receive" and "send" rings, since from the port's viewpoint they are entirely general-purpose. However, for clarity we will continue to use the terms "response" and "command".

The communications area is aligned on a 16-bit word boundary. Its layout is:



note: n = response ring size
m = command ring size

4.1.1 Interrupt Indicators and SA Register

Words [ringbase -6,-4,-2] are used as indicators which are zeroed by the host but which, when the port interrupts the host, will have been set non-zero by the port to indicate the reason for the interrupt. The word meanings are:

ring base-6: Port is requesting a bus adapter purge. The non-zero value is the adapter channel number and is contained in the high-order byte. It is derived from the triggering command.

The host responds by performing the purge. It then signals purge completion by writing zeroes to the SA register.

-4: Command ring transitioned full to not-full.
Set now-zero by port. Cleared by host.

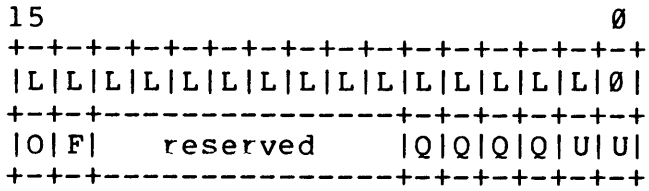
-2: Response ring transitioned empty to not-empty.
Set now-zero by port cleared by host

However entered, the port driver should examine the SA register regularly to verify normal port/controller operation. A port/controller self-detected fatal error is reported in the SA register as discussed in Section "SELF DETECTED FATAL PORT/CONTROLLER ERRORS".

4.1.2 Command and Response Rings

The command and response rings each are organized into a ring of 32-bit descriptors. The length of each ring is determined by the relative speeds with which the host and the port/controller generate and process messages and is unrelated to the controller command limit discussed in Section "Message Credits". The host sets the ring lengths at initialization time as discussed in Section "DETAILS OF INITIALIZATION".

A formatted descriptor has this layout:



where:

Bit Ø is zero. This is because the envelope address [text+Ø] is word-aligned. Note that the port/controller will always assume that bit Ø is set to zero.

L is a bit in the low-order envelope address, for all systems.

U is a bit in the high-order portion of an 18-bit Unibus address.

Q is reserved

O is an "ownership" bit indicating whether the descriptor is owned by the host (O=Ø) or by the port (O=1) and which interlocks the descriptor against premature access by either party.

F is a "flag" bit whose meaning varies depending on the state of the descriptor. These are described below.

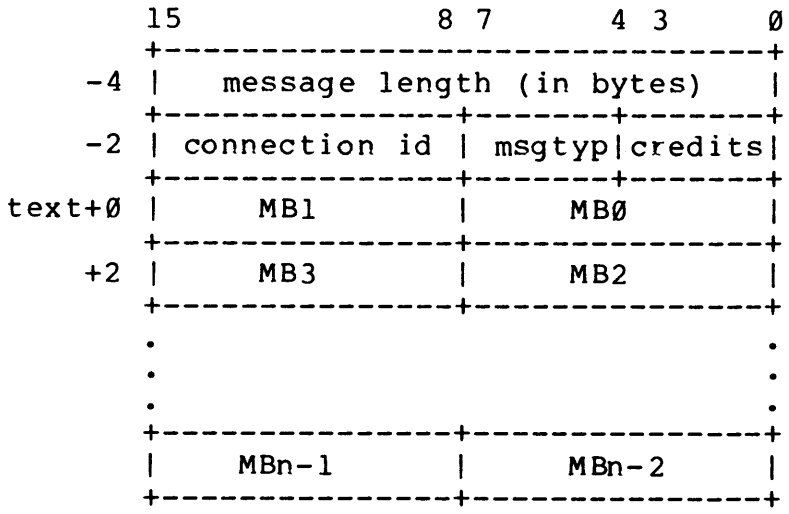
When the port returns ownership of a descriptor to the host, it sets F=1 to indicate that the port has completed action on the descriptor.

When the port acquires ownership of a descriptor from the host, F=1 indicates that the host is requesting a ring transition interrupt. If F=Ø, the host is not requesting a ring transition interrupt. The ring transition interrupt will occur only if this descriptor causes a ring transition and transition interrupts were enabled during initialization.

Since the port always will set F=1 when returning a descriptor to the host, a host desiring to override ring transition interrupts must always clear F when passing ownership of a descriptor to the port.

4.1.3 Message Envelopes

The command or response descriptor points to word [text+Ø] of a 16-bit word-aligned message envelope formatted as follows:



The fields have the meaning shown below.

msg length gives the length of the message text, in bytes.

For commands, this length is equal to the size of the command, in bytes, beginning with [text+Ø].

For responses, the host sets the length equal to the size of the response buffer, in bytes, beginning with <text+Ø>. The minimum acceptable size is 6Ø bytes of message text (64 bytes overall). Before actual transmission of a response, the controller reads the length field in the message envelope. If the port's response is longer than the response buffer, the port will fragment its response into as many response buffers as necessary.

The port sets the resulting value into the message length field. The host therefore must keep re-initializing the value of this field for each proposed response.

Note that if a controller's responses are less than or equal to 6Ø bytes, then the controller need not check the size of the response slot.

MBj is a byte of message text.

connection id identifies the connection serving as source of, or destination for, the message in question.

credits gives the credit value associated with the

message, as discussed in Section "Message Credits".

msgtyp indicates the message type, as follows:

Ø - Sequential Message: The "credits" and "message length" fields are valid.

1 - Datagram: "Credits" must be zero; "message length" is valid.

2 - Credit notification: "Credits" is valid; "message length" must be zero.

3-14 - Reserved (unassigned).

15 - Maintenance.

Note that the port provides no information protection for the text of the message beyond bus parity, which is not implemented by all systems.

4.1.4 Message Credits

The higher-level protocol is required to use a credit-based command limit mechanism. The credits field of the envelope supports the algorithm explained below.

1. In its first response the controller will return, in the credits field of that response, a number N. This number is one more than the controller's limit for non-immediate commands, the "extra" being to allow the host always to be able to issue an immediate-class command. The class driver is to remember the delivered number in its "credit account".

2. Each time the class driver queues a command, it decrements the credit account by one.

3. Each time the class driver receives a response, it increments the credit account by the value contained in the credits field of that response. For unsolicited responses this value will be zero; for solicited responses it normally will be one (an exception is discussed below).

4. If the credit account has a value of one, the class driver may issue only an immediate-type command. If the account balance is zero then the class driver may not issue any commands at all.

Note that for a controller having N > 15(10), responses beyond the first will have [credits] > 1, allowing the controller to "walk" the class driver's credit balance up to the correct value.

Note also that for a well-behaved class driver, enlarging the command ring beyond the value N+1 provides no performance benefits, and that in this situation command ring transition interrupts will not occur since the class driver will never fill the command ring.

4.2 MESSAGE TRANSMISSION

4.2.1 Command Transmission

For a command descriptor, the ownership bit transition O Ø-->1 means that the host has filled the descriptor and is releasing it to the port. The transition O 1-->Ø means that the port has emptied the command descriptor and is returning the empty descriptor to the host. Thus to send a command, the host sets O=1; the port clears O when the command has been received, returning the empty slot to the host.

To guarentee that the port/controller sees each command, the host must read the IP register whenever it inserts a command in the command ring. This forces the port to poll the command if it was not already accessing the command ring.

4.2.2 Response Transmission

For a response descriptor, the transition O 1-->Ø means that the port has filled the descriptor and is releasing it to the host. The transition O Ø-->1 means that the host has emptied the response descriptor and is returning the empty descriptor to the port. Thus to send a response the port clears O; the host sets O=1 when the response has been received, returning the empty slot to the port.

Just as the port must poll for commands, so must the host poll for responses. This is especially true because of the possibility of unsolicited responses. See Section "Host Polling".

4.2.3 Interrupts

The transmission of a message will result in a host interrupt if and only if interrupts were armed suitably during initialization (see Section "OVERVIEW OF INITIALIZATION") and one of the following conditions has been met.

1. The message was a command with F=1 and the port's fetching it caused the command ring to transition from full to not-full. This interrupt means that the host may place another command in the command ring.

2. The message was a response with F=1 and the port's depositing it caused the response ring to transition from empty to not-empty. This interrupt means that there is a response for the host to process.

3. The port is interfaced to the host via a bus adapter and a command required the port/controller to re-access a given location during data transfer. This interrupt means that the port/controller is requesting the host to purge the indicated channel of the bus adapter.

4.2.4 Polling

4.2.4.1 Port Polling -

An initially idle port/controller will not poll for commands or response slots until stimulated by the host's reading of the IP register. When that happens, the port/controller begins reading commands out of host memory (the controller may have an internal command buffering capability). The port will continue to poll for full command slots until the command ring has been found to be empty. Thus it is extremely important that the class driver adhere to the message credits policy discussed in Section "Message Credits".

Having ceased polling, the port will resume polling either when it delivers a response to the host, or if no responses will be delivered, when the host reads the IP register.

Response polling (for empty slots) continues until any commands buffered within the controller have been completed and the responses sent to the host. At that time the controller will cease accessing the communications area.

Such I/O rundown is a necessary (but insufficient) condition for the host to be able to reallocate the memory occupied by a transient driver. The additional conditions to be met are listed below.

1. Unsolicted responses must be disabled (higher-level protocol-dependent).

2. At least 100 milliseconds must have elapsed since the host last read the IP register.

The port/controller will cease accessing the communications area when it is initialized.

4.2.4.2 Host Polling -

Because of the possibility of unsolicited responses, it is not generally sufficient for the host to cease polling for responses when all outstanding commands have been acknowledged. An accumulation of such unsolicited messages would first saturate the response ring and then any controller internal message buffers, blocking the controller and preventing it from processing additional commands.

Thus the host must at least occasionally scan the response ring, even though it may not be expecting a response. The best way to mechanize this requirement is through the use of the ring transition interrupt facility described earlier, supplemented with the advice that the host should pull from the response ring as many responses as it finds, ceasing polling only when the response ring is found to be empty. This policy not only guarantees that the host will see every response in a timely way, it also eliminates unnecessary host polling.

CHAPTER 5

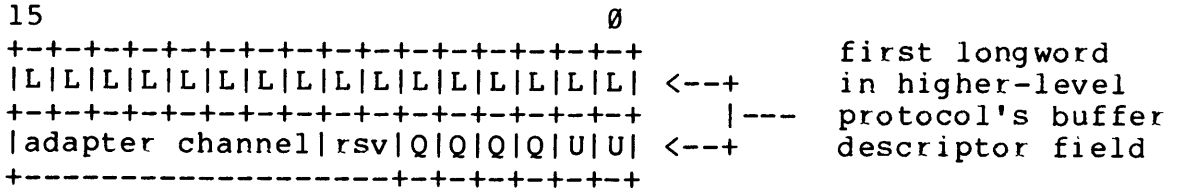
DATA TRANSMISSION

5.1 DATA TRANSMISSION

The details of data transmission are port/controller dependent, however the general philosophy will be outlined here.

In the command ring, the command descriptor points to a command packet. Within the command packet there is a buffer descriptor which contains a pointer and a byte or word count. The buffer descriptor points to the data buffer. The data buffer serves as a source or sink for data transfers. The data is moved by the port in to/out of the buffer as DMA transfers to/from Unibus addresses (host memory being the most likely source/destination).

A buffer descriptor begins at the first word allocated for this purpose in the formats of higher-level commands. The Unibus port uses a two-word buffer descriptor formatted as follows:



in which:

L is a bit in the low-order buffer address, for all systems. Note that the higher-level protocol may require buffers to begin on word boundaries; the port makes no such restriction.

U is a bit in the high-order portion of an 18-bit Unibus address.

Q is reserved

To support such higher-level protocol functions as transfer restarts, compares, etc., the host memory interface must allow repeated access to a given host memory location for both reads and writes. On purely

Unibus systems such as 11/44, this requirement is trivially met with no participation by the host CPU.

On systems with bus adapters such as the 11/78Ø, the repeated access requirement means that the relevant adapter channel may have to be purged, requiring the active cooperation of the host CPU. The port signals its desire for an adapter channel purge by interrupting the host. The host writes zeroes to the SA register to indicate purge completion.

The port architecture allows the host to require the port/controller to break a lengthy DMA transfer into a series of bursts, each limited to a host-specified number of word. See Section "Step 4".

CHAPTER 6

TRANSMISSION ERRORS

6.1 TRANSMISSION ERRORS

Four classes of errors are considered in the Unibus Port Architecture. They are listed below.

1. Failure to become Bus Master: This can arise whenever the port attempts to access host memory for any reason.

   To deal cleanly with this condition before requesting the bus, the port always sets up a corresponding "last fail" response packet (see Section "Purge and Poll Test") before actually requesting the bus. The port then requests the bus and waits "forever" for bus grant. Should the grant fail to come, the host eventually will experience a timeout and will reinitialize the port/controller. At that time the port will report the Bus Master error via the previously set up "last fail" response packet, assuming such packets were enabled during the reinitialization.

2. Failure to become Interrupt Master: This can arise whenever the port attempts to interrupt the host for any reason.

   The treatment and reporting of this error are analogous to those of failure to become Bus Master.

3. Bus Data Timeout error: Whether retries are performed is controller-dependent. If they are, a persistent error results in differing action depending on whether the offending operation was a control or data transfer.

   1. If a control transfer, the port is to set a failure code into the SA register and then to terminate the connection with the port. The port/controller will have to be reinitialized.

   2. If a data transfer, the port/controller remains online to the host. The failure is to be reported in the response to the offending operation.

4. Bus Parity error: The action here is the same as for bus data timeout errors.

CHAPTER 7

SELF-DETECTED FATAL PORT/CONTROLLER ERRORS

7.1 SELF-DETECTED FATAL PORT/CONTROLLER ERRORS

Various fatal errors may be self-detected by the port or controller. Some of these may also arise while the controller is operating its attached device(s).

In the event of a fatal error, the port posts the followin bits in SA.

1. Bit 15 = 1 Fatal Error Indicator.

2. Bits 10-0: Fatal Error Code.

Port-generic fatal error codes are binary numbers occupying the value range 1-199 (10). Currently-assigned codes are:

001 - Envelope/Packet Read (parity or timeout).
002 - Envelope/Packet Write (parity or timeout).
003 - Controller ROM and RAM parity.
004 - Controller RAM parity.
005 - Controller ROM parity.
006 - Ring Read (parity or timeout).
007 - Ring Write (parity or timeout).
008 - Interrupt Master.
009 - Host Access Timeout (higher-level protocol-dependent).
010 - Credit Limit Exceeded.
011 - Unibus Master Error
012 - Diagnostic Controller Fatal Error.
013 - Instruction Loop Timeout.
014 - Invalid Connection Identifier.
015 - Interrupt Write.
016 - MAINTENANCE READ/WRITE Invalid Region Identifier.
017 - MAINTENANCE WRITE Load to non-loadable controller.
018 - Controller RAM error (non-parity)
019 - INIT sequence error
020 - High-level protocol incompatibility error
021 - Purge/poll hardware failure
022 - Unassigned
 * - "
099 - Unassigned

Codes 100(10) and higher are assigned to specific controllers in groups of 100 so that a given controller uses the code range N00-N99 (10). Appendix B contains a table of currently-assigned code ranges.

CHAPTER 8

PORT PERFORMANCE CONSIDERATIONS

8.1 PORT PERFORMANCE CONSIDERATIONS

As earlier mentioned, ring interrupts are generated by the port when the command ring makes the transition full to not-full or when the response ring makes the transition empty to not-empty.

Since a full condition in either ring implies that the intelligence responsible for filling that ring is blocked in some way, ring sizes should be specified large enough to keep the incidence of full rings small. For the command ring, the optimal size depends on the latency in the controller's polling the command ring. For the response ring, the optimal size depends on the latency in invoking the host software process which will empty the ring. The message credit scheme generally ensures that resource allocation will not increase these latencies.

CHAPTER 9

INITIALIZATION

9.1 OVERVIEW OF INITIALIZATION

The initialization procedure serves the following purposes.

1. Identify the parameters of the host-resident communications region to the port.

2. Provide a confidence check of port/controller integrity.

3. Bring the port/controller online to the host. Note that this action does not bring the devices online to the controller.

The initialization procedure consists of a hard initialization during which the port/controller runs preliminary diagnostics and which either fails, requiring reinitialization, or succeeds and is followed by the following four-step procedure.

1. The host writes into the SA register the lengths of the rings, whether interrupts are to be armed and, if so, the address of the interrupt vector. The port/controller then runs a complete internal integrity check and signals either success or failure.

2. The host reads from SA an echo of the ring lengths, and then writes into SA the low-order portion of the ring base address and whether the host is one which requires purge interrupts.

3. The interrupt vector address and the master interrupt arming signal are echoed in the SA register. The host then writes the high order portion of the ring base address to the SA register along with a signal that conditionally triggers an immediate test of the polling and adapter purge functions of the port.

4. The port tests the ability of the I/O bus to perform DMA transfers. If successful, the port then zeroes the entire communications area. The port then signals the host that initialization is complete. At this point, the port driver sets the response slot ownership bits so that the port owns

all the slots. The controller then waits for a signal from the host to begin normal operation.

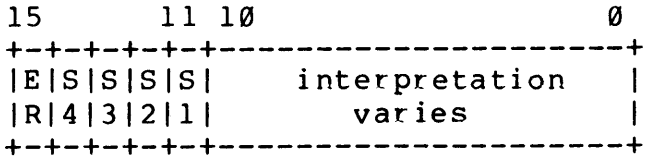
At each step the port informs the host of either failure (requiring a restart of the initialization sequence) or success (and therefore willingness to progress to the next initialization step). The host maintains timeouts on the various port/controller responses.

At various points information is echoed by the port to the host. The intent here is to echo all bit positions but not necessarily to echo all fields.

9.2 DETAILS OF INITIALIZATION

During initialization, the detailed format and meaning of the SA register depends on the initialization step and whether SA is being read or written.

When being read, certain aspects of the SA format are constant and apply to all steps. This constant SA read format is:



The bits S1-S4 are set separately by the port to indicate which step it is ready to perform. If the host detects more than one S-bit set at any time, it should reinitialize the port/controller. If this happens a second time, the host should consider the port/controller to be "dead".

If ER=1, then either a port/controller-based diagnostic test has failed or there has been a fatal error. Bits 10-0 display an error code which may be either port-generic or controller-dependent. See Section "SELF DETECTED FATAL PORT/CONTROLLER ERROR.

Thus in the event of an error, the host can determine not only the nature of the error but also the step during which the error occurred. If ER=1 and a step bit is set, then a fatal error was detected during hard initialization.

In the event of an initialization error, the port driver must retry the sequence at least once. It is suggested, however, that a second failure be considered as meaning that the port/controller is "down".

Note also that during both initialization and normal operation an error is always posted in SA as described above. In addition it may be recorded within the controller and reported at a higher level when initialization completes (see the LF bit discussion in Section "Step 4").

The host begins the initialization sequence either by issuing a bus INIT or by writing any value to the IP register. The port must guarantee that the host will read zeroes in SA on the next bus cycle. Initialization then sequences through Steps 1-4 as described on the following pages.

At the beginning of Step n, the port is to clear bit Sn-1 before setting bit Sn so that the host will never see bits Sn-1,Sn set simultaneously.

From the host's viewpoint, Step n is deemed to have begun when reading SA shows the transition Sn Ø-->1. Of course, Step n ends when Step n+1 begins as just defined. This transition from Step n to Step n+1 may be accompanied by an interrupt, depending on whether interrupts are enabled.

Steps 1-3 each are required to complete within 1Ø seconds. If any of these steps fails to complete within that period, this is to be treated as a host-detected fatal error.

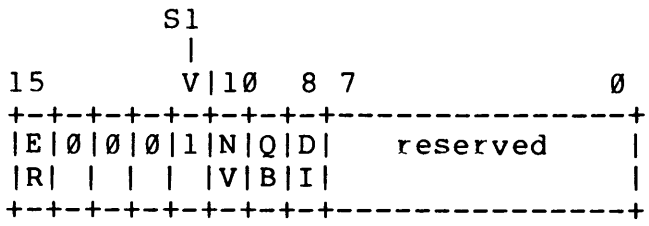
There is no explicit signal for the completion of Step 4. Rather, the host observes either that controller operation has begun or that a higher-level protocol-dependent timer has expired.

During initialization, the host must wait 1ØØ microseconds after any interrupt before reading the SA register to see if there was an error. This is because the port may use the SA register to deliver the vector address to the processor interrupt sequence. If it does, then time will be required by the port to set SA to the value to be read by the host initialization code.

Assuming that ER continues to be zero, the remainder of initialization is as described in the following sections.

9.2.1 Step 1

The host knows that Step 1 has begun when S1 makes the transition Ø-->1. At that time the following pattern may be read from SA:



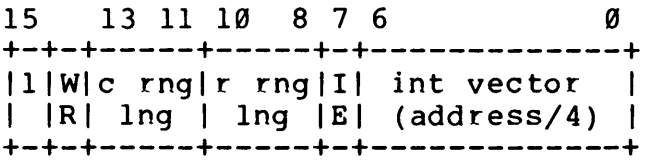
This pattern should appear within 1ØØ microseconds after the hard-initialize. Bits 1Ø-8 are controller-dependent. Their interpretations are shown below.

NV=1 means that the port does not support a host-settable interrupt vector address.

QB reserved.

DI=1 means that the port implements enhanced diagnostics, i.e. wraparound, purge and poll tests.

The host responds by writing the following pattern into SA:



Note that bit 15=1. This is to guarantee that the port does not interpret the pattern as a host "adapter purge complete" response (after a spontaneous reinitialize).

Note also that the vector address pertains to all interrupts generated by the port.

The remainder of the host-generated Step 1 SA pattern has this interpretation:

WR=1 means that the port should enter diagnostic wrap mod. See Section "PORT DIAGNOSTIC FACILITIES".

The port will ignore WR if it delivered DI=Ø at the beginning of Step 1.

c rng lng is the number of slots (32 bits each) in the command ring, expressed as a power of two. Thus for a maximal command ring the host would set this field equal to seven, requesting 2**7=128 slots.

r rng lng is the number of response ring slots, again expressed as a power of two.

IE=1 means that the host is requesting an interrupt at the completion of each of Steps 1-3.

Note that no interrupt will be generated at the completion of Step 4 since this step requires only a small number of microseconds.

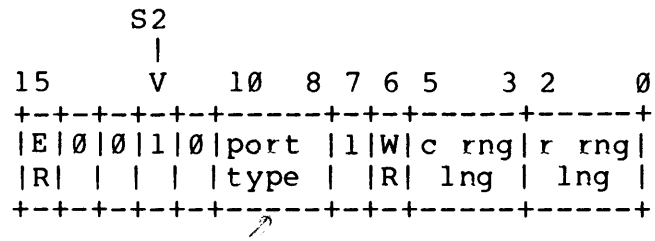
int vector determines if interrupts will be generated by the port. If this field is non-zero, interrupts will be generated during normal operation and, if IE=1, during initialization. If this field is zero, then interrupts will not be generated during normal operation and initialization (regardless of the "IE" bit state). When this field is non-zero, those ports which accept an interrupt vector from the host use the value in the field as the address/4 of the interrupt vector.

Upon receipt of the above data the port/controller begins running its integrity check diagnostics. When finished, the port conditionally interrupts the host as described above. If enabled, the interrupt will take place whether the diagnositics succeeded or failed.

Step 1 must complete within 1Ø seconds after the host writes to the SA register. The completion will result in an interrupt if IE was set to one in Step 1.

9.2.2 Step 2

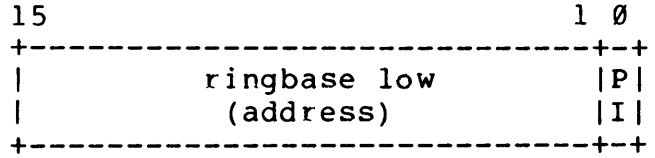
The host knows that Step 2 has begun when the transition S2 Ø-->1 takes place. At that time the following pattern may be read from SA:



Bits 1Ø-8 are a port type number. All controllers conforming to this port specification will use the value assigned to "Unibus Storage Systems Port". (See Appendix C for a table of assigned port type numbers. The controller type is reported via the higher-level protocol.)

Bits 7-Ø are echoes of the data written by the host to SA bits 15-8 during Step 1.

The host responds by writing into SA the following pattern:



The above data has the following interpretation:

ringbase low is the low-order portion of the address of word [ringbase+Ø] of the communications area. This is a 16-bit byte address whose low-order bit is zero implicitly.

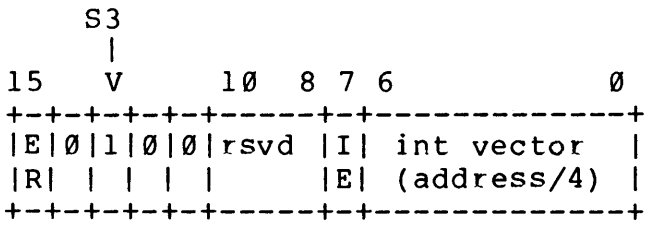
Note that the high-order portion of this address is written to SA by the host during Step 3.

PI=1 means that the host is requesting adapter purge interrupts.

Step 2 must complete within 1Ø seconds from the time the host writes the Step 2 data to SA. The completion will result in a host interrupt if IE was set to one in Step 1.

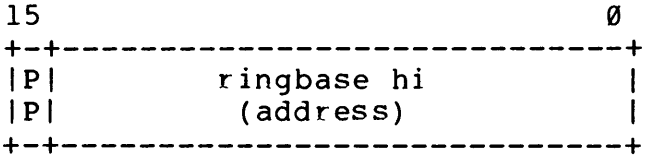
9.2.3 Step 3

The host knows that Step 3 has begun when the transition S3 Ø-->1 takes place. At that time, the following may be read from SA:



In this pattern, bits 7-Ø are the echo of bits 7-Ø of the data which was written to SA by the host during Step 1.

The host responds by writing to SA the following pattern:



This pattern has the following interpretation:

PP=1 means that the host is requesting execution of "purge" and "poll" tests as described below.

The port will ignore PP if it responded with DI=Ø at the beginning of Step 1.

ringbase hi is the high-order portion of the address [ring base+Ø].

Note that the low-order portion of this address will have been written to SA by the host during Step 2.

If PP has been set, then immediately upon writing to SA the host must wait for SA to transition to a zero value. The host must then do two things.

1. Write zeroes to the SA register. (This simulates a "purge completed" host action).

2. Read (and then disregard) the IP register. (This simulates a "start polling" command from the host to the port.)

   The host must complete this sequence within 1ØØ milliseconds from the time SA was first written during Step 3.

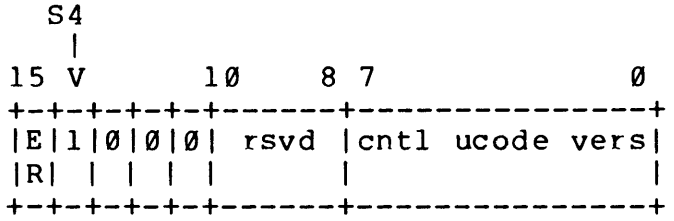
While the host is performing the above steps, the controller, having seen PP=1, does the following functions.

1. Load zeroes into SA.
2. Wait for SA to be written by the host.
3. Wait for IP to be read by the host.
4. Announce in SA the transition to Step 4.

Step 3 must complete within 10 seconds. When it does complete, an interrupt will be generated if IE was set to one in Step 1.

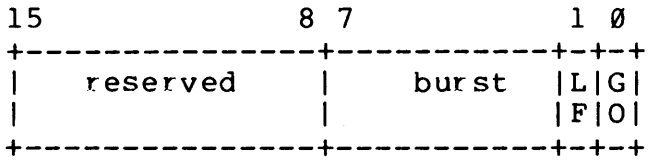
9.2.4 Step 4

The host knows that Step 4 has begun when the transition S4 Ø-->1 takes place. At that time, the following pattern may be read from SA:



Bits 7-Ø give the version number of the port/controller microcode.

The host responds by writing the following pattern into the SA register:



This data is interpreted as follows:

burst is one less than the maximum number of longwords the host is willing to allow per DMA transfer. If this field is zero, then the port is to use its default burst count.

The values of both the default and the maximum the port will accept are controller-dependent.

LF=1 means that the host wants a "last fail" response packet when initialization is complete. The state of LF does not have any effect on the enabling/disabling of unsolicted responses in the higher-level protocol.

GO=1 means that the controller should enter its functional microcode as soon as initialization completes.

Note that if GO=Ø when initialization completes, the port will continue to read SA until the host forces SA bit Ø to make the transition Ø-->1.

Note that there will be no explicit interrupt at the end of Step 4. Instead, if interrupts were enabled, the next interrupt will be due either to a ring transition or to an adapter purge request.

CHAPTER 1Ø

PORT DIAGNOSTIC FACILITIES

1Ø.1 PORT DIAGNOSTIC FACILITIES

1Ø.1.1 Diagnostic Wrap Mode

Diagnostic Wrap Mode (DWM) provides host-based diagnostics with a means of verifying the lowest levels of host/controller communication via the port.

In DWM, the port attempts to echo in the SA register any data written to SA by the host. DWM is a special path through initialization Step 1; Steps 2-4 are suppressed and the port/controller is left disconnected from the host.

DWM is initiated by the setting of the 'WR' bit in the host's Step 1 response (see Section "Step 1") and takes effect immediately, causing the host's Step 1 response to be echoed in the SA register.

DWM is terminated by the host's re-issuing a hard initialize (bus INIT or host write to IP). Assuming the host found the results of DWM to be satisfactory, the host would then bypass DWM during the second initialization sequence.

It is recommended that a diagnostic program which uses DWN not enable interrupts at Step 1. A port/controller failure could result in interrupts to unintended host addresses.

For each datum to be echoed, the host does the following functions.

1. Write any bit pattern to SA.
2. Wait for SA to match the data written. (This must happpen within 1Ø seconds or the port/controller has failed.)
3. Loop back to 1 above for another pass.

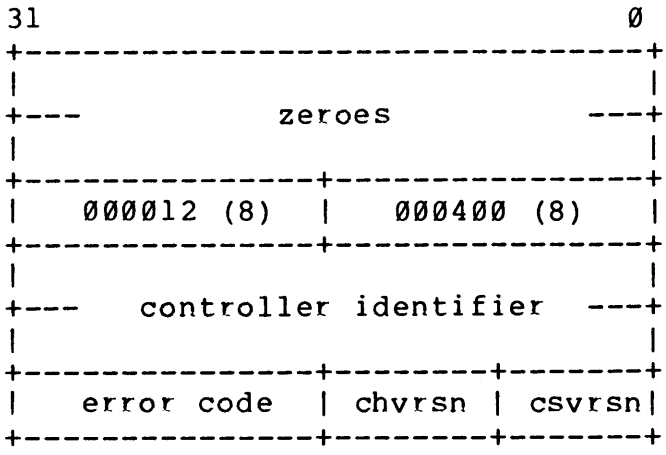
The sequence is terminated by re-initializing the port/controller.

10.1.2 Purge and Poll Tests

These tests are described under Initialization Step 3, Section "Step 3".

10.1.3 Last Fail

The port will send a Last Fail Error Packet to the host if an error was detected during a previous "run" and the LF bit was set in Step 4 of the current initialization. The Last Fail Error Packet has the format shown below.



The error code is discussed in Section "SELF DETECTED FATAL PORT/CONTROLLER ERRORS" and "controller identifier", "chvrsn" and "csvrsn" are discussed in the MSCP Basic Disk Functions Manual under "Error Log Message" formats.

10.1.4 MAINTENANCE READ and MAINTENANCE WRITE

The controller is required to support some variant of the MAINTENANCE READ and MAINTENANCE WRITE commands. Their intended purpose is to allow host-based diagnostics to read and write controller-internal storage.

These commands are not formally part of any higher-level protocol; nevertheless, their formats and many of their field meanings coincide with those of MSCP and the MSCP specification should be consulted for details.

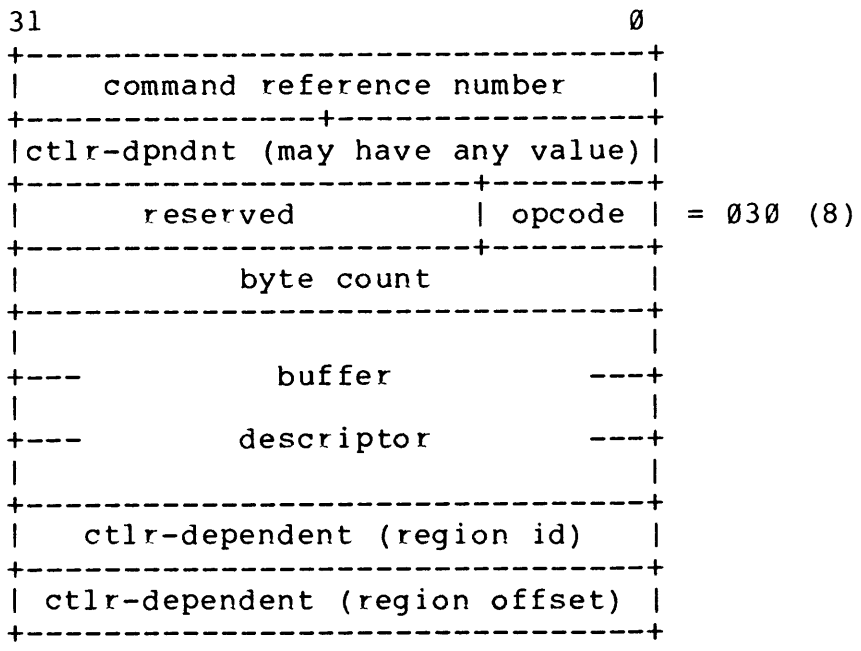
The remaining fields are controller-dependent and only their general outlines will be discussed here.

10.1.4.1 MAINTENANCE READ -

The format of the MAINTENANCE READ command packet is given below. The displayed packet comprises the text of the message contained in the earlier-specified command/response envelope (begining at word [text+0]).

Command Category: Sequential

Command Packet Format:

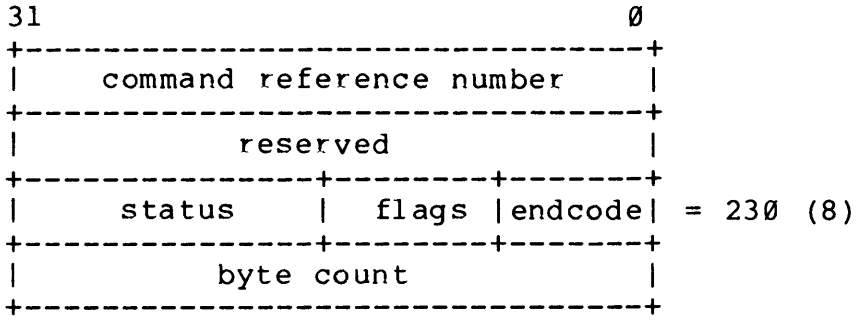


The "region id" serves to select among various classes of storage within the controller. The "region offset" creates sub-classes within a class. At present only one region id is defined.

1. ID = 1: The contents of controller buffer storage, without regard to how the contents got generated.

The controller will respond to the MAINTENANCE READ by delivering the requested data to the requested host buffer (to the limit of the specified byte count) and then delivering to the specified response envelope the following text:

End Packet Format:



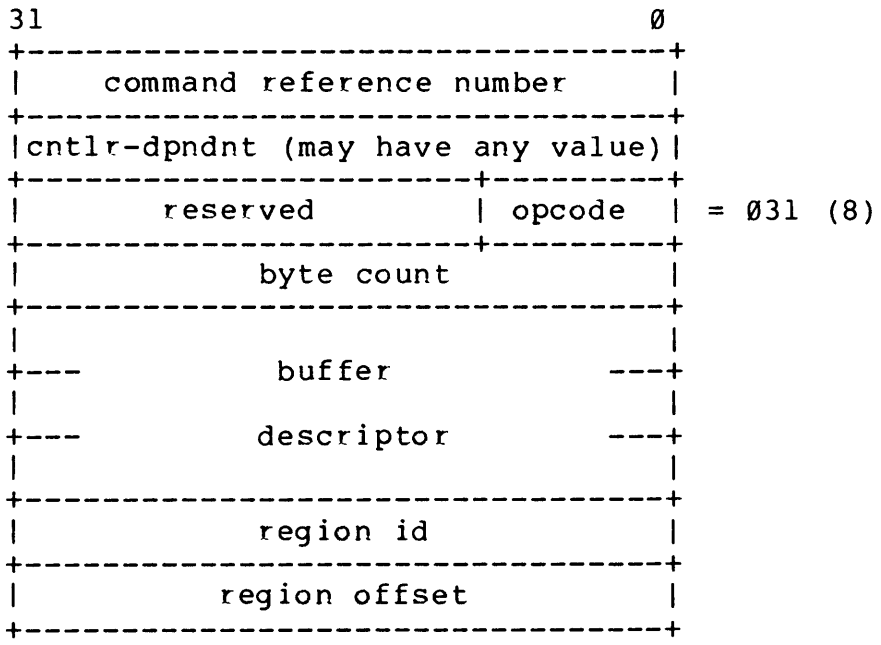
10.1.4.2 MAINTENANCE WRITE -

The MAINTENANCE WRITE command is used by the host to send to the controller information which is outside the scope of the higher-level protocol.

The general MAINTENANCE WRITE command packet format is given below:

Command Category: Sequential

Command Packet Format:

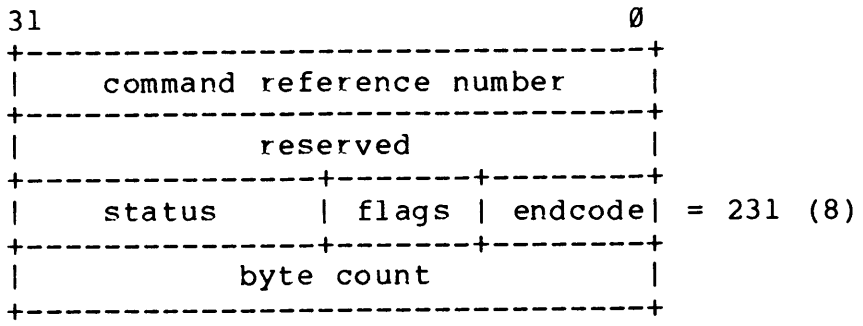


The distinction among data types is made via the region id and region offset fields as shown below.

1. ID = 1: Transfers [byte count] worth of data into the controller at the (virtual) address specified by the region offset.

The controller responds to MAINTENANCE WRITE by fetching the indicated data and then delivering to the indicated response envelope this End Packet text.

End Packet Format:



APPENDIX A

TABLE OF ASSIGNED CONNECTION ID'S AND USES

ID Number	Purpose
Ø (1Ø)	Disk MSCP
1	Reserved
2	DUP
3-254	Unassigned
255	Implementation specific maintenance protocol

APPENDIX B

TABLE OF ASSIGNED ERROR CODE RANGES

Range	Associated Controller
Ø	Reserved
ØØ1-Ø99	Generic, all controllers
1ØØ-199	UDA
2ØØ-299	Reserved
3ØØ-399	Reserved
4ØØ+	Unassigned

APPENDIX C

TABLE OF ASSIGNED PORT TYPE NUMBERS

Number	Corresponding Port Type
Ø	Unibus Storage Systems Port
1-7	Unassigned