96769390

CONTROL DATA CORPORATION

SOFTWARE PERIPHERAL DRIVERS REFERENCE MANUAL

CONTROL DATA®

MASS STORAGE OPERATING SYSTEM

REAL TIME OPERATING SYSTEM

REVISION RECORD

REVISION	DESCRIPTION
A	Manual released.
(6/76)	
B	Manual revised to reflect added information on auto data transfer. Also added drivers for CB104 Card Reader,
(9/96)	storage module disk (SMD), flexible disk, 1827 Line Printer, low cost tape transport (LCTT), and cassette tape.
C	Revised to incorporate PSROs 3253, 4254, and include 1827-2 Line Printer Driver, 1811-2 Conversational Display
(10/77)	Terminal Driver, 1843-2 CLA Driver, LCTT/FORMATTER Driver, MSOS Console Driver, and FDUTIL.
D	Revised to incorporate new peripherals for release of Peripheral Drivers 1.3C. Pages revised are vii/viii, 3-6, 3-8,
(1/78)	3-21, 3-22, C-2, C-3, C-19 and Index-1 through Index-3. New pages are 3-8.1 and C-20.
Publication No. 96769390	

REVISION LETTERS I, O, Q AND X ARE NOT USED

Address comments concerning this manual to:
Control Data Corporation
Publications and Graphics Division
4455 Eastgate Mall
LaJolla, California 92037

or use Comment Sheet in the back of this manual.

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual, are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV
Cover	—	C-5 thru C-18	C						
Title page	—	C-19	D						
ii	D	C-20	D						
iii/iv	D	D-1 thru D-3	B						
v/vi	B	E-1	B						
vii	D	F-1	B						
viii	D	G-1	B						
1-1 thru 1-3	A	G-2	B						
1-4	B	H-1	B						
1-5	A	H-2	B						
1-6	A	H-3	C						
2-1 thru 2-7	B	I-1 thru I-10	C						
3-1	C	Index-1 thru Index-3	D						
3-2	C	Comment sheet/	D						
3-2.1	C	Mailer	—						
3-2.2	C	Back cover	—						
3-3	A								
3-4	A								
3-5	C								
3-6	D								
3-7	B								
3-8	D								
3-8.1	D								
3-9	B								
3-10	C								
3-10.1	C								
3-11 thru 3-14	B								
3-15	C								
3-16	C								
3-16.1	C								
3-17	B								
3-18	B								
3-19	C								
3-20	C								
3-21	D								
3-22	D								
3-22.1	C								
3-23	B								
3-24	C								
3-25	C								
3-26	B								
3-27	B								
3-28	C								
3-29 thru 3-39	B								
4-1	A								
4-2	C								
5-1	B								
6-1	A								
6-2	A								
A-1 thru A-6	B								
B-1	C								
C-1	D								
C-2	C								
C-3	D								
C-4	B								

PREFACE

This manual provides a detailed description of the software I/O drivers for CYBER 18/1700 Series peripherals. These I/O drivers are also utilized by the Mass Storage Operating System (MSOS) and the Real-Time Operating System (RTOS).

This manual is intended to be used by programmers of CYBER 18/1700 software that require a real-time, multiprogramming environment.

Additional information concerning MSOS and RTOS can be found in the following manuals:

Publication	Publication Number
Real-Time Operating System (RTOS) Version 3 CYBER 18-17 Reference Manual	96769560
Mass Storage Operating System Version 5 Reference Manual	96769400
1700 MSOS 5 Diagnostic Handbook	96769450
1700 MSOS 5 Installation Handbook	96769410
274 Interactive Graphics System Version 2 Reference Manual	60358800

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features or parameters.

CONTENTS

APPENDIXES

INDEX

FIGURES

TABLES

INTRODUCTION 1

Each peripheral device in a computer system is associated with a device driver, the only piece of software that is allowed to give direct commands to the device. The driver controls execution of the read, write, and motion requests that are passed to the monitor by the user programs.

Each driver normally has three entries: initiator, continuator, and timeout (error). Variable parameters relating to the device and the driver's working storage are contained in the physical device table in a format common to all drivers. Functionally, the initiator initializes the working storage in the physical device table and initiates input/output on an idle device; the continuator drives the device to perform the actual requested task. If the diagnostic timer detects a device hang-up (lost interrupt), a timer entry is entered.

Whenever a program requires input or output (I/O) for data it is processing, it makes a monitor request to effect the desired transfer. The monitor queues the request for processing by an I/O driver. A driver may handle more than one device of the same type, but requires a separate physical device table for each device.

When a request is queued, the request processor, RW, determines if the driver is available. If the driver is not busy, its initiator is scheduled, and the request exit processor returns to the caller.

The tape motion requests are handled in a similar way through the T14 motion request processor.

Upon entry to the initiator, a call is made to the find-next-request routine, which decodes the requestor's parameter list and places the information in the physical device table. The driver initiates the I/O operation and selects some interrupt condition (end-of-operation, data, etc.). A diagnostic clock value also is set in the physical device table, and an exit is made to the dispatcher.

When the I/O device completes the operation, an interrupt is generated. When the interrupt mask allows the interrupt, the program that is currently executing is stopped, its registers and overflow states are stored in the interrupt stack, and control is given to the interrupt response routine, which enters the driver's continuator entry point. The driver acknowledges the interrupt and performs the I/O command or, if the request is complete, the complete request routine is called, followed by a jump to the initiator.

If there is a hardware malfunction and the device fails to give an interrupt at the end of an operation, the time-out entry is scheduled by the diagnostic timer routine when the clock value in the physical device table has expired (if a timer is present in the system). The driver uses the MAKQ routine to set the error flags and then calls the device error logging routine. If the logical unit number is not that of a diagnostic logical unit, the alternate device handler may be called by a jump or scheduler request. If the request was performed on a diagnostic logical unit, the complete request routine is called, instead of the alternate device handler, followed by a jump to the initiator part of the driver. The diagnostic clock is set negative when a device is inactive.

QUEUEING OF I/O REQUESTS

Input/output requests are queued by logical unit number. Requests for the same logical unit are queued on a thread in the third word of the parameter list. This word contains the first word address of the parameter list of the next request (zero for unqueued requests). The beginning of each queue is identified by an entry in the table of logical units (LOG2) which contains the address of the first word of the first request parameter list. The end of the queue is identified by FFFF$_{16}$ in the third word of the parameter list for the last request in the queue.

READ/FREAD/WRITE/FWRITE REQUESTS

READ/WRITE instructions transfer data between the specified input/output device and core. The word count specified in the request determines the end of the transfer.

FREAD/FWRITE requests read/write records in a specific format for each device.

The macro format for READ/WRITE/FREAD/FWRITE requests (1,2,4,6) is shown below:

$$\left.\begin{array}{l}\text{READ}\\\text{FREAD}\\\text{WRITE}\\\text{FWRITE}\end{array}\right\}\quad \text{lu,c,s,n,m,rp,cp,a,x,d}$$

Where:

lu	is	the logical unit.
c	is	the completion address.
s	is	the starting address.
n	is	the number of words to transfer.
m	is	the mode.
rp	is	the request priority.
cp	is	the completion priority.
a	is	the absolute/indirect indicator for the logical unit.
x	is	the relative/indirect indicator (affects parameters c, s, and n).
d	is	the part 1 request indicator (absolute parameter addresses).

A detailed description of each of the above parameters follows after the calling sequence generated by the macro.

The request codes are 1 (READ), 2 (WRITE), 4 (FREAD), and 6 (FWRITE). The calling sequence generated by the macro is as follows:

	15	14	13	12	11	10	9	8	7	4	3	0
	RTJ-($F4)											
0	0	d	rc					x	rp		cp	
1	c											
2	thread											
3	v			m	a		lu					
4	n											
5	s											

The field descriptions for the calling sequence generated by READ/WRITE macros are:

rc	The request code
thread	The thread location used to point to the next entry or the threaded list
v	The error code passed to the completion address in bits 15 through 13 of the Q register and set in the request by the system at completion

Detailed parameter descriptions for the requests are:

lu is the logical unit; an index to the LOG1A table of physical device table addresses that may be modified by parameter a.

c is the completion address of the core location to which control is transferred when an I/O operation is completed. If omitted, no completion routine is scheduled and control is returned to the interrupted program. The notation (c) represents an index to the system library directory, indicating the program to be executed upon completion of the requested I/O operation. Use of the (c) option by unprotected programs results in job termination.

Completion routines are operated by threading the I/O requests on the scheduler thread. A three-bit code in the v field of the fourth word of the request indicates completion status:

15	14	13	Description
0	0	0	No error condition detected by driver; number of words is requested read or written; device not ready
0	0	1	No error; requested number of words read or written; device ready
0	1	0	No error condition detected by driver; fewer words read than requested; device not ready
0	1	1	No error; fewer words read than requested; device ready
1	0	0	Error condition; requested words read; device not ready
1	0	1	Error condition and/or end-of-tape detected by driver; number of words requested is read or written; device ready
1	1	0	Error condition and/or end-of-file detected; fewer words read than requested; device not ready
1	1	1	Error condition and/or end-of-file or end-of-tape detected; fewer words read than requested; device ready

When control is returned to the completion address, these bits are set in similar positions in the Q register. If less than n words were transferred on a read, the location following the last word filled is placed in the last word of the user's buffer.

An end-of-file can be verified by checking bit 11 of word 12 in the physical device table.

s is the starting address; the address of the first block location to be transferred (see parameter x).

n is the number of words to be transferred.

(n)	Number of words to be transferred is determined by parameter x.
0	The minimum information is transferred (one word or one character), depending on the device.

NOTE

For FREAD and FWRITE, n cannot be zero. Some devices signal zero words as an illegal request.

m is the mode; it determines the operating condition (binary/ASCII) of a driver.

Macro	
A	Data is converted from ASCII to external form for output; from external form to ASCII for input.
B	Data is transferred as it appears in core or on an I/O device.

Coding

0	Binary
1	ASCII

rp is the request priority (15 through 0, with 0 as the lowest) with respect to other requests for this device. This request establishes the order in the I/O device queue. It is automatically zero for unprotected requests.

cp is the completion priority (15 through 0), the level at which the sequence of the code specified by parameter c is executed. It is automatically 1 for unprotected requests.

a is the absolute/indirect indicator for the logical unit.

Macro

blank	The first parameter (lu) specifies the logical unit.
R	lu is a signed increment ($-1FF_{16} \leq lu \leq 1FF_{16}$) which is added to the address of the first word of the parameter list to obtain the core location containing the logical unit number.
I	lu is the address of the core location number ($lu \leq 3FF_{16}$).

Coding

0	lu is a logical unit number.
1	lu is a signed increment ($\pm 1FF_{16}$); not allowed if d = 1.
2	lu is a core address containing the logical unit number.

x is the relative/indirect indicator; this parameter affects parameters c, s, and n as shown here. Because of the wrap-around feature, computed addresses may be before or after the parameter list.

(c) is indirect	x is meaningless and (c) represents an index to the system directory.
0 or blank and c is direct	c is the completion address.
0 or blank and s is direct	s is the starting address. If the request is on mass memory, the mass memory address follows the request.
0 or blank and (s) is indirect	(s) is a core location that contains the starting address. If the request is on mass memory, the mass memory address follows the core location that contains the starting address.†
≠ 0 or not blank and c is direct	c is a positive increment that is added to the address of the first word of the parameter list to form the completion address.
≠ 0 or not blank and s is direct	s is a positive increment added to the address of the first word of the parameter list to form the starting address. If the request is on mass memory, the mass memory address follows the request.
≠ 0 or not blank and (s) is indirect	(s) is a positive increment added to the address of the parameter list to form the address of a location containing another positive increment. If the request is on mass memory, the location containing the second increment is immediately followed by two words which contain the mass memory address.
n is direct	x is meaningless and n is the length of the block to be transferred.
x is 0 or blank and (n) is indirect	(n) is the core location containing the block size.
x is ≠ 0 or ≠ blank and (n) is indirect	(n) is a positive increment added to the address of the first word of the parameter list to obtain the location containing the block size.

d is the part 1 request indicator; this parameter indicates that the request requires the use of 16-bit address arithmetic.

0 or blank	Preceding description of parameter applies

†If bit 15 is set for (n) or (s), incrementing continues indirect until bit 15 is not set.

1	x	is ignored.
	n	is the number of words
	c and s	are 16-bit absolute addresses.
	lu	is processed the same as d = 0.
	a	cannot be set to R.

Mass Memory Address Format:

15	14	0	
Most Significant Bits of Mass Storage Address (MSB)			
0	Least Significant Bits of Mass Storage Address (LSB)		(LSB)

The mass storage address specifies that a mass memory word address (READ/WRITE) or a mass memory sector (96-word size) address (FREAD/FWRITE) return is to the location following the mass storage address.

The user is reminded that the assembler that executes under RTOS does not support macros. The user must therefore code the required monitor calls directly. For example, assume that a message at location MESS of length MESSL is to be printed on the comment device. A sample coding could be the following:

```
        RTJ-    (AMONI)     Call monitor
        NUM     $4C00       Request code, absolute
        ADC     COMPLT      Completion address
        NUM     0           Thread
        NUM     $18FC       ASCII, logical unit in
                            location FC16
        ADC     MESSL       Length
        ADC     MESS        Start of message
        JMP-    ($EA)       Call dispatcher
COMPLT  NOP     0           I/O completion address
```

MOTION

This request (14) is used to control motion and end-of-file processing. The macro format is:

MOTION lu,c,p_1,p_2,p_3,dy,rp,cp,a,x,d,m

Where:

lu	is	the logical unit.
c	is	the completion address.
p_1, p_2, p_3,	are	the motion control parameters; each of these results in a specific action that is defined in table 1-1. Up to three motion commands may be defined in a MOTION request; they are executed in the sequence p_1,p_2,p_3. The first command with a value of zero terminates the request.
dy	is	the density parameter.

0	No change	
1	800 bpi	External rejects result when an illegal density selection is attempted.†
2	556 bpi	
3	200 bpi	
4	1600 bpi	

rp	is	the request priority.
cp	is	the completion priority.
a	is	the absolute/indirect indicator for the logical unit.
x	is	only related to the completion address.
d	is	set to 0 All parameters are processed as described.
		1 A part 1 request is indicated (c is a 16-bit absolute address and must not equal R for the a parameter).
m	is	the mode.
		A ASCII
		B Binary

The MOTION control request code is 14. The calling sequence generated by the macro is:

	15	14	13	12	11	10	9	8	7	4	3	0
	RTJ-($F4)											
0	0	d	rc					x	rp		cp	
1	c											
2	thread											
3	v			m	a		lu					
4	p1				p2				p3		dy	

† In addition to the attempt to set a density that is not legal for a unit (e.g., 200/556 bpi on a 609), the drivers do not allow a density change if the unit is not at load point.

TABLE 1-1. MSOS DRIVER ACTION FOR MOTION REQUEST PARAMETERS p_1, p_2, p_3

Code	Description	MT	CR	CP	LP	TTY	PTR	PTP	MSD	PTD	CD†
0	First zero terminates processing the request	X	X	X	X	X	X	X	X	X	X
1	Backspace one record	X								X	X
	Do nothing		X	X	X	X	X	X	X		
2	Write one end-of-file mark	X								X	X
	Punch one end-of-file mark			X				X			
	Page eject; reset line count				X	X					
	Punch leader							X			
	Do nothing		X				X		X		
3	Rewind to loadpoint	X									X
	Set pointer to start-of-tape									X	
	Do nothing		X	X	X	X	X	X	X		
4	Rewind and unload; terminates request	X									X
	Terminates processing the request					X	X	X			
	Sequence count goes to zero; terminates request		X	X							
	Reset line count; terminates request				X						
	Set pointer to start-of-tape; terminate this request									X	
	Do nothing								X		
5	Skip one file forward	X					X			X	X
	Slew cards to end-of-file		X								
	Do nothing			X	X	X		X	X		
6	Skip one file backward	X								X	X
	Do nothing		X	X	X	X	X	X	X		
7	Advance one record	X								X	X
	Do nothing		X	X	X	X	X	X	X		

Key:	MT	Magnetic tape	PTR	Paper tape reader
	CR	Card reader	PTP	Paper tape punch
	CP	Card punch	MSD	Mass storage driver
	LP	Line printer	PTD	Pseudo tape driver
	TTY	Teletypewriter	CD	COSY driver

†Assumes use of the magnetic tape physical device; for detailed information refer to COSY Driver in section 3.

Where:

rc	is	the request code.
thread	is	the thread location used to point to the next entry on the threaded list.
v	is	the error code setting.

One MOTION control can be repeated for magnetic tape. In this case the macro request is as follows:

```
        MOTION lu,c,r,p,n,0,rp,cp,a,x,d,m
```

Where:

r	is	the repeat function indicator that must equal R in the a parameter.
p	is	the motion code.
n	is	the number of times to be executed, not to exceed 4,095.
0	is	a null parameter.

All of the parameters are the same as in the preceding MOTION request except for r, p, n, and 0, that replace p_1,p_2,p_3, and dy.

The coding sequence generated is the same as above except for the last word, which is generated as follows:

15	14 12	11 0
1	p	n

Where 1 indicates that the request can be repeated.

The following macros can also be used for MOTION requests; each macro can perform only one MOTION request.

```
        BSR*    lu,a,n,c,p

        EOF*    lu,a,n,c,p

        REW*    lu,a,n,c,p

        UNL*    lu,a,n,c,p

        ADF*    lu,a,n,c,p

        BSF*    lu,a,n,c,p

        ADR*    lu,a,n,c,p
```

Where:

*		specifies a relative completion address. If left blank, there is absolute completion. (The macro computes the relative address constant.)
lu	is	the logical unit number of the device.
a	is	the absolute/indirect/relative indicator for the logical unit.

blank	lu is the actual logical unit number.
R	lu is a signed increment ($-1FF_{16} \leq lu \leq 1FF_{16}$) added to the address of the first word address of the parameter list to obtain the address of a location containing the actual logical unit number.
I	lu is a core address (0 to $3FF_{16}$) that contains the logical unit number.

n	is	the number of iterations. If blank, 1 is assumed (not to exceed 4,095).
c	is	the completion address. If the macro call terminator is an *, completion is relative (only the label name is required). If the macro call terminator is a blank, the completion is absolute. If c is left blank, there is no completion.
p	is	the priority level; it defines both the request and completion priority. If left blank, the priority is zero.

All parameters are optional and may be left blank with the exception of lu.

Examples:

The following parameters are common to the examples:

NEXT	is	the completion address.
6	is	the logical unit of the magnetic tape.
10	is	the logical unit of the card punch.
MT	is	the program location containing a 6.
FA_{16}	is	the low core location containing the standard binary output device.

1. A backspace macro with the following: A relative location containing the logical unit number, backspace 3 records, a relative completion address, and a request and completion priority of 3.

```
        BSR*    MT,R,3,NEXT,3
```

2. Same as the above, except that the completion address is absolute.

```
        BSR     MT,R,3,NEXT,3
```

3. An end-of-file macro with the following: The actual logical unit number, write one end-of-file, zero completion, and a priority of 0.

```
        EOF     10
```

4. Same as above, except that the logical unit number is in a low core location.

        EOF FA_{16},I,,,0

SYSTEM INTERFACES FOR I/O DRIVERS

Several system functions are common to most I/O drivers. The programs that perform the functions are resident in macro memory. The programs are always called with the address of the physical device table in the computer I register. The remainder of this section describes the functions that are performed.

FIND-NEXT-REQUEST

The find-next-request (FNR) subroutine is used by all driver initiator modules to find the next request for a device and to fill the physical device table with information from the request. FNR is entered by an indirect return jump through $B5_{16}$ with the core address of the physical device table entry in the I register.

DEVICE SHARED

The FNR subroutine scans the logical unit table, starting with logical unit one, to locate other logical units related to the same device. When a logical unit with a waiting request is encountered, FNR initiates the input/output device in the same manner as unshared devices. The lowest numbered logical unit with a request waiting for that device has the highest priority; i.e., the device priority completely supersedes the scheduling priority. Scheduling priority becomes operational only within the thread of requests when the logical unit is activated; i.e., if two logical units sharing the same physical device both have requests queued (in priority order), all requests from the lower numbered logical unit are executed before the first request from the higher numbered logical unit starts processing. If no requests are waiting on a device, FNR exits to the caller at the address of the call plus one.

DEVICE NOT SHARED

FNR examines the queue to obtain the next request. If none exists, FNR exits to the caller at the address of the call plus one. If another request is found, FNR updates the queue, fills the physical device table, and returns to the caller at the address of the call plus two. Upon return, the I register is unchanged and the A, Q, and overflow registers are destroyed.

COMPLETE REQUEST

The complete request (COMPRQ) subroutine is entered by an indirect return jump through B6 from input/output drivers to complete requests. This causes interrupts to be inhibited and the completion address to be scheduled with the error field from the physical device table, replacing the error indicator (v field) of the I/O request parameter list for logical units that do not share devices. Q is set negative if an error occurs. The request parameter list (containing a request code designating it as an I/O call) is interpreted as a secondary scheduler call by setting bit 15 of the first word to 1. The scheduler resets it to 0, and the device is released from its request assignment. When the driver has completed the request, control is given to the dispatcher. The dispatcher then passes control to the highest priority interrupted program or scheduled program. The latter might be the completion address if one was specified and is the highest priority program awaiting execution.

The complete request is entered by a return jump to COMPRQ that terminates the request by executing the following:

1. Resets the diagnostic clock counter (EDCLK) to $FFFF_{16}$
2. Transfers the error field in the physical device table (ESTAT1) to the v field of the request
3. Clears the operation in the progress bit (EREQST)
4. Clears the thread and returns to the driver if there is no completion address (C = 0)
5. Schedules the completion address if there is one, passing any error condition in Q and in the v field of the request, and returns to the driver

SET ERROR FLAG

The MAKQ subroutine is used by the drivers to set up the v field of the logical unit word of the request and to place the address of the last valid data into the last word of the caller's buffer.

DIAGNOSTIC TIMER MODULE

Initially, the diagnostic timer is operated after system start-up. Thereafter, it is periodically reactivated by a TIMER request. The frequency of operation is dependent on a parameter internal to the diagnostic timer program (normally 1 second).

An input/output hang-up error occurs when a driver fails to get a completion interrupt on an operation that it initiated. The diagnostic timer module detects hang-ups. The following features must be available for proper operation of the diagnostic timer module or these errors cannot be detected.

- A hardware device that gives periodic interrupts to measure time
- The timer request module
- The diagnostic timer module

The driver establishes a time differential (in increments of seconds) for each input/output operation; upon differential expiration a hang-up is assumed. This differential is entered in the physical device table slot for the device. The diagnostic timer then decrements the time differential each time the module is set into execution. When the differential becomes negative after decrementing, a hang-up is assumed. If the time differential is negative before decrementing, either the operation is complete or no operation has occurred.

When a hang-up occurs, the diagnostic timer accesses the physical device table entry for that device to obtain the driver core location to be executed in case of a hang-up. This location is executed by a SCHDLE request at the same priority level as the driver. Q contains the core address of the physical device table entry for the device. The driver takes any necessary action to clear the device involved in the hang-up. If recovery is not possible, it transfers control to the alternate device handler. The parameters passed are the logical unit number and error code.

The devices to be supervised by the diagnostic timer are specified by a table of physical device table addresses. This table (DGNTAB) is included in the SYSDAT program.

ALTERNATE DEVICE HANDLER

When a driver detects an irrecoverable failure of an associated driver, the following actions take place:

1. The driver sets the error field in bits 15 through 13 at word 9 (ESTAT1) of the physical device table for the device.

2. The controller hardware is cleared and an error word is set in the Q register.

15 6	5 0
Logical Unit	Error Code

3. The driver transfers control to the alternate device handler by a jump or a scheduler request with the error word in the Q register.

The following are typical errors:

Input/output hang-up (diagnostic timer)

Alarm

Parity error

Checksum error

Internal reject

External reject

The alternate device handler determines if the device has an operational alternate; if so, the request for the device that failed is assigned to it at the priority level of the driver operating the device that failed. The logical unit that failed is marked down, and the alternate is set active (bit 13 of LOG1 is set). The request is then rethreaded to the requested logical unit. If no alternate exists, the program reschedules itself at a low priority level to request operator intervention. In either case, all message output is executed from a low priority level section of the program.

The alternate device handler continues to assign alternates to devices that failed without waiting for completion of input/output messages. Therefore, the buffer table (ALTERR) must be provided to store the error words on entry. This table is included in SYSDAT. The size of the table is included in the first location of the table and is equal to the number of devices that can malfunction at one time. If this table size is not adequate for the system, the ADEV program hangs (18FF) when the table is filled. Identical device failures are not accumulated in the error table.

If the alternate is also inoperative and it does not specify an alternate, the procedure is the same as if no alternate were available for the original unit and it repeated until an operative alternate is found or until the handler determines that one is not available.

When an operative alternate is found, pointers are set so that requests from the failed device are automatically transferred to the alternate. The comment device has the following format:

L,lu FAILED ec

ALT, aa

Where:

lu	is	the logical unit number of the failed device.
ec	is	the error code.
aa	is	the logical unit number of the alternate.

If no alternate is found, the handler issues the following diagnostic:

L,lu FAILED ec

ACTION

Where:

lu	is	the logical unit number of the failed device.
ec	is	the error code.

The operator must respond to the error with one of the following and then press RETURN.

RP	Directs the request to repeat
CU	Reports the error to the requesting program; the device is allowed to continue processing requests.
CD	Causes any future programs calling the device to be informed of the failure upon completion; the error is reported to the calling program and the

device is marked down. No subsequent attempt is made to operate this device. The message

LU xx DOWN

is printed on the comment device.

DU Activates CU and terminates the current job being processed; the input unit attempts to slew to the next job to be executed.

DD Activates CD and terminates the current job being processed; the input unit attempts to slew to the next job to be executed.

If job processing is not in progress when the DU and DD options are selected, no action is taken, the word ACTION is retyped, and another option may be selected.

Mass storage device drivers do not use the alternate device handler. Mass storage device errors are logged in the engineering file.

The comment device must never be marked down because it is required to bring devices back up once they are operational. The dummy device driver, acting as an alternate for the comment device, restores the downed comment device.

If a downed device is requested by a program and if this device contains no alternate, the following message is typed on the comment device:

L,lu DOWN

Where: lu is the logical unit number.

This message occurs only the first time it is requested after being downed.

The completion address is always scheduled with an error. The requesting program should not repeatedly request downed units.

MASS-STORAGE RESIDENT DRIVERS

Most standard 1700 drivers are released with the capability to operate from core or mass memory residency with the exception of the following drivers:

- Teletypewriter/conversational display terminal keyboard devices
- 364-4 Communications Multiplexer
- Mass memory devices (except the flexible disk)
- 1747 Data Set Controller
- 1744 Digigraphics Controller
- 1500 Series
- Software buffer
- Dummy

All mass-storage resident drivers execute in a shared fixed buffer area located in SYSDAT. The allocation of this buffer is controlled by a core-resident executive routine, MMEXEC. It is possible to load either one or two drivers in the buffer, depending on its size.

The core buffer should be at least as large as the largest driver that is used in the system. The maximum size required that allows two drivers in core simultaneously is the combined size of the two largest drivers in the system. Whenever the DCOSY driver is used in a system, the maximum size criterion (the two largest drivers) should be used since DCOSY makes I/O requests upon another driver that has to be in core at the same time to complete the request for the COSY driver.

When a driver is mass-memory resident, the driver's physical device tables must declare their initiator, continuator, and time-out entry addresses to be the corresponding entries in MMEXEC (i.e., MASDRV, MASCON, and MASERR).

When an I/O request is made upon a mass-memory resident driver, control is routed to the entry point MASDRV. MASDRV determines if the driver is already in core and passes control to it. If the driver is not in core, it is determined if there is sufficient core available in the buffer for the new driver. If so, the driver is read in from mass memory and placed in execution. When there is not sufficient core for the driver, it is queued for later execution when space is released by drivers that currently occupy the buffer.

When a driver completes its input/output for all of the devices it controls, it releases its space in the core buffer by jumping to MASEXT. At this time MMEXEC resets all initiator, continuator, and time-out addresses for this driver's physical device tables to point to the corresponding entries in MMEXEC. If any other drivers are waiting in the queue, the first one encountered is read from mass memory and placed in execution.

When MMEXEC enters a driver, the Q register contains the physical device table address and the A register contains the first word address of the driver.

The mass memory location and size of each mass-memory driver must be contained in words 13 and 14 of the physical device table. These parameters are supplied by *S initializer control statements during system installation. If a driver is core resident, its mass memory size must be set to zero and its length to $7FFF_{16}$. This is not required for mass memory device drivers such as disk or drum.

INTERRUPT RESPONSE ROUTINES

Individual interrupt processing routines are used for interrupt lines that are assigned to only one device. These routines consist of setting Q to the address of the physical device table for that device and then transferring control to the driver continuator.

Example:

```
R17331     LDQ      =XP73310
           JMP*      (P73310+2)
```

The address of the interrupt response routine (R17331) is contained in word 3 of the interrupt trap for the interrupt line.

If several devices are assigned to one interrupt line, the interrupt processor must identify the device (usually by reading the status on each device) that interrupted. For some special devices the interrupt processing routine is an integral part of the driver. The address in word 3 of the trap is then set to the address of the processor for this specific interrupt.

ENGINEERING FILE LOGGING

All hardware failures detected by the I/O drivers are logged in the engineering file by the program log. The logical unit and error code are defined in the computer Q register (see Alternate Device Handler above). The call is:

```
        RTJ+    LOG
```

SCMM DIAGNOSTICS

The input/output for SCMM diagnostics is performed by the drivers. When errors are detected by a driver, a check is made to determine if the current logical unit assigned to the device is a diagnostic logical unit. If so, the error is logged and the request is completed with an error indicated. No call is made to the alternate device handler.

BUFFERED DATA CHANNEL ALLOCATION

The 1700 Computer System uses the 1706 Buffered Data Channel to buffer data transfers involving A/Q devices. This is useful for decreasing software overhead on input/output operations. Up to three 1706 units may be present in a system with up to eight devices on each unit. Since a 1706 may only transfer data to one device at a time, the buffered data channel allocator program (AL1706) is used to queue the use of the 1706 by drivers whose devices share one unit. The following drivers are designed to accommodate a shared 1706:

- 1726/1706/405 Card Reader
- 1731/1706/601 Magnetic Tape
- 1732-1/1706/608/609 Magnetic Tape

The following drivers control devices which each require a dedicated 1706 Buffered Data Channel and cannot use the 1706 allocator:

- 1747/1706 Data Set Controller
- 1744/1706/274 Digigraphics Console

The 1706 allocator must be requested prior to a 1706 operation. The request is:

```
        RTJ+    RQ1706
```

The I register contains the address of the driver physical device table. When the 1706 is available, a return is made to the driver with the physical device table address in the Q register. If the 1706 is not currently allocated, an immediate return is made to the driver. If the buffered data channel is currently in use, the driver return address, I register (physical device table address), and priority level are saved in the wait stack in SYSDAT. All parameterization for allocation is done through SYSDAT.

When the driver has completed usage of the buffered data channel, the channel is released with the call:

```
        RTJ+    RL1706
```

The I register contains the physical device table address. Following a release, the next user (first-in, first-out) is given control. With the physical device table address in the Q register, a return is made to the releasing driver. The allocator is set up to ignore requests for a 1706 (return to the caller with no action); if the requesting driver has already received a 1706, requests to release a 1706 are ignored if one has not already been allocated.

A/Q CHANNEL ALLOCATION

The 1700 computer input/output is an unbuffered operation performed via the A/Q channel. For devices where the data is contained on transportable media such as cards, paper tape, and magnetic tape and where the controller does not buffer the data, data can be lost if inadequate response time is provided to service a data interrupt. To avoid lost data, the drivers of these devices must run at a higher priority than the system hardware timer, if present.

The following types of devices are subject to this data handling restriction:

- Paper tape reader
- Paper tape punch
- Card reader
- Card punch
- Unbuffered magnetic tape
- Keyboard/conversational display terminal

The A/Q channel allocator (ALAQ) program is provided to allocate the A/Q channel if more than one of these devices is present in a system. The A/Q allocator is functionally equivalent to the 1706 Buffered Data Channel allocator, except that the ALAQ request entry is RQAQ instead of RLAQ.

AUTO-DATA TRANSFER (ADT)

Auto-data transfer (ADT) provides pseudo direct memory transfers of data blocks to or from a device. At the macro level, the transfer appears as a direct memory/storage access (DMA/DSA) transfer; however, at the micro level, the 1700 emulator processes each data interrupt and inputs or outputs the next word of data in a singular fashion. ADT takes less time than input/output via the INP, OUT, or SIO instructions, but more time than a true DMA/DSA transfer.

To use ADT for a particular device:

- The device and its controller must be capable of operating in the ADT mode and must adhere to the ADT specifications.
- The macro programmer must execute a DMI instruction (which specifies the location of the ADT table). Information in the ADT table specifies the beginning and end of the data block in main memory, the direction (input or output) of data flow, the equipment code (address) for selecting the device, and whether the data transfer is a word (16 bits) or a character (eight bits).
- The ADT operation must be initiated by an OUT (or SIO) instruction as specified by the particular device.

While the ADT operation is in progress, the emulator is executing macro instructions. However, after each macro instruction is executed, interrupts are checked. If the particular ADT micro interrupt has become the highest active interrupt, the next data word is input or output. After the interruption, the next macro instruction is executed unless there is another interrupt active.

When the ADT operation is completed (or if there is an error), a macro interrupt is generated. The macro programmer may then disable the ADT micro interrupt or initiate another ADT operation to or from the device. Note that for MO5 devices, an SPS instruction must be done to clear the macro interrupt.

Four types of ADT tables are specified by DMI instructions. They are described in the following sections.

ADT TABLE FOR SINGLE A/Q DEVICE

The ADT table for a single A/Q device is as follows:

	15	14	13	12	11	10 … 7	6 … 0
1	0	0	w/c	0	r/w	E	S/D
2	CWA						
3	LWA						
4	Not Used						

The ADT table for a single consists of four words:

Where: w/c is	type of operation	
	0	Word operation. For a word operation, data is transferred one word at a time. Normally a total of (CWA-FES+1) words are transferred.
	1	Character operation. For a character operation, data is transferred eight bits at a time. The first character is stored in the most significant half (bits 8 through 15) of the current word address; the second character in the least significant half (bits 0 through 7). Subsequent pairs of characters are input in the same fashion. Normally a total of 2*(CWA-FWA+1) characters are transferred.
r/w is	read/write	
	0	Read
	1	Write
E is	equipment number of the device. (It cannot conflict with any MO5 I/O port numbers.)	
S/D is	station/director bits of the device. Director bits should specify a data (not a status/function) transfer.	
CWA is	current word address counter during the ADT operation (which always points to the last data word read from or stored into main memory). During a data transfer, the emulator increments the current word address counter before the data is transferred. Therefore, it is necessary for the macro programmer to initially set word 2 of the ADT table to the first word address minus 1 of the data block in order for the transfer to occur correctly. Word 2 is used in conjunction with word 3 (by the macro programmer) to ascertain if all	

data (words or an even number of characters) was transferred after the ADT operation is completed. (If the current word address equals the last word address when the software driver is recalled with a macro interrupt, all data was transferred.) In character mode, word 2 in the ADT table is not actually updated until the second character has been transferred.

LWA is last word address of the data block to be transferred.

ADT TABLE FOR MULTIPLE A/Q DEVICES

The ADT table for multiple A/Q devices is as follows:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	0	0	E				Not used					1	0
2	T_{15}	T_{14}	T_{13}	T_{12}	T_{11}	T_{10}	T_9	T_8	T_7	T_6	T_5	T_4	T_3	T_2	T_1	T_0
3	T_{31}	T_{30}	T_{29}	T_{28}	T_{27}	T_{26}	T_{25}	T_{24}	T_{23}	T_{22}	T_{21}	T_{20}	T_{19}	T_{18}	T_{17}	T_{16}
4	Not used															
5	0	1	w/c	0	r/w	E				S/D						
6	CWA															
7	LWA															
8	Not used															
.	.															
.	.															
.	.															
I*4+1	0	1	w/c	0	r/w	E				S/D						
I*4+2	CWA															
I*4+3	LWA															
I*4+4	Not used															

The ADT table for this type consists of I*4+4 words, where I is the number of multiple A/Q devices (up to 32) on one micro interrupt.

Where: E is equipment number of the device. (It cannot conflict with any MO5 I/O port numbers.)

Th is termination bits for the 32 devices. Initially, they must be all zero. When a macro interrupt occurs (after RTERM is generated), one or more of these bits is set to indicate that one or more ADT operations have terminated. Thus $T_7 = 1$ indicates that station (or channel) number 7 has terminated its ADT operation. After receipt, the bit should be cleared via an instruction that locks memory (e.g., a CLF instruction).

Words 5, 6, 7, and 8 are same as words 1 through 4 of a single A/Q device except bit 14 of the first word must be 1. (Words I*4+1, I*4+2, I*4+3, and I*4+4 are defined in the same way.)

ADT TABLE FOR CLOCK

The ADT table for the clock is as follows:

	15	14	13	12	11	10 … 7	6 … 0
1	1	0	0	0	0	E	S/D
2	cc						
3	cl						
4	Not used						

The ADT table for this type consists of four words:

Where: E is equipment number of the clock (always equal to 1)

S/D is station/director bits of the clock (always equal to 70_{16}. (Thus, word 1 should equal $80F0_{16}$.)

CC is clock counters initially set to zero. Whenever the clock has been enabled, the clock counter is incremented every 3-1/3 milliseconds.

CL is clock limit, which is interpreted as a multiple of 3-1/3 milliseconds. When the clock counter equals the clock limit and the macro clock interrupt is enabled, the macro clock interrupt occurs. (If the clock limit is five, the clock interrupt interval is 16-2/3 milliseconds or 60 times a second.) To continue the process, the clock counter should be reset to zero, or the limit counter incremented by its original value (five). In this later method, the clock counter can function as an elapsed time counter.

ADT TABLE FOR SINGLE OR MULTIPLE MO5 DEVICES

The ADT table for single or multiple MO5 devices consists of (I-1)*4+4 words, where I is the number of MO5 devices, up to 8, on one micro interrupt. The ADT table follows.

	15	14	13	12	11	10	9 7	6	5	4 2	1	0
1	1	0	w/c	0	r/w	1	port	0	0	Not used	0	0
2	CWA											
3	LWA											
4	Not used											

.

.

.

(I-1)*4+1	1	0	w/c	0	r/w	1	port	0	0	Not used	0	0
(I-1)*4+2	CWA											
(I-1)*4+3	LWA											
(I-1)*4+4	Not used											

Where: w/c is single A/Q device

r/w is read/write

0	Read
1	Write

port is	port number of the device, where bit 10 is always 1. Port numbers are analogous to the A/Q I/O equipment numbers and cannot conflict with them.
CWA is	current word address. Initially set to the first word address minus 1 of the data block to be transferred. This is the current word address while the ADT operation is in progress and points to the last data word read/stored. Each time a word (or two characters, if character operation) is transferred, the current word address is incremented. The current word address can be used to ascertain if all the data was transferred after the ADT operation is completed (i.e., if the current word address equals the last word address, all data has been transferred).
LWA is	last word address of the data block to be transferred

Words (I-1)*4+1, (I-1)*4+2, (I-1)*4+3, and (I-1)*4+4 (where 2 < I < 8) are defined the same as words 1, 2, 3, and 4, respectively.

DATA FORMATS 3

Formatted requests cause data to be transferred in a format that is specific for each device type. The length of the transfer can be determined by the word count or the format. Unformatted requests transfer the data defined by the word count. This section describes the data formatting performed by the drivers for each device type. MOTION requests are handled by most drivers. The meaning and function of each MOTION command may differ for each device.

DUMMY DRIVER

The dummy driver performs no physical input/output. When the dummy driver is defined as an alternate, it completes failed I/O requests that have an error indication without operator intervention. The dummy driver also restores the failed device to an up condition. This technique is always used to prevent system hang-ups on the standard comment device. Programs should always check for I/O errors at their completion address and take appropriate action when errors occur (bit 15 of the Q register equals 1).

The dummy driver logical unit can also be used in place of any other system logical unit. In this case, the I/O request is completed with no error. This facility is useful for slewing through records.

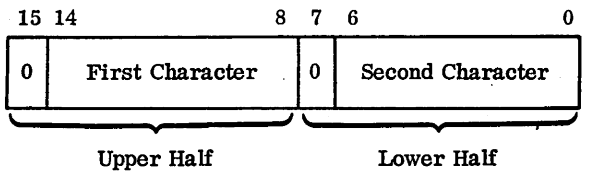
TYPEWRITER KEYBOARD DRIVERS

Four types of requests, READ, WRITE, FREAD, and FWRITE, are honored from the keyboard. Each request specifies the core starting address location being read into or written from the number of words and the completion address. All data is in ASCII format.

READ

The number of words specified by the READ request is filled, starting at the specified core location. Two characters fill one word; the first character is put into the upper half of the word. Bit 7 of each character is an even parity bit; it is set to zero after it is checked and before it is packed. If the parity bit is incorrect, a hardware error is indicated.

If zero words are specified, only one character is read into the upper half of the specified core location. The lower character is filled with a hexadecimal FF.



FREAD

Words in core are filled, starting at the specified core location and continuing until the number of words specified is filled or a carriage return is encountered. Two characters are packed in each word. Bit 7 of each character is interpreted as an even parity bit before being cleared to zero. Line feed characters are ignored. If a cancel character is encountered, characters are passed and no information is stored until a carriage return is detected.

The request is then repeated from the beginning. If a carriage return is not encountered before the specified number of characters is read, characters are passed until a carriage return is detected. A carriage return before the specified number of words is read constitutes a short read.

WRITE

The specified number of words is printed starting at the specified core location. Each word causes two characters to be printed with the upper half being printed first. If zero number of words is specified, only one character is printed from the upper half of the specified core location. If an ASCII end-of-text character (03_{16}) is encountered, the request is terminated at that point, regardless of the number of words specified.

FWRITE

This operation is the same as WRITE except that before any words are printed, a carriage return and line feed function are executed by the teletypewriter driver.

MOTION

A write end-of-file MOTION request to the keyboard is honored by executing a top-of-form function. All other MOTION request parameters cause no action with normal completion of the request.

ITOS 1 TERMINAL DRIVER

The ITOS 1 Terminal Driver provides communications between the ITOS executive and up to 16 752 CRT terminals. Each terminal is connected to the ITOS CYBER computer by way of an 1843-2 Communications Line Adapter (CLA). This driver also provides communication between the ITOS computer and the 1811-2 Console CRT.

The ITOS 1 Terminal Driver appears to the MSOS 5 system as a single logical unit. The request structure allows the caller to specify the particular CRT with which he wants to communicate. A subrequest structure allows requests in addition to the normal READ, FREAD, WRITE, and FWRITE requests.

REQUEST FORMAT

The request format is similar to normal MSOS 5 I/O requests, except for two parameters that have different meanings. The parameters are:

- Mode (bit 12 of word 3)

  The MSOS 5 request ASCII/binary mode bit is interpreted by this driver as specifying the meaning of the n parameter. If the mode bit m equals 0 (binary), n specifies the number of characters. If the mode bit m equals 1 (ASCII), n specifies the number of words.

- s (word 5)

  The s parameter in the MSOS 5 request is the address of the extended request parameter block, not the address of the data buffer.

The extended request parameter block consists of five words as shown:

	15 — 8	7 — 0
0	Reserved	Port
1	Completion Status	
2	Cursor (x)	{ Cursor (y) / Termination code
3	nc - Number of characters	
4	s - Address of data buffer	

Detailed parameter descriptions are:

Reserved:	Reserved for future use.
Port:	Logical port number. Set by the caller on READ, FREAD, WRITE, or FWRITE requests.
Status:	Status as at the completion of the request

Bit	Meaning
15	Subsystem down
14	Spare
13	Spare
12	Lost data
11	Framing error
10	Request timeout
9	Illegal request
8	Parity error
7	Spare
6	Spare
5	Spare

NOTE

If any status bit is set, bit 15 of Q is also set at the completion of the request.

4-0 Subrequest code as follows:

0	Normal mode
1	Logical connect
2	Logical disconnect
3	WRITE-READ
4-31	Reserved

Cursor (x):	Horizontal cursor position prior to input (WRITE-READ request)
Cursor (y):	Vertical cursor position prior to input (WRITE-READ request)
Termination code:	On READ, FREAD, and WRITE-READ, the driver returns the termination code. (See ITOS reference manual.)
nc:	Number of characters requested to be read for WRITE-READ requests set by driver to actual number of characters input on completion of READ, FREAD, and WRITE-READ requests.
s:	Address of the data buffer for the request.

Seven types of requests are recognized. They are described below.

MOTION

All MOTION requests are rejected.

READ/FREAD

The number of characters specified by n are input into the data buffer until a termination character is encountered. Any characters input in excess of the buffer length are ignored. Any characters stored in the buffer are echoed to the screen unless echo is disabled (see WRITE-READ). Entered characters may be erased by means of the ← (backspace) key. Termination characters are not stored in the buffer.

WRITE

The number of characters specified by n are output to the screen.

FWRITE

The number of characters specified by n are output to the screen. A CARRIAGE RETURN, LINE FEED sequence is output before any character. A LINE FEED character is added following each CARRIAGE RETURN in the buffer.

CONNECT

The CONNECT request is issued as a READ request and causes the driver to return all succeeding input to the buffer specified. Completion is scheduled at the address specified and at the priority specified. CONNECT sets unsolicited input mode. No completion is scheduled for a CONNECT request.

DISCONNECT

DISCONNECT request is issued as a READ request; it causes the port to be disconnected from the request buffer. Completion is specified as in the CONNECT request. DISCONNECT sets the port to non-unsolicited input mode. No completion is scheduled for a DISCONNECT request.

WRITE-READ

The WRITE-READ request is issued as a WRITE or FWRITE request; it delays write completion until a succeeding unsolicited input is received. No read completion is scheduled. WRITE-READ is rejected if the port is not connected. An 80$_{16}$ character in the output buffer disables the echoing of input characters until the completion of the unsolicited input.

CURSOR POSITIONING

The driver provides cursor positioning by use of the direct cursor addressing of the 752 CRT. Two methods are provided:

- A cursor positioning sequence (1B$_{16}$, 31$_{16}$, x, y) may be embedded in any write buffer.
- For a WRITE-READ request, word 2 of the extended request parameter block requests the cursor position that is to be used prior to receiving input data. A -1 (FFFE$_{16}$) in word 2 disables this feature.

CONVERSATIONAL DISPLAY TERMINAL (CDT) DRIVER

This driver is used only for diagnostic purposes. It communicates with the CDT through the 1843-2 CLA. The CDT driver works as a pseudo driver since it makes a non-standard MSOS request to the 1843-2 CLA driver to transfer the data on the CLA channel attached to the CDT.

The driver handles data as described above under the ITOS 1 Terminal Driver.

If an error occurs it is reported by the alternate device handler and logged in the engineering file. Fault code values are as follows:

Code	Meaning
0	Timeout
2	Alarm/status
3	Parity

PAPER TAPE READER DRIVERS

The paper tape reader honors two types of requests, READ and FREAD. Each of these requests may be in ASCII or binary mode, thus providing four different combinations. In addition to mode, each request specifies the starting core location being read into, the number of words to read, the completion address, the request priority, and the completion priority.

READ BINARY

The calling sequence specifies the number of words in core that are filled, starting at the specified core location. Two frames of tape fill one word, with the first frame packed into the upper half of a word. If zero number of words is specified, only one frame is read into the upper half of the specified word; the lower half is filled with a hexadecimal FF.

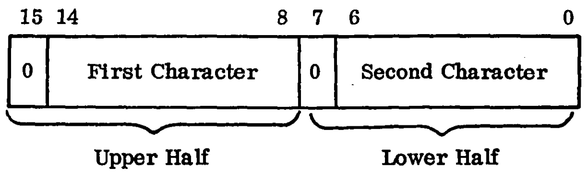
15 8	7 0
First Frame of Binary Data	Second Frame of Binary Data

PARITY CHECKING

Two logical units can be employed for READ and FREAD ASCII requests. One logical unit performs exactly as READ and FREAD ASCII requests. The logical unit identified by the name SKPAR (a word in the physical device table for the paper tape reader) performs READ and FREAD ASCII requests without parity checking. Bit 7 of each frame is set to zero without checking parity.

READ ASCII

The specified number of words is filled, starting at the specified core location. Two tape frames fill one word. Bit 7 of each frame is interpreted as an even parity bit (set to zero). If zero number of words is specified, only one frame is read into the upper half of the specified word. The lower half word is filled with a hexadecimal FF.



FREAD ASCII

Words in core are filled, starting at the specified core location and continuing until the specified number of words is filled or an ASCII carriage return character is encountered. If the number of words requested is read first, tape frames are passed with no information being stored in core until a carriage return is encountered.

Two tape frames are packed in each word with bit 7 of each frame interpreted as an even parity bit before being set to zero. Line feed, cancel, and null characters are ignored. If a carriage return is read before the word count is depleted, no further data is transferred. If the number of characters read is odd, then the lower half of the last word is filled with a hexadecimal FF.

FREAD BINARY

If the first frame read during FREAD binary is an ASCII asterisk, the request is changed to ASCII mode. If not, the first word is interpreted as the complement of the number of words in this formatted record. This number or the number of words requested in the calling sequence (whichever is smaller) determines the number of words being filled. After the entire record is read, the next word is a checksum which balances the sum of the header word and the information in the record to zero. If the sum is incorrect, a hardware error is indicated. If the word count is depleted before the end of the record, the tape is passed until the end of the record is found and the checksum is checked. If the end of the record is found before the word count is depleted, no further data is transferred. If the word count of the binary format record is more than 22,016, then 256 is subtracted to obtain the actual word count. This is necessary because the system cannot recognize word counts between 21,760 and 22,015; these numbers appear as an asterisk. Therefore, the system adds 256 to the word count of all paper tape format records of length 21,760 and larger on output.

EOF PROCESSING AND MOTION REQUESTS

The paper tape reader drivers provide the capability for handling end-of-files. An end-of-file is specified with 001C as the first frame of a record. If an end-of-file is detected during an ASCII or formatted binary request, the driver sets bit 11 of word 12 of the physical device table and completes the request with an error code. The alternate device handler is not called, and control is given to the caller's completion routine with the error code set in bits 15 through 13 of the Q register. End-of-files are treated as data in unformatted binary requests.

PAPER TAPE PUNCH DRIVERS

Two types of requests, WRITE and FWRITE, are honored by the paper tape punch driver. Both of these requests may specify ASCII or binary mode, providing four different combinations. In addition to mode, each request specifies the core location to be punched from, the number of words to punch, and the completion address.

WRITE BINARY

The number of words punched is the same as the number of words specified, starting at the specified core location. Each word is punched in two frames of tape with the upper eight bits being punched first. If the number of words specified is zero, only one frame is punched and it is the upper half of the specified word. If the number of words specified cannot be punched, a hardware error is indicated.

WRITE ASCII

The number of words requested is punched, starting at the specified core location. Each word is punched in two tape frames; bit 7 of each frame is changed to provide even parity. If the number of words specified is zero, only one frame is punched from the upper half of the specified core location.

FWRITE ASCII

This operation is the same as WRITE ASCII except that after all words have been punched, a carriage return and a line feed character are punched; when a carriage return is encountered, a line feed character is inserted following the carriage return.

FWRITE BINARY

The first word punched is the complement of the number of words being punched. The specified number of words is then punched at two frames per word, starting at the specified core location. A final word, which is a checksum, is then punched and, when added to the sum of the header word and the information in the block, balances to zero. If the proposed format record length is more than 21,760, then 256 is added to the word count prior to punching the record. This is necessary to avoid record lengths between 21,700 and 22,015 that appear as asterisks.

MOTION REQUEST

A write end-of-file MOTION request is honored by punching a file mark followed by leader. All other parameters cause no action with normal completion of the request.

ERROR CONDITIONS

Drivers recognize the following irrecoverable errors:

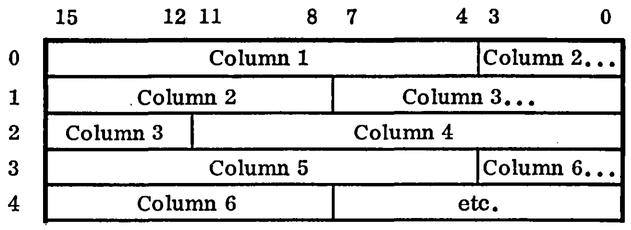
- Internal reject on input or output instructions
- External reject on input or output instructions
- Failure to interrupt
- Alarm
- Parity error on input
- Checksum
- Lost data
- Validation error while punching

The driver sets the error fields in the physical device table and the error parameter in the request. Upon entry to the completion program, the Q register is negative, indicating an irrecoverable error to the user. The alternate device handler is called by the driver for error recovery.

CARD READER/PUNCH DRIVERS

READ BINARY

The calling sequence specifies the number of words in core to be filled, starting at a requested beginning address. Card columns are packed, leaving no unused bits. After reading in binary mode, the buffer appears as:



If zero words are requested, only the first column of the card is read. The unused bits are set to ones; the remainder of the card is unavailable. If a READ ends in the middle of a card, motion continues but the unread portion of the card is unavailable.

READ ASCII

Words in core are filled, starting at a given address, until the number of requested words is filled. READ ASCII proceeds in the same manner as READ binary except that each column is converted from Hollerith to a 7-bit ASCII equivalent before being stored. These ASCII characters are stored two per word, leaving bits 7 and 15 as zero. If the number of words being read is zero, one column is read and the character is placed in the upper half of the word with ones placed in the lower half. See FREAD ASCII above for conversion code information.

FREAD BINARY

Although a formatted read operation may be specified with the request as binary or ASCII, the format of the card actually determines the mode. If column one of the card contains a 7/9 punch, it is read as a formatted binary record; otherwise, it is read as a formatted ASCII regardless of the mode specified in the request.

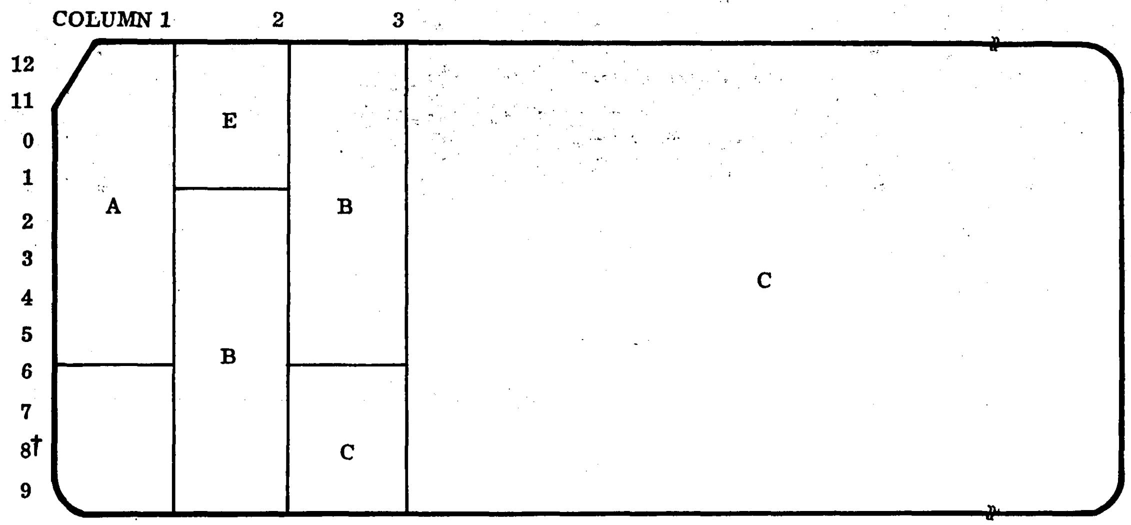
Figure 3-1 is the format for the binary record cards.

All cards within a record must have their sequence numbers in order. If a record requires multiple card storage and a sequence error is detected on any card within the record, the error is fatal and the alternate device handler is entered.

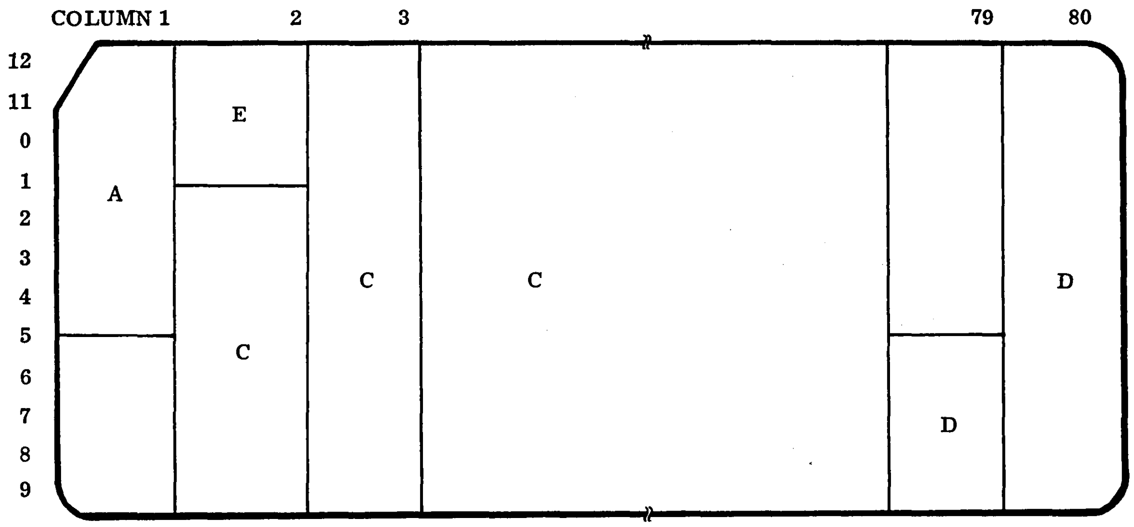
If the number is out of sequence, sequence numbers between records can be handled in two possible ways:

- The operator can resequence the cards, rereading the out-of-sequence card. Respond with RP to the sequence error.

FIRST CARD



SUBSEQUENT CARDS



WHERE: A IS THE SEQUENCE NUMBER (COLUMN 1, ROWS 12 THROUGH 5).

B IS THE COMPLEMENTED RECORD LENGTH (COLUMN 2, ROWS 2 THROUGH 9; COLUMN 3, ROWS 12 THROUGH 5 ON THE FIRST CARD).

C IS THE DATA (FIRST CARD STARTS IN COLUMN 3, ROW 6; OTHER CARDS START IN COLUMN 2, ROW 2).

D IS A 16-BIT CHECKSUM (IT FOLLOWS THE LAST DATA WORD OF A RECORD).

E IS RESERVED (COLUMN 2, ROWS 12 THROUGH 1). IT IS BLANK UNLESS A CHECKSUM OVERRIDE IS INDICATED IN COLUMN 1; OVERRIDING THE CHECKSUM IS NOT RECOMMENDED.

†IF ROW 8 IN COLUMN 1 IS PUNCHED, THE DRIVER IGNORES THE CHECKSUM.

Figure 3-1. Binary Format Record Cards

- The operator can read the cards out of sequence, rereading the out-of-sequence card twice. Respond with RP to the sequence error.

When an ASCII card is read, the driver sets the sequence number to zero. Thus, if an ASCII card is embedded in a formatted binary deck, a sequence error is reported when the next formatted binary card is read. If the operator uses the RP response to cause the unit to reread the binary card that caused the sequence error, the ASCII record is retained as part of the block of data and the driver recovers. Otherwise, this type of sequence error is fatal.

If the number of words requested is less than the length of the record, cards are passed with no data transferred until the entire format record is passed.

If the number of words requested is greater than the length of the record, data transfer ceases at the end of the record and no further cards are read for that request.

On a formatted binary card, row 8 of column 1 is the checksum override bit. When the driver detects this condition, a card's checksum is ignored. The following example illustrates how this function could be used.

Assume that while the user is loading a binary deck the card reader jams, damaging a card. The user can successfully duplicate the card, using the LIBEDT *T processor; however, the card's sequence number is incorrect. This is corrected by punching the sequence number on a keypunch machine, which causes the checksum to be incorrect. The user then punches the checksum override bit, allowing the card to be read properly.

FREAD ASCII

Columns are read to ASCII mode until either one entire card is read or the number of words requested is filled, whichever occurs first.

If the number of words requested is depleted prior to reading one card, the remainder of the card is unavailable and the read operation is in READ ASCII mode.

If a binary card is read when an FREAD ASCII is specified, either the card is read in binary (the first card of a record) or a 7/9 punch error occurs (not the first card). The Hollerith code conversion to ASCII can be done using ASCII63 (026 type) or ASCII68 (029 type). The conversion table is appended to the driver for this conversion:

- CR026 – ASCII63 conversion
- CR029 – ASCII68 conversion.

In addition, the 1726 Card Reader Controller provides for hardware conversion of Hollerith to ASCII for ASCII63. This hardware conversion is used if bit 15 of word 16 in the physical device table is set to one.

WRITE BINARY

The number of words specified is punched in the format illustrated in figure 3-2. When processing relocatable binary decks, the NAM card is offset for each deck punched.

WRITE ASCII

Each word in core is converted into two Hollerith columns. Characters not within the range 20_{16} through $5F_{16}$ are

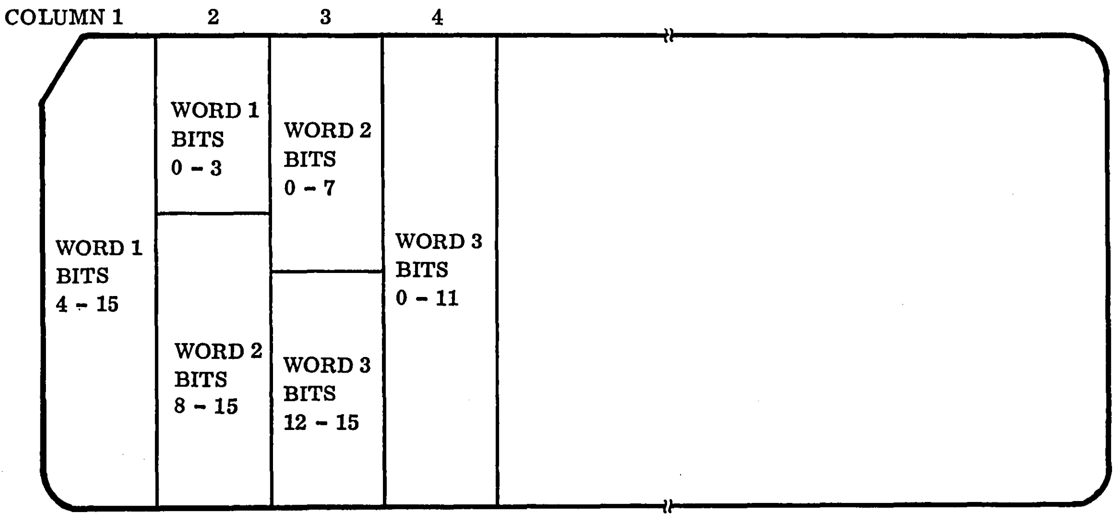


Figure 3-2. WRITE Binary Punching

converted to blanks. The character in the upper half of the word is punched first, followed by the character in the lower half. This continues until the number of words specified is punched. See FWRITE ASCII above for code conversion options.

FWRITE BINARY

Cards are punched in the format previously specified in this section.

The sequence base is reset to zero on detection of a formatted binary NAM block, and the card is offset. If the upper eight bits of the first data word are $2A_{16}$ (asterisk), the driver changes the request to FWRITE ASCII.

FWRITE ASCII

The ASCII FWRITE capabilities are the same as WRITE in ASCII mode, except that a maximum of one card is punched.

The code conversion from ASCII to Hollerith can be done from ASCII63 (026 type) or ASCII68 (029 type). A table program is appended to the driver for this conversion:

- CP026 – ASCII63 conversion
- CP029 – ASCII68 conversion

EOF Processing and Motion Requests

The card reader and controller drivers provide the capability of handling end-of-files. If an end-of-file record is detected by the driver on input, the driver offsets the record, sets bit 11 in word 12 of the physical device table, and completes the request with an error code. The alternate device handler is not called, and control is given to the caller's completion routine with the error code set in bits 15 through 13 of the Q register.

An end-of-file is a card with column 1 punched with the configuration set into bits 11 through 0 of word 16 of the physical device table. Normally it is a 6/7/8/9 punch; therefore, PHYSTB word 16 contains $000F_{16}$.

The MOTION request to skip file forward (parameter code 5) is honored by the driver. All other MOTION requests cause no action.

ERROR CONDITIONS

Errors detected by the driver are caused by equipment malfunctions or improper card decks.

The following conditions are detected:

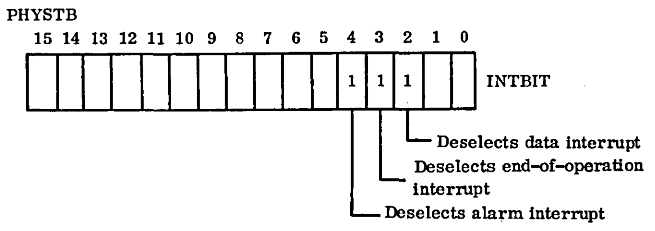
- Internal or external reject to any INP or OUT command
- Alarm interrupt
- Illegal Hollerith punch detected during an ASCII READ
- Data interrupt after column 80 on READ
- End-of-operation interrupt before column 80 on READ
- Incorrect checksum at the conclusion of READ
- Preread error
- Incorrect record sequence number
- Bit 15 of the complemented length is not 1
- 7/9 punch error

When one of these malfunctions occurs, the alternate device handler is entered by the driver. The alternate device handler functions in handling error codes, and the possible options available to the user for recovery are described in Alternate Device Handler in section 2. Refer to the 1700 MSOS Diagnostic Handbook for I/O error codes and descriptions.

CB104 CARD READER DRIVER

This driver processes READ, FREAD, and MOTION requests for the CB104 Card Readers. The CB104 Card Reader Driver is written in kernel format, and is initially mass storage resident. The nondiagnostic logical unit portion of this driver handles two or more card readers. The input or motion requests are handled successively, rather than concurrently.

The driver is interrupt driven unless the operator selects the status driven option. The status driven feature is selected by changing two words in the physical device table for that card reader (see appendix C). INTBIT (word 33) contains deselecting bits as shown:



If any INTBIT is set, WAITAD (word 34) must contain the absolute address of the status routine that checks completion of the I/O event. The kernel driver executes the routine and receives control after execution at the driver's continuator entrance. Transfers are made only on the A/Q channel.

The mechanical nature of card readers limits error recovery. The few operator-aided error recoveries are discussed below.

DATA FORMATS

The input data formats for the card reader are as shown above for the card reader/punch drivers.

READ/FREAD/MOTION

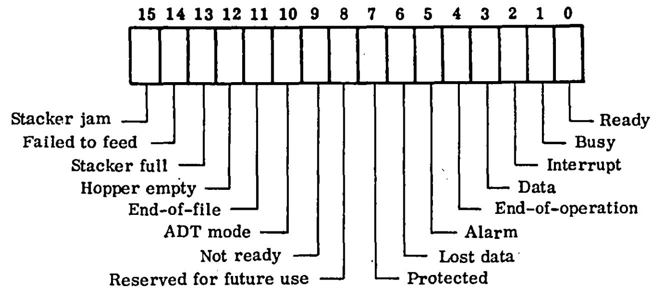
The READ/FREAD are as described above for the card reader/punch drivers. READ and FREAD may be performed in either binary or ASCII modes.

If the cards are out of order and the reader stops for that status error, the operator presses the RESET button, making the card reader not ready. The operator then removes the remaining cards from the hopper, causing a hopper empty status condition. The out-of-order card is reinserted in the correct sequence and the remainder of the deck is read when the operator represses the RESET button (making the card reader ready). The card reader then reads the remainder of the deck, now properly sequenced.

STATUS AND ERROR HANDLING

As discussed above, the operator has the option of driving the card reader from status information as well as from various interrupts.

Actual status (director status 1 and director status 2) is saved on the initiator, continuator, and timeout entries. Separate locations in the physical device table save the latest status. This feature is available for the diagnostic logical unit, as well as for the nondiagnostic logical unit. In addition, a combined director status 1 and director status 2 is provided. The results are placed into word ESTAT2 (word 12) in the physical device table. Combined status is shown below.



A diagnostic logical unit is supported by the driver. When using the diagnostic logical unit, a maximum of one card can be read even if the user's request specifies a word count greater than one card. A zero word request updates the two director status words in the physical device table but does not feed a card. For a nonzero request, raw data (12 bits right-justified per word representing a card column) is stored in the user's buffer. A request saves a maximum of 80 words of raw data. Errors detected by the kernel driver are not handled by the alternate device handler.

Timeout errors, bad status, internal rejects, and external rejects are reported in the physical device table. Detailed analysis of bad status is not performed. The fault code defining the error that has been detected is located in the physical device table, word 16. The possible fault codes are shown below:

Code	Meaning
0	Timeout
1	Lost data
	Bad initiator status (diagnostic logical unit
2	Alarm
	Bad continuator status (diagnostic logical unit)
4	Checksum error
5	Internal reject
6	External reject
8	Hollerith punch bad
9	Sequence error
10	Length error (formatted binary)
12	7/9 punch error
14	Not ready
22	Stacker full
23	Hopper empty
24	Feed failure
25	Card jam
34†	Data status at end-of-operation
35†	Premature end-of-operation status
53†	No end-of-operation
54†	No data status before end-of-operation
61†	No interrupt status
62†	Status shows ADT mode
63†	Busy after end-of-operation
64†	No busy before end-of-operation

†Indicates fault codes not currently in other MSOS card reader drivers.

TAB 501 CARD PUNCH DRIVER

This driver processes WRITE, FWRITE, and MOTION requests for the TAB 501 card punch. The driver is mass storage resident. More than one card punch can be handled by the driver, and requests are processed concurrently at the same priority level.

The driver communicates with the card punch through the 1843-2 Communication Line Adapter (CLA). The card punch driver works as a pseudo driver since it reformats data intended for the punch and then makes a nonstandard MSOS 5 request to the 1843-2 CLA driver indicating the CLA channel attached to the punch.

WRITE BINARY/FWRITE BINARY

The characteristics of binary punching are the same as those in the preceding description except that the TAB punch has no offset capability.

WRITE ASCII/FWRITE ASCII

The driver complies with the preceding standard description except in the method of Hollerith conversion. The conversion is done by the hardware and is available in ASCII 68 (029 type) only.

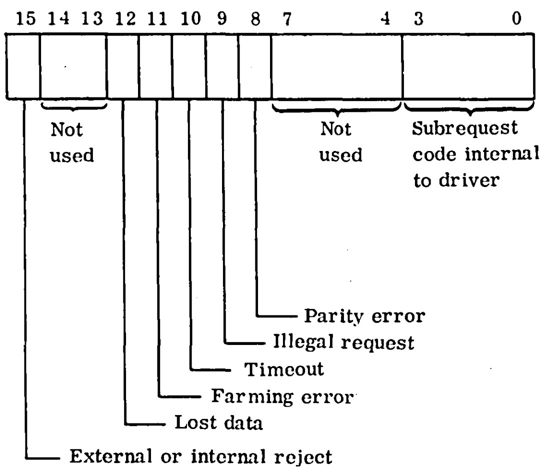
MOTION

An end-of-file request causes the end-of-file code contained in the punch physical device table to be punched and the sequence number to be set to zero.

The rewind/unload request will zero the sequence number and terminate the request.

STATUS AND ERROR HANDLING

Card punch status is not available. The status returned in the physical device table is the formatted status from the 1843-2 CLA driver. Bit assignment is:



If an error occurs, it is handled by the alternate device handler and logged in the engineering file. Fault code values are as follows:

Code	Meaning
0	Timeout
2	Alarm/Status
3	Parity

MASS MEMORY DRIVERS[†]

Data is transferred to and from disk/drum mass-storage drivers. These device drivers perform formatted (sector addressing) and unformatted (word address simulation) requests and detect motion requests that result in no operation. All mass-memory drivers also detect an overlay of the requester's parameter list by a read operation. In that event, sufficient information is moved from the user's parameter list to the physical device table to allow the completion routine to be operated.

The hardware compare feature is optional for disk drivers. It is enabled by resetting bit 15 of the unit select code in the unit's physical device table to a zero; it is disabled in the standard delivered system.

When addressing mass-memory devices, the first logical address is sector 1. Since the first five sectors (0 through 4) are reserved for the autoload program, logical sector 1 is physically located at sector 5. Mass memory drivers automatically bias all addresses by four sectors; all mass-memory requests, which originate in unprotected core, are biased to the beginning of scratch by the protect processor.

For devices having more than one drive, overlap seek is utilized for maximum data transfer efficiency. The software drivers for these devices initiate all required seek operations before starting a data transfer for a device that is on cylinder, thus overlapping the seek on several devices with data transfer on another device.

Cartridge disks utilize a single fixed disk and a single removable disk in a cartridge case. The disks are referred to as disk 0 and disk 1 and each has two recording surfaces. They are individually addressed by the hardware controller and differentiated by a single bit designator within the file address word. Autoload is always from disk 0, which is ordinarily the removable disk (disk 1 is the fixed disk). A toggle switch on the breakpoint panel of the 1739-1 Cartridge Disk Drive Controller allows disk addressing to be reversed; disk 0 becomes the fixed disk and disk 1 the removable disk. Note that autoload is still from disk 0. On the 1733-2/856-x Cartridge Disk, positioning is controlled by jumpers on the 1733-2 Cartridge Disk Controller in slot 15 that determine the drive type (-2 or -4) and the position of the data cables from the computer to the drive desired to be unit 0.

Software users reference the entire cartridge disk drive as a single logical unit with disk 0 containing the lowest sector address and disk 1 containing the highest. The disk referenced is dependent on the toggle switch position for disk addressing.

[†]Excludes storage module drive for CYBER 18 and micro processor flexible disk driver.

NOTE

The 856-2/856-4 Cartridge Disk Drives have a switch labeled WRITE PROTECT. When this switch is set, no data can be written on the storage device. There is no hardware status to indicate the setting of the switch and no alarm status given when a write is performed with the switch set. Therefore, it is the operator's responsibility to place the switch in the desired position.

DATA TRANSFER REQUEST FORMATS

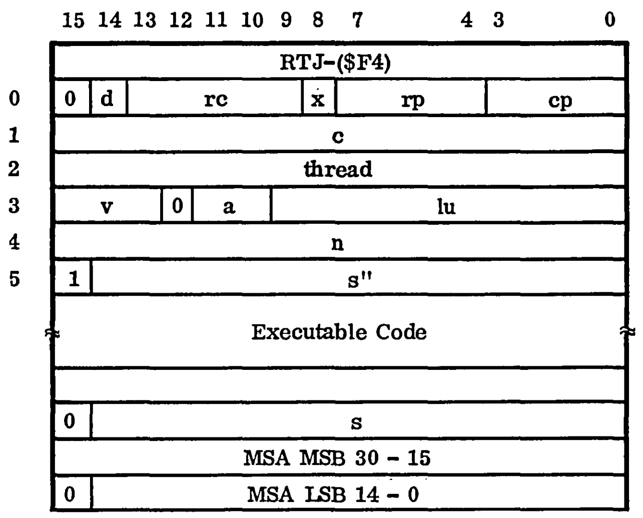
Execution of the data transfer request transfers n words from mass storage (READ/FREAD) or to mass storage (WRITE/FWRITE), starting at any core first word address indicated by s and the mass storage address indicated by MSA. If n is zero, one word is transferred. No data formatting is involved since corresponding core and mass storage locations contain identical 16-bit images.

The first request format is consistent with normal requests (the x parameter is not set) by providing a seven-word format.

If the x parameter is set, it indicates an indirect reference to the first word address increment contained in the s parameter. This positive increment is added to the address of the parameter list to form the address of a location containing another positive increment. The second increment is added to the address of the parameter list to obtain the starting address. This second increment is immediately followed by two words which contain the mass storage address (MSA) and which must comply with the first request format.

If parameter x is zero, both s and s" are absolute addresses; otherwise, they are 15-bit positive increments that are added to the address of the first request parameter (word zero) to form absolute addresses. Control is returned to the location following word 6 after the request is made.

The first request format is as follows.



The second request format adds two additional words that contain mass storage address in line with the conventional data transfer format and that are identified by a direct reference to s (bit 15 equals 0).

The conventions defined previously for s in relation to x and d also apply here. Control is returned to the location following word eight after the request is made. The second request format is as follows.

	15 14 13 12 11 10 9 8 7 4 3 0					
	RTJ-($F4)					
0	0	d	rc	x	rp	cp
1	c					
2	thread					
3	v	0	a	lu		
4	n					
5	0	s				
6	MSA MSB 30 - 15					
7	0	MSA LSB 14 - 0				

DISK AND DRUM DRIVER REQUESTS

The disk/drum driver processes requests from the user programs for data transfer to and from mass storage (READ/WRITE/FREAD/FWRITE), provides a program overlay capability, and handles the transfer of mass storage resident system directory programs into core (SCHDLE); mode has no meaning.

The number of words specified in the calling sequence is transferred to or from core, beginning at the specified starting address and sector number. Sectors are read or written sequentially until the requested number of words has been transferred. If zero words are requested, the driver transfers one word to or from core.

READ/WRITE

READ and WRITE requests provide the ability to simulate the word address by allowing the mass storage address to be any word address within the size range of the disk/drum. The driver converts the word address to sector and word in the sector by dividing by 96.

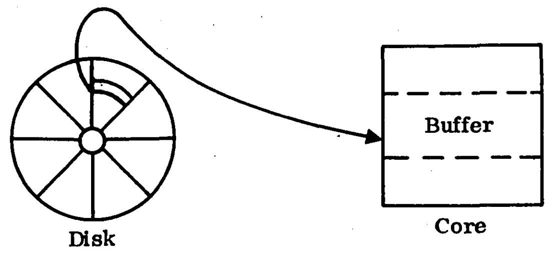
READ – The READ request fills core with the specified number of words starting at a specified address. If zero words are requested, one word is transferred. Transfer is initiated from the disk word address specified by the most significant bits (MSB) and least significant bits (LSB) of the request (least significant bit is a 15-bit value). A carry into bit 15 of the least significant bits should be treated as an overflow condition and the most significant bits should be incremented by one.

The following is an example of a READ request in which C is the completion address.

```
READ      8,C,BUFFER,15,B,4,4,,1
  •
  •
  •
ADC       $1,$6D59
JMP-      ($EA)
BSS       BUFFER(15)
```

As a result of this request, 15 words, starting at disk-word address 1,6D59$_{16}$, are read from logical unit 8 (disk) into core, beginning with the first-word address buffer. Disk-word address 1,6D59$_{16}$ is the same as sector 632, word 88 (divide 00016D59$_{16}$ by 96 for sector and word).

This example is illustrated by the following:

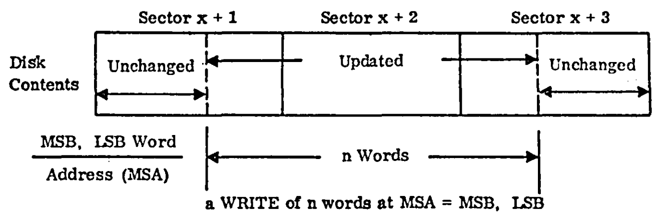


WRITE – The WRITE request transfers the requested number of words from core to disk. The disk starting address (most significant bits, least significant bits) is interpreted by the driver as a word address. When writing on the disk in this mode, the remainder of partially updated sectors is preserved.

A partial sector WRITE request causes:

- The entire sector to be read into a buffer in the driver
- The user's data to be moved into the appropriate portion of that buffer
- The entire buffer to be written onto the disk

The following is an example of a word-oriented WRITE across several sectors:



An indirect WRITE request is similar to the indirect READ request. Fifteen words are written on the disk at sector 632, word 88; other words in the sector remain unchanged.

FREAD/FWRITE

FREAD and FWRITE requests utilize the sector orientation of the disk. The formats of FREAD and FWRITE are the same as READ and WRITE. The mass storage address represents a sector number; n represents the number of words to be transferred. If n is not a multiple of 96 for an FWRITE request, the unused words of the last sector are set to zero.

FREAD – FREAD fills the core, starting at a requested address, with the specified number of words. If zero words are requested, one word is transferred.

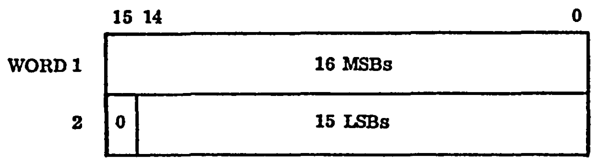
FWRITE - This request transfers the specified number of words from core to disk. The starting disk address is interpreted by the driver as a sector address; the most significant bits must be zero. If a zero number of words is requested, one word is transferred. The remainder of a partially updated sector is not preserved.

Using the same symbolic conventions as the previous example, a normal FWRITE request appears as:

```
FWRITE   8,COMP,BUFFER,113,B,4,4,1
  .
  .
  .
ADC      0,103
  .
  .
  .
BSS      BUFFER(113)
```

In this case, 113 words are written from the core first-word address buffer onto the disk, starting at sector 103.

The cartridge disk drive has more than $7FFF_{16}$ sectors and requires both words of the request to specify the mass memory sector address. The sector address is defined in the same manner as a word address (the 16 most significant bits in word 1 and the 15 least significant bits in word 2 with bit 15 of word 2 set to zero).



MOTION

MOTION requests to the disk result in no action, and return from the disk is through a normal completion procedure.

ERROR CONDITIONS AND RECOVERY

The following errors are detected by the drivers:

- Internal and external rejects
- Parity error
- Seek error
- Address error
- Lost data error
- Protect fault
- Checkword error
- Defective sector error
- Compare error
- Time-out error

Error recovery is not attempted with parity, protect fault, and time-out errors.

Several methods of error detection are employed during disk transfers. After data reads and writes, a hardware compare function can be issued to compare the data read or can be written with the data contained on the file. When an error is detected, a reposition and retry can be attempted up to ten times.

When an irrecoverable error occurs, the driver sets the error field of the disk physical equipment table and the error parameter in the request. The Q register is negative upon entry to the completion program and indicates an irrecoverable error to the user. No information about the nature of the error is passed to the user.

The alternate device handler is not used for error recovery since no meaningful alternate device exists.

1833-4/1866-12/1866-14 CARTRIDGE DISK DRIVER (CDD)

Data is transferred to and from disk mass-storage drivers. These device drivers perform formatted (sector addressing) and unformatted (word address simulation) requests and detect motion requests that result in no operation. All mass-memory drivers also detect an overlay of the requestor's parameter list by a READ operation. In that event, sufficient information is moved from the user's parameter list to the physical device table to allow the completion routine to be operated.

The hardware compare feature is optional for disk drivers. It is enabled by resetting bit 15 of the unit select code in the unit's physical device table to a zero; it is disabled in the standard delivered system.

For devices having more than one drive, overlapping seek operations are utilized for maximum data transfer efficiency. The software drivers for these devices initiate all required seek operations before starting a data transfer for a device that is on cylinder, thus overlapping the seek on several devices with data transfer on another device.

Cartridge disks utilize a single fixed disk and a single removable disk in a cartridge case. The disks are referred to as disk 0 and disk 1; each has two recording surfaces. They are individually addressed by the hardware controller and differentiated by a single bit designator within the file address word.

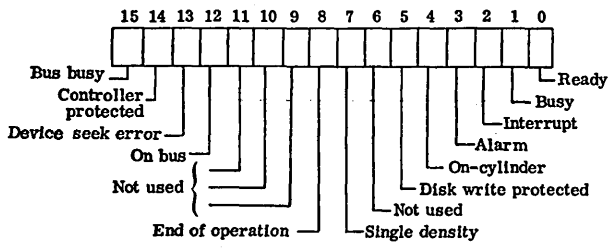
Software users reference the entire cartridge disk drive as a single logical unit with disk 0 containing the lowest sector address and disk 1 containing the highest. The position of the toggle switch determines which disk is to be addressed.

DATA TRANSFER REQUEST FORMAT

The READ/FREAD, WRITE/FWRITE, and MOTION commands have been described above under Mass Memory Drivers.

STATUS AND ERROR HANDLING

Status bit definition for the cartridge disk drive controller is as follows:



The following errors are detected by the driver:

- Internal and external rejects
- Parity error
- Seek error
- Address error
- Lost data error
- Protect fault
- Checkword error
- Defective sector error
- Compare error
- Time-out error
- End of medium
- Bus is relinquished
- Disk is write protected
- Errors in current word address, bank status, true cylinder and true sector

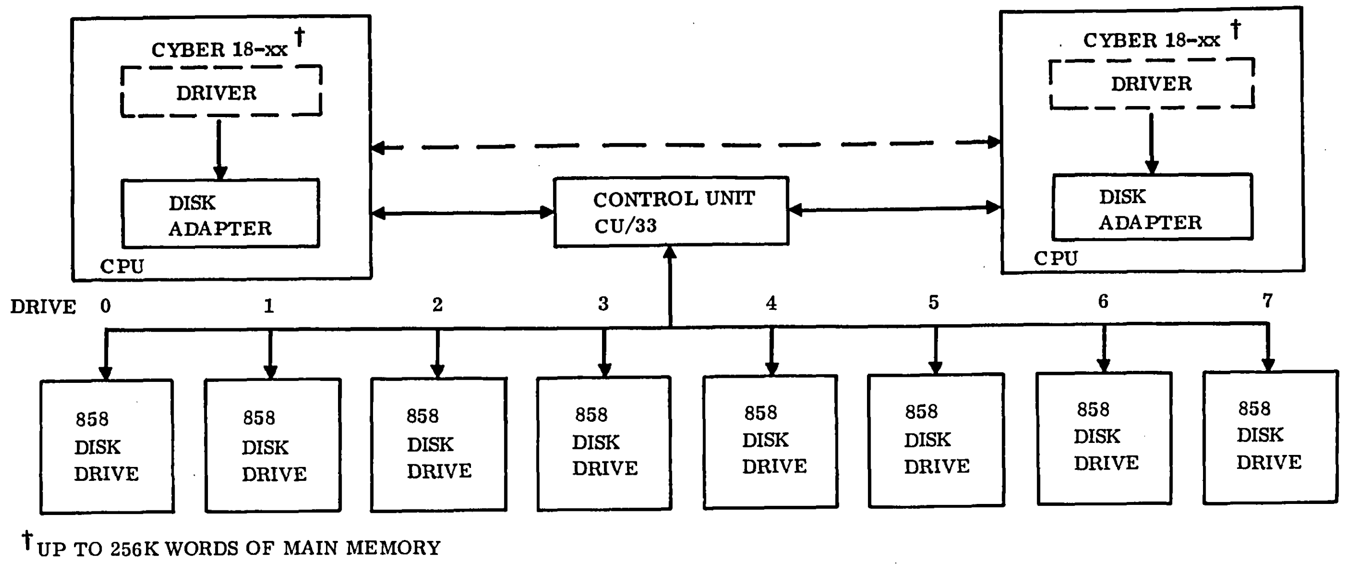
STORAGE MODULE DRIVER (SMD)

This driver handles CYBER 18 data transfers to and from disk. The driver is written in kernel fashion so that portions may be deleted. This MSOS 5 driver provides seven major functions:

- MSOS driver (kernel structure)
- Pseudo disk handler (to provide full addressing for CPUs that are restricted to a single 16-bit mass storage address)
- System initializer driver
- Disk-to-tape driver
- SYSCOP bootstrap routine to transfer core image to disk
- Write disk addresses routine
- Initialize disk (track sync logic and clear disk)

The driver controls the CU/33 Disk Control Unit. That unit in turn controls up to eight 858 Disk Drives. The driver may be installed in a single or dual-CPU configuration. Each CPU that accesses the disk must contain disk adapter hardware, which is used primarily to check track and sector address. If the requested address differs from the address read, the data transfer is inhibited. The maximum hardware configuration for this driver is shown in figure 3-3.

The use of the single controller by the two CPUs is serial, not parallel. Disk control is regulated by the drive status usage table. At autoload time, the SPACE program of the driver in one CPU takes control of the table, clears it, and autoloads. The other CPU must not attempt to access disk during this period. Thereafter, both CPUs' drivers use the



0346

Figure 3-3. Storage Module Driver Maximum Hardware Configuration

table to gain and release control of the control unit. Safeguards are provided to prevent one CPU from locking out the other CPU. These are as follows:

- Normal: When one CPU has terminated its use of the controller, an alternate disk adapter interrupt is generated for the second CPU. The second CPU may then process its drive requests.

- Forced: When the non-using CPU has a mass storage request, it starts a five-second diagnostic timer. If this preset time elapses before a normal release of disk by the using CPU occurs, the requesting CPU generates a demand control command that disconnects the control unit from the using CPU. This function prevents the disk from being captured by a CPU that has failed. On the other hand, since this forced interrupt condition destroys any data transfer or seek in progress, the operation is catastrophic with respect to the using CPU if it is not already in a failed condition.

When a CPU controls the disk, all seek operations must be handled first. The control unit allows overlapping seek operations on each disk drive except the one on which a read/write operation is currently taking place. The seek operations cannot be commanded by the user, i.e., a MOTION command results in a no-operation. The driver itself develops a seek command (if necessary) as the first step of a READ, WRITE, FREAD, or FWRITE operation. Further, the seek complete interrupt is transparent to the user, since this is handled entirely within the driver. If the seek command positioned the driver heads properly, the read/write transfer is initiated when it is the first such transfer in queue.

In addition to the five-second diagnostic timeout, three other timeout periods are provided by the driver: one second to complete a seek operation, one second to complete a read/write operation, and three seconds to complete the alternate disk adapter interrupt. The latter, therefore, requires the requesting CPU to take control of the control unit within three seconds of the time when the using CPU has indicated it is ready to release the disk.

DISK PACK INITIALIZATION

The disk packs for the disk drivers consist of five platters; an inner and outer protective platter and three inner platters (six surfaces) for magnetic information. One full surface is preformatted for head positioning (die kits). This feature allows the head to be offset perpendicular to the track. As a result, slight misalignments from pack to pack and drive to drive may be automatically compensated by the driver to preserve the stored conformation. This is discussed in the Error Recovery section.

Packs may be low density (410 tracks per surface) or high density (821 tracks per surface). The five data tracks directly over or under one another comprise a disk cylinder. Each track is divided into 64 sectors of 96 words each (192 8-bit bytes per sector). The tracks per cylinder are numbered consecutively (0 through 4) and the tracks per surface are also numbered consecutively (0 through last equals 409 or 820). Track identification is a hardware function.

Sector identification is determined at system ordering/configuration time. The sequence of sector identifications is held in a sector addressing table (part of the address tag routine). Normally the order is 0 through 63 consecutively. However, should the application encounter an I/O timing problem such that the CPU has difficulty in providing or accepting data at the storage module drive transfer rate (1.65 microseconds or 2.48 microseconds per byte), sectors could be interlaced so that a full sector would be skipped between each pair of sectors holding sequential data. This would effectively decrease the maximum data transfer rate to/from the CPU by 50 per cent, since two disk revolutions would be required to transfer a full track of data. Normal sector addressing and the alternate mode discussed are shown in figure 3-4.

Two driver routines are needed to fully prepare a disk pack: the pack initializer routine and the address tag routine.

Pack Initializer

This is a stand-alone program to initialize the tracks and to zero the data portion of the track.

Initialization need be done only once to set track identifications for the pack. Two methods of calling the initializer are provided:

- *SMDMPI generates calling instructions on the comment device so that initialization can be executed from a bootstrap.
- *JOB calls the job processor. Then *SMDINT executes the initializer (i.e., this routine is a file on the program library).

In the first case, instructions are presented to the operator to set register values. In the second case, A and Q registers must be set as shown below:

A	Drive number holding pack
Q	WES code

Where: W = S = 0 always
E = 3 normally. The driver accepts Q = 0 or Q = 0700_{16} for equipment code 3. If E does not equal 3, then the WES code must be entered to designate the disk equipment identification.

At completion, with the selective stop switch on, the Q register equals 0 if the initialization was proper. Otherwise, Q does not equal 0 and the pack must be reinitialized. If this cannot be done by repeating the above procedure, disk diagnostics should be run to isolate the hardware errors.

The address tag routine must follow pack initialization.

Address Tag Routine

This is a stand-alone program to write sector addresses in the sequence specified by the sector addressing table. The routine must be called after pack initialization, but it may also be called if diagnostics indicate addressing problems. After the sector addresses are written on the tracks, the

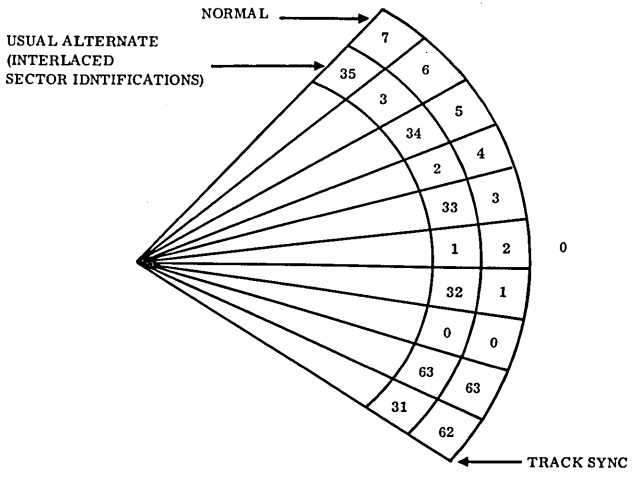


Figure 3-4. Normal Sector Addressing and Alternate Mode

data portion of the track is filled with 4142 (AB in ASCII character mode). The user may use this value to check data writing if the errors exceed the 11-bit driver error compensation limit, which is discussed below.

The address tag routine may be called by any of three methods:

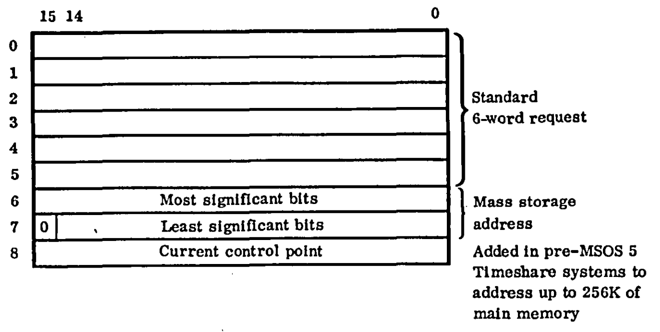
- *SMDMPT generates calling instructions on the comment device so that the address tag routine may be executed from a bootstrap.
- *SILP calls the system initializer (see the MSOS Version 5 Reference Manual); then *G executes the address tag routine.
- *JOB calls the job processor. Then a *SMDTGS executes the address tag routine (i.e., this routine is a file in the program library).

In all cases, the A and Q registers must be set as they were for the pack initializer (described in the Pack Initializer section above).

At completion, with the selective stop switch on, the Q register equals 0 if all sector addresses were written properly. Otherwise, the attempt to rewrite the tags must be repeated. If Q does not equal 0 for each of several retries, diagnostics should be run to isolate the hardware error.

READ/WRITE/FREAD/FWRITE

The standard request formats for the read and write requests (formatted or unformatted) are used and are described in section 1 with the following changes (direct format shown).



The first added words (words 6 and 7) are necessary to specify the disk address. However, if the system used is restricted to single word sector addressing to disk, only word 6 is used. To address the entire disk, a pseudo disk handler is provided. This handler partitions the disk drive into four equal parts: 0 through $7FFF_{16}$, 8000_{16} through $FFFF_{16}$, 10000_{16} through $17FFF_{16}$, and 18000_{16} through $1FFFF_{16}$. The segments are addressed as pseudo disks 0, 1, 2, or 3 respectively plus disk address 0 through $FFFF_{16}$. The user program specifies the pseudo disk in the WES code; the pseudo disk handler employs its own physical device table to convert the pseudo address to a true 31-bit disk address for the control unit.

Current control point (CCP) indicates the segment of core from which the request originated. The data transfer must remain within this segment. MSOS Version 5 has a mass memory manager, hence word 8 (CCP) is not required.

There are no buffer size constraints for either input or output disk transfers under MSOS 5, nor are the buffer size constraints on output (write to disk) under any operating system which interfaces with the disk. If a write transfer extends acrosss a cylinder boundary, the disk handler divides the request into two sections, but this operation is transparent to the using program. There may be buffer size (or placement) restrictions on MSOS 4 if the main memory is greater than 65K in size. The buffer in that case must remain within the 65K sector of main memory designated by the control point in the read request.

The driver automatically biases the sector addresses by five to compensate for the first five physical sectors (0 through 4) that are reserved for autoload and the sixth sector that is also used by the system. The first sector address is designated as 0_1. Mass memory requests originating in unprotected core are saved in the protected scratch area by the protect processor unless swapping is inhibited by the unprotected program.

ERROR RECOVERY

The driver provides automatic recovery for four of the five types of errors that can be encountered on mass storage requests. In the recoverable cases, an error code is logged in the engineering file if the operation failed after the maximum number of error recovery attempts. The error code is also saved in the status word for return to the user program (status values are described below). The maximum number of error recovery attempts for each type of failure is a prestored parameter in the physical device table (PHYSTB). Except for the error correction code (ECC) generated by write errors (described below), no indication of corrected errors is saved. No error will be logged if the unit is inactive or if it is the diagnostic logical unit.

Control Unit Connection Error

The connection is attempted five times. If the control unit is currently being used by the second computer, the diagnostic clock (5 seconds delay) is started and no error is logged. If the control unit should be available but cannot be connected after five retries, the transfer is aborted and an error code of 70 is returned to the user.

Drive Connection/Seek Error

Five retries are allowed to connect the disk drive. The error code is 70 if the connection fails. After the drive is connected, the driver checks ready status. If the unit cannot be made ready, the error code is 14. Then seek status is checked when the seek interrupt is received. After the maximum number of retries (a physical device table parameter) has failed, the error code is 17.

Error Correction Code Error

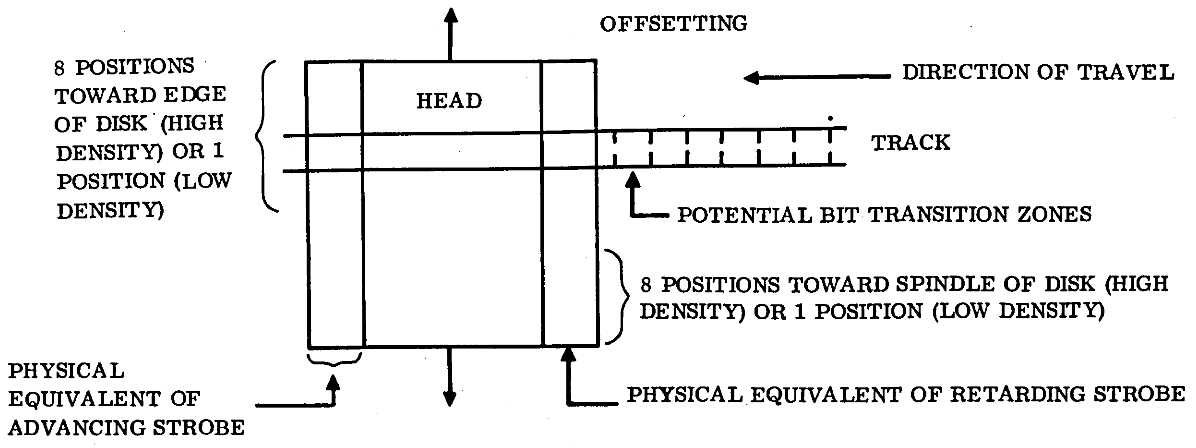
If a write operation fails, it is retried and a special code is generated in line with the data within the sector. The code can detect an error up to 22 bits in length per sector and can automatically correct an error up to 11 bits in length. The error correction code recovery attempt is selected by a flag set in the physical device table. If the error exceeds 11 bits in length and is therefore uncorrectable, the error code is 71.

Note that this system does not declare bad tracks or bad sectors. Instead, the attempt is made to correct a bad spot on the track by use of the error correction code. On reading the data, the correction code is applied to the bad data spot, and the output from the driver is the reconstituted (correct) data. This reconstitution is transparent to the user program.

Data Transfer Error (Read)

Because of the high density of data packing, positioning of the head over the track is critical, both in line with the data in the track (strobing variations) and perpendicular to the track (head offset variations). This type of positioning error is most frequently caused by recording data on one disk drive and reading it back from another. See figure 3-5.

To counter this type of error, the driver automatically retries reading the data using numerous combinations of


OFFSETTING
8 POSITIONS TOWARD EDGE OF DISK (HIGH DENSITY) OR 1 POSITION (LOW DENSITY)
HEAD
DIRECTION OF TRAVEL
TRACK
POTENTIAL BIT TRANSITION ZONES
8 POSITIONS TOWARD SPINDLE OF DISK (HIGH DENSITY) OR 1 POSITION (LOW DENSITY)
PHYSICAL EQUIVALENT OF ADVANCING STROBE
PHYSICAL EQUIVALENT OF RETARDING STROBE

0345

Figure 3-5. Head Positioning

offset positions and strobing variations (see figure 3-5), the total number being controlled by parameters in the physical device table.

If no offset/strobing combination succeeds in reading the data, the error code is set to 41.

Force Release Error

This error results in the using CPU from the 5-second diagnostic clock timeout in the requesting CPU. It should never occur unless the using CPU is mass memory I/O bound. The error is unrecoverable since the using CPU's interruption is hard; i.e., no time is provided to complete or save the interrupted operations (the programming overhead to do this would be prohibitively wasteful of space and time). Hence, the interrupted CPU has only a bare indication of the reason for the I/O failure, and the driver variables/flags are left in an uncleared condition for the next set of data transfers. As explained earlier, the purpose of the demand function is to recapture control of the control unit from a failed CPU.

Error Codes

The error codes for all errors are:

Code	Meaning
0	I/O hang up (1-second data transfer timeout)
2	Alarm error (hardware)
5	Internal reject
6	External reject
14	Unit not ready (disk drive)
17	Seek error
41	Unsuccessful request (data transfer error)
44	Guarded address (write protect switch set)
70	Connect error (control unit or disk drive)
71	Uncorrectable by error correction code (read or write)
72	Extraneous (ghost) interrupt
73	Forced release error
82	Control unit error
83	Mass storage address error (disk adapter addresses nonexistant main memory)

DIAGNOSTIC FEATURES

Some diagnostic features included in the driver are optional. These diagnostic features must be defined during the system ordering/configuration phase. The optional diagnostic buffer in which the parameters are defined becomes a part of the normal physical device table for the disk. The physical device table for the disk is included in appendix C.

The additional requirements specified in the test unit's physical device table during initialization are:

- The diagnostic unit (word 17, DIAGLU) must be designated as the read/write request logical unit number.
- The diagnostic request type code (6, 9, and 10 for format write pack initialize, read address tags, and write address tags respectively) must be inserted into bits 3 through 0 of word 52 (DIAGSP) of the requested unit's physical device table.

- The standard driver specifies that one retry should be made for each combination of strobe/offset. However, there is the option of selecting combinations for diagnostic operation and retrying selected combinations more than once. The allowed data transfer error retrial value must be set to zero. The strobe and offset value (labeled STROBE, words 64 in the unit's physical device table) is set to a desired value (upper eight bits) before calling the driver.

The value(s) changed by the diagnostic unit must be restored when the diagnostic function is completed. When the caller regains control, the last status of the disk adapter, control unit, etc., is stored in the 40-word diagnostic status buffer (see appendix C).

For read/write address tags operation, it is advisable to set up the MSOS read/write request as formatted. This eliminates the odd word format (five words per address tag data), which can create errors if the unformatted MSOS type read/write request is used. Furthermore, the address tag operation is a track type function, so it is advisable to force any starting mass memory address to be the beginning address of a track.

Following a diagnostic operation, the caller regains control regardless of the error detected during the I/O operation. The error encountered is not logged in the engineering file; instead, the caller regains control as if a successful operation occurred. However, the error code is stored in word 16, labeled FLTCOD, in the unit's physical device table. The proper error retrial count, if applied, is stored in its type counter (such as word 40 for seek error count, etc.). It is caller's responsibility to examine these counters in conjunction with the status words to determine the proper result.

STORE CORE IMAGE

This CPU-resident program copies the current main memory to disk. Any error forces the routine into a selective stop loop with Q equal to $FFFE_{16}$. The image must be written in the lower 32K sectors of disk, and the CPU memory size is assumed to be 65K words or less. The transfer is always made to drive unit 1.

DISK-TO-TAPE DRIVER

This program provides the capability of transferring data in either direction between disk and magnetic tape. The storage module drive interfaces with the disk-to-tape utility routines (DTLP), which use a single word sector address, limiting disk access to the lower 32K sectors. The program utilizes error correction code recovery techniques as well as strobe/offset head positioning to minimize data transfer errors.

FLEXIBLE DISK DRIVER

This driver handles data transfers to and from the flexible disk. Two flexible disk units (diskettes) may be controlled from the same flexible disk control unit. Each diskette is identified as a separate logical unit. The identity of a diskette (0 or 1) is determined by a switch in the controller. Since each diskette has its own physical device table that contains parameters determining the addressing structure of the diskette as well as current and recent data transfers, the switch position cannot be changed without interchanging the diskettes.

In a dual-diskette system, one diskette can be transferring data while the other unit is performing a seek operation.

The driver is written in kernel fashion. The entire driver may be initially mass storage resident if the program library is on another disk device (e.g., storage module drive).

The driver performs seven major functions:

- Kernel driver
- Data formatting and transfers
- Error recovery and logging
- Diagnostic logical unit support
- CPU main memory bank addressing
- Overlay processing
- Verify write operation

The flexible disk unit (diskette) consists of a single surface with 77 tracks. To perform a data transfer, the head is positioned over the appropriate track and a loading device presses the disk against the head so the transfer can take place.

Two separate data formats may be used as shown in table 3-1.

The diskette initialization must be performed by the flexible disk drive utility processor (FDUTIL). The procedure writes track and sector address tags and the cyclic redundancy check (CRC) bits for the sectors. Only 74 of the 77 tracks may be used for data storage at any one time. During initialization, tracks are checked for data transfer equality. If a track is found to be bad, it is marked bad and the next track is assigned as the alternate track. This assignment is transparent to the user program, so the mass storage addresses delivered to the user program are unaffected by use of the alternate track. A maximum of two alternate tracks can be assigned. At reinitialization, alternate tracks are released (FDUTIL attempts to reinitialize using only tracks 1 through 75.).

FDUTIL is described in appendix I.

TABLE 3-1. DATA FORMATS

Format	Word/ Sector	Sectors/ Track	Tracks/Drive	Number of Drives (Maximum)	Total Words
CDC	96	19	77 } 74 only are available to user programs	2	281K
IBM	64	26	77	2	256K

Sector numbering for word addressing purposes begins at logical sector 0, which is normally physical sector 1 on physical track 1 (physical sector numbering begins at 1, not 0). This is also transparent to the user program, except that the program must not attempt to do a format write operation (sector addressed transfer) to the autoload sectors.

When a diskette containing data is loaded into the system, the bad track sectors read/write and write compare options are obtained from the newly loaded diskette. Because the two diskettes are totally separate logical units, it is possible to have one diskette formatted for IBM compatibility and one formatted to CDC standards. Likewise, a CDC formatted diskette could be followed by an IBM formatted diskette in the same physical slot, etc.

The transfer of data may be made using direct memory access. However, if the optional diagnostic package is a part of the driver, A/Q data transfers are also used. Auto data transfer is not available. The diskette may be used as a word addressable device (READ or WRITE requests) or as a sector addressable device (FREAD or FWRITE requests). Locating a word or sector physical (for diagnostic purposes) is described below.

To increase the speed of A/Q read operations, for more than one continuous sector transfer, diskettes may be interlaced at the time they are initialized. This procedure noticeably speeds up A/Q reading of binary data (about 500 percent) but slows down reading of deadstart read slightly. Interlacing is done only at the time a diskette is intialized. It is transparent for all FDUTIL functions. (For example, an interlaced diskette can be copied to a non-interlaced diskette with no change of data.)

Sectors are interlaced as follows:

Sequential Address	IBM Interlaced	CDC Interlaced
1	1	1
2	14	11
3	2	2
4	15	12
.	.	.
.	.	.
.	.	.
18	22	19
17	10	10
.	.	.
.	.	.
.	.	.
25	13	
26	26	

The diagnostic package also accepts a MOTION command.

Preset error recovery processes are available for data transfers and preliminary seek operations. Error procedures are usually triggered by a cyclic redundancy check (CRC) comparison error. The CRC code is generated for both sector addresses and for data. If an unrecoverable error is encountered, a composite of the errors is generated as a transfer status, and this information is returned to the using program.

READ/WRITE

Unformatted I/O requests are treated as word addressed requests. The request format is substantially the same as described above for storage module drive requests. However, the control point logic used for addressing CPU main memory is not available. Instead (and only if the bank addressing option is present), the driver uses the 18-bit address of the buffer area to interface with the MSOS or RTOS that controls the CPU. The user need not supply any special bank addressing information in the request.

Word addressable transfers start at the next word on disk and continue uninterrupted across sector and track boundaries.

CAUTION

WRITE with the ASCII bit set writes a deleted record. A deleted record may be read or written; such a record appears the same as a normal record except that bit 13 of the v field in word 3 equals 1.

To calculate a disk sector/track physical address from a 31-bit disk word address, divide the word number by 96 (CDC) or 64 (IBM). Next divide the resultant sector number (discarding the word remainder) by 19 (CDC) or 26 (IBM) to obtain the offset track identification (discarding the sector remainder). Any bad track in this track address must be compensated for. Bad tracks are designated in word 27 of the physical device table (appendix C). Add the calculated track address x (x = 1 plus the number of bad tracks below this track) where $1 \leq x \leq 3$. Using this track address, the driver commands a seek operation. When the seek has been completed, the driver handles the seek complete interrupt. All of this is transparent to the user so long as no seek error occurs. If an unrecoverable seek error occurs, the user program is notified of an error, but not of the type of error. (However, if the diskette is a diagnostic logical unit, the type of error is specified to the user.)

FREAD/FWRITE

These read and write commands are oriented to the sector/track formatting scheme and are the same as described above for storage module drive requests except for the control point word. The least significant bit of the mass storage address indicates that the address is to be interpreted as a logical sector address (not a track/sector address). The n parameter determines the number of words to transfer. If n is not a divisor of a sector (96 for CDC or 64 of IBM), the remainder of the sector is transferred anyway on an FWRITE request. However, the remainder of the sector is filled with zeroes. On a FREAD request, only the specified number of words is transferred; this in effect masks out the remainder of the last sector.

CAUTION

An FWRITE with the ASCII bit set results in the flexible disk driver writing address tags (diskette initialization) to a specific track. The least significant bit of the mass storage address contains the logical sector address that references the first sector of a track. The most significant bit contains the number of sectors/track in bits 0 through 7 and the words/sector bits 8 through 15. The I/O request must be preceded immediately by a motion request (with a code of 1). The motion request permits the initialization to occur. If the I/O request is not preceded by the motion request, a deleted record is written.

Notice that other user's I/O request cannot be issued to the same logical unit while a program is using the privileged motion request to change the state of the I/O request.

MOTION

If the diagnostic logical unit option is included, the driver recognizes the MOTION request discussed in section 1.

However, the fourth hexadecimal bit always equals 0. The three motion codes (p1, p2, and p3) are processed in that order. Values for the p codes are as follows:

Code	Meaning
0	Terminate request
1	Prime I/O request for initialization, logically referencing track 0 or seeking a specified track
2	Request no data compare during write
3	Request data compare during write
4	Change diskette mode to READ/WRITE
5	Change diskette mode to read only (option)
6	Not used
7	Not used

The I/O options are summarized in table 3-2.

TABLE 3-2. FLEXIBLE DISK COMMANDS

Definition	I/O[1]	Format[2]	Mode[3]	Count[4]	Prime[5]	Logical Unit[6]
READ	0	0	0	0	0	0
READ	0	0	0	0	X[7]	X
FREAD	0	1	0	0	0	0
FREAD TRACK 1	0	1	0	0	X	X
FREAD TRACK 0	0	1	0	0	1	1
WRITE	1	0	0	0	0	0
WRITE	1	0	0	0	0	X
DEL. REC. WRITE	1	X	1	X	0	X
FWRITE	1	1	0	0	0	0
FWRITE TRACK 1[8]	1	1	0	0	0	X
FWRITE TRACK 0[8]	1	1	0	0	1	1
DEL. REC. FWRITE[9]	1	X	1	X	0	X
INITIALIZATION	1	1	1	0	1	X
SEEK ONLY	0	X	X	0	1	1
STATUS ONLY[10]	0	X	X	0	0	1

NOTES:

1. I/O:	0 = Read operation 1 = Write operation
2. Format:	0 = Unformatted sector read/write 1 = Formatted word addressable read/write
3. Mode:	0 = Binary 1 = ASCII
4. Count:	I/O word count
5. Prime:	0 = Not proceeded with a motion code of 1 1 = Proceeded with a motion code of 1
6. Logical unit:	0 = Not a diagnostic logical unit 1 = Diagnostic logical unit
7. X:	Not applicable
8. Track 1: Track 0:	Logical sector addressing starts with track 1 Logical sector addressing starts with track 0
9. DEL REC:	Deleted record
10. Status:	Located in physical device table after request

ERROR RECOVERY

The user program has no control over the error recovery sequence.

For read errors the error recovery sequence is as follows:

1. The driver tries to read at the given address 10 times.
2. The driver seeks a track two tracks away, then reseeks the specified track and attempts to write. This is done five times.
3. The driver seeks logical sector 0, then reseeks the specified track and attempts to write. This is done five times.
4. The driver develops the error status word and places it in the physical device table. (See appendix C.)

For write errors the error recovery sequence is as follows:

1. The driver tries to write to the specified address two times.
2. The driver seeks logical sector 0, then reseeks specified track and attempts to write. This is done twice.
3. The driver develops the error status word and places it in the physical device table.

The failure, but not the failure type, is reported to the normal user. If the diskette is being used as a diagnostic unit, the nature of the error is saved in the diagnostic expansion of the physical device table. The error codes appear as a bit assignment. The last equipment status appears in ESTAT2 (word 12) of the physical device table and has the following meaning:

Code	Meaning
0	Unit ready
1	Unit busy
2	Head loaded
3	Seeking
4	Reading/writing
5	Interrupt
6	Interrupt selected
7	Direct memory access parity error
8	Direct memory access protect fault
9	Direct memory access memory address fault
10	Lost data
11	Seek error
12	Data cyclic redundancy check error
13	Deleted record
14	Protect switch on
15	Controller busy

This is not actual hardware status but a composite status formed by a driver module based on the information supplied by the hardware status.

The hardware status consists of the controller and unit status.

Not all errors are reported in the engineering log for this logical unit. The physical device table contains a threshold value for error retries. When the number of recovery attempts of a given type reaches that value, the error count is cleared and an entry is made in the engineering log.

The error messages (numbers appearing on the comment device) are listed below:

Code	Meaning
0	Time out
1	Lost data
1	Bad initiator pseudo status
2	Bad continuator pseudo status
3	Bad timeout pseudo status
3	Parity error
4	Status fault(s) after I/O
5	Internal reject fault code
6	External reject fault code
13	Read only diskette
14	Unit not ready
16	Track/sector fault
18	Invalid sector address
19	Protect error
20	Data compare error
48	Controller address error
61	No interrupt
65	No interrupt selected
66	Memory address error
68	Unexpected interrupt
69	Initialization not enabled

76	Fault indicator for logging recovered errors
77	Expected reject did not occur
78	Short/long transfer error
79	Unit busy
80	Unit seeking
81	Unit doing I/O

SPECIAL FEATURES

Overlay requests cause the driver to check the location of the requester's parameter list (e.g., READ request). If it is overwritten by the projected overlay, sufficient information is saved from the request in the physical device table to allow the request to be completed.

NOTE

Not enough information is saved to repeat the request using the alternate device handler.

Unprotected requests are saved at the beginning of scratch mass memory in logical sector 1.

The optional compare data logic compares data just written to the diskette with the data in the user's write buffer. The user must supply his own buffer for this function.

LINE PRINTER DRIVERS

WRITE/FWRITE

WRITE and FWRITE requests are honored by the line printer drivers. Binary/ASCII mode has no significance and is ignored.

The drivers print up to 136 characters per line. The requestor output buffer may be any length, provided it contains embedded control characters. If more than 136 characters are supplied for one line, the additional characters are ignored.

These drivers can use either FORTRAN or non-FORTRAN mode. The printer's physical device table contains the logical unit number of the FORTRAN line printer. If the logical unit number specified in the request is the same, the FORTRAN mode of operation is used. All other logical unit numbers are handled in the non-FORTRAN mode.

The FORTRAN mode interprets the first character of the record as carriage control information only for FWRITE. Carriage control characters are the following.

Character	Action Before Printing
0	Space two lines
1	Page eject
+	No space
all others	Space one line

The non-FORTRAN mode upspaces one line before printing, and the first character of the record is printed for FWRITE.

In both the FORTRAN and non-FORTRAN modes, the unformatted WRITE does not cause a preceding upspace. It prints the buffer only when a control character that causes a print or paper motion is encountered.

The 1742-120 Line Printer and Controller requires that a train image table be appended to the driver (T5954).

MOTION

A MOTION request to write end-of-file is honored as a page eject function. A REWIND/UNLOAD resets the line count. All other MOTION requests cause no action with normal completion of the request.

CHARACTER EDITING

All characters are edited as follows before they are sent to the print buffer.

Character	Action
20_{16} – $5F_{16}$	Send to buffer.
60_{16} – $7E_{16}$	Change to 40_{16} – $5E_{16}$.
03_{16} – EOT	Print buffer, upspace one line, and terminate request.
04_{16} – EOT	Same as 03_{16}
09_{16} – HTAB	Simulated TAB, send blanks to buffer.
$0A_{16}$ – Line feed	Ignore.
$0B_{16}$ – VTAB	Print, select format tape level two, and continue.
$0C_{16}$ – Form feed	Select format tape level one, top of form.
$0D_{16}$ – Carriage return	Print buffer and upspace one line.
$1B_{16}$ – Escape	Used for direct function control of the line printer.

Character	Action	
$1B_{16}$ – Escape (continued)	The next character is interpreted as follows:	
	00_{16} – $2F_{16}$	Ignore.
	30_{16}	Printer buffer, no upspace, next line starts at the beginning.
	31_{16}	Printer buffer, single space, next line starts at the beginning.
	32_{16}	Print buffer, double space, next line starts at the beginning.
	33_{16} – $3E_{16}$	Print buffer, select format tape level (01 through 12), and continue printing from the next printing position.
	$3F_{16}$	Select eight lines per inch.
	40_{16}	Clear controller and continue.
	41_{16} – $7F_{16}$	Ignore.

Tab stops for tab simulation are assumed to exist every n characters of the print line. Each time a tab character is encountered, sufficient space characters are sent to the print buffer to advance the character counter to the next tab-stop position. In the released version n is 20.

ERROR CONDITIONS

The following errors are detected by the driver:

Internal or external reject

Hang-up

Alarm

When the driver detects an irrecoverable failure, it sets the error field in bits 15 through 13 of word 9 of the physical device table for the device, sets the error word in the Q register, and transfers control to the alternate device handler. Refer to the MSOS Diagnostic Handbook for I/O error codes and descriptions.

1827 LINE PRINTER DRIVER

This driver processes WRITE, FWRITE, and MOTION requests for the 1827 Line Printer. The driver is written in kernel format and is initially mass storage resident. There are no options in the driver; all modules must be included. More than one line printer can be handled by the driver. For each logical unit associated with the physical device, there must be a separate physical table (see appendix C). The logical units commonly assigned to the line printer are normal logical units, diagnostic logical units, and FORTRAN logical units. The driver provides FORTRAN format conversion to support the last named logical unit.

The driver communicates with the line printer exclusively on the A/Q channel.

WRITE/FWRITE/MOTION

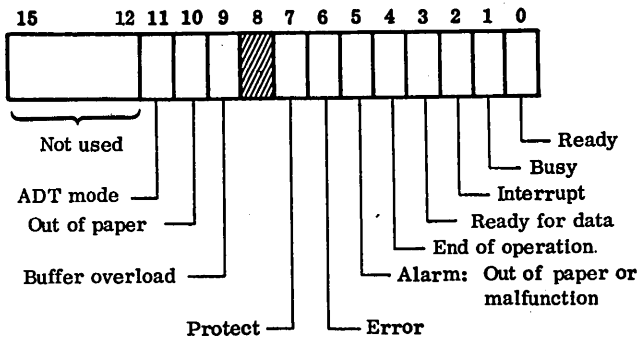
The characteristics of these requests are discussed above.

CHARACTER EDITING

The character editing capability is discussed above.

STATUS AND ERROR HANDLING

The printer controller status bits are as follows:



If an unrecoverable error occurs, it is reported by the alternate device handler (if any). The error fault value is returned in the A-register as well as being stored in word 9 of the physical device table. Values are as follows:

Code	Meaning
0	Timeout
2	Alarm
3	Parity
5	Internal reject
6	External reject

1827-7 LINE PRINTER DRIVER

This driver processes WRITE, FWRITE, and MOTION requests for the 1827-7 Line Printer. The driver is mass storage resident. There are no options in the driver; all modules must be included. More than one line printer can be handled by the driver.

The driver communicates with the line printer through the 1843-2 CLA. The line printer driver works as a pseudo driver since it reformats data intended for the printer, then makes a non-standard MSOS 5 request to the 1843-2 CLA driver indicating the CLA channel attached to the printer.

WRITE/FWRITE/MOTION

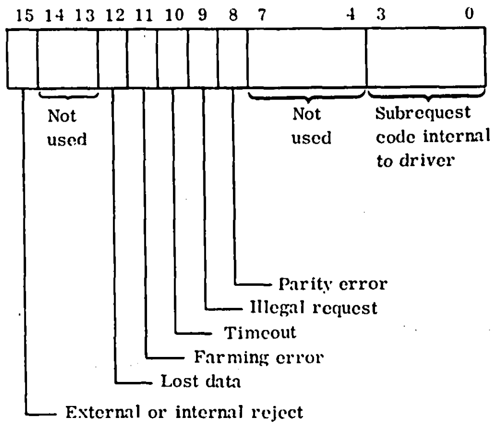
The characteristics of these requests are discussed above, except that the FORTRAN logical unit number is contained in word 24 of the printer's physical device table rather than in word 19.

CHARACTER EDITING

Characters are edited as discussed above except that the escape character ($1B_{16}$) when followed by 33_{16} – $7F_{16}$ is ignored.

STATUS AND ERROR HANDLING

Printer controller status is not available. The status returned in the physical device table is the formatted status from the 1843-2 CLA driver. Bit assignment is:



If an error occurs, it is reported by the alternate device handler, and logged in the engineering file. Fault code values are as follows:

Code	Meaning
0	Timeout
2	Alarm/status
3	Parity

MAGNETIC TAPE DRIVERS

Magnetic tape drivers process the standard READ/WRITE/FREAD/FWRITE/MOTION requests. A data transfer can be accomplished with one of the three types of transfer modes: buffered, unbuffered, and auto-data transfer (ADT). During an unbuffered transfer, interrupts are inhibited to prevent lost data. Interrupts are enabled during the other types of transfer.

DATA FORMATTING

Magnetic tape drivers can transfer data to seven- or nine-track tape drives. Data on nine-track devices requires no special handling. All data is written in odd parity and no code conversion is required. One computer word comprises two data frames on tape.

Data transfers on seven-track drives require special processing. There are only six data bits in each frame; therefore, internal ASCII data is converted to external BCD, a six-bit code, when data is to be written to tape. The reverse process is used on read transfer – BCD to ASCII conversion.

Binary transfers on seven-track drives also require additional reformatting. The hardware assembly/disassembly utilizes only the six least significant bits of each half word. Therefore, prior to an output transfer, data is reformatted to conform to this requirement. On input, a reverse procedure is performed. The reformatting of the data is performed in a buffer area different from the user buffer. This repacking buffer must also be bigger than the user's buffer. Therefore, a maximum buffer size limitation must be imposed on the user buffer length. The release software provides a buffer that allows the user to specify a maximum record size of 192 words.

FORMATTED REQUESTS

The FREAD and FWRITE requests are handled as single record requests; one request for one physical record. The maximum record size limitation for seven-track drives is enforced. Any length in excess of this maximum is truncated. No length restriction applies to nine-track drives. When the record is shorter than the requested length in reading operations, the short read indicator is set and the address of the last location containing data is placed in the last word of the user's buffer.

UNFORMATTED REQUESTS

READ and WRITE requests are handled as logical record requests. A logical record length is that number of words defined in the user's parameter list. However, if that size exceeds the maximum size allowable on seven-track drives, the driver breaks up the logical record into several physical tape records. Therefore, when writing on nine-track drives, one logical record equals one physical record. On seven-track drives, one logical record can equal several physical records.

For READ requests on either seven- or nine-track, one logical record may be one or more physical records. The driver continues reading records until the user's buffer is filled or until a file mark is encountered.

MOTION REQUESTS

Tape drivers perform all of the normal motion commands. Special handling is performed on a rewind request. When a tape is rewinding, the subsystem is capable of performing data transfers on other drives. Therefore, the interrupt is not selected on the rewind operation. Instead, a status checking routine is scheduled at priority 3. When a load point condition is detected, the driver's initiator entry is scheduled.

ERROR CONDITIONS

The following error conditions are recognized by the driver:

- No write ring on a write or end-of-file mark
- Tape transport not ready
- Tape transport number not dialed
- Parity error
- Failure to interrupt (required TIMER package)
- Buffer channel not operative
- Lost data switch mode
- Missing processing module

These are considered irrecoverable and are reported to the alternate device handler. The user may continue, repeat the request, or down the driver.

LOW-COST TAPE TRANSPORT†

The low-cost tape transport driver processes READ, WRITE, FREAD, FWRITE and MOTION commands for the LCTT. Several tape units may be controlled at the same time; however, only one request is executed at a time. A single copy of the LCTT driver can control several LCTT units even though the units may have differing characteristics.

Since the LCTT driver is written in kernel format, unneeded options may be deleted from the system to conserve space. If an option is deleted, the deleted program must be replaced by a dummy program that returns control to the caller.

If only nine-track units are present, modules TK7 and TK7DAT can be omitted. Seven-track capability requires module TK7. ASCII conversion and binary reformatting for seven-track tapes requires module TK7DAT also.

† This is to be distinguished from the Low Cost Tape Transport (LCTT)/FORMATTER, 1860-5/6, which is described later.

If recovery capability is desired, the module RECVRY must be included. If raw hardware status rather than composed status is desired, the module FORMIT may omitted.

There is no logical diagnostic unit for the LCTT; instead the alternate device handler has the ability to report failed device errors.

The LCTT driver operates in the buffered data mode; data transfers using ADT and A/Q are not supported. Tape recording density is 800 fpi. Neither reverse read nor nonstop read capability are present.

All transfers are timed by a diagnostic clock. Failure of the transfer to complete in the allotted time triggers the timeout error (and recovery) logic.

DATA AND RECORD FORMAT

Data Format

Track Data Transfers - The basic transfer (9-track) does not need reformatting. All data transfers on a 9-track LCTT are treated as binary and are recorded in odd parity at 800 frames per inch (fpi). Two 8-bit tape characters (frames) compose a single CPU word. ASCII is transferred without conversion. The data in core corresponds to the data on tape and is packed as shown:

15 8	7 0
Frame 1	Frame 2
Frame 3	Frame 4
Frame 5	Frame 6

Seven-Track Data Transfers - This option requires the TK7 module. If there is no module TK7DAT in the system, data is transferred to and from core without conversion. Since the hardware operates in assembly/disassembly mode only, the data appearance in core (both ASCII and binary) is as shown:

15	14	13 8	7	6	5 0
0	0	Frame 1	0	0	Frame 2
0	0	Frame 3	0	0	Frame 4
0	0	Frame 5	0	0	Frame 6

Bits 6, 7, 14, and 15 must be zero whether reading into core or writing from core. If any of these bits are set during a write-to-tape operation, a hardware program error occurs. This error appears in the V field asa short record error. For raw data transfers, the logical record length is defined by the 15-bit parameter n of the request.

If the module TK7DAT is included, 7-track converted data transfers are available. Binary data is packed or unpacked in core; ASCII data is converted to or from external binary coded decimal (BCD).

Seven-track binary mode assumes core data to have the following format:

15   13 12 11 10 9 8 7 6 5 4 3 2   0

Frame 1	Frame 2	Frame 3	
Frame 3	Frame 4	Frame 5	Frame 6
Frame 6	Frame 7	Frame 8	

Seven-track ASCII mode causes the ASCII data in core to be converted to or from external BCD. The ASCII characters in core correspond to the BCD tape from as shown:

15 8	7 0
Frame 1	Frame 2
Frame 3	Frame 4
Frame 5	Frame 6

Record Constraints

READ/WRITE records - Record lengths for 9- and 7-track LCTTs are defined by the requestor; the number of words specified in a READ/WRITE request defines a logical record.

For 9-track write requests, a logical record is written as a physical record. For read requests, the user defines the logical record as any length that is unrelated to the length of the physical record. The driver reads through record gaps until the logical record is complete or until it encounters a file mark.

For 7-track requests, the maximum length of a physical record is set to the physical record size parameter (PHSREC) to limit core use. (PHSREC is 192 words for this discussion.) If a logical record is longer than PHSREC, it is segmented and written as a series of physical records. An analogous procedure is used for reading a logical record. For raw data transfers, the record size is limited only by the 15-bit parameter n of the request. Any unused portion of the last physical record is lost on subsequent read operations.

FREAD/FWRITE records - The formatted 7-track requests are physical record oriented, with all records defined to be the same PHSREC length (usually 192 words). Any record longer than that is truncated.

The formatted 9-track requests are not similarly restricted. Instead, the user defines the logical record length. For FREAD operation, the driver reads through record gaps until the logical record is complete, or until a file mark is read.

READ/WRITE/FREAD/FWRITE

The calling sequence and parameter lists for these requests are described in section 1. At execution time, the external indications are also the same as those described in that section. However, SYSDAT must contain a buffer for 7-track data conversion (if TK7DAT is included). The size of this buffer (which is defined at assembly time) is (PHSREC x 4/3) + 2. Note that the buffer must be large enough to accommodate the maximum record size as defined by PHSREC.

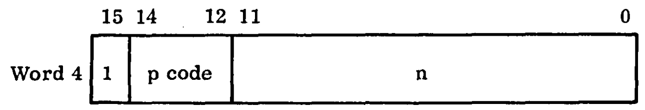
The use of buffers for the four acceptable transfer requests is shown in table 3-3.

MOTION

The calling sequence and parameter list for the MOTION command are as shown in section 1, except that the dy parameter in word 4 always is zero. The p1, p2, and p3 codes are executed from left to right. The codes are:

Code	Meaning
0	First zero terminates request
1	Backspace record
2	Write file mark
3	Rewind to load point
4	Rewind and unload
5	Advance one file
6	Backspace and erase file
7	Advance record

If a motion is to be repeated, word 4 of the MOTION request is:



where n is the number of repeats. n must be 4095 or less.

ERROR RECOVERY

If the error recovery option (RECVRY) is included, the driver attempts to correct errors using the standard CDC tape recovery techniques. These include write and read operations.

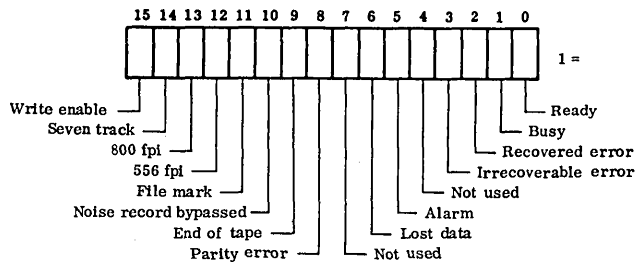
Write operations are as follows:

1. Back over the area just used, erasing and rewriting.
2. Write system noise blocks to pass over a bad area in tape.
3. Verify the rewritten information.
4. Indicate an unrecoverable error because tape has run out, to a good block (record) on tape has not been found, allowable number of consecutive erasures has been exceeded, and erase error has occurred, or the block just written has not been identified.

Read operations are as follows:

1. Attempt to reread the data several times using normal, high, and low signal clipping levels.
2. Reset parity mode.
3. Indicate an unrecoverable error because there are parity errors or the number of allowable retries has been exhausted.

If the composite status option (FORMIT) is included, word 12 of the physical device table for an LCTT (see appendix C) indicates the status bits shown below:



This is not the actual hardware status but is a composed status formed by the driver module called FORMIT based on the information supplied by the hardware status and word 24 (unit/mode) in the physical device table.

There is no diagnostic logical unit capability in the LCTT driver. Instead, the alternate device handler sends error messages concerning the failure to the comment device. Values for the alternate device fault code are as follows:

Code	Meaning
0	Logical unit timed out
1	Lost data
2	Alarm
3	Parity
13	No write ring on write request
14	Not ready
15	Noise record was bypassed
31	Short record write requested
41	Incomplete error (unrecoverable)

TABLE 3-3. USE OF BUFFERS FOR READ, FREAD, WRITE, FWRITE REQUESTS

Request	Mode	7-Track with S18326 (LCTT/FORMATTER) 7-Track with TK7DAT (LCTT)	9-Track and 7-Track without TK7DAT (LCTT) or without S18326 (LCTT/FORMATTER)
READ or FREAD	Binary	LCTT→ PHYSTB buffer → packed 4 to 3 into requestor's buffer	LCTT → requestor's buffer
	ASCII	LCTT → PHYSTB buffer; convert external BCD to ASCII in place → requestor's buffer	LCTT → requestor's buffer
WRITE or FWRITE	Binary	Requestor's buffer → unpacked 3 to 4 into PHYSTB buffer → LCTT	Requestor's buffer → LCTT
	ASCII	Requestor's buffer → PHYSTB buffer; convert ASCII to external BCD in place → LCTT	Requestor's buffer → LCTT

LOW COST TAPE TRANSPORT/ FORMATTER DRIVER

The LCTT/FORMATTER Driver processes READ, WRITE, FREAD, FWRITE, and MOTION commands for the LCTT/FORMATTER. Several tape units may be controlled at the same time; however, only one request is executed at a time. A single copy of the LCTT/FORMATTER driver can control several LCTT units even though the units may have differing characteristics.

Since the LCTT/FORMATTER Driver is written in kernel format, unneeded options may be deleted from the system to conserve space. If an option is deleted, the deleted program must be replaced by a dummy program which returns control to the caller.

If only nine-track units are present, the S18326 module can be omitted. Seven-track capability requires the S18326 module.

If recovery capability is desired, the module R18326 must satisfy the appropriate conditional assembly.

The LCTT/FORMATTER driver uses the DSA channel for data transfer and the A/Q channel for its other functions. It supports nonstop read and nonstop write. Densities allowed are 800 or 1600 frames per inch (fpi) for nine-track units, and 556 or 800 fpi for seven-track units. All transfers are timed by a diagnostic clock. Failure of the transfer to complete in the allotted time triggers the timeout error/recovery logic.

DATA FORMAT

Nine-Track Data Transfers – The basic transfer (nine-track) does not need reformatting. All data transfers on a nine-track LCTT/FORMATTER unit are treated as binary and are recorded in odd parity at 800/1600 fpi. Two 8-bit tape characters (frames) compose a single CPU word. ASCII is transferred without conversion. The data in core corresponds to the data on tape and is packed as shown:

15 8	7 0
Frame 1	Frame 2
Frame 3	Frame 4
Frame 5	Frame 6

Seven-Track Data Transfers – This option requires the S18326 module. If the module is not in the system, data is transferred to and from core without conversion. Since the hardware operates in assembly/disassembly mode only, the data appearance in core (both ASCII and binary) is:

15	14	13 8	7	6	5 0
0	0	Frame 1	0	0	Frame 2
0	0	Frame 3	0	0	Frame 4
0	0	Frame 5	0	0	Frame 6

Bits 6, 7, 14, and 15 must be zero whether reading into core or writing from core. If any of these bits are set during a write to tape operation, a hardware program error occurs. This error appears in the v field as a short record error. For raw data transfers, the logical record length is defined by the 15-bit parameter n of the request.

If the module S18326 is included, seven-track converted data transfers are available. Binary data is packed or unpacked in core; ASCII data is converted to or from external BCD.

Seven-track binary mode assumes core data to have the following format:

15 13 12 11 10 9 8 7 6 5 4 3 2 0

Frame 1	Frame 2	Frame 3	
Frame 3	Frame 4	Frame 5	Frame 6
Frame 6	Frame 7	Frame 8	

Seven-track ASCII mode causes the ASCII data in core to be converted to or from external BCD. The ASCII characters in core correspond to the BCD tape frame as shown:

15 8	7 0
Frame 1	Frame 2
Frame 3	Frame 4
Frame 5	Frame 6

Record Constraints

READ/WRITE Records: Record lengths for nine- and seven-track LCTT/FORMATTER units are defined by the requestor: the number of words specified in a READ/WRITE request defines a logical record.

For nine-track write requests, a logical record is written as a physical record. For READ requests, the user defines the logical record as any length that is unrelated to the length of the physical record. The driver reads through record gaps until the logical record is complete or until it encounters a file mark.

For seven-track requests, the maximum length of a physical record is set to PHSREC to limit core use. (PHSREC, the physical record size parameter, is 192 words for this discussion.)

If a logical record is longer than PHSREC, it is segmented and written as a series of physical records. An analogous procedure is used for reading a logical record. For raw data transfers, the record size is limited only by the 15-bit parameter n of the request. Any unused portion of the last physical record is lost on subsequent read operations.

FREAD/FWRITE Records: The formatted seven-track requests are physical record oriented, with all records defined to be the same PHSREC length (usually 192 words). Any record longer than that is truncated.

The formatted nine-track requests are not similarly restricted. Instead, the user defines the logical record length. For FREAD operation, the driver reads through record gaps until the logical record is complete, or until a file mark is read.

READ/WRITE/FREAD/FWRITE

The calling sequence and parameter lists for these requests are described in section 1. At execution time, the external indications are also the same as those described in that section. However, SYSDAT must contain a buffer for seven-track data conversion if S18326 is included. Size of this buffer, which is defined at assembly time, is (PHSREC x 4/3) + 2. Note that the buffer must be large enough to accommodate the maximum record size as defined by PHSREC.

The use of buffers for the four acceptable transfer requests is shown above in table 3-3.

MOTION

The calling sequence and parameter list for the MOTION command are as shown in section 1, except the dy parameter in word 4 is always zero. The p1, p2, and p3 codes are executed from left to right. The codes are:

Value	Operation
0	First 0 terminates request
1	Backspace record
2	Write file mark
3	Rewind to load point
4	Rewind and unload
5	Advance one file
6	Backspace and erase file
7	Advance record

If a motion is to be repeated, word 4 of the MOTION request is:

	15	14 12	11 0
Word 4	1	p code	n = number of repeats

The number n must be 4095 or less.

ERROR RECOVERY

If the error recovery option R18326 is included, the driver attempts to correct errors using the standard CDC tape recovery techniques. These include:

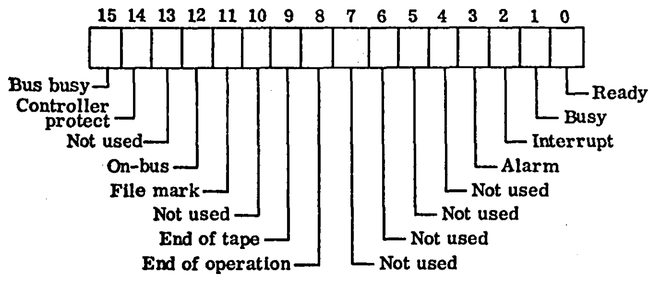
Write Operations:

1. Back over the area just used, erasing and rewriting,
2. Indicate an unrecoverable error due to running out of tape, or being unable to find a good block (record) on tape, or exceeding the allowable number of consecutive erasures, or an erase error occurring.

Read operations:

1. Attempt to reread the data several times using normal, high, and low signal clipping level levels.
2. Reset parity mode.
3. Indicate an unrecoverable error due to parity errors or to exhausting the number of allowable retries.

The status bit assignment is:



Meanings of the fault codes are:

Fault Code	Failure
0	Lu times out
1	Lost data
2	Alarm
3	Parity
5	Internal reject
6	External reject
13	No write ring on write request
14	Not ready
31	Short record write requested
32	Tape defect
36	Transmission parity error
50	No ID burst
51	Illegal density, or an attempt to change density when tape not at load point
60	Illegal motion code
84	Bus relinquished
86	Switch mode error
87	No character read in 25 feet (7.6 meters)

CASSETTE DRIVER

The cassette driver processes READ, WRITE, FREAD, FWRITE, and MOTION requests for the cassette units. The driver passes commands and receives status from the cassette controller which may in turn control either one or two cassette units. Control of two units is serial; the nonactive unit is idle.

The cassette driver is written in kernel format so that unneeded options may be deleted from the system. If an optional module is deleted, it must be replaced with a dummy program whose sole function is to return control to the caller. The cassette driver may be CPU resident or it may reside on mass storage.

The optional features are:

- Error recovery
- Forming composite status for the physical table

Data transfers are accomplished in the ADT mode. All transfers are timed by a diagnostic clock. Failure of the operation to complete in the allotted time triggers the timeout error (and recovery) logic.

DATA AND RECORD FORMATS

All writing on the cassette itself uses bit serial mode in Manchester code format. This code is generated by the controller and is the same bit length as the input or output code (e.g., 8-bit ASCII). A cyclic redundancy check (CRC) code is also generated for each record and constitutes the end of that record. Data is recorded at a single density: 100 8-bit characters per inch. Two serial tracks are available for data transfers.

The physical and logical records for WRITE and FWRITE requests are of equal size. For ADT mode, this is a minimum of two CPU words (four bytes). The maximum is limited by the size of the user program's write buffer. If the cassette is a diagnostic logical unit, a single transfer (one CPU word of two bytes) is permitted.

The user specifies the logical length for READ and FREAD buffers. Three cases are encountered:

- The logical record is greater than the physical record: The driver reads through interrecord gaps on READ commands until it reads the requested number of words or until a file mark is encountered. For an FREAD command, a short read error is reported in the v field.
- The logical record equals the physical record: The CPU read buffer is filled.
- The logical record is less than the physical record: The CPU buffer is filled; then the tape continues to read to end of physical record without transferring data. If the composite status option is present, the overflow error bit is set in the status word. This is not considered to be a system error.

READ/WRITE/FREAD/FWRITE

The calling sequence and parameter lists for these requests are described in section 1.

MOTION

The calling sequence and parameter list for this request are described in section 1 except that the m field in word 0 is always zero. The motion codes are processed from left to right. Code values are as follows:

Code	Meaning
0	The first zero terminates request.
1	Backspace one record

Code	Meaning
2	Write file mark
3	Rewind to load point (unless diagnostic logical unit) Erase (diagnostic logical unit)
4	Rewind
5	Search tape mark forward
6	Search tape mark reverse
7	Advance one record

If a MOTION request is to be repeated, word 4 of the parameter list is:

	15	14 12	11 0
Word 4	1	p code	n

where n is the number of repeats. n must be 4095 or less.

ERROR RECOVERY

If the error recovery option (RECCAS) is included, the driver attempts to correct errors using the standard CDC tape recovery techniques. These include write and read operations:

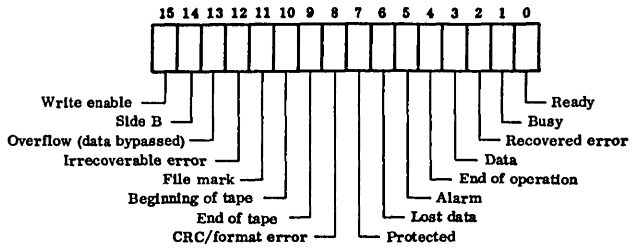
Write operations are as follows:

1. Back over the area just used, erasing and rewriting
2. Write system noise blocks to pass over a bad area in tape
3. Verify the rewritten information
4. Indicate an unrecoverable error because tape has run out, a good block (record) on tape has not been found, the allowable number of consecutive erasures has been exceeded, and erase error has occurred, or the block just written cannot be identified.

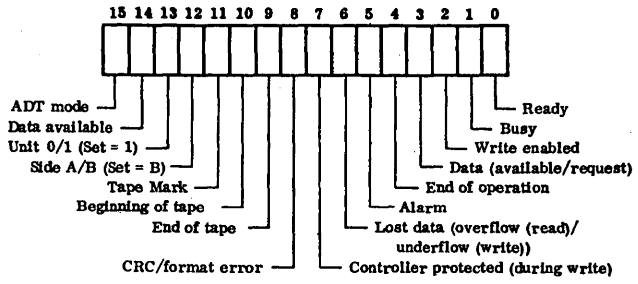
Read operations are as follows:

1. Attempt to reread the data several times.
2. Indicate an unrecoverable error because there are parity errors or the number of allowable retries is exhausted.

If the composite status option (FS2CAS) is included, word 12 of the physical device table for a cassette (appendix C) indicates the status bits shown below.



If the composite option is not included, word 12 of the physical device table reflects actual hardware status. The bit assignment is as follows:



If the cassette is not designated as a diagnostic unit, error messages are sent to the comment device. The error code values are listed below:

Code	Meaning
0	Timeout error
1	Lost data
2	Alarm due to runaway
3	Parity error
5	Internal reject
13	Write not enabled
14	Not ready
21	End of tape (unrecoverable error; tape automatically rewinds on next back motion command)
31	Short record
41	Incomplete request
46	External reject (on output)
47	External reject (on input)
53	End-of-operation not set after interrupt

COMMUNICATIONS DRIVERS

COMMUNICATIONS MULTIPLEXER DRIVER

The 364-4 Communications Multiplexer driver (D3644) is designed to operate up to 32 low-speed, nonsynchronous communications adapter channels at rates of up to 30 characters per second. These channels may be connected to low-speed ASCII terminals, such as teletypewriters or conversational display terminals. There may be up to 32 361-1 private line communications adapters, up to 16 361-4 dial-up communications adapters, or any combination of either of these connected to the multiplexer. Refer to appendix F for a description of the proper setting of the hardware options.

Structure

The 364-4 Communications Multiplexer generates an interrupt from a free-running clock that causes entry to the driver on each cycle. The driver examines the status of each adapter and inputs or outputs data as required. All communications adapter physical device tables are linked together via a thread word, so that all adapters are checked at every clock cycle. The clock interrupt must be adjusted to allow the highest speed terminal that is connected to the multiplexer (normally 30 characters per second) to be serviced.

Communications Driver Requests

MOTION

All MOTION requests are honored and result in no action. These requests are completed without error.

READ/FREAD

READ and FREAD requests are treated similarly to MOTION requests since the driver is always in a read condition via the clock interrupt. These requests differ in that READ and FREAD start the input timeout on the diagnostic timer. Data is read into the 40-word buffer associated with each communications adapter until a carriage return is detected. When this occurs, the entry point of a core-resident input processor, specified by word 21 of the physical device table, is scheduled if it is nonzero.

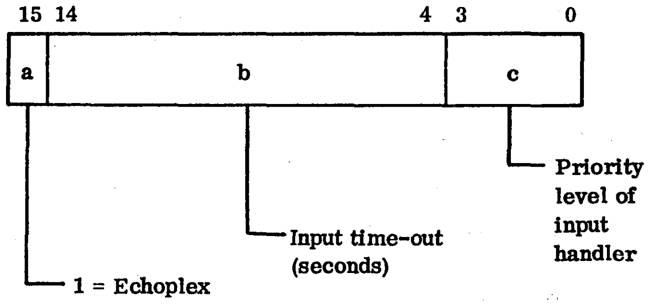
The entry point is scheduled at a priority level specified by word 20 of the physical device table with the Q register containing the logical unit from which the data was read. To aid in the processing of the data, the location immediately following the input buffer contains the word count of the input. In the standard release system, word 21 is 0 and the priority level is 4. The input processor program must contain a table of input buffer addresses, arranged in logical unit order, to allow data to be obtained from the buffer. In the standard release system, the first 16 communications adapter logical units are CABF00 through CABF15. If more than 80 characters are entered before a carriage return, the extra characters are slewed. If a rub-out character is sensed anywhere in the data, the input is slewed until a carriage return is executed, and the communications adapter indices are reset so that a new input can be accepted.

In addition to the carriage return, it is possible to specify two special input termination characters that are contained in word 17 of the physical device table. If either or both are zero, the character is ignored.

Input characters are always translated to uppercase if lowercase characters are entered.

It is possible to specify an echoplex mode where the driver automatically outputs all input characters as they are received. Lowercase characters are always echoed as uppercase. This option is selected by setting bit 15 of word 20 of the physical device table to a 1.

It is possible to specify an input time-out period so that a time-out error condition is indicated if two successive inputs are not received within the defined period. A period of zero disables this feature. The time-out period is contained in bits 4 to 14 of word 20 of the physical device table. The following is the arrangement of word 20 of the physical device table.



NOTE

In the standard release system, a = 1, b = 60, and c = 4.

WRITE/FWRITE

Both types of output requests are honored. Mode has no meaning and is assumed to be ASCII. Each request specifies the core location from which it is being written, the number of words, and the completion address. Output is always with even parity generated by the driver, and the communications adapter must be strapped to accommodate this. If the communications adapter is a 361-4 Full or Half Duplex Nonsynchronous Adapter and a disconnect character (14_{16}) is encountered anywhere in the message, the terminal is disconnected from the multiplexer, and the request is completed without error.

WRITE

The number of words specified is transmitted, beginning at the starting address. Each word causes two characters to be transmitted; the upper half is sent first. If zero words are specified, only one character is sent from the upper half of the core location specified as the starting address.

FWRITE

This request is processed in the same manner as the WRITE request, except that two control characters are sent before the requested message is output. These characters are located in word 16 of the physical device table and, in the standard release system, are specified as line feed and carriage return.

Error Conditions

The driver recognizes internal or external reject, lost data, and line break errors and notifies the operator. These errors are fatal and are considered irrecoverable. Refer to the MSOS diagnostic handbook for I/O error codes and descriptions.

All error conditions result in the standard error return to the requester. In addition to the error conditions defined above, two nonfatal errors are detected by the driver. If a disconnect (as opposed to a line break) is detected during an input or output operation on a 361-4 Communications Adapter, the request is completed with the error, and bit 15 of the status word in the physical device table is set to a 1. This error is not reported on the system comment device since it may occur frequently and is nonfatal.

A time-out error may occur on input, as previously described. In this event, the input processor entry point is scheduled with the Q register containing the logical unit and bit 15 set to a 1.

If a time-out occurs during output, a lost interrupt is indicated. The standard error return is made to the requester and processing may continue.

ASYNCHRONOUS COMMUNICATIONS CONTROLLER AND SERIAL I/O DRIVERS

These drivers provide a flexible communications interface between the 1700 Series computer and serial data, RS232-C compatible devices. Each 1743-2 Asynchronous Communications Controller provides eight channels, and each 1595 Serial I/O board provides one channel. The release software allows up to 16 devices in a system configuration. The user may add additional devices by specifying the required physical device tables. The tables are threaded together through word 75 of the table. These drivers are capable of processing unsolicited input from a timeshare device. This mode of operation is defined by setting bit 15 of word 29 in the physical device table.

READ/FREAD Requests

The drivers process these requests in a manner identical to that of the I/O-TTY driver. The drivers can operate in an echoplex mode. Input data is retransmitted to the sending device. This mode is defined by setting bit 15 of word 17 of the physical device table. If an error is detected on input, a rubout is transmitted to the device and the bad data is not placed in the user buffer. On a timeshare device, the requests are completed immediately after the diagnostic clock value is updated, unless the logical unit is that of the diagnostic unit. In this case the request is processed normally.

WRITE/FWRITE Requests

The drivers process these requests in the same manner as the I/O-TTY driver.

UNSOLICITED INPUT—TIMESHARE DEVICES

For timeshare devices, the physical device table contains a buffer of 40 words. These devices are always in an input mode except when a WRITE, FWRITE, or MOTION request is in progress. When data is received, it is placed in the physical device table buffer. The end of input is signified by a carriage return or one of the two end-of-text characters defined in word 18 of the physical device table. For timeshare, these characters are defined to be an exclamation point (!) and the escape character. The 1595 Serial I/O board also can detect an end-of-text character that is jumperable on the card. At the termination of input, the user address specified in word 31 of the physical device table is scheduled at the priority level defined in bits 3 to 0 of word 29 of the physical device table. The Q register contains the logical unit number. In the event of a time-out error, Q is negative.

MOTION Requests

Words 22 to 28 of the physical device table define the motion requests for each device. MOTION requests resulting in no action are completed without error. The following are some examples of MOTION requests:

<u>713 Conversational Display Terminal</u>

Backspace record	Backspace cursor one character
Write end of file	Reset cursor to home position
Rewind	No action

<u>713 Conversational Display Terminal</u>

Rewind/unload	Reset cursor to beginning of line
Advance file	Move cursor down one line
Backspace file	Move cursor up one line
Advance record	Advance cursor one record

1711 Teletypewriter

Backspace file	No action
Write end of file	Top of form
Rewind	No action
Rewind/unload	Terminate request
Advance file	No action
Backspace file	No action
Advance record	No action

Error Conditions

The drivers detect the following errors:

00	Timeout (fatal)
01	Lost data
03	Parity error
05	Internal reject (fatal)
06	External reject (fatal)
33	Line break

All errors are logged in the engineering file. Fatal errors result in the standard error return to the requester with the Q register bit 15 set. The other errors result in a rubout being transmitted to the device; on onput requests, the operator may re-enter the desired character. Diagnostic logical unit capability is provided.

1843-2 COMMUNICATION LINE ADAPTER

The CLA as it is used in ITOS 1 terminal communication is described above.

REAL-TIME PERIPHERAL DRIVERS

This section defines the operation of the following device drivers:

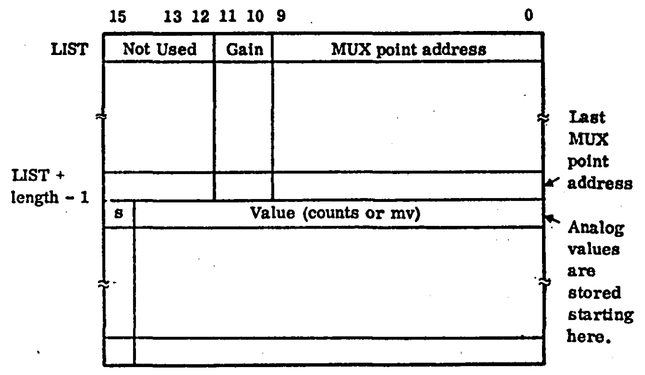
- D1536 – 1536-2/1525-3/1502-80 Relay Multiplexer Analog Input Subsystem driver
- D1501A – 1501-10/1525-3/1501-11 Solid-State Multiplexer Analog Input Subsystem driver
- D1544A – 1544 Digital Input driver
- D1553A – 1553 Digital Output driver
- D1555A – 1555 Relay Output driver
- D1547A – 1547 Events Counter driver
- D1566A – 1566 Digital/Analog Conversion driver
- L15721 – 1572-1 Sample Timing Unit Line Synchronous Timer driver
- S15721 – 1572-1 Sample Timing Unit Sample Rate Generator driver
- D1590 – 1590 Remote I/O Local Adapter driver

RELAY MULTIPLEXER ANALOG INPUT SUBSYSTEM DRIVER

This driver responds to READ or FREAD requests, returning data from the analog input subsystem to the request buffer. A READ request causes the analog input to be returned as the actual analog digital converter (ADC) reading in left-justified ones-complement form. An FREAD request causes the analog input to be converted to millivolts and returned as an integer value equal to analog input millivolts by gain (mv x gain). Thus, the full-scale integer value is always 5000 (decimal) representing:

5000 mv x 1
500 mv x 10
50 mv x 100
5 mv x 1000

The following is the format of the request buffer (LIST):

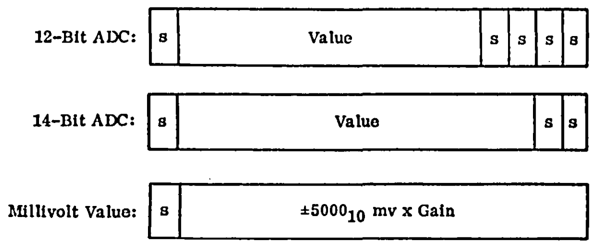


Where:

gain is 0, multiply by 1
1, multiply by 10
2, multiply by 100
3, multiply by 1000

s is the sign

If the request was completed without error, the value returned to the buffer is as follows.



If the request resulted in an error, the data value returned is:

Error Code	Description
8000_{16}	Timeout
8001_{16}	Multiplexer reject
8002_{16}	ADC reject

MOTION requests are ignored by the driver. Refer to appendix E for further analysis.

SOLID-STATE MULTIPLEXER ANALOG INPUT SUBSYSTEM DRIVER

This driver is a re-entrant subroutine that is directly called by the user to return the value of one analog input channel.

The FORTRAN function is:

```
    CALL AIRD (INDEX, IVAL, IERR)
```

The assembly call is:

```
    LDQ INDEX
    RTJ AISB
    A = value
    Q = error
```

Where:

INDEX	is	the logical index for all analog inputs (1 to n). The index is contiguous whether the analog input stations are contiguous or installed on the 1750-1/2 Computer Interface Unit/Computer Interface Expander or on the 1590 Remote I/O Local Adapter. Driver control tables located in SYSDAT enable the driver to translate this logical index to an actual hardware address and to determine if the device is connected on the 1590 Remote I/O Local Adapter.
IVAL	is	the analog digital converter reading returned as a left-justified ones complement number.
IERR	is	the error return.

0	No error
8000_{16}	INDEX is not legal
8001_{16}	Reject

When connected to the 1590, the operation of this hardware subsystem is handled by this driver with no effect on the user interface.

DIGITAL INPUT DRIVER

This driver is a re-entrant subroutine that is called directly by the user to return the value of one digital input word. The FORTRAN function is:

```
    CALL DIRD (INDEX, IVAL, IERR)
```

The assembly call is:

```
    LDQ     INDEX
    RTJ DISB
    A = value
    Q = error
```

Where:

INDEX	is	the logical index to all digital inputs (1 to n). The index is contiguous whether the digital input stations are contiguous or installed on the 1750-1/2 or on the 1590. Driver control tables located in SYSDAT enable the driver to translate this logical index to an actual hardware address and to determine if the device is connected on the 1590.
IVAL	is	the value of the digital input word.
IERR	is	the error return.

0	No error
8000_{16}	INDEX is not legal
8001_{16}	Reject

When connected to the 1590, the operation of 1544 digital inputs is handled by this driver with no effect on the user interface.

DIGITAL OUTPUT DRIVER

This driver is a re-entrant subroutine that is called directly by the user to output one word or one bit of digital output.

The FORTRAN function is:

```
        CALL DORD (INDEX, IVAL, IERR)
```

The assembly call is:

```
        LDQ INDEX
        LDA value
        RTJ DOSB
        Q = error
```

Where:

INDEX	is	the logical index to all digital outputs (1 to n). The index is contiguous whether the digital output stations are contiguous or installed on the 1750-1/2 Computer Interface Unit/Computer Interface Expander or on the 1590 Remote I/O Local Adapter. Driver control tables located in SYSDAT enable the driver to translate this logical index to an actual hardware address and to determine if the device is connected on the 1590.
Bit addressing	is	specified by setting bit 15 of the index to 1. INDEX then becomes a logical index to digital output bits numbered from the first digital output station to the last and from bit 0 through bit 15 of each output word.
IVAL	is	the desired state to be output to the digital output word. If bit addressing is used, bit 0 of the value is used by the driver to indicate the desired state of the selected output bit.
IERR	is	the error return.

0	No error
8000_{16}	INDEX is not legal
8001_{16}	Reject
8002_{16}	Bit addressing with no data buffer specified in the control table (SYSDAT)

When connected to the 1590, the operation of 1553 digital outputs is handled by this driver with no effect on the user interface.

RELAY OUTPUT DRIVER

This driver is a re-entrant subroutine that is called directly by the user to output one word (16 relays) or one bit (one relay) of relay output. The FORTRAN format is:

```
CALL RORD (INDEX, IVAL, IERR)
```

The assembly call is:

```
LDQ INDEX
LDA value
RTJ ROSB
Q = error
```

Where:

INDEX	is	the logical index to all relay outputs (1 to n). The index is contiguous whether the relay output stations are contiguous or installed on the 1750-1/2 or on the 1590. Driver control tables located in SYSDAT enable the driver to translate this logical index to an actual hardware address and to determine if the device is connected on the 1590.

Although the user treats 1555 relay outputs as if they contained 16 relays per station, the hardware stations actually contain eight relays each. The driver compensates for this by making two outputs per call. The eight lower order bits of the value are output to the second station. If the second station is not installed (as indicated in the control table in SYSDAT), the second output is not executed.

IVAL	is	the output value desired. For bit addressing, only bit 0 is used. For bit-addressed momentary relays, IVAL is ignored by the driver.
IERR	is	the error return.

0	No error
8000_{16}	INDEX is not legal
8001_{16}	Reject
8002_{16}	No data buffer for non-momentary bit-addressed relays

Bit Addressing

If bit addressing is required, bit 15 of INDEX is set to 1 and the lower 15 bits become a logical bit index to all relay outputs numbered from the first station to the last and from bit 0 through 15 of each word. When bit addressing is specified, the driver picks up the last output value (16 bits) from the data buffer in SYSDAT, changes the specified bit as specified by bit 0 of the value, and outputs the 16-bit word in two separate outputs as in word addressing.

Momentary Relays

The momentary 1555 Relay Output Stations may be treated the same as latching relay outputs: they may be word or bit addressed. From the control table in SYSDAT, the driver determines if the 1555 stations are momentary relays. The driver outputs IVAL if word addressing is specified or sets the bit specified by IVAL if bit addressing is specified. A data buffer in SYSDAT is not required.

Restrictions (Momentary Relays)

The 1555 momentary relay outputs provide an adjustment for the length of time the relays remain closed when addressed. When doing outputs to momentary relays, the user must not output to the same address again until the expiration of the closure time delay or uncertain results occur on the relay outputs.

When connected to the 1590 Remote I/O Local Adapter, the operation of the 1555 relay outputs is handled by this driver with no effect on the user interface.

EVENTS COUNTER DRIVER

This driver is a re-entrant subroutine that is called directly by the user to return the value of one counter operating in EPUT (events per unit time) mode.

The actual reading of the counters is done by a section of the driver running periodically on the timer. Each time it is scheduled, it reads all counters specified in the control table and places the results in a data buffer in SYSDAT. When the driver is called by a user, the value from the data buffer is returned. The periodic scan of all counters is started by the first user call to the driver.

The 1547 Events Counters are connected to gate from the 1572 Sample Timing Unit Sample Rate Generator (SRG). The driver initializes the sample rate generator multiplier with 1000, hence the gate time for the counters will be the selected frequency of the sample rate generator divided by 1000. The 1547 Events Counters are configured to clear when read.

The counters are treated as if they were 16-bit counters.

The FORTRAN format is:

```
CALL CTRD (INDEX, IVAL, IERR)
```

The assembly call is:

```
LDQ INDEX
RTJ CTSB
A = value
Q = error
```

Where:

INDEX	is	the logical index to all counters (1 to n). The index is contiguous whether the counter stations are contiguous or installed on the 1750-1/2 Computer Interface Unit/Computer Interface Expander or on the 1590 Remote I/O Local Adapter. Driver control tables located in SYSDAT enable the driver to translate this logical index to an actual hardware address and to determine if the device is connected on the 1590.
IVAL	is	the reading from the counter that represents the number of counts received within the gate time established between two pulses from the sample timing unit. The value is treated as one 16-bit counter but actually may be two 8-bit counters. The user must handle the separation in this case.
IERR	is	the error returned.

0	No error
8000_{16}	INDEX is not legal
8001_{16}	Reject
8002_{16}	Returned on first call; periodic reading is not in process and data is bad.

When connected to the 1590, the operation of the 1547 counters is handled by this driver with no effect on the user interface.

DIGITAL/ANALOG CONVERSION DRIVER

This driver is a re-entrant subroutine that is called directly by a user to output a value through the 1566 Digital/Analog Conversion Unit. The driver supports current output only. The FORTRAN format is:

```
CALL AORD (INDEX,IVAL,IERR)
```

The assembly call is:

```
LDQ INDEX
LDA value
RTJ AOSB
Q = error
```

Where:

INDEX	is	the logical index to all analog outputs (1 to n). The index is contiguous whether the analog input stations are contiguous or installed on the 1750-1/2 or on the 1590. Driver control tables located in SYSDAT enable the driver to translate this logical index to an actual hardware address and to determine if the device is connected on the 1590.
IVAL	is	the value converted to an analog output by the 1566 Digital/Analog Conversion Unit. The transfer function is:

0_{16} = 0 percent of full scale

FFF_{16} = 100 percent of full scale

The full-scale ranges available are:

1 to 5 milliamperes

4 to 20 milliamperes

10 to 50 milliamperes

IERR is the error returned.

0	No error
8000_{16}	INDEX is not legal
8001_{16}	Reject

When connected on the 1590, the operation of the 1566 outputs is handled by this driver with no effect on the user interface.

REMOTE I/O DRIVER

This driver responds to WRITE requests to operate any of the following devices:

1544 Digital Input Units

1553 Digital Output Units

1555 Relay Output Units

1547 Events Counter Units (EPUT mode)

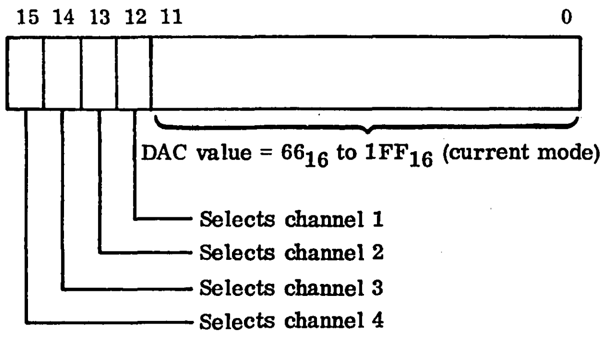
1501/1525 Solid-State Multiplexer, Analog Input Subsystem

1536/1525 Relay Multiplexer, Analog Input Subsystem

1566 Digital/Analog Conversion units

The request buffer specified by s and n in the request has the same format as shown for Relay Multiplexer Analog Input Subsystem Driver above with the following device differences:

1544	The addresses specified in the buffer are the actual station addresses of the devices on the remote.
1553	The addresses specified in the buffer are the actual station addresses of the devices on the remote. The corresponding values to be output are loaded into the second half of the buffer.
1555	The addresses specified in the buffer are the actual station addresses. The corresponding values to be output are loaded into the second half of the buffer. These are eight-bit values.
1547	Same as for the 1544 Digital Input Units.
1501/1525	The addresses specified are the channel addresses of the analog input. The driver calculates the multiplexer station address by adding the integer result of dividing the channel number by 16 to the base station address in the physical device table. The values are returned into the second half of the buffer as the analog digital converter reading in left-justified ones-complement form.
1566	The addresses specified are the actual station addresses of the analog outputs. The corresponding values to be output must be combined with the channel addresses and placed in the second half of the buffer. The format of this value/channel address word is as follows.



Although all of the above devices can be operated by MSOS WRITE requests; the normal mode of operation and the one used by all of the subroutine drivers (D1544A, D1553A, D1555A, D1547A, D1501A, and D1566A) is to interface through a periodic scanning program (ATOSAN) that operates all devices connected to the 1590 Remote I/O Local Adapter on an independent basis using data buffers in SYSDAT. ATOSAN runs every 10 seconds and generates the WRITE requests described above for all devices on the 1590.

When the subroutine drivers encounter a device that is remote in the control tables, they make a direct call to a subroutine (BUFEXC) that interfaces with the data buffers maintained by ATOSAN.

The 1536/1525 Analog Input Subsystem, which is connected to the 1590 Remote I/O Local Adapter subsystem, is not operated by ATOSAN. The user must make an MSOS WRITE request to an appropriate logical unit using a request buffer formatted as shown for the locally connected 1536/1525 Relay Multiplexer Analog Input subsystem. The analog input values are returned to the user as the actual analog digital converter reading in left-justified ones-complement form. If the request results in an error, the data value returned will be as follows:

Error Code	Description
8000_{16}	Timeout
8001_{16}	External reject on multiplexer or analog digital converter
8002_{16}	Internal reject on multiplexer or analog digital converter.

In addition to these data errors, the v field of the request is set negative (bit 15 equals 1) if an irrecoverable error occurs in the 1590 subsystem.

LINE SYNC TIMER DRIVER

The line sync timer driver allows the user to enable or disable the line sync timer if it is not designated as the system timer. It will generate interrupts and/or sync pulses at one, two, four, eight, or 16 times the line frequency (50 or 60 Hz) and is selectable only at the hardware level.

Line Sync Timer Driver Requests

Since this drive is actually a subroutine, a request is made by making a direct call. The FORTRAN function is:

```
ISTAT = ILST(IVALUE)
```

The assembly call is:

```
RTJ ILST              RTJ ILST
ADC IVALUE      or    ADC (IVALUE*)
```

Where:

IVALUE has	bit 0 set at 1	Enable LST interrupt
	0	Disable LST interrupt
	bit 1 set at 1	Enable LST sync
	0	Disable LST sync

Line Sync Timer Driver Error Conditions

The only errors the drive can detect are internal or external rejects. An error code is returned in the A register:

A = 0	No error
+1	External reject
−1	Internal reject

SAMPLE RATE GENERATOR DRIVER

The sample rate generator driver allows the user to have programmable periodic interrupts and/or sync pulses. The rate is determined by the product of a 16-bit data register and a precision time base of 1, 10, 100, or 1000 kHz.

Sample Rate Generator Driver Requests

Since this driver is actually a subroutine, a request is made by making a direct call. The FORTRAN function is:

```
ISTAT = ISRG(ICOMND,IVALUD)
```

The assembly call is:

```
RTJ ISRG              RTJ ISRG
ADC ICOMND      or    ADC (ICOMND-*)
ADC IVALUE            ADC (IVALUE-*)
```

Where:

ICOMND is 1	Input the status to IVALUE.
2	Input the value of counter to IVALUE
3	Output the value from IVALUE to the register.
4	Output a function from IVALUE where IVALUE has all bits set at zero except:

Bit 0 = 1	Enable sample rate generator interrupt
0	Disable sample rate generator interrupt
Bit 1 = 1	Enable sample rate generator sync
0	Disable sample rate generator sync

Sample Rate Generator Driver Error Conditions

The only errors the driver can detect are internal or external rejects. An error code is returned in the A register:

A = 0	No reject
+1	External reject
−1	Internal reject

PSEUDO TAPE DRIVER

The pseudo tape driver drives pseudo devices that have the external characteristics of a magnetic tape. The pseudo devices are sequential files that are created and accessed by the driver for calls to the file manager. Using job control statements several set-up operations must be performed prior to using pseudo tapes from the unprotected file. The file must be defined, opened, etc., using:

```
*DEFINE
*OPEN
*CLOSE
*RELEAS
*PURGE
*MODIFY
```

These statements cause the necessary initialization of the pseudo tape's physical device table. After this set-up process, the devices may be accessed as normal magnetic tape by using the monitor requests:

```
READ
WRITE
FREAD
FWRITE
MOTION
```

as well as the job control statements:

```
*REW
*UNL
*EOF
```

NOTE

When writing on a pseudo tape that is at load point, the driver releases the current file space and redefines the file, causing the information in the old file to be lost. If the tape is positioned at some record between the load point and end-of-tape and a writing operation is performed, the existing record is lost.

There is also a provision for pseudo tapes to be used in the protected foreground. The physical device tables for these devices are initialized at system configuration time, rather than through use of the job control statements. The file number must be predefined, the status initially defined as ready at load point, and WRITE enabled. The file number must have the system status reflect that the device is not available for writing from the unprotected file. (Refer to the 1700 MSOS 4 Installation Manual for specific details.)

File numbers $7FFD_{16}$ through $7FFF_{16}$ are reserved for the MSOS verification tests, and file numbers $7FFS_{16}$ through $7FFC_{16}$ are reserved for the foreground pseudo tapes. The background pseudo tape file numbers are in the preceding block.

NOTE

If it is allowable to read the foreground pseudo tapes from the background, it is the user's responsibility to ensure that the condition described in the preceding caution does not occur.

PSEUDO TAPE DRIVER REQUESTS

READ/FREAD and WRITE/FWRITE have a maximum record length that is specified at the time the file is equated to the logical unit defined. Any attempt to write a longer record results in an error. Any attempt to read a longer record results in a short read condition; format and mode have no meaning. All information is transferred in binary mode. The number of words specified determines the record length. If the number of words exceeds the maximum of 192, the record is truncated to 192 words.

The MOTION request format is described under Motion in section 1. Density has no meaning. All other motion codes are processed when the driver is advancing or backspacing records and a file mark is encountered. The current MOTION command is terminated with the end-of-file status (bit 11, word 12) set in the physical device table.

ERROR CONDITIONS

The following error conditions are recognized by the pseudo tape driver:

- Attempt to write on a file that is not enabled for a write
- File not defined or not assigned to this logical unit
- Disk hardware failure
- Request specifies a record length that is longer than the maximum record length specified in the file definition
- Not enough file space available for this request

COSY DRIVER

The COSY driver is actually a COSY compress/decompress module. Input through this driver is from the standard COSY input unit; output is to the standard COSY output unit.

COSY INTERFACE DRIVER REQUESTS

READ

There is no difference between READ and FREAD. The first record read is a CSY/ record; the associated deck identification determines the manner in which the reset of the deck is processed. If the deck identification is blank (generated by the write logic), no sequence numbers are added to the requester's record length. If the deck identification is not blank (generated by COSY), sequence numbers are added, along with one deck identification if specified. 192 words are then read from the standard COSY input unit, and the first record is unpacked in the requester's buffer. Subsequent read requests are filled by unpacking the next record in the buffer until the buffer is exhausted and another 192-word block is read in. Read requests are processed until an END/ record is read that is translated into the user's buffer. Information transferred must be in ASCII mode.

WRITE

There is no distinction between formatted and unformatted writes. Information in the user's output buffer is packed together until a 192-word block has been filled. The block is written out on the standard COSY output unit, and another block is started. The end-of-file motion control request indicates the end of the writing process. The buffer is written out, although it may be less than 192 words. An END/ record is then written and the end-of-file MOTION request is passed on to the COSY output unit. The first write request to the driver causes a CSY/ record to be written on the output unit. An END/ card is output following the last block, and the end-of-file MOTION request is passed on to the COSY output unit.

Motion

The only motion control request recognized by this routine is the write end-of-file that indicates the end of data compression and causes output of the COSY buffer, even if it is only partially filled. All other motion control requests are simply passed on to the standard COSY input or output unit. MOTION requests to skip records (forward/backward) skip physical COSY records (192 words) rather than logical records.

Status

The hardware status is picked up from the device table of the standard COSY unit being used.

OPERATION

The user can perform both read and write functions with one physical device table. However, because the driver data buffer must be kept intact for an entire deck, read and write must be alternated at the deck level and not at the record level (read one deck, write one deck, read one deck, etc.). If two physical device tables are used, alternate concurrent read and write is allowed. Two concurrent reads or writes address the same physical unit and therefore are not allowed because only one COSY input device and one COSY punch device can be defined for each job. The *CSY control statement must precede the *K control statement when the COSY standard logical units and system standard logical units are being defined.

ERROR CONDITIONS

The COSY driver sets bit 15 of the v field when error conditions occur. The alternate device handler outputs the error message and specifies the COSY driver logical unit number and an error code. The reply for the error message is a CU or DU. The user program must check bit 15 of the v field to determine if an error occurred.

SOFTWARE BUFFER DRIVER

The software buffer driver is capable of improved core memory utilization in situations where programs require data output to system devices that are classified as having slow data rates (e.g., I/O-TTY, paper tape punch, etc). The following definition of terms is presented to clarify the description of the software buffer driver.

Term	Description
Buffer logical unit	A logical unit upon which a user program requests data output
Buffer area	An area on mass storage reserved to store data temporarily for the software buffer driver until the data can be output on the buffer output logical unit
Buffer output logical unit	The actual output logical unit upon which the software buffer driver makes the request to output data
Buffer physical device table	The table that defines the limits of the mass memory buffer area and the buffer output logical unit and contains the character buffer and current pointers into the buffer area
Character buffer	An area in the buffer physical device table where the data is placed while it is being output on the buffer output logical unit
Data	A block of information whose starting address, number of words, and mode (binary or ASCII) are defined in the user's request on the buffer logical unit

The request section of the software buffer driver is initiated when a user makes a formatted WRITE request on a buffer logical unit. The driver completes the request immediately if it is for motion. MOTION requests cannot be buffered. The driver requests that the requester's data be written in the buffer area. The data on mass memory is preceded by a one-word header that contains the length and the mode in bit 15. When the data has been buffered on mass memory, the request is complete and the user's buffer is now available for further I/O operations. The output section of the driver is initiated if it is not already active.

The output section requests that data be read from the buffer area to the character buffer. It then requests that the character buffer be output on the buffer output logical unit in the mode defined by the original user. The pointers are adjusted, and the cycle continues until all data in the buffer area has been output on the buffer output logical unit.

The driver checks for such error conditions as buffer area over-subscription, I/O errors on mass memory, and buffer output devices that are logged in the engineering file.

NOTE

Data cannot be input by the software buffer driver.

Consult the MSOS Customization Manual for further information on buffer set-up and organization.

BUFFERED LOCAL TERMINAL CONTROLLER DRIVER

Below is a description of the operation of the buffered 1745-2 Local Terminal Controller driver. One to 12 120-display stations can be operated through a 1706 Buffered Data Channel.

USER INSTRUCTIONS

To communicate with the computer, the operator presses the SEND key on the conversational display terminal keyboard. The computer indicates that operator input is required with the alert indicator. After the operator has input his commands and pressed the SEND key, the KEYBOARD LOCK indicator illuminates and remains lit until the computer causes an actual data transfer. The computer reads the requested number of words from the last STX or from the upper-left corner of the screen if no STX is displayed. The computer stops receiving data when the requested number of words has been received or until an ETX is transmitted, whichever is shorter. The only exception is when an ETX immediately follows an STX; ETX is not transmitted, and data is transferred from the STX to the next ETX on the screen or the requested number of words has been transferred, whichever is shorter.

Note that it is not necessary for the ALERT indicator to be on before pressing the SEND key. The computer will recognize all send interrupts and respond by transferring data.

FORMAT WRITE

The driver is FORTRAN-compatible on format write operations; i.e., the first character of the output buffer or format statement can be one of the following control functions:

First Character	Control Function
1	Top of form (clear-reset)
0	Double-line skip
Blank	Single-line skip
+	No line advance

These characters are not printed; however, any other character can be printed and a single-line advance is executed before the message is written.

UNFORMATTED WRITE

Any write request can perform control operations by including control characters preceded by a hardware escape code (ASCII $1B_{16}$). See Control Characters below. Some conversational display terminal functions, however, must be output as actual function commands or software-simulated functions. To perform these operations, the driver provides pseudo-escape codes as part of the regular write feature. The action indicated below is performed:

Pseudo-code	Function
$1B0C_{16}$	Clear-reset
$1B07_{16}$	Alert
$1B1A_{16}$	Suppress send

The suppress send function causes any unattended send operations at that station to be ignored. Any new request to that station's logical unit clears the send suppress condition. Any or all of these pseudo codes may be used in a single message, but there must be no more than three and they must precede any functional escape codes or data. Pseudo-escape codes are replaced with synchronous (nonfunctional) characters while the message is output and replaced in the user's message buffer before completing the request.

FORMAT READ

A format read executes a line skip, puts an STX character at the beginning of the new line, and performs an alert function to indicate that the computer is waiting for input. The request times-out if the operator does not press the SEND key within two minutes.

UNFORMATTED READ

A regular READ request causes an immediate transfer from the screen. The program RD1745 is expected to do a regular read from the screen in response to an unattended send. This feature can be used as the basis for a conversational package with minimum system load (e.g., the DAC console handler for conversational display terminals). This technique is for more efficient than a format read/format write sequence.

CONTROL CHARACTERS

The following control functions must be preceded by an escape code:

ASCII Code	Function
11	Skip
12	Line skip
13*	Selective clear
14*	Line clear
0B	Reset cursor

The following control functions do not require a preceding escape code:

ASCII Code	Function
1D†	Tab protect
1E†	Tab access
0A	New line
16	Synchronous
19†	End tab
1F†	Start blink
5F†	Start tab
09	Horizontal tab
0C	Carriage return

†Edit keyboard only

Note that the carriage return displays an arrow in the current marker position and does not put the entry marker at the beginning of the next line.

Start blink causes all following characters up to the next blank or new line character to flash on the screen.

ERROR CONDITIONS

In the case of a ghost interrupt from the 1706 Buffered Data Channel, the driver prints the message:

GI 1706

on the standard comment medium.

In the case of a ghost interrupt from the 1745 Local Terminal Controller, the driver prints the message:

GI CRT

on the standard comment medium.

The driver does not call the alternate device handler; it produces its own error messages and recovery. If any error occurs while core is allocated, it is released. All errors result in the completion of any pending request. The error message format is:

CRT yx

where y is the station number of the conversational display terminal on which the error occurred and x is as follows:

0	Diagnostic timeout
1	Reject in initiator
2	Reject doing function output
3	Reject attempting buffered I/O
4	Reject issuing write terminate function
5	Reject in 1745 interrupt response (station interrupt)
6	Reject in 1745 interrupt response (end-of-operation interrupt)
7	Reject in send portion of continuator
8	Reject in end-of-operation portion of continuator
9	Allocatable core not sufficient for this format read size
A	Zero length request
B	Software cannot identify this end-of-operation. It is followed by a ghost interrupt message.

Rejects are separated by the execution area of the driver. This makes it easier to identify problems and the time of their occurrence. Error 1 is usually the most common reject message; it indicates that the unit is unavoidable. This error is typed out if the unit is turned off.

Error 9 should occur only if there is a basic MSOS customization problem. The error is generated on an error return from a SPACE request. This occurs only when the core allocator discovers that there is not enough core (allocatable plus unprotected) to satisfy this SPACE request. This is extremely unlikely, since the driver never asks for more than 522 words, far below the minimum required by the job processor.

Error A results from the receipt of a zero length request. The request is completed immediately without any error flag.

Error B results when an end-of-operation interrupt is received at the driver continuator and the software flags (word 13 of the physical device table) indicate that the interrupting unit was not expecting an interrupt. The printout is followed by the ghost interrupt message.

LIMITATIONS AND RESTRICTIONS

There are several places in the driver where it talks directly to the 1706. Therefore, this driver assumes that the 1745 is on the first 1706 in the system. This driver does not allow the conversational display terminal to be used as a comment device.

This program makes space requests for format reads.

The driver requests allocatable core at request and completion priorities equal to the running priority of the driver; the operating system will swap to satisfy this request. Therefore, it is unadvisable to use this driver to execute format READ requests from unprotected core, especially if the system is not set up with an area 4 (i.e., protected allocatable core area longer than the minimum required for the job processor). One way to safeguard this operation is to set aside an area n, where n is the priority of the driver. The disadvantage is that this area is restricted to programs with a request priority greater than or equal to n. If this technique is used, area n should be the size of the largest screen plus 2 (522 words), since the driver never makes a larger space request.

This driver is for buffered operation only.

DIGIGRAPHICS DRIVER

This driver handles the I/O transfers between the 1700 Series and the 1744 Digigraphics Controller. It can drive up to six 1744/274 Interactive Graphics Consoles (refer to the 274 Interactive Graphics Manual Version 2. A 1706 Buffered Data Channel is used to handle buffered inputs and outputs. The alternate device handler is not used by this driver.

NOTE

The 1700 Series user cannot make direct calls to the 1744/274 driver since all communications are performed by the interactive graphics system.

EXTERNAL CHARACTERISTICS

Each controller is identified by a unique logical unit number and each has an entry in the PHYSTB giving the equipment code and the console number associated with it. Figure 3-7 is the assembly language calling sequence.

15 14 13 12 11 10 9 8 7 4 3 0

RTJ-($F4)				
0	rc	x	rp	cp
c				
thread				
v	h	a	l	
n				
s				
b				

Where:

rc is the request code for the desired function.

READ	A normal read from the 1744 controller is attempted. If unsuccessful, the request is completed with error.
WRITE	A normal write from the 1744 controller is attempted. If unsuccessful, the request is completed with error.
FWRITE	A write is made to the 1744 controller at the next available 1744 location. If unsuccessful, the request is completed with error.
ID READ	1744 memory is scanned and a read is made of the first ID block encountered. If unsuccessful, the request is completed with error.
COMPUTER DISPLAY	The 1744/274 hardware is put into computer display mode, which allows the 1700 to drive the 274 console. If unsuccessful, the request is completed with error.

NOTE

Since the Digigraphic I/O requests are eight words long, they should always be issued through an indirect request. An attempt to execute this request by a return jump to MONI, followed by the request, results in a MONI return to the last word of the request.

rp is the request priority.

cp is the completion priority.

c is the completion address.

v is the error code.

h is the restart display, if equal to 1.

a is 0.

l is the logical unit number.

n is the number of words to transfer.

s is the 1700 starting address.

b is the 1744 starting address. If $b > 2000_{16}$, start at the end of the previous call.

INTERNAL CHARACTERISTICS

If more than three internal or external rejects occur, location A3 or A4 is set for the translator and the request is completed with error. The v field error codes are used to indicate the problem to the requester:

v = 2	Buffer overflow
v = 3	Internal reject
v = 4	External reject
v = 6	Short transfer

The following are the error messages:

BDC NOT READY	Buffered data channel not ready
BDC BUSY	Buffered data channel busy
DGC NOT READY	Digigraphic console not ready
DGC EXT REJ	Digigraphic external reject
DGC INT REJ	Digigraphic internal reject

DATA SET CONTROLLER DRIVER

NOTE

The 1700 Series user cannot make direct calls to the 1747 Data Set Controller driver since all communications are performed by the high-speed IMPORT system.

DRIVER REQUEST HANDLING

If there is a request subroutine at the initiator entry, DSCHK, the data set controller check routine is called to see if the hardware is operational. If it is, the type of request being made is checked. The appropriate action is shown below for each type of request.

FREAD

The interrupt on the interrupt word is selected. Completion without error is made if this interrupt is received. If it is not received within 7 seconds, the request times out and is completed with error and short transfer.

FWRITE

The interrupt word is sent, and the request is completed without error.

READ

A normal READ from the transmission line is attempted. If successful, the request is completed without error; if unsuccessful (due to cyclic code errors), the request is completed with error. If it is completed after 2 seconds, the request is completed with error and short transfer.

WRITE

A normal WRITE over the transmission line is attempted. If successful, the request is completed without error; if unsuccessful (due to a sync word acknowledge signal not being set or an alarm interrupt), the request is completed with error. If not completed after 2 seconds, the request times out and is completed with error and short transfer.

HARDWARE MALFUNCTIONS

The following hardware malfunctions cause a diagnostic to be printed out and the time-out counter to be set to 10 seconds. This immediately notifies the user so that he may remedy the malfunction or consult a customer engineer. The time-out interval is changed to 10 seconds to avoid overloading the teletypewriter with repeated diagnostic messages.

Hardware Failure	Message
Reject	DSC REJECT
Buffered data channel not ready	BDC NOT READY
Buffered data channel busy	BDC BUSY
Data set controller not ready	DSC NOT READY
Data set controller busy	DSC BUSY
Data set controller in test mode	TEST MODE
No carrier signal on the data set	NO CARRIER

All these conditions result in completion of the request with error and short transfer after 10 seconds. The alternate device handler is not used by this driver. The driver attempts the rejected, busy, or not ready conditions 100 times before assuming a hardware failure.

Other errors that can occur are:

- Sync word not acknowledged – A counter in the driver allows it to wait for the sync word acknowledge bit SNCNAK to be set. The user can adjust SNCNAK to suit his conditions; normally, the longer the distance between the terminal and the central, the larger (negative) this value should be.
- Cyclic code errors – Another adjustable parameter, CYCNT, is provided to allow the driver to await the arrival of the cyclic code word after a transmission. If this delay is too short, cyclic code errors are not detected.

SYSTEM INITIALIZER DRIVER 4

The system initializer requires device drivers for many of the peripherals. One driver of any device type may be present in a given system initializer. The drivers in the system initializer do not use interrupts but operate the devices on the basis of the device status condition. In this mode of operation, device operation and timing may differ from the interrupt-driven driver. The logical units used by the initializer are predefined in the range 1 through 8. The minimum initializer requirement is an input driver, comment driver, and mass-memory driver. Dummy routines are provided to satisfy external linkages for missing routines. Table 4-1 defines the logical units, driver names, devices supported, and dummy names.

DRIVER OPERATION

INPUT DRIVERS

The input drivers are controlled by the input driver interface, IDRIV. IDRIV calls the device driver by means of a return jump instruction that passes the following to the driver:

- A register – The buffer first word address for read
- Q register – The number of words to be read

The driver returns to IDRIV with the error indicator (0 for error and 1 for no error) in the A register. If an error occurred, the following error information will be passed:

- Q register – The error code
- I register – The last hardware status

MASS-MEMORY DRIVERS

The mass-memory drivers are controlled by the mass-memory driver interface, MDRIV. MDRIV calls the device driver by means of a return jump instruction that passes the following to the driver:

- A register – If set positive, buffer first word address
  If set negative, write address tags (disk only)
- Q register – If set positive, number of words to be read
  If set negative, complement of the number of words to write
- I register – The starting sector address

The driver exits with the error indicator (0 for error and -0 for no error) in the A register. If an error occurred, the following error information is passed:

- Q register – The error code
- I register – The last hardware status

DRIVER ERRORS

The IDRIV module of the initializer reports device failures on the initializer comment device as:

```
L,xx FAILED yy (zzzz)
ACTION
```

Where:

xx	is the failed logical unit.
yy	is the error code (the same as for MSOS drivers).
zzzz	is the last hardware status.

The response to the error takes one of two forms:

- RP – Repeat the operation.
- CU – Abort the operation and return to the comment unit for a subsequent control statement.

DISK-TO-TAPE UTILITY DRIVERS

The disk-to-tape utility program requires drivers for three types of devices: magnetic tape, mass memory, and the comment device. These drivers are status-driven and do not use interrupts. In general, calls to these drivers specify two parameters:

A register = Buffer address

Q register = Length of buffer.

COMMENT DEVICE

The length of the data buffer is 120 characters. The driver tests each character as a carriage return; if the test is true,

TABLE 4-1. HARDWARE DEVICE DRIVERS

Logical Unit	Driver Name	Hardware Devices	Equipment Code/WES	Dummy Name †	Other Requirements
1	QPTAPE	1721/1722	1/00A1	QPTDMY	
2	QCARD	1726/405	11/0581	QCDDMY	ASCII63 conversion requires CR026. ASCII68 conversion requires CR029.
		1726/1706/405	11/1581		
		1728/430	11/05A1		
		1729-2	11/0581		
		1829-30	11/0581		
		1729-3	11/0581		
3	QMT9TK	1732-1/609	7/0381	QMTDMY	Unit 0 used
		1732-1/1706/609	7/1381		
		1732-2/615-93	7/0381		
3	QMT7TK	1731/601	7/0381	QMTDMY	Unit 0 used
		1731/1706/601	7/1381		
		1732-1/608	7/0381		
		1732-1/1706/608	7/1381		
		1732-2/615-73	7/0381		
3	QMLS9	1860-5/6	7/0601	QMTDMY	Unit 0 used
4	QDK85X	1733-1/853/854	3/0181		Unit 0 used
		1738/853/854			
4	Q17391	1739-1	3/0181		
4	Q17332	1732-2/856-2/856-4	3/0181		Unit 0 used
4	Q1751	1751	2/0101		
4	Q1752	1752	2/0101		
4	Q18334	1833-4	14/0700		Unit 0 used
6††	Q1711	1711	1/0091		
		1712			
		1713			
		713-10			
		1810-1			
7	Q40421	1740/501	4/0201	QPRDMY	
		1742-1			
7	Q42312	1742-30	4/0201	QPRDMY	1742-120 requires train image T5954
		1742-120			
		1827-30	4/0201		
7	Q18277	1827-5/7			1843-2 CLA
8	DUMMY†††				

†Used to allow linkage of unused driver entry points
††Logical unit 5 is reserved for future use.
†††Used when no device action is desired

the data transfer is complete. On input, lowercase alphabetic characters are converted to uppercase.

MASS MEMORY

The length of the data transfers is normally 16 sectors of 96 words each. All data is transferred in a buffered mode.

MAGNETIC TAPE

In addition ot the normal parameters defined above, if the A register is equal to zero, a file mark is written and the tape is rewound to the load point. When tape is being read, a check for a file mark is made on each read. A file mark signifies the end data for a given dump.

DSKTAP UTILITY DRIVERS

These drivers transfer data between disk and tape. Transfer may be made in either direction. The transfer starts to/from the tape at the current file mark, and on input runs to the end of tape. Transfer from disk starts at the beginning and runs to the designated sector. Transfer to disk starts at the beginning and runs to the end of the tape records. After transfer, the option to verify the data on the two storage media is provided.

There are three types of input/output subroutines used in the DSKTAP utility: magnetic tape, mass memory, and comment devices.

The comment device driver is called:

```
RTJ CDRIVE
```

The A register contains the data buffer address. The Q register contains the length of the data buffer for an output or zero for an input request.

I/O on mass memory is performed by a call to MDRIVE:

```
RTJ MDRIVE
```

The A register contains the data buffer address, and the Q register equals the positive word count for a read and the negative word count for a write.

To read data from magnetic tape, a call to MGREAD is made:

```
RTJ MGREAD
```

The A register is equal to the buffer address, and the Q register is equal to zero. To write data, call MGDRIV:

```
RTJ MGDRIV
```

The A register is equal to the buffer address, and the Q register is equal to the number of data words. If the A register is equal to zero upon entry to MGDRIV, an end of file is written on the tape and the unit is rewound.

Errors can be detected by the calling program by testing the A register to see if it is equal to zero on return from the I/O routines.

SYSTEM CHECKOUT PACKAGE BOOTSTRAP DRIVERS 6

System checkout is an on-line program that diagnoses failures in a 1700 Mass Storage Operating System (MSOS).

The failed system image is written on mass storage by a bootstrap program. This bootstrap operation is followed by a system restart (autoload) and call-up of the system checkout program (SYSCOP) via MIPRO. This program executes at a low priority level, obtaining its information from the image on mass storage in an attempt to isolate the system problems.

System checkout is written as a series of overlays to minimize core requirements.

CHECKOUT BOOTSTRAP PROGRAMS

The checkout bootstrap programs write the core image on mass storage. They are self-contained, status-driven drivers.

ASSUMPTIONS AND RESTRICTIONS

The user is responsible for ensuring that the bootstrap program is intact and in core. The bootstrap must be core-resident since it is referenced absolutely by the user.

The bootstrap program transfers the number of words specified by the user via SYSDAT to mass storage.

The starting sector number is specified by an EQU in SYSDAT.

When using a cartridge disk, the failed image must reside completely on either disk 0 or disk 1.

The released system has standard values specified for memory size and the mass memory location of the failed image. Refer to the MSOS customization manual for procedures to alter these parameters. The system's A, Q, and I registers are saved by the bootstrap program for reference by SYSCOP.

COMPLETION AND ERRORS

After transferring the failed image to mass storage, the bootstrap stops (loops on a selective stop) with the Q register set to 0 if no errors occurred during the transfer or negative if errors occurred.

BOOTSTRAP OPERATION

When the system fails, the following steps are used to bootstrap the failed system onto mass memory.

1. Stop the computer. Do not press MASTER CLEAR.
2. Clear the M, P, Y, and X registers.
3. Set the P register to the address 142_{16}.
4. Set the SELECTIVE STOP switch. Select the Q register.
5. Place the computer in run. The computer will stop when the failed image has been transferred.
6. Autoload the system.
7. After system start-up, request SYSCOP via MIPRO.

SYSTEM ABORT DUMP

When a system condition causes abnormal termination of system operation, the core-resident system abort dump program can be executed to dump the contents of selected core locations. There is a unique dump routine for each type of printer. These routines do not use interrupts; they are status-driven. The procedure to operate the program is:

1. Press MASTER CLEAR.
2. Enter the starting address of the location to be dumped in the A register.
3. Enter the ending address of the locations to be dumped in the Q register.
4. Set the P register to the address 140_{16}.
5. Execute the program by setting the RUN/STEP switch to RUN.

The following result after the program is executed:

1. The paper is set to the top of the form.
2. There is an absolute/relative heading of 16 columns at the top of each page.
3. Absolute and relative addresses and 16 words are printed per printer line.
4. Lines having 16 words the same as the last line printed are ignored by printing a line of asterisks.
5. Sixty lines are printed per page.
6. The program hangs when the requested number of words are printed.

The system abort dump program can be executed as many times as necessary to dump the selected contents of core by repeating the operating procedure.

The printout is shown in figure 6-1.

```
AESL   REL    50†    61     72     83     94     A5     B6     C7     D8     E0     FA     0B     1C     2D     3E     4F

1F75   1A70   1CF7   0000   0000   0000   0000   0180   0B00   08FB   48FB   E105   0FA6   0D03   0A0F   5400   1797   C8F3
1F85   1A80   E8F3   0F61   46EF   0121   184F   C800   FC67   09FC   0100   D800   FC63   5802   182D   0B00   5800   FD79
1F95   1A90   0FC7   0132   08E2   0337   1CF8   C800   FC58   09FB   0111   1CDC   D800   FC53   0A00   68D3   58EE   5800
1FA5   1AA0   FDOR   18FD   D8CE   AC3D   011F   58E7   5800   ED04   18FD   A02D   0119   D8C5   58E0   5800   FCFD   1800
1FB5   1AB0   FFFC   A02D   0111   1804   C8BC   09FE   688A    8B9   0113   E888   0FA1   1A12   09FE   68B3   5800   FD49
1FC5   1AC0   C02D   0309   0DFE   0202   18FE   0B00   5800   ECE4   18FA   18ED   0B00   1800   FCAE   1800   FF1C   1800
1FD5   1AD0   FD43   1800   FE15   0800   FC19   09FD   0101    8B1   C80C   FC15   D800   FC13   09FD   0138   0111   1820
1FE5   1AE0   0CC0   4S00   FC0B   4800   FC0A   1C90   58A6   E110   0FAA   C02C   0172   C000   022C   2000   000C   6813
1FF5   1AF0   E106   54BF   0DFE   0F6F   380E   680D   5800   PD11   C80B   03DC   5892   C807   0102   09FE   18F6   6800
```

NOTE: TO CONSERVE CORE MEMORY, THE TRAIN IMAGE IS NOT LOADED WHEN THE 1742-120 LINE PRINTER AND CONTROLLER IS USED (ASSUMING THERE IS AN EXISTING TRAIN IMAGE).

†THE 5 IS ABSOLUTE; THE 0 IS RELATIVE.

Figure 6-1. System Abort Dump Printout

GLOSSARY

A register	A CPU register used for addition. It is also used for one-word input/output transfers.
Absolute address	1. An address that is permanently assigned by the machine designer to a storage location 2. A pattern of characters that identifies a unique storage location without further modification 3. Synonomous with machine address, specific address
Address	1. An identification, as represented by a name, label, or number, for a register, location in storage, or any other data source or destination; e.g., the location of a station in a communication network 2. In general, any part of an instruction that specifies the location of an operand for the instruction
Advance file	Command to magnetic tape to advance tape until the next file mark is found
ADT	Auto data transfer – an I/O mode that allows A/Q transfers to be made in blocks as if a buffered transfer were undertaken. ADT mode drivers notify the requesting program only when the end-of-block transfer mark interrupt occurs. Micro interrupts for each A/Q transfer are treated internally by the driver.
Alarm	1. Error signal in the status word, usually indicating a serious hardware malfunction 2. An audible signal; e.g., sent to an interactive terminal
Alternate device handler	A driver for an alternate hardware device. An alternate device is used for the I/O transfer when an unrecoverable error occurs on the scheduled I/O device. The alternate device handler usually also processes the error indications for the device that failed.
A/Q channel	A Control Data CYBER 18/1700 computer data channel that can handle input/output only through the A register is called the A/Q channel.
Auto-data transfer	See ADT
Back read	A magnetic tape command that causes magnetic tape to be read in the reverse direction
Backspace	A magnetic tape command that causes magnetic tape to be wound in the reverse direction for the number of records specified
Batch	In MSOS, an object program running in a stacked job manner that shares the CPU with the priority program when a priority program is present and executes only when the priority program is not in control of the processor. Batch interrupts have lowest priority in the interrupt processing priority scheme.
Batch processing	Pertaining to the technique of executing a set of computer programs so that each is completed before the next program of the set is started. Batch jobs are not considered to be time-critical since they do not need a particular response time (batch jobs have the lower priority).
BCD	Binary coded decimal
Binary mode	A method of reading data one binary bit at a time. Binary readouts are given in BCD or hexadecimal format for CYBER 18/1700 data.
Block	Consecutive machine words or characters considered or transferred as a unit, particularly applicable to I/O
Bootstrap	A technique or device designed to bring itself into a desired state by means of its own action; e.g., a machine routine whose first few instructions are sufficient to bring the rest of itself into the computer from an input device

BOT	Beginning of (magnetic) tape. A physical mark on the tape
Buffer	1. A routine or storage used to compensate for a difference in rate of flow of data or time of occurrence of events, when transmitting data from one device to another 2. An isolating circuit used to prevent a driven circuit from influencing the driving circuit
Buffered data	Data that is transferred via the I/O channel using the buffered data channel
Buffered data channel	The high speed I/O transfer channel for CYBER 18/1700 systems. Buffered data controllers transfer blocks of data. Micro interrupts are handled by hardware or firmware without driven intervention.
Cassette tape	A two-track magnetic tape with a single track of sequential data in each direction
Character mode	A data buffering mode in which the basic byte is usually a 7-bit ASCII character plus a parity bit
Checksum	A summary of digits or bits used primarily for checking purposes and summed according to an arbitrary set of rules
Completion priority	A field in the standard I/O request parameter list. It designates the priority to returning control to the requestor after the task is completed.
Continuator entrance	An entrance to a program used to return control after a request has been completed
Control point	A boundary for main memory. Control point is a necessary parameter for 16-bit addressing in main memory when the main memory is larger than 65K words. MSOS 5 requests do not need control point parameters since the location of the request is a saved value.
Controller	A hardware device that controls access and data transfer to I/O units that are connected to it
Core	1. The main memory of a core type computer 2. The main memory of core saved in mass memory
Core image	A word-for-word image of core saved in mass memory
COSY	A compress/decompress module used to maintain programs
CPU	Central processing unit
CRC	Cyclic redundancy check – a hardware or firmware generated check code. It is stored on mass storage following the data and is returned with data read from mass storage. It is used for error checking to define data quality.
CYBER 18/1700	A series of CDC mini-computers systems including the CYBER 18-10, 18-20, 18-30 Timeshare and SYSTEM 17. The standard operating systems for the CYBER 18/1700 computers are RTOS 3 and MSOS 5.
Cyclic redundancy check	See CRC
Cylinder	A set of tracks on a multisurfaced disk pack. Each track in a cylinder is on a different disk pack surface and lies above/below all other tracks on the same cylinder. The read/write heads are simultaneously positioned over all tracks in a single cylinder.
Data packing	A method of storing data within a data word to minimize non-used bits; e.g., packing 6-bit bytes into a 16-bit word would cause eight bytes to be packed into three words. (If the data was not packed, this would require one word per byte or per two bytes.) Bytes would be split between words 0 and 1 and between words 1 and 2.
Diagnostic logical unit	A logical device devoted to detecting hardware errors. Many physical devices will have two logical unit identifications: a main logical unit identification for data transfer purposes, and a diagnostic logical unit identification for checking data transfer quality and storage. Each logical unit has its own physical device table.
Diagnostic timer	A clock value set at the start of I/O transfers. If the requested operation has not completed before the preset time elapses, an error is declared and the driver initiates recovery/abort routines to terminate the request.

Disk	A mass storage device consisting of a controller and one or more disk packs each with one or more rotating surfaces for storing data. One movable pair of heads (read/write) is provided for each surface. These heads read or write data to a single track at a time.
Diskette	A single surface flexible disk pack
Driver	A program whose main function is to perform a physical I/O transfer of data between one storage medium and another (e.g., between central memory and mass storage, between central memory and magnetic tape)
DSKTAP	A disk-to-magnetic tape module
DTLP	A disk-to-magnetic tape module
Dummy driver	A routine substituting for a driver that has substantive tasks to perform. The sole function of a dummy driver is to return control to the calling module.
ECC	Error correction code. A data quality checking code
Engineering file	A software controlled error logging file that stores information on hardware failures. An unrecoverable I/O error is usually logged in the engineering file.
EOF	End-of-file
EOP	End-of-operation
EOT	End of (magnetic) tape. A physical mark locating the end of usable tape
Error entrance	A entrance to a program used to return control after an external request has failed
Error recovery	Software that attempts to use data checks and special techniques to complete malfunctioning I/O transfers. If the recovery techniques succeed, no indication of an error is usually given to the requester. If the recovery techniques fail (unrecoverable error), the user program is often notified, the transfer is usually attempted on the alternate device (if any), and the error may be logged in the engineering file.
ETX	End-of-text
External reject	Reject sent by an I/O controller when it cannot perform a specified operation within the allotted period
FDD	Flexible disk driver
File mark	Mark imposed on tape (as a separate record) to signify end-of-file
Find next request	RTOS or MSOS routine that checks request queues and logical tables to find the next request to be executed. This routine assures that I/O channels remain busy so long as there are unstarted I/O requests.
FNR	Find next request
Formatted data	Data that is written in sector mode (e.g., uniform records)
fpi	Frames per inch
FREAD	The formatted read request
Function command	Command to an I/O controller that prepares the peripheral device for an I/O transfer
FWRITE	The formatted write request
Ghost interrupt	An unsolicited interrupt from a peripheral device or an unused line
Hang-up	When a request is unable to be completed because a peripheral device is not able to issue the necessary interrupt, the condition is called an I/O hang-up
Head	The magnetic read or write head for a magnetic disk, drum, or tape
Hollerith	A code
Indirect	An addressing mode
Internal reject	The reject generated by the CPU when the I/O controller fails to answer a request within the allotted time
Initial entrance	The initial entry point used when the program is first called
Initialize disk	The process of writing track and sector addresses on a disk pack
I/O	Input/output
Job processor	The batch processing subsystem that initiates, monitors, and terminates all jobs executed in unprotected core
Kernel drivers	Drivers that are written in a special modularized fashion so that some modules may be excluded from the driver at system initialization, thereby saving CPU space during execution

LCTT	Low cost tape transport
LOG1 LOG1A LOG2	Logical unit tables
Logical unit tables	Three tables that are used to locate alternate devices and other logical unit information
lu	Logical unit
Macro instruction	An instruction in a source language that is equivalent to a specified sequence of machine instructions; usually a mnemonic instruction that a programmer can write in a source program to call for library or special routines
Macro interrupt	An interrupt signaling an error or completion of an I/O operation (e.g., a word transfer on the A/Q channel or a data block transfer on buffered channel or ADT)
Main memory	The principal memory of the CPU; it may be core or solid state (depending on the CPU) and includes all addressable memory within the CPU (excluding micro memory)
MAKQ	The request queuing program
Mass memory	The external magnetic memory, usually disk. It may also be drum. In cases where neither disk nor drum are included, it can be a magnetic tape device.
Micro interrupt	An interrupt that signals an error or completion of a single I/O operation; e.g., a single word transfer of an ADT block transfer. Micro interrupts are often handled by the driver without notification to the requesting module.
MO5	NCR operating mode (set/sample instructions)
MONI	An entry point for the monitor
Monitor	The supervisory routine in an operating system that coordinates and controls the operation of user and system programs
MOTION	The motion request
MSA	Mass storage address
MSOS	Mass Storage Operating System
Offset	For the storage module drive, the ability to position the head slightly off the track in a direction perpendicular to the track. This may allow data recovery if the read head is not aligned perfectly with the track.
Overflow	For buffered transfers – the condition that occurs where an I/O device attempts to transfer more words of data than are provided for in the buffer
p1, p2, p3	The motion request fields. These appear in the motion word of the request and are processed in the order given; i.e., p1 before p2; p2 before p3. No more than three requests can be made in the same request.
Page eject	A line printer control command causing the printer to skip the remainder of the current page and to move to the top of the next page
Parameter	1. A variable that is given a constant value for a specific purpose of process 2. A quantity in a routine that specifies a machine configuration, subroutines to be called, or other operating conditions
Parameter list	A part of a READ, WRITE, FREAD, FWRITE, or MOTION request. The parameter list holds all the unique parameters necessary to prepare the driver for executing this specific request.
Parity check	A mode of checking data by adding a binary bit so that the sum of all binary bits (e.g., in a byte) is always even or always odd. If parity is not satisfied, the user program is usually notified that data quality is degraded.
Physical device tables	The table (PHYSTB) that holds most information needed by the driver to control an I/O transfer for the specified logical unit. One physical device may have several logical unit designations. Each logical unit must have its own physical device table. Some I/O transfer parameters are saved in the physical device table at request time; last device status is saved at the end of the transfer.
PHYSTB	Physical device table
Protect mode	Areas of main memory or mass storage may be protected from write operations (or even read access) so that important programs and data cannot be inadvertently destroyed
Pseudo disk	A method of partitioning disk for addressing purposes so that a bit limited addressing field (e.g., 16 bits) may be used to address (in word mode) all of disk. The disk driver translates pseudo addresses to true disk addresses.

Pseudo tape	A method of treating data so that no matter where it is stored (e.g., in core), it appears to have the same format to the user as if it had been stored on magnetic tape
Q register	One of the CPU arithmetic registers. It is also used for I/O transfers in conjunction with the A register.
Queue	A list of requests waiting to be processed; requests are ordered by priority level
READ	The normal input request (non-formatted)
Real time	Pertaining to a program for which time requirements are particularly stringent
Relative	A mode of addressing
Request	The method used by a program to start an I/O transfer. Standard requests are READ, WRITE, FREAD, FWRITE, and MOTION.
Request priority	The priority assigned to a request. This determines the order in which a group of waiting requests are processed. Completion priority may differ from the request priority.
Rewind	To return a tape or disk file to its beginning
RTOS	The Real Time Operating System
Scheduler	The portion of the monitor that schedules programs to be executed (may be an initial, continuator, or error entry to the program). Other entry points are normally reached by a jump or return jump to a program entry point without using the scheduling function of the monitor.
SCMM	Small Computer Maintenance Monitor
Sector	A number of contiguous words in main or mass memory. Disks and drums have a preset sector size. When used in sector mode, sectors are consecutively numbered throughout the mass storage device. Initialization writes the sector addresses on the disk or drum.
Seek	The operation that positions the disk read/write heads over the track where the I/O transfer is to be made. All heads over all surfaces move as a unit; hence a seek operation positions heads over all tracks in a single cylinder.
Segment	A portion of the main memory lying in the same 65K set of core address; e.g., a 262K main memory would have four segments: 0 through 65K, 65K through 131K, 131K through 196K, and 196K through 262K.
Slew	To pass data until the desired end of input pattern is sensed
SMD	Storage module drive
Status	A state or condition of hardware or a task; e.g., busy or not busy. I/O devices normally send status of the operation just finished to the CPU.
Strobing	Timing of data; e.g., from a disk. By advancing or delaying the strobing of a data bit, the disk controller may be able to achieve a better data transition state. Variable strobing is one of the storage module drive modes of error correction.
System data	A region in low core reserved for data used by several programs. Physical device tables are located in the system data region.
SYSDAT	System data region
Tape	A data storage medium. Magnetic tape (2-, 7-, or 9-track) and punched paper tape are used by various CYBER 18/1700 systems.
Thread	A list of entries that each contain a pointer to the next entry; e.g., logical unit thread. I/O requests (i.e., parameter lists) that are a part of the requesting program are usually threaded in place, and are therefore scattered throughout core. An entry may be threaded to the beginning (highest priority), end (lowest priority), or someplace within the thread. In the last named case, the thread is broken and the new entry is threaded to both ends of the break in the thread.
Timeout	The expiration of a preset period of time. I/O transfers are normally allowed a predefined period. If one operation is not completed during this period, a timeout is declared and the driver enters error processing.
Time-sharing	The capability of a computing system to accommodate more than one user during the same interval of time without apparent restriction caused by the existence of other users; a given device is used in rapid succession by a number of other devices, or

	various units of a system are used by different users or programs. The sharing is controlled automatically and may or may not include a priority scheme by using multiprocessing. The time-sharing may reduce total processing time from that required to do batch processing.
Timer requests	The stack of requests awaiting time-out checks. The timing program periodically checks requests. If the request is still in the stack at the end of the period, a timeout is declared for that operation. Completed operations are removed from the timer stack before timeout.
Track	A data storage region on disk, drum, or tape. Tape tracks are linear and do not require addresses; disk and drum tracks are circular and must be addressed to establish a start of track mark.
Unload	To remove a tape from ready status by rewinding it beyond the load point; the tape is then no longer under control of the computer.
User program	An object program loaded and entered under MSOS control; includes batch and priority programs and library routines.
v field	An error field in the parameter list used by the driver to notify the user program of I/O failures
WES	The address code in the Q register used to activate an I/O device. A director field is also included in the I/O device address.
WRITE	The normal output request (non-formatted)
Write enable	A command or physical device (e.g., write enable button on disk controller; write enable ring on magnetic tape reel) allowing data to be written onto the device, or to a region of the device
Write protect	A switch setting or program command that disables write operations into the protected region of mass storage or main memory

STANDARD EQUIPMENT/INTERRUPT ASSIGNMENTS FOR CYBER 18-10, 18-20, AND 18-30 TIMESHARE

TABLE B-1. STANDARD EQUIPMENT/INTERRUPT ASSIGNMENTS FOR CYBER 18-10, 18-20, AND 18-30

Peripheral	Equipment Code	Macro Interrupt	Micro Interrupt
Teletypewriter/conversational display terminal	1	1	1
Paper tape reader	2	2	2
Paper tape punch	2	2	2
Card punch	2	2	2
None	3	3	3
Line printer	4	4	4
None	5	5	5
None	6	6	6
Cassette	7	7	7
Clock	1	8	8
Magnetic tape transport (NRZI only)	9	9	9,0
Eight-channel communications line adapter	10	10	10
Dual-channel communications line adapter	10	10	10
Card reader	11	11	11
Magnetic tape transport (NRZI and phase encoded)	12	12	N/A
IOM	13	13	N/A
Storage module disk (SMD)	14	14	N/A
Cartridge disk drive (CDD)	14	14	N/A
Flexible disk drive (FDD)	15	15	N/A

PHYSICAL DEVICE TABLE

The physical device tables are included in SYSDAT (the system tables and parameters that are located at the beginning of core).

PHYSICAL DEVICE TABLE

Each physical device has a device table (PHYSTB) that contains the interfacing information specified by the user to the device. It contains the entry addresses to the driver responsible for operating the device, the station address that tells the driver which device to use, and the information that allows the driver to fulfill the current request. For all drivers the table contains at least 16 words for a device. Words 0 through 15 have a standard function for all devices. Additional words are added for use by the output message buffer package and special use by drivers. For kernel drivers, words 16 through 23 are also predefined. A detailed description of these common kernel driver words is given in figure C-1.

Following the two standard PHYSTB tables (figures C-1 and C-2), PHYSTBs for selected specific devices are shown in figures C-3 through C-13. Following the PHYSTBs, the logical unit tables (LOG1A, LOG1, and LOG2) are defined and discussed.

The following information gives a detailed description of each of the words for all drivers (figure C-2).

Word 0 ELVL

$520x_{16}$; a scheduler request to operate the driver initiator address at level x, the driver priority level

Word 1 EDIN

The driver initiator address

Word 2 EDCN

The driver continuator address; control is transferred to EDCN on interrupt at the priority level assigned to the interrupt trap region. This priority level must be the same as the priority level specified by word 0.

Word 3 EDPGM

The driver error routine address; control is transferred to EDPGM when the diagnostic clock is counted down to negative by the diagnostic timer at the driver priority level.

Word 4 EDCLK

The diagnostic clock; this location is set by the driver and counted down by the diagnostic timer for a hardware completion interrupt. It is set idle (-1) by a complete request.

Word 5 ELU

The logical unit currently assigned to the device; it is zero if the device is not in use. It is set by the request processor and may be reassigned by find-next-request and cleared by find-next-request or complete request.

Word 6 EPTR

Call parameter list location for current request; it is set by find-next-request.

Word 7 EWES

Hardware/address

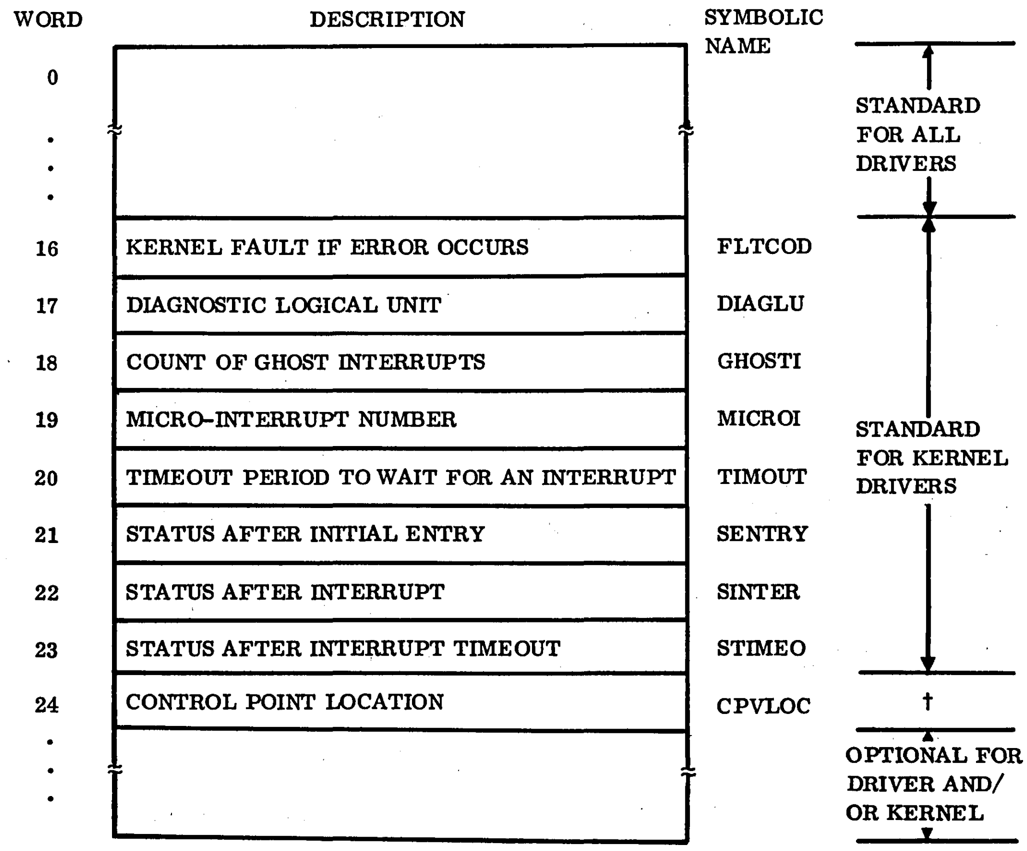
Bits	Code
0 through 6	Station
7 through 10	Equipment (refer to appendix A)
11 through 15	Converter

The equipment status is obtained by loading this word into Q followed by input. Status is saved in word 12, ESTAT2.

Word 8 EREQST

Request status

Bits	Code	Device
15	1	If operation is in progress
	0	If operation is complete
14	1	If driver detects I/O hardware failure



WORD	DESCRIPTION	SYMBOLIC NAME	
0 . . .			STANDARD FOR ALL DRIVERS
16	KERNEL FAULT IF ERROR OCCURS	FLTCOD	STANDARD FOR KERNEL DRIVERS
17	DIAGNOSTIC LOGICAL UNIT	DIAGLU	
18	COUNT OF GHOST INTERRUPTS	GHOSTI	
19	MICRO-INTERRUPT NUMBER	MICROI	
20	TIMEOUT PERIOD TO WAIT FOR AN INTERRUPT	TIMOUT	
21	STATUS AFTER INITIAL ENTRY	SENTRY	
22	STATUS AFTER INTERRUPT	SINTER	
23	STATUS AFTER INTERRUPT TIMEOUT	STIMEO	
24	CONTROL POINT LOCATION	CPVLOC	†
. . .			OPTIONAL FOR DRIVER AND/ OR KERNEL

Figure C-1. Common Words in PHYSTB for All Kernel Drivers

Bits	Code	Device
13, 12, 11		The equipment class
	0	Class undefined
	1	Magnetic tape
	2	Mass storage
	3	Card
	4	Paper tape
	5	Printer
	6	Teletypewriter
	7	Reserved for future use
10 through 4		Equipment type constant (T)
	0	1711
	1	1721/1722
	2	1723/1724
	3	1752
	4	713-10/711-100 713-120
10 through 4 (continued)	5	1738/853
	6	1751
	7	1739-1
	8	1738/854
	9	1731/601
	10	Software buffer
	11	COSY driver
	12	1728/430
	13	Core allocator
	14	1733-1/854
	15	1733-2/856-2
	16	1733-2/856-4
	17	1742-30
	18	1742-120
	19	1740/501
	20	1732-2/615-73
	21	1732-2/615-93
	22	1732-1/1706/608

†Standard for DMA kernel drivers, except for SMD.

Bits	Code	Device
10 through 4 (continued)	23	1726/405
	24	1732-1/608
	25	1732-1/609
	26	1713 Keyboard
	27	1713 Punch
	28	1713 Reader
	29	1729-2
	30	1732-1/1706/609
	31	Software Dummy
	32	364-4/361-1
	33	364-4/361-4
	34	1742-1
	35	1777 Reader (Paper Tape Station)
	36	Pseudo Tape
	37	1777 Punch (Paper Tape Station)
	38	1729-3
	39	1733-1/853
	40	1731/1706/601
	41	1726/1706/405
	42	1747
	43	1744/274
	44	1536 (Local)
	45	1501 (Remote)
	46	1536 (Remote)
	47	1544 (Remote)
	48	1553 (Remote)
	49	1555 (Remote)
	50	1566 (Remote)
	51	1547 (Remote)
	52	1595 Serial I/O Card
	53	1732-3/616/72
	54	1732-3/616-92/95
	55	1743-2
	56	1745/210
	57	1725-1 Card Punch
	58	1720-1 Reader
	59	1720-1 Punch
	60	Magnetic Tape Simulator
	61	1732-2 Long Record Driver
	62	1810-1/1811-1 CDT/Printer
	63	1829-30/60 Card Reader
	64	1827-30 Line Printer
10 through 4 (continued)	65	1860-72 7-Track Tape
	66	1860-92/95 9-Track Tape
	67	1832-5 Cassette Tape
	68	1833-5 Flexible Disk
	69	1833-1/1867-10 Disk
	70	1833-1/1867-20 Disk
	71	Extended Core Driver
	72	Pseudo Disk
	73	1843-2 8-Channel CLA
	74	1866-14 Cartridge Disk
	75	1866-12 Cartridge Disk
	76	1827-7 Matrix Printer
	77	TAB 501 Card Punch
	78 – 89	Reserved
	90	Reserved – OCR
	91	915 OCR Page Reader
	92	929 Document Reader
	93	936 Document Reader
	94	Reserved – OCR
	95	955 Page/Document Reader
	96	Reserved
	97	979 Reader/Sorter
	98	Reserved – OCR
	99	Reserved – OCR
	100 – 127	For user assignment
3	Not used	(reserved)
2	1	If device may be written by unprotected programs
1	1	If device may be read from unprotected programs
0	1	If device is not available to unprotected programs

Word 9 ESTAT1

Status word number 1

Bits	Code	Device	
15	1	If error condition and/or end-of-file is detected	Driver
14	1	If fewer words are read than requested	Driver
13	1	If device remains ready after detecting an error or end-of-file or both	Driver
12, 11, 10, 9, 8, 7, 6	Reserved for special use by individual drivers		
5	0	If this is a control character	Driver
4	0	If this is the first character	Driver
3	1	If ASCII; 0 if binary mode	Driver
2	1	If lower character; 0 if upper character	
1	1	If format READ or WRITE; 0 if unformatted	FNR
0	1	If WRITE; 0 if READ	FNR

Word 10 ECCOR

The location where the driver next stores or obtains data; it is set initially by FNR and updated by the driver (refer to Find-Next-Request in section 2).

Word 11 ELSTWD

Last location + 1 where the driver is to store or obtain data to satisfy the request

Word 12 ESTAT2

Status word 2; the last value of the equipment status (refer to word 7)

Word 13 MASLGN

The length of the driver for this device when the driver is mass-storage resident. This word is zero if the driver is core-resident.

Word 14 MASSEC

Contains the name associated with the sector number on mass storage; if the driver is core-resident, this name is patched with $7FFF_{16}$.

Word 15 RETURN

Used for a return address by NFNR, MAKQ, and NCMPRQ

Word 16 FLTCOD

Kernel fault code if an error occurs

Word 17 DIAGLU

Diagnostic logical unit. If ELU equals DIAGLU, then the request is completed with error. The error is not logged in the engineering file and ALTDEV is not called.

Word 18 GHOSTI

A count of the number of ghost (unexpected) interrupts that have occurred

Word 19 MICROI

The device's micro-interrupt number, if any

Word 20 TIMOUT

The amount of time in seconds to wait for an expected interrupt

WORD	15	14 11 10 9	8	7 4	3 0	SYMBOLIC NAME
0		1 0 1 0 0 1	0	0 0 0 0		ELVL
1	DRIVER INITIATOR ADDRESS					EDIN
2	DRIVER CONTINUATOR ADDRESS					EDCN
3	DRIVER I/O HANG-UP DIAGNOSTIC ADDRESS					EDPGM
4	DIAGNOSTIC CLOCK					EDCLK
5	LOGICAL UNIT CURRENTLY ASSIGNED TO THIS DEVICE					ELU
6	CURRENT REQUEST PARAMETER LIST LOCATION					EPTR
7	CONVERTER	EQUIP CODE		STATION CODE		EWES
8	REQUEST STATUS BITS					EREQST
9	STATUS BITS[1]					ESTAT1
10	CURRENT LOCATION FOR DRIVER					ECCOR
11	LAST LOCATION +1 FOR DRIVER					ELSTWD
12	LAST EQUIPMENT STATUS READ[4]					ESTAT2
13	DRIVER LENGTH[2]					MASLGN
14	MASS STORAGE ADDRESS OF DRIVER[3]					MASSEC
15	USED FOR RE-ENTRANCY BY FNR, MAKQ, COMPRQ					RETURN

STANDARD FOR ALL DEVICES (words 0–15)

OPTIONAL BY DRIVER (words following)

NOTES: 1. REFER TO WORD 8 DESCRIPTION.
FOR RTOS WORD 9 IS REQUEST TYPE
MM - MASS MEMORY FLAG
A/$\overline{B}$ - ASCII OR BIANRY
F/$\overline{F}$ - FORMATTED OR UNFORMATTED

2. FOR RTOS,WORD 13 IS THE ERROR CODE AND ERROR COUNT (ENGINEERING FILE ENTRY)

3. FOR RTOS,WORD 14 IS THE STATUS ON LAST ERROR

4. REFER TO THE 1700 DIAGNOSTIC HANDBOOK

Figure C-2. Common Words in PHYSTB for All Drivers

Word 21	SENTRY	The status after the initial entry into the kernel
Word 22	SINTER	The status after the device has interrupted (entry into the kernel's continuator entry)
Word 23	STIMEO	The status after the device's interrupt has timed out
Word 24	CPVLOC	Control point location. Used for DMA devices only (except SMD).
Word 25...		These words may be added to the device table if required for special purposes. For example, they can be used to count the lines per page of output, link several tables together all using the same driver/kernel, or save multiple status words (the auxiliary status words should start at word 24).

WORD	DESCRIPTION	SYMBOLIC NAME
0 ≈ 23		STANDARD FOR ALL KERNEL DRIVERS
24	FLAG WORD	FLAGWD
25	CLA LOGICAL PORT NUMBER FOR THIS DEVICE	PORT
26	START ADDRESS FOR CONVERTED BUFFER	BEGBUF
27	LAST LOCATION + 1 OF CONVERTED BUFFER	ENDBUF
28	READ FIRST WORD ADDRESS OF CALLER'S BUFFER	RDBUF
29	NEW REQUEST CODE, PRIORITIES	NEWREQ
30	NEW COMPLETION ADDRESS	NEWCOM
31	NEW THREAD	NEWTHD
32	NEW V-FIELD, LOGICAL UNIT	NEWVLU
33	NEW NUMBER OF CHARACTERS	NEWNCH
34	NEW START ADDRESS OF HEADER	NEWSHD
35	PDT THREAD	ELINK
36	PSEUDO COMPLETION FOR NEW REQUEST	

Figure C-3. ITOS 1 Terminal/1811-2 CDT/1843-2 CLA Pseudo Driver PHYSTB

WORD	DESCRIPTION	SYMBOLIC NAME	
0–23	WORDS 0 THROUGH 23 HAVE THE STANDARD KERNEL DRIVER MEANINGS. NOTE HOWEVER THAT WORDS 21 THROUGH 23 ARE CALLED BY THE SPECIAL NAMES OF: SENTRY — WORD 21, INITIATOR DIRECTOR STATUS 1 SINTER — WORD 22, CONTINUATOR DIRECTOR STATUS 1 STIMEO — TIMEOUT DIRECTOR STATUS 1		STANDARD FOR ALL KERNEL DRIVERS
24	INITIATOR DIRECTOR STATUS 2	RAWSI2	
25	CONTINUATOR DIRECTOR STATUS 2	RAWSC2	
26	TIME OUT DIRECTOR STATUS 2	RAWSE2	
27	CARD COLUMN COUNTER	COLNUM	
28	3.3 MS COUNTER AT FEED TIME	TFEED	
29	3.3 MS COUNTER AT COLUMN 1	TCOL01	
30	3.3 MS COUNTER AT EOP (DIAGNOSTIC LOGICAL UNIT)	TCOLEP	
30	16-BIT PACKED RAW DATA (NON-DIAGNOSTIC LOGICAL UNIT)	WORD	
31	CURRENT ADDRESS FOR RAW DATA (SAME AS WORD 10, ECCOR, IF DIAGNOSTIC LU)	FWA	
32	LAST WORD ADDRESS PLUS ONE OF RAW DATA BUFFER (SAME AS WORD 1, ELSTWD, IF DIAGNOSTIC LU)	LWA	
33	DESELECTS INTERRUPT CONDITION WORDS IN COMBINATION WITH ALL POSSIBLE INTERRUPT STATES: 4 - DESELECTS INTERRUPT ON DATA 8 - DESELECTS INTERRUPT ON EOP $10 - DESELECTS INTERRUPT ON ALARM	INTBIT	
34	ADDRESS OF ALTERNATE WAIT ROUTINE IF VALUE NONZERO	WAITAD	
35	NONZERO VALUE INDICATES TIMEOUT OCCURRED	TIMFLG	
36	CONVERTED WORD LENGTH FOR BINARY READS	LENGTH	
37	EXPECTED SEQUENCE NUMBER FOR FORMATTED BINARY CARDS	SEQ	
38	EXPECTED SEQUENCE NUMBER (SEQ SAVED VALUE)	OLDSEQ	
39	ACCUMULATIVE CHECKSUM VALUE	CHKSUM	

Figure C-4. CB104 Card Reader PHYSTB (Sheet 1 of 2)

WORD	DESCRIPTION	SYMBOLIC NAME
40	LOOP COUNTER USED TO COUNT NUMBER OF PASSES REQUIRED TO CONVERT RAW DATA	CYCLE
41	MOTION OPTIONS TEMPORARY LOCATION	MOTREQ
41	FLAG TO DETERMINE AT WHAT POINT RAW DATA IS BEING CONVERTED FOR FORMATTED BINARY. -0 = PROCESSING COLUMN 1 OR 2 OF FIRST CARD 0 = PROCESSING COLUMN 1 FOR SUBSEQUENT CARDS READ 0 ≠ ALL OTHER CASES	NEWCRD
42	END OF FILE INDICATOR	EOF
43 122	80 WORD RAW DATA BUFFER	BUFR

NOTE: INITIAL VALUE OF WORDS 24-41 AND 43-122 IS 0. INITIAL VALUE OF WORD 42 IS F_{16}.

Figure C-4. CB104 Card Reader PHYSTB (Sheet 2 of 2)

WORD	DESCRIPTION	SYMBOLIC NAME
0-24		STANDARD FOR ALL KERNEL DRIVERS
24	CONTROL POINT LOCATION	CPVLOC
25	LAST DATA TRANSFER FUNCTION	FUNCT
26	BUFFER SIZE FOR SPLIT TRANSFERS	BUFSIZ
27	CYLINDER ADDRESS	CYLADR
28	MASK FOR SEEK COMPLETE	SEKBIT
29	ADDRESS FOR 96 WORD	ABUFF
30	TEMPORARY FIRST WORD - FOR WORD ADDRESSING	TEMFWD
31	TEMPORARY LAST WORD - FOR WORD ADDRESSING	TEMLWD
32	NUMBER OF WORDS - FOR WORD ADDRESSING	WORDNO
33	REQUEST CODE	TEMREQ
34	REQUEST PRIORITY	PRILVL

Figure C-5. Cartridge Disk Driver (CDD) PHYSTB (Sheet 1 of 2)

WORD	DESCRIPTION	SYMBOLIC NAME
35	START SECTOR FOR COMPARE OR RETRY	SVFLAD
36	FIRST WORD ADDRESS FOR COMPARE OR RETRY	SAVFWD
37	ERROR COUNTER - FOR RECOVERY	ERCONT
38	DATA TRANSFER FUNCTION CODE	FDATAF
39	NUMBER OF SECTOR CURRENTLY IN BUFFER	BUFSEC
40	LAST VALUE OF TRUE SECTOR ADDRESS	STAT12
41	LAST VALUE OF TRUE CYLINDER ADDRESS	STAT14
42	LAST VALUE OF CWA	STAT2
43	LAST VALUE OF CURRENT BANK STATUS	STAT3
44	RETURN INDEX AFTER DATA TRANSFER	EXTRA
45	SOFTWARE SECTOR NUMBER	FILEAD
46	UNIT-SELECT PARAMETER	FCONN
47	TEMPORARY SECTOR BUFFER - FOR WORD ADDRESSING	TEMSEC
48	COMPARE OR CHECK WORD - CHECK MOTIONS PARAMETER	COMCHC
49	RESERVED FOR OVERLAY ROUTINE	ECALL1
50	RESERVED FOR OVERLAY ROUTINE	ECALL2
51	RESERVED FOR OVERLAY ROUTINE	ECALL3
52	RESERVED FOR OVERLAY ROUTINE	ECALL4
53	RESERVED FOR MOTIONS	TMOPAR
54	NUMBER OF SECTOR ON DISK (BETWEEN 0-28)	SECTAD
55	SELECTED BANK	BANK
56	ALARM - STATUS	ALRMST
57	RTZS INDICATOR AFTER DEVICE SEEK ERROR	RTZS
58	LAST Q REGISTER	LASTQ
59	LAST A REGISTER	LASTA
60	HARDWARE DYNAMIC STATUS	DYNMIC
61	FIRST SECTOR ADDRESS ON DISK 1	DISK1
62	LINK TO THE NEXT PHYSTB	OTHER
63	96 WORD BUFFER	BUFF

Figure C-5. Cartridge Disk Driver (CDD) PHYSTB (Sheet 2 of 2)

WORD	DESCRIPTION	
0		STANDARD FOR ALL KERNEL DRIVERS
23		
24	CYLINDER ADDRESS FOR TRANSFER	
25	TRACK AND SECTOR FORMATTED	
26	UPPER FIELD OF ADDRESS (BITS 17-16 OF DMA)	
27	TEMPORARY CYLINDER FOR WORD ADDRESSING	
28	USED BY WORD ADDRESSING (TRACK/SECTOR)	
29	USED BY WORD ADDRESSING (FWA)	
30	USED BY WORD ADDRESSING (LWA)	
31	USED BY WORD ADDRESSING (WORD IN SECTOR)	
32	LAST VALUE OF CONTROL UNIT STATUS	
33	LAST VALUE OF DRIVE 1 STATUS	
34	LAST VALUE OF DRIVE 2 STATUS	
35	DATA TRANSFER FUNCTION CODE	
36	LAST DATA TRANSFER FUNCTION CODE	
37	RETURN ADDRESS INDEX FOR DATA TRANSFER	
38	DA NUMBER	
39	LOGICAL DRIVE NUMBER OF DISK	
40	COUNTER FOR SEEK ERROR	
41	RETURN SEEK TO ZERO FLAG	
42	ERROR COUNTER	
43	REQUEST PRIORITY	
44	REQUEST CODE	
45	RESERVED FOR OVERLAY	
46	RESERVED FOR OVERLAY	

Figure C-6. Storage Module Driver PHYSTB (Sheet 1 of 2)

WORD	DESCRIPTION
47	RESERVED FOR OVERLAY
48	RESERVED FOR OVERLAY
49	NOT USED
50	INDEX THROUGH OFFSET JUMP TABLE
51	LAST VALUE OF OFFSET VALUE
52	SPECIAL WORD FOR SPECIFYING SYSTEM CONFIGURATION AND FOR DIAGNOSTIC PROGRAM BIT 15 = 0 1 DA = 1 2 DA BIT 14 = 0 1 DRIVE = 1 MULTIPLE DRIVES BIT 13 = 0 NON-TIMESHARE SYSTEM = 1 TIMESHARE SYSTEM 3-0 = SET BY DIAGNOSTIC PROGRAM = 0 NORMAL TRANSFER = 6 FWRITE = OTHER ADDRESS TAGS
53	ECC ERROR RECOVERY FLAG (1 = DO RECOVERY)
54	MAXIMUM ERROR RETRIES FOR SEEK OPERATION
55	MAXIMUM ERROR RETRIES FOR DATA TRANSFER
56	MAXIMUM ERROR RETRIES FOR CONTROL UNIT CONNECTION
57	MAXIMUM TIME COUNTS LOOP NO. FOR CONTROL UNIT
58	DIAGNOSTIC TIMER VALUE FOR DATA TRANSFER
59	DIAGNOSTIC TIMER VALUE FOR CU WAIT
60	DIAGNOSTIC TIMER VALUE FOR ALT CHANNEL WAIT
61	DIAGNOSTIC TIMER VALUE FOR SEEK OPERATION
62	FORCE RELEASE COUNT
63	TIMESHARE CONTROL POINT (CP) VALUE
64	THESE BITS ARE SET BY DIAGNOSTIC PROGRAM BIT 15 - EARLY STROKE 14 - LATE STROKE 2 - SEEK ONLY REQUEST 1 - DO RETRIES ON ALARM ERROR 0 - DO RECOVERY ON ALARM ERROR
65	ADDRESS OF 96 WORDS BUFFER
66	LINK FOR MULTIPLE PHYSICAL DEVICE TABLE
67	RETURN FROM CONNECT CU SUBROUTINE

Figure C-6. Storage Module Driver PHYSTB (Sheet 2 of 2)

WORD	DESCRIPTION	SYMBOLIC NAME	
0	≈		STANDARD FOR ALL KERNEL DRIVERS
23			
24	CONTROLLER INITIATOR STATUS	RAWST2	
25	CONTROLLER CONTINUATOR STATUS	RAWSC2	
26	CONTROLLER TIMEOUT STATUS	RAWSE2	
27	DEFINE BAD TRACKS OF DISKETTE: BITS 15-8 - FIRST BAD TRACK NUMBER BITS 7-0 - SECOND BAD TRACK NUMBER	BADTKS	
28	NUMBER OF SECTORS PER TRACK	SECTRK	CDC OR IBM FORMAT
29	NUMBER OF WORDS PER SECTOR	WRDSEC	
30	CURRENT TRACK/SECTOR BITS 15-8 - TRACK NUMBER 0-76 BITS 7-0 - SECTOR NUMBER 1 - SECTRK	TRKSEC	MSA IN TRACK/ SECTOR
31	STARTING LOGICAL SECTOR FOR REQUEST	SECTOR	MSA ASKED
32	LAST LOGICAL SECTOR FOR REQUEST. CALCULATES MSA LAST.	LGLSEC	
33	WORD ADDRESSABLE SECTOR OFFSET, 1ST SECTOR	WRDFWA	
34	SECTOR OFFSET FOR END OF DATA IN REQUEST	WRDLWA	
35	CURRENT 1ST WORD ADDRESS	CWABUF	
36	FWA WITHIN USER'S BUFFER FOR KERNEL CALL	FWABUF	
37	LWA WITHIN USER'S BUFFER FOR KERNEL CALL	LWABUF	
38	NFDD CURRENT FWA	CFWA	
39	NFDD CURRENT LWA	CLWA	
40	PSEUDO COMPLETION PARAMETER LIST WORD 0: REQUEST CODE 1: COMPLETION ADDRESS 2: THREAD WORD (SCRATCH LOCATION) 3: LU WORD - V FIELD (SCRATCH LOCATION)	PARLST	

Figure C-7. Flexible Disk Drive PHYSTB (Sheet 1 of 4)

WORD	DESCRIPTION	SYMBOLIC NAME
42	MOTION OPTIONS 4 BIT BYTE CODE: 0 - TERMINATES REQUEST 1 - PRIMES REQUEST 2 - REQUEST NO DATA COMPARE 3 - DATA COMPARE ON WRITE 4 - READ AND WRITE MODE DISKETTE 5 - READ ONLY DISKETTE 6 - NOT USED 7 - NOT USED	MOTREQ
42	SAVED BAD STATUS BITS	BADBIT
42	TRANSFER ROUTINE RWA	FWA
43	TRANSFER ROUTINE LWA	LWA
43	THRESHOLD RECOVERY COUNT IN LFDD	RCVCNT
44	READ RECOVERY OPTIONS BITS 15-11 - ATTEMPTS TO ZERO SEEK 10-6 - NUMBER OF OFF TRACK SEEKS 5-0 - NUMBER OF CONTINUOUS REREADS	RDCNT
45	WRITE RECOVERY OPTIONS BITS 15-11 - ATTEMPTS TO ZERO SEEK 10-6 - NUMBER OF OFF TRACK SEEKS 5-0 - NUMBER OF CONTINUOUS REWRITES	WRCNT
46	ERROR COUNT OF MEDIA ERRORS WITHIN REQUEST	ERRCNT
47	NUMBER OF ERROR RECOVERIES FROM MEDIA ERRS	ERRCOV
48	LOGGING RECOVERY INTERVAL	LOGRCV
49	PASS COUNTER FOR I/O PROGRESSION IN NFDD	PASCNT
50	STARTING SECTOR IN WORD ADDRESSABLE BUFFER	SCRSFC
51	USEABLE SECTOR COUNT IN ABUFF BUFFER	SECNUM
52	MAXIMUM SECTOR COUNT FOR ABUFF BUFFER	SECCNT
53	ADDRESS OF WORD ADDRESSABLE BUFFER	ABUFF
54	BUFFER SIZE IN WORDS OF ABUFF	BUFSIZ

Figure C-7. Flexible Disk Drive PHYSTB (Sheet 2 of 4)

WORD	DESCRIPTION	SYMBOLIC NAME
55	ADDRESS OF WRITE COMPARE BUFFER	BUFADR
56	BUFFER LENGTH IN WORDS OF BUFADR	BUFLEN
57	PRIMED REQUEST FLAG - START OF OPTWRD TABLE	PRIME
	DISKETTE OPTION TABLE OPTION NOT IN EFFECT IF WORD IS ZERO WORD 0: PRIMED REQUEST 1: WRITE COMPARE 2: READ ONLY 3: NOT USED FBTBUF (PRIME+1) 58-63 BAD TRACK SECTOR BUFFER WORDS 1-3 OF OPTWRD TABLE OVERLAP BAD SECTOR OR BUFFER WORD 3: 1ST BAD TRACK 4: NOT USED 5: 2ND BAD TRACK	OPTWRD
64	KERNEL FLAG BITS OPTION WORD	AQDMAF

BYTE NAME	BIT	FUNCTION
AQDMAM	0	0 - DMA XFER, 1-A/Q
RGFILE	1	1 - READ/WRITE REG. FILE (FOR DIAGNOSTIC LU) 0 - ABOVE OPTION IGNORED
IOFLAG	2	0 - I/O NOT OCCURRING FOR UNIT 1 - I/O IN PROGRESS
SUSPND	3	0 - NOT WAITING FOR I/O TO COMPLETE ON OTHER UNIT 1 - WAITING FOR I/O TO COMPLETE ON OTHER UNIT
ILLFCN	4, 5	00 - NO ILLEGAL FUNCTION CODE 1 - ILLEGAL READ FCN CODE ISSUED 10 - ILLEGAL WRITE FCN CODE ISSUED
INPOUT	6	0 - INPUT CONTROLLER FCN 1 - OUTPUT CONTROLLER FCN
RWREQ	7	SAVED READ/WRITE BIT FOR WORD ADDRESSABLE I/O
INTFCN	8	0 - LAST I/O WAS NOT INITIALIZATION 1 - LAST I/O WAS FOR INITIALIZATION

Figure C-7. Flexible Disk Drive PHYSTB (Sheet 3 of 4)

WORD	DESCRIPTION	SYMBOLIC NAME
65	LAST FUNCTION CODE ISSUED	FUNC
66	TIME COUNTER AT START OF SEEK (3.3 MS)	SEEKT1
67	TIME COUNTER AT END OF SEEK (3.3 MS)	SEEKT2
68	ILLEGAL FUNCTION CODE TO BE ISSUED	DIAFCN
69	RWFDD RETURN ADDRESS	RWRTN
70	OFFSET RETURN ADDRESS	OFFRTN
71	BUFFIO RETURN ADDRESS	BUFRTN
72	FLEXIBLE DISK DRIVE PHYSICAL DEVICE TABLE THREAD	FDDPTH

Figure C-7. Flexible Disk Drive PHYSTB (Sheet 4 of 4)

WORD	DESCRIPTION	SYMBOLIC NAME	
0 ≈ 23			STANDARD FOR ALL KERNEL DRIVERS
24	FORTRAN LOGICAL UNIT NUMBER	FINLU	
25	PAPER MOTION COMMAND WORD	MOTCMD	
26	COUNT FOR SPACE FILL	BLNKCT	
27	CHARACTER OUTPUT COUNT	CHARCT	
28	NUMBER OF CHARACTERS PER LINE	LINLEN	
29	LINE COUNT	LINCTO	
30	MAXIMUM NUMBER OF LINES PER PAGE	MAXLIN	
31	MOTION REQUEST WORD SAVED HERE	MTNREQ	
32	ZERO IF ALL BLANKS IN LINE	BLNKDT	

Figure C-8. 1828 Line Printer PHYSTB

WORD	DESCRIPTION	SYMBOLIC NAME
0 ... 23		STANDARD FOR ALL KERNEL DRIVERS
24	FORTRAN LOGICAL UNIT NUMBER	FTNLU
25	CLA LOGICAL PORT NUMBER FOR THIS DEVICE	PORT
26	START ADDRESS FOR CONVERTED BUFFER	BEGBUF
27	LAST LOCATION + 1 OF CONVERTED BUFFER	ENDBUF
28	NEXT CHARACTER FOR OUTPUT BUFFER	NEXCAR
29	NEW REQUEST CODE, PRIORITIES	NEWREQ
30	NEW COMPLETION ADDRESS	NEWCOM
31	NEW THREAD	NEWTHD
32	NEW V-FIELD, LOGICAL UNIT	NEWVLU
33	NEW NUMBER OF CHARACTERS	NEWNCH
34	NEW START ADDRESS OF HEADER	NEWSHD
35	NUMBER OF BLANKS TO OUTPUT FOR TAB	TABCNT
36	N, SUCH THAT TAB STOPS EVERY N CHARACTERS	TABSET
37	CURRENT PAGE LINE COUNTER	LINCT
38	MAXIMUM LINES PER PAGE	MAXLIN
39	LINES TO ADD TO LINE COUNT FOR VERT TAB	VTABLF
40	PHYSTB THREAD	ELINK
41	PSEUDO COMPLETION FOR NEW REQUEST	NCOMP

Figure C-9. 1827-7/1843-2 Matrix Printer PHYSTB Pseudo Driver

WORD	DESCRIPTION	SYMBOLIC NAME	
0			STANDARD FOR ALL KERNEL DRIVERS
24	UNIT AND MODE SELECT	UNTMOD	
25	TEMPORARY STORAGE	ETEMP1	
26	HALF WORD FLAG	HAFURD	
27	ADT TABLE CONTROL WORD	ADTCW	
28	ADT TABLE FWA-1	ADTFW	
29	ADT TABLE LWA	ADTLW	
30	REQUEST TYPE FLAG	MFLG	
31	RECOVERY RETURN ADDRESS	RRETAD	
32	RECOVERY CHECKSUM	RCKSUM	
33	RECOVERY COUNT FLAG	RCNTFG	
34	RECOVERY FLAG BIT = 1 (NO RECOVERY)	RFLAG	
35	REQUEST CODE	QSTCOD	
36	WORD 4 OF REQUEST	QSTWD4	
37	MAXIMUM PHYSICAL RECORD SIZE	PHSREC	
38	PACK/UNPACK BUFFER ADDRESS	ABUFF	
39	PHYSICAL DEVICE TABLE THREAD	ELINK	

Figure C-10. LCTT PHYSTB

A physical device table is required for each LCTT unit (drive). The PHYSTB is located in SYSDAT.

WORD	DESCRIPTION	SYMBOLIC NAME	
0 ⋮ 23			STANDARD FOR ALL KERNEL DRIVERS
24	CONTROL POINT LOCATION	CPVLOC	
25	RETURN FOR RECOVERY	RTRECV	
26	A REGISTER AT LAST OUTPUT	OUTARG	
27	Q REGISTER AT LAST OUTPUT	OUTQRG	
28	UNIT AND MODE SELECT CODE	UNTMOD	
29	WORD 4 OF REQUEST	QSTWD4	
30	RECOVERY COUNT FLAG	RCNTFG	
31	RECOVERY FLAG BIT 15 = 1 DISABLE	RFLAG	
32	MAXIMUM PHYSICAL RECORD SIZE (7 TRACK)	PHYREC	
33	PACK/UNPACK BUFFER ADDRESS (7 TRACK)	ABUFF	
34	ALARM STATUS WORD	ALRMST	
35	TRANSPORT STATUS	TRNSPT	
36	PHYSTAB THREAD	ELINK	

Figure C-11. LCTT/FORMATTER (1860-5/6) PHYSTB

WORD	DESCRIPTION	SYMBOLIC NAME	
0 23			STANDARD FOR ALL KERNEL DRIVERS
24	UNIT SELECT	UNTSEL	
25	TEMPORARY STORAGE	ETEMP1	
26	ADT TABLE (CONTROL WORD)	ADTCW	
27	ADT TABLE (FWA-1)	ADTFW	
28	ADT TABLE (LWA)	ADTLW	
29	REQUEST CODE FLAG	MFLG	
30	RECOVERY RETURN ADDRESS	RRETAD	

Figure C-12. Cassette Drive PHYSTB (Sheet 1 of 2)

WORD	DESCRIPTION	SYMBOLIC NAME
31	RECOVERY CHECKSUM	RCKSUM
32	RECOVERY COUNT FLAG	RCNTFG
33	RECOVERY FLAG BIT =1 (NO RECOVERY)	RFLAG
34	REQUEST CODE	QSTCOD
35	WORD 4 OF REQUEST	QSTWD4
36	PHYSICAL DEVICE TABLE THREAD	ELINK

Figure C-12. Cassette Drive PHYSTB (Sheet 2 of 2)

WORD	DESCRIPTION	SYMBOLIC NAME
0 … 23		STANDARD FOR ALL KERNEL DRIVERS
24	FLAG WORD	FLAGWD
25	CLA LOGICAL PORT NUMBER FOR THIS DEVICE	PORT
26	START ADDRESS FOR CONVERTED BUFFER	BEGBUF
27	LAST LOCATION + 1 OF CONVERTED BUFFER	ENDBUF
28	NEXT WORD FOR OUTPUT BUFFER	NEXWRD
29	NEW REQUEST CODE, PRIORITIES	NEWREQ
30	NEW COMPLETION ADDRESS	NEWCOM
31	NEW THREAD	NEWTHD
32	NEW V-FIELD, LOGICAL UNIT	NEWVLU
33	NEW NUMBER OF CHARACTERS	NEWNCH
34	NEW START ADDRESS OF HEADER (BUFFER)	NEWSHD
35	BINARY PACKING CYCLE ADDRESS OFFSET	CYCLE

Figure C-13. TAB 501 Card Punch PHYSTB, Pseudo Driver (Sheet 1 of 2)

36	SEQUENCE NUMBER	SEQUEN
37	REQUEST MOTION CYCLE	MOTION
38	CHECKSUM WORD	CHKSUM
39	END-OF-FILE CODE	EOFCOD
40	PHYSTB THREAD	ELINK
41	PSEUDO COMPLETION FOR NEW ADDRESS	NCOMP

Figure C-13. TAB 501 Card Punch PHYSTB, Pseudo Driver (Sheet 2 of 2)

LOGICAL UNIT TABLES

Three logical unit tables (LOG1, LOG1A, and LOG2) specify correspondences between logical and physical units, and between logical units and the threads of I/O tasks awaiting execution on those logical units.

LOG1 TABLE - ALTERNATE DEVICE TABLE

The logical unit table indicates whether a logical unit has an alternate physical device that can be used for the source I/O transfer. No more than one alternate unit can be used for a given logical unit.

	15	14	13	12	11	10	9 … 0
LOG1	Largest legal logical unit number						
L1							Alternate logical unit number
L2							
L3							
•							
•							
•							

Where:

Bit 15	is reserved.
Bit 14	is 0 if the logical unit does not share the device. 1 if the logical unit shares a device with another logical unit.
Bit 13	is 0 if the logical unit is operative. 1 if the logical unit is out of service; the alternate, if any, is in use.
Bits 12 – 10	are reserved for future use.
Bits 9 – 0	are the alternate logical unit number.

LOG1A TABLE - LOGICAL/PHYSICAL UNIT TABLE

This table contains pointers that relate each logical unit to its physical device table. Since one entry is provided for each logical unit, one physical device may have more than one associated physical device table.

	15 … 0
LOG1A	Largest legal logical unit number
L1	Address of PHYSTB slot corresponding to this logical unit
L2	
L3	
•	
•	
•	

LOG2 TABLE - TASK THREADS

This table contains pointers to the top of the request thread for each logical unit. The threads themselves are ordered by request priority. At threading time, the requests may be in unprotected core. At execution time, a request being processed is moved to protected core unless swapping is prohibited.

LOG2	Largest legal logical unit number
L1	Top of thread for this logical unit number
L2	
L3	
:	

ASCII CONVERSION TABLE

The 1963 American Standard Code for Information Interchange (ASCII) is used by the 1700 MSOS. ASCII code uses eight bits: bit 8, which is always zero, is omitted in table D-1. Bits 1 through 4 contain the low-order four bits of code for the character in that row. Bits 5 through 7 contain the high-order three bits of the code for the character in that column. The code is given in ascending sequence.

TABLE D-1. CODE CONVERSION TABLE

ASCII Symbol	Bit Configuration	Hexadecimal Number	Meaning
NULL	000 0000	0	Null/idle
SOM	000 0001	1	Start of message
EOA	000 0010	2	End of address
EOM	000 0011	3	End of message
EOT	000 0100	4	End of transmission
WRU	000 0101	5	Who are you
RU	000 0110	6	Are you
BELL	000 0111	7	Audible signal
FE_0	000 1000	8	Format effector
HT/SK	000 1001	9	Horizontal tab skip (punched card)
LF	000 1010	A	Line feed
V_{TAB}	000 1011	B	Vertical tabulation
FF	000 1100	C	Form feed
CR	000 1101	D	Carriage return
SO	000 1110	E	Shift out
SI	000 1111	F	Shift in
DC_0	001 0000	10	Device control/data link escape
DC_1	001 0001	11	Device controls
DC_2	001 0010	12	
DC_3	001 0011	13	
DC_4 (STOP)	001 0100	14	Device control/stop
ERR	001 0101	15	Error
SYNC	001 0110	16	Synchronous idle
LEM	001 0111	17	Logical end of media
S_0	001 1000	18	Information separators
S_1	001 1001	19	
S_2	001 1010	1A	
S_3	001 1011	1B	
S_4	001 1100	1C	
S_5	001 1101	1D	
S_6	001 1110	1E	
S_7	001 1111	1F	

TABLE D-2. ASCII AND BCD CHARACTER SET

8-Bit ASCII Codes	171x-1 TTY Array	171x-2 TTY Array	026 Punches	029 Punches	6-Bit EXT. BCD Magnetic Tape	8-Bit ASCII Codes	171x-1 TTY Array	171x-2 TTY Array	026 Punches	029 Punches	6-Bit EXT. BCD Magnetic Tape
20_{16}	Space	Space	No Punch	No Punch	20_8	40_{16}	@	@	0-8-7	8-4	37_8
21†	!	!	11-8-2	12-8-7	52	41	A	A	12-1	12-1	61
22	"	"	8-7	8-7	17	42	B	B	12-2	12-2	62
23†	#	#	12-8-7	8-3	77	43	C	C	12-3	12-3	63
24	$	$	11-8-3	11-8-3	53	44	D	D	12-4	12-4	64
25†	%	%	0-8-5	0-8-4	35	45	E	E	12-5	12-5	65
26†	&	&	8-2	12	00 (35)††	46	F	F	12-6	12-6	66
27†	'	'	8-4	8-5	14	47	G	G	12-7	12-7	67
28†	(	(	0-8-4	12-8-5	34	48	H	H	12-8	12-8	70
29†	)	)	12-8-4	11-8-5	74	49	I	I	12-9	12-9	71
2A	*	*	11-8-4	11-8-4	54	4A	J	J	11-1	11-1	41
2B†	+	+	12	12-8-6	60	4B	K	K	11-2	11-2	42
2C	,	,	0-8-3	0-8-3	33	4C	L	L	11-3	11-3	43
2D	-	-	11	11	40	4D	M	M	11-4	11-4	44
2E	.	.	12-8-3	12-8-3	73	4E	N	N	11-5	11-5	45
2F	/	/	0-1	0-1	21	4F	O	O	11-6	11-6	46
30	0	0	0	0	12	50	P	P	11-7	11-7	47
31	1	1	1	1	01	51	Q	Q	11-8	11-8	50
32	2	2	2	2	02	52	R	R	11-9	11-9	51
33	3	3	3	3	03	53	S	S	0-2	0-2	22
34	4	4	4	4	04	54	T	T	0-3	0-3	23
35	5	5	5	5	05	55	U	U	0-4	0-4	24
36	6	6	6	6	06	56	V	V	0-5	0-5	25
37	7	7	7	7	07	57	W	W	0-6	0-6	26
38	8	8	8	8	10	58	X	X	0-7	0-7	27
39	9	9	9	9	11	59	Y	Y	0-8	0-8	30
3A	:	:	8-5	8-2	15	5A	Z	Z	0-9	0-9	31
3B	;	;	11-8-6	11-8-6	56	5B†	[	[	12-8-5	12-8-2	75
3C†	<	<	12-8-6	12-8-4	76	5C†	\	\	0-8-2	0-8-2	36
3D†	=	=	8-3	8-6	13	5D†	]	]	11-8-5	11-8-2	55
3E†	>	>	8-6	0-8-6	16	5E	↑	^	11-8-7	11-8-7	57
3F†	?	?	12-8-2	0-8-7	72	5F†	←	—	0-8-6	0-8-5	32

†Refer to note 2 below.
††Refer to note 4 below.

NOTES

1. The 171x-2 TTY array is the ASCII 68, 64-character subset. This array is the same as used on the 171x-3 devices which receive from a 1774.
2. To operate in 026 punched card mode, ASCII 64 options are selected. To operate in 029 punched card mode, ASCII 68 options are selected. These options are assembly-time options for each driver affected.
3. The CDC Standard 1.10.003 is supported by an assembly option. For CDC ASCII mode of operation, the card punches 12-8-2 and 12-0 are stored internally as 7B. The card punches 11-8-2 and 11-0 are stored internally as 7D. For line printer operations, the internal codes 7B and 7D are converted to 5B and 5D to allow printing the hardware compatible graphic characters [(left bracket) and] (right bracket).
4. Since 173x magnetic tape controllers do not provide any code conversion, BCD code 00 is illegal and causes a noise record or BCD code 35 is substituted for the illegal 00 code to prevent tape errors.

   On tape write operations the ASCII codes 25_{16} (%) and 26_{16} (&) are written as BCD 35_8.

   On tape read operations the BCD code 35_8 is always translated to an ASCII $25 (%).

TABLE D-3. HOLLERITH-TO-ASCII CONVERSION

Card Punch	Left Character	Card Punch	Right Character
1	1	-	Blank
3	3	2	2
5	5	4	4
7	7	6	6
9	9	8	8
8-3	'	8-2	&
8-5	:	8-4	/
8-7	"	8-6	>
0-1	/	0-	0
0-3	T	0-2	S
0-5	V	0-4	U
0-7	X	0-6	W
0-9	Z	0-8	Y
0-8-3	¬	0-8-2	/
0-8-5	%	0-8-4	{
0-8-7	∧	0-8-6	_
11-1	J	11-	-
11-3	L	11-2	K
11-5	N	11-4	M
11-7	P	11-6	O
11-9	R	11-8	Q
11-8-3	$	11-8-2	!
11-8-5	]	11-8-4	*
11-8-7	∧	11-8-6	;
12-1	A	12-	+
12-3	C	12-2	B
12-5	E	12-4	D
12-7	G	12-6	F
12-9	I	12-8	H
12-8-3	.	12-8-2	?
12-8-5	[	12-8-4	}
12-8-7	#	12-8-6	<

MAGNETIC TAPE RECOVERY

HARDWARE REQUIREMENT

The recovery procedure provides a uniform recovery algorithm for the 1731, 1732-1, and 1732-2 Magnetic Tape Controllers. This procedure provides an equal recovery capability when using the 1706 Buffered Data Channel and when using the unbuffered A and Q channels.

SOFTWARE REQUIREMENT

The magnetic tape drivers (D1732U, D1732B, D1731U, D1731B, and D17322) provide a capability for maximum recovery by utilizing standard noise records. A system noise record is predefined in size and marks off a bad spot on the tape; it is used as an aid for error recovery.

During a read operation the length and parity of the record are checked first. If there is no parity error and the record length is that of a system noise record, the record is assumed to be standard noise record and is discarded. Another read is then issued to fulfill the user's request.

The standard noise record reduces the likelihood that unerased noise will cause a read error. Standard noise records absorb small amounts of noise and cause the driver to discard them. Without standard noise record the unerased noise would be merged with the following record, causing an incorrect or irrecoverable read.

SWITCHED MODE

When a read error recovers by switching the mode of operation, the v field bits are set (bits 15 through 13), resulting in Q being set negative upon return to the requester. The data read is not placed in the requester's buffer. The tape is positioned after the last record read.

NOTE

It is the user's responsibility to backspace the record and issue a new request in the opposite mode to obtain data.

SUCCESSFUL RECOVERY

On a read function the data is passed to the user as if it were an error-free read. The write function passes control back to the user as if no error had occurred.

UNSUCCESSFUL RECOVERY

Control is never given back to the user without operator intervention; it goes to the alternate device handler where the appropriate message and code are typed. The message format and code are as follows:

L,lu FAILED ec

Where:

lu is	the logical unit number.
ec is	the error code from the alternate device handler.

00	Failure to interrupt (requires timer package)
01	Lost data
03	Parity error
05	Internal reject
06	External reject
13	No write ring
14	Device not ready
31	Processor error
37	1706 Buffered Data Channel address error

Operator's response:

RP	Repeats the request
CU	Allows the processing to continue
DU	Suspends any further processing

SCALING CONSIDERATIONS

The maximum theoretical range of the analog digital converter used is:

12 bit: $\pm 800_{16} = \pm 5.12$ volts

14 bit: $\pm 2000_{16} = \pm 5.12$ volts

However, the hardware cannot produce the most positive value (i.e., 800_{16} or 2000_{16}); the most positive value that can be represented is one count less than full scale (i.e., $7FF_{16}$ for 12-bit analog digital converter or $1FFF_{16}$ for 14-bit analog digital converter. For this reason and to provide the user with overrange detection capability, the full-scale instrument range used is ±5.00 dc. The following relationship exists between analog input and digital output values:

	Left-Justified Value	
Range	12 Bit	14 Bit
Full scale analog digital converter range (-1 analog digital converter count)	$7FF0_{16}$	$7FFC_{16}$
Full scale instrument range	$7D00_{16}$	$7D00_{16}$
0	0	0
Full scale instrument range	$82FF_{16}$	$82FF_{16}$
Full scale analog digital converter range	$800F_{16}$	8003_{16}

The full-scale instrument range in volts is a function of the amplifier gain in the analog input subsystem, and may be as follows.

Gain	= 1000	5 millivolts full scale
	= 100	50 millivolts full scale
	= 10	500 millivolts full scale
	= 1	5000 millivolts full scale

CONVERSION AND CALCULATION OF MILLIVOLTS

The physical significance of counts is as a fraction, since the maximum absolute value 2^{15} of a left-justified analog digital converter value corresponds to 100 per cent or unity. Therefore, any smaller value is a fraction of the full-scale value (maximum absolute value). This fraction F of full scale is:

$$F = \frac{I \text{ (input counts)}}{I_{FS} \text{ (full-scale counts)}} = \frac{I}{2^{15}}$$

Expressing F as a function of analog input voltage V, amplifier gain G, and full-scale analog digital converter gives:

$$F = \frac{VG}{5.12}$$

Solving the two equations yields:

$$VG = 5120 \frac{I}{2^{15}}$$

or

$$VG \times 2^{16} = 10240\, I$$

The value returned to the user is the contents of the Q register after this computation, and hence, the 2^{16} multiplier is absorbed, leaving the result in units of millivolts x gain as:

$$VG = 10240\ I$$

In order to allow the 364 Communications Multiplexer driver (D3644) to operate properly, the following hardware options must be selected. Refer to the 1700 A/Q Communications Multiplexer (DJ814-A) Customer Engineering Manual, Publication No. 41612800, the 361-1 Communications Adapters Reference Manual, Publication No. 41612200, and the 361-4 Communications Adapter Reference Manual, Publication No. 41612200. The following are descriptions of the communication adapters.

- 361-1 – An exclusive, full- or half-duplex, nonsynchronous adapter with an EIA RS232-C interface that requires one address location in a communications multiplexer. Data byte sizes are adjustable from five to eight bits and are transmitted over communications facilities at speeds ranging from 50 to 3000 bits per second.
- 361-2 – An exclusive, simplex, receive-only, nonsynchronous adapter with an EIA RS232-C interface that requires one address location in a communications multiplexer. Data byte sizes are adjustable from five to eight bits and are received over communications facilities at rates ranging from 50 to 3000 bits per second.
- 361-3 – An exclusive, simplex, send-only, nonsynchronous adapter with an EIA RS232-C interface that requires one address location in a communications multiplexer. Data byte sizes are adjustable from five to eight bits and are transmitted over communications facilities at rates ranging from 50 to 3000 bits per second.
- 361-4 – A full- or half-duplex, nonsynchronous adapter with an EIA RS232-C interface that requires two addresses in a communications multiplexer. It uses computer-controlled automatic answering, disconnect, and modem carrier and provides status. Character parity is selectable and data byte sizes are adjustable from five to eight bits and are transmitted at speeds of 50 to 3000 bits per second.
- 361-5 – A full- or half-duplex, synchronous adapter with an EIA RS232-C or current switching mode interface (Bell System type 303) that requires two addresses in a communications multiplexer. It uses computer-controlled automatic answer, disconnect, and modem carrier and it provides status. It provides selection of an odd or even parity character and logical longitudinal sum parity or operates in universal mode. Data characters are eight-bit bytes transmitted at 600 to 230.4K bits per second, depending on the modem and system configuration.
- 361-6 – A full- or half-duplex synchronous adapter with an EIA RS232-C or current switching mode interface (Bell System type 303) that requires two addresses in a communications multiplexer. It uses computer-controlled automatic answer, disconnect, and modem carrier and it provides status. It provides selection of an odd or even parity character and cyclic message parity or operates universal mode. Data characters are eight-bit bytes transmitted at 600 to 230.4K bits per second.
- 361-7 – An automatic calling control adapter that requires one address in a communications multiplexer. It interfaces with an 801A1 or 801C1 Bell System Auxiliary Data Set and enables automatic dialing into the DDD switched telephone network. One unit is required for each dialer controlled.

364-4 COMMUNICATIONS MULTIPLEXER

The clock interrupt must be adjusted to accommodate the highest speed terminal being serviced, which is normally 30 characters per second (300 baud). The clock cycle shorting block must be set from 8 to 40 milliseconds (P2 and P7), and the clock must be adjusted to 33.3 milliseconds. If 364-5 Communications Multiplexer Expansion Modules are present, their interrupts should not be used, since all units are checked by the driver at each clock interrupt.

In the standard release system, the communications adapters are always configured so that all dual channel adapters (361-4) are assigned to the lowest channel numbers, followed by all single channel adapters (361-1).

The following describes the required set-up for the 361-1 Communications Adapter boards.

Board Identification	Shorting Block Selections
Send one (9EPM)	Disable break
	Full duplex
Send two (9EQM)	Disable restraint
	Eight-bit signal generator
	One-bit stop pulse
	Clock speed †
Receive two (9ESM)	Eight-bit signal generator
	Clock speed †

† The bit clock should be set to 3.33 milliseconds for 30-character-per-second operation or 9.09 milliseconds for 11-character-per-second operation. Each character is represented by ten bits.

The following describes the required set-up of the 361-4 Communications Adapter boards.

Board Identification	Shorting Block Selections
Send control (9CUM)	Select ACA
	Disable break
	Disable restraint
	Even parity
	Full duplex
Send one (9CXM)	Clock speed †
	Two-bit stop pulse
	Eight-bit signal generator
Receive control (9CZM) (0FVM)	200 Series Data Set
Receive one (9CWM)	Disable SOM/EOM
	Disable EOM/break
	Clock speed
Receive two (9CVM)	No SOM character
	No EOM character

TERMINAL UNIT

Where selection is present, even parity and full-duplex mode should be employed.

† The bit clock should be set to 3.33 milliseconds for 30-character-per-second operation or 9.09 milliseconds for 11-character-per-second operation. Each character is represented by ten bits.

DRIVER CODING STRUCTURE

A driver in the Mass Storage Operating System (MSOS) is the direct software interface to a hardware device. In some special cases, a driver is used for a pseudo software device. MSOS provides monitor components that aid the driver in performing functions normally common to all drivers. In order to interface to these monitor modules, drivers must conform to a set structure. The purpose of this appendix is to define the structural characteristics of a typical driver and to examine some of the common deviations from this typical form.

DRIVER PRE-INITIATOR FUNCTIONS

When a user makes one of the following monitor requests, a driver is entered: READ, WRITE, FREAD, FWRITE, or MOTION. The read/write request processor (RW) handles read/write requests while T14 handles the tape motion requests. This typical driver can be core- or mass-memory resident as are most MSOS drivers. The driver is scheduled with an indirect monitor request using words 0 and 1 of the driver physical device table.

The address in word 1 of the physical device table causes entry to the driver initiator (core-resident driver) or to the initiator handler entry of MMEXEC (mass-resident driver). MMEXEC moves the driver from mass memory to its driver buffer (when space is available) and jumps to the first location of the driver (not the initiator), passing the physical device table address in the Q register and the driver memory location in the A register. The pre-initiator driver function is to compute the actual memory locations of its initiator, continuator, and error sections and place them in the physical device table.

The following is a typical sequence:

```
START  STQ-   I              SAVE PDT ADDRESS
       TRA    Q              HOLD DRIVER LOCATION
       ADD    =XI1799-START
       STA-   EDIN,I         SAVE INITIATOR ADDRESS
       TRQ    A
       ADD    =XC1799-START
       STA-   EDCN,I         SAVE CONTINUATOR
                             ADDRESS
       ADQ    =XE1799-START
       STQ-   EDPGM,I        SAVE DIAGNOSTIC ERROR
                             ADDRESS
       JMP*   to initiator section
```

NOTE

The I register is the address of the physical device table.

DRIVER INITIATOR FUNCTIONS

The driver initiator sets up the hardware for its desired function including positioning the device to the start of data area if needed and initiating the user request. The following is a typical event sequence:

1. Store the address of the physical device table in the I register.

```
I1799   STQ-    I     SAVE PDT ADDRESS
```

2. Enter the find-next-request module (NFNR) to set up the request and fill in the physical device table (PDT) information. This includes the type of transfer and address of the I/O buffer (or data word) in core.

```
EFNR    RTJ-    (AFNR)
        JMP*    EXIT
        (initiator functions)
```

   A return is made to JMP* EXIT if no requests remain to be serviced or to one location beyond if a request is active. At EXIT:

```
EXIT    JMP+    MAS300      EXIT - NO MORE
                            REQUESTS
```

   The FNR routine will set up physical device table information as defined in appendix B for driver use.

3. The driver should then check if this is a MOTION request by examining the physical device table:

```
LDQ-    EPTR,I      REQUEST ADDRESS
LDA-    (ZERO),Q    GET REQUEST CODE WORD
ARS     9           ISOLATE RC
AND-    LPMSK+5
INA     -14         MOTION = RC 14
SAN     NOMOTN      SKIP, NO MOTION
```

   (Process motion request or, if motion requests are not performed by the driver, go to complete request.)

4. The driver then initiates the hardware operation by some function command.

5. The diagnostic clock is set to allow for a diagnosis of lost interrupts. The time interval is in increments of seconds and is typically very long compared to expected response time.

```
ENA     3           3 SECOND TIMEOUT
STA-    EDCLK,I     PERIOD IN PDT
JMP-    (ADISP)     EXIT TO DISPATCHER
```

6. The driver is now inactive until an interrupt occurs.

DRIVER CONTINUATION FUNCTIONS

The driver continuator section responds to the device interrupts and continues operations begun by the driver initiator section. Two general types of continuator exist:

- The positioning phase is completed and data transfer can be initiated.
- Data transfer is completed and should be checked.

The following is a typical event sequence:

1. The driver is entered from the SYSDAT interrupt response routine for the device with the physical device table address in the Q register. The driver first saves the physical device table address:

```
C1799   STQ-    I   SAVE PDT ADDRESS
```

2. It is possible to get extraneous interrupts (interrupts not caused by driver requests) from a hardware device. The driver first checks for these extraneous interrupts:

   a. Examine the logical unit word of the physical device table to see if it is zero; if so, no request is in progress.

```
LDA-    ELU,I           LOOK AT LU WORD
SAN     NGHOST          SKIP IF NON-ZERO
(Clear controller of device)
JMP-    (ADISP)         IGNORE INTR
```

   b. Examine the device status for interrupt.

```
LDQ-    EWES,I
INP     REJECT-*
(Look at the status bit.)
```

   Clearing the controller at this point may not be advisable, depending on the device. Something must be done to keep the current request progressing and not allow further ghost interrupts.

   If the interrupt status is not present, an eventual exit should be made to the dispatcher (if the above action has been taken).

3. Clear the diagnostic clock physical device table word so the diagnostic timer module does not check for timeouts.

```
ENA     -1          SET CLOCK MINUS
STA-    EDCLK,I
```

4. Continue the request processing until completion.

5. When the request is completed, the complete request module is entered,

```
RTJ-    (ACOMPR)        COMPLETE REQUEST
JMP*    EFNR            GO TO LOOK FOR MORE
                        REQUESTS
```

DRIVER DIAGNOSTIC ERROR FUNCTIONS

The driver error entry is used when the diagnostic timer counts the EDCLK word of the physical device table down to zero. Entry is with the physical device table address in Q:

```
E1799   STQ-    I   SAVE PDT ADDRESS
```

The error code (code = 0) for timeout should then be passed to the driver error handling section.

DRIVER ERROR HANDLING

The driver should diagnose as many specific errors as possible for the hardware device. The driver should save the hardware status in the ESTAT2 physical device table word. All additional hardware status words should be saved in additional physical device table words for reference.

The normal driver error sequence is:

1. Set bits 13 through 15 of the physical device table ESTAT1 word, as noted in appendix B.

2. Use the MAKQ routine to set up short data transfers and the failed Q register.

```
RTJ+        MAKQ
```

3. Use the engineering file to log the error.

```
RTJ+        LOG
```

4. Exit to ADEV to report the error (see Alternate Device Handler in section 2).

```
JMP+        ALTDEV
```

   The request is completed with error, and the initiator of the driver is rescheduled.

DRIVER DIAGNOSTIC UNIT HANDLING

If a driver is to make use of a diagnostic logical unit, appropriate coding is needed in SYSDAT as well as in the driver. SYSDAT must include appropriate entries in all the log tables for the diagnostic logical unit and the physical device table must use a word to store the diagnostic lu for the driver.

For example, the LOG1A table needs an entry like this:

```
     X17990        ADC     P17990      DIAGNOSTIC
                                       1799 UNIT 0
```

(P17990 is the start of the physical device table for 1799 unit 0.)

LOG1 and LOG2 also need proper entriesfor the diagnostic lu. In the physical device table, an equals statement is used to calculate the diagnostic lu. For example:

```
     EQU     U17990        (X17990-LOG1A)
```

Then, in the physical device table some additional word, for instance 19 (any word beyond 15 can be used) is used to store the diagnostic logical unit. For example:

```
     ADC     U17990       19  DIAGNOSTIC LOGICAL
                              UNIT
```

Now the driver can make use of the diagnostic lu in SYSDAT. In the driver's equal statements region there should be an EQU to locate the PHYSTB diagnostic logical unit. For example:

```
     EQU     DIAGLU(19)
```

Then at the end of its error processing would be the following typical sequence:

```
             LDA     DIAGLU,I   Get diagnostic lu.
             SUB     LU,I       Compare against cur-
                                rent lu.
             SAN     NOT DLU
             RTJ-    COMPRQ     Complete request.
             JMP*    INI+1      To find next request
   NOT DLU   RTJ+    MAKQ       set up error info
             RTJ+    LOG        transfers to log error
             JMP+    ALTDEV     to alternate device.
```

GENERAL COMMENTS

The following general information items should be considered:

- Drivers must not hang on INP or OUT instructions, but exit with error on rejects.
- Drivers are made effectively re-entrant by MSOS so that each individual driver need not be re-entrant.
- Drivers that handle multiple devices on a single controller must take care of the overlap operations and interrupt handling with additional logic.
- Drivers should use words 0 through 15 of the physical device table only as specified in appendix C. Additional words can be added for special needs.

FDUTIL FORMATTING FLEXIBLE DISK

This appendix describes the flexible disk drive (FDD) utility program FDUTIL. Throughout the discussion the FDD is referred to as a diskette. Its capabilities include:

- Initializing a diskette
- Inputting data from an external media and writing it onto a diskette
- Copying data from one diskette to another
- Verifying data on one diskette with another

The program executes as a job on a CYBER 18 computer using RTOS or MSOS or MSOS/ITOS. It requires an FDD controller (1833-5) and at least one FDD transport (1865-x).

This appendix describes the FDUTIL requests, error processing, operator intervention procedure, bad track information, FDD output formats, and diskette addressing.

FDD UTILITY REQUESTS

When FDUTIL is executed, a request record is input from the standard input device. It is then output on the standard comment device and processed. Assuming there were no errors, another request record is input and the procedure is repeated until a terminate request record is input. FDUTIL terminates and returns control to the operating system. An option that causes a pause between request records allows operator intervention.

The set of FDD utility requests is:

- *I - Initialize a diskette
- *A - Absolutize relocatable binary programs and write to a diskette
- *B - Input absolute binary programs and write to a diskette
- *H - Input ASCII records and write to a diskette
- *C - Copy diskette to diskette
- *V - Verify diskette with diskette
- *F - Define the initialize format
- *S - Set for operator intervention
- *R - Reset for no operator intervention
- *Z - Terminate the FDUTIL program

Each input request record contains an asterisk in the first character position and an alphabetic character (I, A, B, C, H, V, F, S, R, or Z) in the second position. The optional parameters that follow are separated by commas. The parameters are located in fixed positions as specified by each request. All numeric parameters are in hexadecimal code except lu, which is in decimal code. A numeric hexadecimal parameter (except for lu which is in decimal) must be right justified in its field; that is, the field is filled with leading zeros or blanks. For example:

...,01A,...

or

...,$_\wedge$1A,...

An alphabetic field must be left justified with trailing spaces. For example, the symbol PGM in a six-digit field must be specified as:

...,PGM$_{\wedge\wedge\wedge}$,...

*I, INITIALIZE DISKETTE REQUEST

This request specifies that a diskette be initialized as defined in the FDD controller (see the FDD description in section 3). A diskette must be initialized before it can be written and read.

One track at a time is initialized; the track is then read and checked for errors. After all tracks have been initialized and checked, the bad track information is written. The define initialize format request (*F) should be specified before initializing if the format is to be changed. The format of the initialize request is:

*I,lu,num,i

Where:

lu	is the logical unit of the diskette to be initialized, represented as three decimal digits (positions 4-6).
num	is the number of times the initialize function is to be repeated, represented as three hexadecimal digits (positions 8-10). If blank or zero, the number of times is one. This field is provided for diagnostic or performance testing.
i =	I if interlacing of diskette sectors is desired. If blank or any other character, diskette is initialized sequentially.

*A, ABSOLUTIZE REQUEST

This carries relocatable binary programs that are input from the standard binary input device, to be absolutized and output to a diskette. Output may be in binary or in deadstart format. Provision is made for starting any program at a particular logical sector address. The format of the absolutize request is:

*A,lu

Where:

lu is the logical unit of the diskette used for output, represented as three decimal digits (positions 4-6).

Program Name Specification Record(s)

Program name specification (*) records may follow the *A request. An * record specifies the next starting logical sector address for a particular program name and/or the format of the output. If there are no * records, the next starting logical sector address is assumed to be the next sector; all data is output in binary format. Only program names which define new sector addresses and/or output format need be specified. The format of the program name specification record is:

*,pgmnam,ssa,o,p

Where:

pgmnam is the name of the program, represented as six alphanumeric characters (positions 3-8), left justified with blank fill. This name must agree with the program name in the NAM record of the corresponding relocatable binary program (see MSOS 5 reference manual).

ssa is the next starting logical sector address, where the program is to be written. It is represented as three hexadecimal digits (positions 10-12). If blank or zero, the starting logical sector address is the next sector.

o is the output format identifier, represented as one character (position 14). If the character is an alphabetic D, the output is in deadstart format to load/execute micro memory; if blank or not a D, the output is in binary format. This field and the p field (described below) remain in force for subsequent binary programs until the next * record matches a program name.

p is the starting micro page number, represented as one hexadecimal digit (position 16). This field has meaning only if deadstart format has been selected (see above). It will cause the deadstart output to start loading and to begin execution at the first micro instruction of the specified micro page. If blank, micro page zero is assumed.

*T, Program Name Specification Termination

A termination record must follow any program name specification records. This record must be present even if there are no * records. The format of the termination record is:

*T

Where:

*T is in positions 1 and 2. Note that neither the * or *T records are listed.

One or more relocatable binary programs must follow the program name specification termination (*T) record. The program names of the relocatable binary programs must be in the same corresponding order as the names which were specified on the program name records in the *A request.

The following information for each binary program input is output to the standard comment device:

Position	Description
2	*, if deadstart output specified; $_\wedge$, if binary output specified.
3-8	Program name of relocatable binary
11-14	Program length in 32-bit hexadecimal format
15	*, if a next starting sector address specified
16-18	Logical starting sector address of program
21-66	Comment information or the NAM record of the program

*T, Relocatable Binary Program Termination

Following the relocatable binary programs must be a terminating record. The format of the terminating record is:

*T

Where:

*T is in positions 1 and 2.

When this record is read, the following message is output to the standard comment device:

Position	Description
1-2	*T
16-18	Logical next sector address (next sector following the end of the program just placed on diskette)

*B, BINARY REQUEST

This request permits previously absolutized binary programs to be input from the standard binary input device and output to a FDD diskette. Provision is made for starting at a particular logical sector address. The format of the binary request is:

*B,lu,ssa

Where:

lu is the logical unit of the diskette used for output, represented as three decimal digits (positions 4-6).

ssa is the starting logical sector address, where the block of binary programs is to be written. It is represented as three hexadecimal digits (positions 8-10). If blank or zero, the starting logical sector address is the next sector.

One or more absolutized binary program(s) follow the binary request (*B) record. (For details, see *P statement of LIBEDT described in the MSOS 5 reference manual.)

The following information for each binary program input is output on the standard comment device:

Position	Description
2-4	Sequential count of binary programs (for instance, 001 is output for program 1).
8-10	Logical starting sector address of program.

The absolutized binary programs must be followed by a termination record. The format of the termination record is:

*T

Where:

*T is in positions 1 and 2.

When the record is read, the following message is output on the standard comment device:

Position	Description
1-2	*T
8-10	Logical next sector address (next sector following the end of the program just placed on diskette)

*H, ASCII REQUEST

This permits ASCII records input from the standard binary input device and output to a diskette. The ASCII characters are to be packed two per 16-bit word. Provision is made for starting at a particular logical sector address, for ignoring spaces (blanks), and for including parity. The format of the ASCII request is:

*H,lu,ssa,q,p

Where:

lu is the logical unit of the diskette used for output, represented as three decimal digits (positions 4-6).

ssa is the starting logical sector address, where the block of ASCII records is to be written. It is represented as three hexadecimal digits (positions 8-10). If blank or zero, the starting logical sector address are the next sector.

q is the ignore spaces (blanks) option, represented as one character (position 12). If the character is an alphabetic I, spaces are ignored; if blank or not an I, spaces are treated as other characters and are output to the diskette.

p is the even parity option, represented as one character (position 14). If the character is an alphabetic E, each ASCII character is output with even parity (using the most significant eighth bit); if blank or not an E, no parity is included.

NOTE

If ASCII records are to be used for the deadstart operation, the even parity option must be selected.

One or more ASCII record(s) follow the ASCII request (*H) record. The following information is output to the standard comment device when the first ASCII record is input.

Position	Description
8-10	Logical starting sector address

The ASCII records must be followed by a termination record. The format of the termination record is:

*T

Where:

*T is in positions 1 and 2.

When the record is read, the following message is output to the standard comment device:

Position	Description
1-2	*T
8-10	Logical next sector address (next sector following the end of the program just placed on diskette)

COPY REQUEST

This request specifies that one or more sectors on a diskette is to be copied to a like number of sectors on another (or the same) diskette. After each sector is written, that sector is read and compared with the input sector to validate that the data is correct. The format of the request is:

*C,lu1,lu2,ss1,es1,ss2,num

Where:

lu1	is	the logical unit of the diskette holding the input (copied) data, represented as three decimal digits (positions 4-6).
lu2	is	the logical unit of the diskette holding the output data, represented as three decimal digits (positions 8-10). It may be the same logical unit as specified by lu1. If blank or zero, logical unit lu1 is used.
ss1	is	the starting sector address of the data to be read from logical unit lu1. It is represented as three hexadecimal digits (positions 12-14).
es1	is	the ending sector of the data to be read from logical unit lu1. It is represented as three hexadecimal digits (positions 16-18). If blank or zero, starting sector address ss1 is used.
ss2	is	the starting sector address where the data is to be written on logical unit lu2. It is represented as three hexadecimal digits (positions 20-22). If blank or zero, starting sector address ss1 is used. Note that the ending sector address where the data is to be written on logical unit lu2 is implied: (ss2 + es1 - ss1).
num	is	the number of times the copy operation is to be repeated. It is represented as three hexadecimal digits (positions 24-26). If blank or zero, the number of times is one. This field is provided for diagnostic or performance testing.

NOTE

Write enable causes both diskettes to be write enabled. Be sure that the unit reverse switch is in the position desired so that the copy proceeds in the direction desired and not in the opposite direction.

*V, VERIFY REQUEST

This request causes one or more sectors on a diskette to be compared (verified) with a like number of sectors on another (or the same) diskette. The format is:

*V,lu1,lu2,ss1,es1,ss2,num

Where:

lu1	is	the logical unit of the first diskette, represented as three decimal digits (positions 4-6).
lu2	is	the logical unit of the second diskette, represented as three decimal digits (positions 8-10). It may be the same logical unit as specified by lu1. If blank or zero, logical unit lu1 is used.
ss1	is	the starting sector address of logical unit lu1 to be verified, represented as three hexadecimal digits (positions 12-14).
es1	is	the ending sector address of logical unit lu1 to be verified, represented as three hexadecimal digits (positions 16-18). If blank or zero, starting sector address ss1 is used.
ss2	is	the starting sector address of logical unit lu2 to be verified, represented as three hexadecimal digits (positions 20-22). If blank or zero, starting sector address ss1 is used. Note that the ending sector address of logical unit lu2 is implied: (ss2 + es1 - ss1).
num	is	the number of times the verify operation is to be repeated. It is represented as three hexadecimal digits (positions 24-26).

If blank or zero, the number of times is one. This field is provided for diagnostic or performance testing.

*F, DEFINE INITIALIZE FORMAT REQUEST

This request specifies the diskette format to be used. It remains in effect until changed by a subsequent *F request. Current supported formats are:

CDC format -	96 (60_{16}) 16-bit words per sector and 19 (13_{16}) sectors per track.
IBM format -	64 (40_{16}) 16-bit words per sector and 26 ($1A_{16}$) sectors per track.

This request does not cause any FDD I/O; it merely specifies the format to the FDUTIL program. The default condition causes CDC format to be used. The format of the request is:

*F,wps,spt

Where:

wps is the number of words per sector, represented as three hexadecimal digits (positions 4-6). If blank or zero, the default is 60_{16} (CDC format).

spt is the number of sectors per track, represented as three hexadecimal digits (positions 8-10). If blank or zero, the default is 13_{16} (CDC format).

*S, SET OPERATOR INTERVENTION REQUEST

This request specifies that before each subsequent request, the FDUTIL pauses (waits) until the operator indicates that the next request is to be processed. The format of the request is:

*S

Note that the operator intervention option is in effect before the first request.

*R, RESET OPERATOR INTERVENTION REQUEST

This request specifies that any previous *S request be ignored; that is, that there be no operator intervention between requests. The format of the request is:

*R

*T, TERMINATE FDUTIL REQUEST

This request specifies that the FDUTIL be terminated. Control is returned to the operating system. The format of the request is:

*Z

FDUTIL ERROR PROCESSING

When FDUTIL detects an error, the following message is output to the standard comment device:

FDD UTILITY PROGRAM, ERROR xxxx, RESTART OPERATION

Where:

xxxx is an error code. The error codes and their meanings are listed in table I-1.

After the error message is output, the FDUTIL is automatically restarted. This causes operator intervention, so that the error can be corrected and processing can continue.

Table I-1 describes the FDUTIL error codes. The characters in column 2 describe the error codes. The characters have the following meaning:

-	An incorrect user record.
*	The resources of the FDUTIL program and/or computer are not sufficient to execute.
+	A possible irrecoverable hardware problem.

OPERATOR INTERVENTION PROCEDURE

If no *R requests have been input, or an *S request or an error followed the last *R request, the FDUTIL program pauses before each request until the operator indicates that the next request is to be processed. The purpose of the pause may be to:

- Remove a diskette and insert another one
- Ready a FFD drive (that is, diskette inserted and drive door closed)
- Change the FDD switches (as in FDUTIL execution)
- Readjust the input records after an error

Whenever a pause occurs, FDUTIL outputs messages to the standard comment device:

READY FDD(S), THEN PRESS CARRIAGE RETURN

After the operator performs the necessary function, he presses the carriage return key.

FDD BAD TRACK INFORMATION

Bad tracks are detected during initialization. As explained in the FDD reference manual, initialization is a DMA track write of approximately 5.4K words containing sync, track, sector, address CRC, data, data CRC, and spacing (along with closing) information. After the write, the track is read and the data is compared; status checks are also performed to verify that the track was initialized properly. Retries are done if errors occur.

If an error is still present after the required number of retries are done, the track is declared bad (defective) and the following message is printed on the standard comment device:

PHYSICAL TRACK xxxx HAS BEEN DESIGNATED A BAD TRACK

Where:

xxxx is physical track number (0-76).

Note that the hardware allows two bad tracks and that track zero must be good.

TABLE I-1. FDUTIL ERROR CODES

Code	Type	Meaning
0110	-	Illegal control record. Position 1 of the request record does not contain an asterisk, or position 2 does not contain a legal character (A, B, C, F, H, R, S, V, or Z).
0120	-	Illegal start or end address. The ending sector address is less than the starting sector address on an *C or *V record.
0130	-	Illegal sector address. The computer last sector address to be written (*C) or compared (*V) is greater than the maximum allowable sector address.
0140	-	Illegal *F request. The specific number of words/sector and/or sector/track is incorrect.
0210	-	Illegal record after an *A record. Position 1 does not contain an asterisk, or position 2 does not contain a comma (program name specification) or T (terminate).
0220	*	Too many program names specified. More than 20 program names have been specified. To increase the number of program specification names, FDUTIL would need to be reconfigured.
0230	-	Illegal record after program name specifications. Record is not a relocatable binary record or an *T record.
0240	-	No binary program entered. An *T record (terminate) was encountered without any relocatable binary program being loaded.
0250	-	Program specification error. One or more of the program specification names are not relocatable binary programs.
0260	-	Illegal NAM record. NAM record encountered was not the first record of a relocatable binary program.
0270	-	Illegal relocatable binary record. An undefined or illegal (BZS or EXT) relocatable binary record has been encountered.
0280	-	Illegal first record of relocatable binary program. The first record of a relocatable binary program was not a NAM record; instead it was an ENT, XFR, or RBD record.
0290	-	No end byte encountered. No end byte encountered on last relocation byte of a RBD record.
0299	*	Program size is too large. The size of the program being loaded (plus the FDUTIL program) is too large to fit in the program memory area. To load such a program, the operating system must be rebuilt to sufficiently increase the program memory area.
0510	*	Not enough memory to initialize. The area needed to properly initialize a diskette (plus the FDUTIL program) is too large to fit in the program memory area. To initialize, the operating system must be rebuilt to sufficently increase the program memory area.
0520	+	Fatal FDD error while initializing. A fatal error has occurred on the FDD. Make sure the FDD unit is ready (diskette inserted and door closed) and the switches are set properly (write enabled, initialize enabled, and unit reverse). If all these actions have been done, retry with another diskette and/or request maintenance support.

TABLE I.1. FDUTIL ERROR CODES (Contd)

Code	Type	Meaning
0530	+	Track zero is bad. Track zero was detected to be bad while initializing. Discard diskette and retry with another diskette and/or request maintenance support.
0540	+	More than two bad tracks. More than two bad tracks have been detected while initializing. Discard diskette and retry with another diskette.
0550	+	Initialized data not read correctly. The written track of initialized data was not read correctly, but the hardware did not detect an error. Retry and/or request maintenance support.
0610	-	Illegal sector address. An attempt to write (via an *A, *B, or *H request) beyond the maximum allowable sector address. Move the program to a lower sector address or place on another diskette.
0620	+	Fatal FDD input/output error. A fatal FDD input/output error has occurred. Make sure that the FDD unit is ready (diskette inserted and door closed) and the switches are set properly (viz., write enabled, initialize enabled, and unit reverse). If the above has been done, retry with another diskette and/or request maintenance support.
0630	+	Written data not read correctly. The written data, whrn read, did not compare exactly. Retry and/or request maintenance support.
0710	-	Illegal FDD logical unit. The specified logical unit is not a FDD.
0810	-	Illegal parameter. One of the parameters of the last read request record is illegal. For example, the sector address may be larger than the maximum allowable sector address.
0910	-	Illegal hexadecimal digit. One of the hexadecimal parameters of the last read request record is not a hexadecimal digit.
1010	-	Illegal diskette format. The format (IBM or CDC) of a diskette to be read or written does not agree with the last *F request record (if no *F record, IBM format is assumed). This error should only occur if a diskette is inserted to be read or written without first being initialized by FDUTIL program.
1110	-	Error in attempt to read input from standard input device.

FDD OUTPUT DATA FORMAT

Data can be stored onto the FDD diskette in four ways: initialization, binary, ASCII, or deadstart.

INITIALIZATION DATA

Each diskette must be properly initialized before it can be used. When it is initialized, data words for all tracks except zero are set to $E5E5_{16}$ (two EBCDIC alphabetic Vs) per IBM standard. For an *I request, all data words in track zero except the first sector are set to the same $E5E5_{16}$ value; for the first sector all data words are set to zero except the bad track words.

BINARY DATA

Binary information is output for any request that causes a diskette binary write (*A, *B, or *C), unless *A specifies deadstart format. Binary information is stored as is, without any extra information.

Used words in the last sector are filled with zeros.

ASCII DATA

ASCII information is output as a result of the *H request. Actually, there is no difference between binary and ASCII except by the user's interpretation of the data.

Spaces are ignored in the text (due to I option). Unused words in the last sector are filled with spaces.

DEADSTART DATA

Deadstart information is output by either the *A request with the deadstart option selected or the *H request with the even parity option selected. Deadstart information is ASCII information for serial loading via the CYBER 18 computer. This method can be used to load and execute programs in deadstart format from a diskette.

For the *H request, the ASCII characters are stored two characters (each with even parity) without any extra information.

For the *A (deadstart option selected) request, the program data words are converted to ASCII and stored with a terminal ASCII colon (for each program word) two characters (each with even parity) per FDD word. This *A format assumes that the program(s) are to be loaded into micro memory. Control information is added to allow the program(s) to be loading into the starting page and each successive half page. Further control information is added at the beginning and end to properly set up the machine's modes and to start program execution. If the program is being loaded on the initial deadstart sector (track 1, sector 1), control information is appended to clear all registers and files.

FDUTIL EXECUTION

The procedures for executing FDUTIL on RTOS and MSOS differ.

RTOS EXECUTION WITH FDUTIL RESIDENT

The program is executed as follows:

1. The operator manually interrupts RTOS (type: CONTROL G at keyboard)
2. RTOS prints: MI
3. The operator types: *F (cr)
4. FDUTIL begins execution

To load the RTOS/FDUTIL system, the operator:

1. Places RTOS/FDUTIL system in the proper deadstart device (for instance, diskette into an FDD) and readies the device.
2. Presses MASTER CLEAR or presses/types ESC to master clear the device.
3. Presses DEADSTART at the computer control panel.

RTOS/FDUTIL system loads and prints the system startup message:

READY FDD(S), THEN PRESS CARRIAGE RETURN

The operator then inserts his diskette(s), closes the FDD door(s), and sets the proper switches:

1. He sets write enabled if writes and/or initialization are to be done.
2. He sets initialize enabled if initialization is to be done.
3. He sets unit reverse if the unit reverse feature is desired.

The input request records are then placed in the standard input device and that device is readied. Then the operator presses carriage return on the standard comment device.

The FDUTIL reads input requests until terminated.

RTOS EXECUTION WITH FDUTIL NON-RESIDENT

The program is executed as follows:

1. Manually interrupt RTOS (type: CONTROL G at the keyboard)
2. RTOS prints: MI
3. Place an absolutized binary copy of FDUTIL in the standard binary input device and ready that device.
4. Type: *L (cr)
5. RTOS loads the FDUTIL program and prints: J
6. Type: *X (cr)
7. FDUTIL begins execution.

The operator then performs the procedure to load the RTOS system, which is identical to the one given above.

MSOS EXECUTION

The program is executed as follows:

1. The operator places the following records before the relocatable binaries of the FDUTIL program:

   *JOB
   *L

2. The operator places the following records after the relocatable binaries of the FDUTIL program:

   *T
   *X

3. The operator places the augmented FDUTIL records in the standard input device and readies the device.

4. The operator manually interrupts (types: CONTROL G at the keyboard)

5. MSOS prints: MI

6. Type: *BATCH (cr)

7. MSOS reads the control records and relocatable binaries, absolutizes them, and executes FDUTIL.

Note that the FDUTIL can also be placed in the program library and executed as a program library program using the job processor (*J) and *FDUTIL (see MSOS reference manual for details of executing program library program).

To load MSOS:

1. The operator master clears the system (either by pressing MASTER CLEAR at the CYBER control panel or typing ESC?).

2. The operator presses AUTOLOAD at the control panel.

3. The operator types: ESC K31000800:

4. The operator types: I@

5. MSOS loads and prints the system startup messages.

6. The operator types: ESC J28@ to protect the system.

7. System responds with the date/time request

8. The operator types the date/time as requested

9. MSOS system is now operational.

FDUTIL DEADSTART CONTROL INFORMATION

FDD DEADSTART BOOTSTRAP

The following FDD bootstrap reads a binary program from unit 0 into macro memory and executes it. The binary program can be output to the diskette using *A or *B requests. The bootstrap can be output to the diskette using the *H request.

For an IBM formatted diskette the command is:

```
*H,lu,01A,I,E
```

For a CDC formatted diskette, the command is:

```
*H,lu,013,I,E
```

Where lu is the logical unit of the diskette.

The FFD bootstrap requires two sectors of space.

FDD MICRO DEADSTART CONTROL INFORMATION

If the FDD is positioned at the deadstart position (track 1, sector 1 = logical sector address $1A_{16}$ for IBM format, 13_{16} for CDC format) and an *A request is executed with the deadstart option selected, deadstart control information is appended to clear all registers and files. Control information is added even if not at the deadstart position. The information is divided into three parts: header, half page, and trailer deadstart control; operation information is shown in figure I-1.

Header Deadstart Control

J	Allow enough spaces to let possible last auto-display to finish. (This and successive data, up to the single J control, is only appended at the deadstart position.)
J48G J10G K4488D000G L0000G K0G	Set console suppress, set micro mode, clear A and Q.
J03G J11G L0000G K0G J05G J12G	Clear X and P.
L0000G K0G J02G L00G J06G L000	Clear F, I, and K.
0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G	Clear register file 1.
0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G	
0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G	
0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G	
0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G	
0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G	
0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G	
0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G	
0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G	
0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G	
0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G	
0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G	
0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G	
0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G	
0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G	
0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G	
J01G L00G J00G L000	Clear N.
0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G	Clear register file 2.
0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G	
0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G	
0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G	
0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G	
0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G	
0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G	
0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G0G	
J1EG K0000G	Clear GR13.
J	Allow enough spaces to let possible last auto-display to finish. (This and successive data is used by all deadstart formatted programs.)
J48G J10G K28089000G K0040G L00G	Set console suppress, set micro mode, enable micro memory write, enable halt mode in SM1, and clear K.

Half Page Deadstart Control

J01G LxxG J0CG L	Set N to next half-page (xx).
XXXXXXXX:	Note: Next half-page of micro program follows (each instruction being eight ASCII hexadecimal digits, with even parity, followed by a colon). There are 256 such instructions.
•	
•	
•	

Trailer Deadstart Control

J1AG K0000G J15G K80003x00G	Clear SM2 and deadstart signal and set MIR to jump to the first upper instruction of first page (x).

Figure I-1. Header, Half Page, and Trailer Deadstart Control Operation

INDEX

COMMENT SHEET

MANUAL TITLE	CONTROL DATA® Software Peripheral Drivers Reference Manual
PUBLICATION NO.	96769390
REVISION	D
FROM NAME:	
BUSINESS ADDRESS:	

COMMENTS: This form is not intended to be used as an order blank. Your evaluation of this manual will be welcomed by Control Data Corporation. Any errors, suggested additions or deletions, or general comments may be made below. Please include page number to which your comment applies.

CUT ALONG LINE

STAPLE

STAPLE

FOLD

FIRST CLASS
PERMIT NO. 333

LA JOLLA. CA.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION
PUBLICATIONS AND GRAPHICS DIVISION
4455 EASTGATE MALL
LA JOLLA, CALIFORNIA 92037

CUT ALONG LINE

FOLD

STAPLE

STAPLE