96769390

CONTROL DATA CORPORATION

SOFTWARE PERIPHERAL DRIVERS VERSION 1.4C REFERENCE MANUAL

CDC® CYBER 18 OPERATING SYSTEMS:

MSOS

ITOS

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, storage
(9/76)	module disk (SMD), flexible disk, 1827 Line Printer, low-cost tape transport (LCTT), and cassette tape.
C	Revised to incorporate PSRs 3253, 4254, and include 1827-2 Line Printer Driver, 1811-2 Conversational Display Terminal
(10/77)	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.
(1/78)	
E	Manual revised to reflect ITOS 2.0 release.
(10/78)	
F	Manual revised to incorporate new products and improve format.
(1/81)	
Publication No. 96769390	

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

Address comments concerning this manual to:

Control Data Corporation
Publications and Graphics Division
4455 Eastgate Mall
La Jolla, 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
Front Cover	-								
Title Page	-								
ii	F								
iii/iv	F								
v/vi	F								
vii	F								
viii	F								
1-1 thru 1-5	F								
2-1 thru 2-6	F								
3-1 thru 3-6	F								
4-1 thru 4-13	F								
5-1 thru 5-6	F								
6-1 thru 6-7	F								
7-1	F								
8-1	F								
8-2	F								
9-1	F								
10-1	F								
A-1 thru A-5	F								
B-1	F								
C-1 thru C-16	F								
D-1 thru D-4	F								
E-1 thru E-3	F								
F-1 thru F-7	F								
Index-1 thru -3	F								
Comment Sheet	F								
Mailer	-								
Back Cover	-								

PREFACE

This manual provides a detailed description of the software I/O drivers for Control Data® CYBER 18 Series peripherals. These I/O drivers are utilized by the Mass Storage Operating System (MSOS) and the Interactive Terminal-Oriented System (ITOS).

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

The following devices are supported by standard drivers common to MSOS and ITOS:

1867-10/20/40 1868-1	Storage module drive
1866-14	Cartridge disk drive
1827-30/32/60/90	Band/drum printer
1829-30/60	Card reader
1860-1/2/3/4	NRZI magnetic tape
1860-5/6	Phase-encoded magnetic tape

The following devices are supported by standard drivers subject to restrictions as noted:

1865-1/2	Flexible disk drive (Separate standard drivers are available for MSOS and ITOS.)
1843-2	8-channel CLA (Separate standard drivers are available for MSOS and ITOS.)
1843-1	2-channel CLA (Supported only on ITOS COMM18 Systems.)
1811-1/2	Console display terminal (Supported via the 1843-2 8-channel CLA on ITOS and MSOS. In addition, a driver for a single 1811-1 or -2 is supported by MSOS only.)
1827-7	Matrix printer (Supported via the 1843-2 8-channel CLA on ITOS only.)

For additional information relative to MSOS and ITOS, refer to the following documents:

Publication	Publication Number
Mass Storage Operating System Version 5 Reference Manual	96769400
MSOS 5 Diagnostic Handbook	96769450
MSOS 5 Installation Handbook	96769410
Interactive Terminal-Oriented System (ITOS) Version 2 Reference Manual	96769240
ITOS Version 2 Installation Handbook	60475200

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

CONTENTS

1. INTRODUCTION 1-1

Queueing of I/O Requests 1-1
READ/FREAD/WRITE/FWRITE Requests 1-1
MOTION 1-4

2. SYSTEM INTERFACES FOR I/O DRIVERS 2-1

Find-Next-Request 2-1
Device Shared 2-1
Device Not Shared 2-1
Complete Request 2-1
Set Error Flag 2-1
Diagnostic Timer Module 2-2
Alternate Device Handler 2-2
Mass-Storage Resident Drivers 2-3
Interrupt Response Routines 2-4
Engineering File Logging 2-4
A/Q Channel Allocation 2-4
Auto-Data Transfer (ADT) 2-4
ADT Table for Single A/Q Device 2-5
ADT Table for Multiple A/Q Devices 2-5
ADT Table for Clock 2-6
ADT Table for Single or Multiple MO5 Devices 2-6

3. DRIVER DESCRIPTION AND DATA FORMATS FOR KEYBOARD AND TERMINAL DRIVERS 3-1

MSOS 5 1811/722 Console Display 3-1
READ 3-1
FREAD 3-1
WRITE 3-1
FWRITE 3-1
MOTION 3-1
ITOS 2/COMM 18 1843-1 Dual Channel Communication Line Adapter 3-1
WRITE/FWRITE 3-2
READ/FREAD 3-2
MOTION 3-2
Mode Bit = 1 3-2
Mode Bit = 0 3-2
Data Reception 3-3
DC CLA Channel Status Bit Definitions 3-3
DC CLA Channel Function Bit Definitions 3-3
DC CLA Input Data Transfer Bit Definitions 3-4
DC CLA Output Data Transfer Bit Definitions 3-4
ITOS 2 1843-2 Terminal Driver 3-4
Request Format 3-4
MOTION 3-5
READ/FREAD 3-5
WRITE 3-5
FWRITE 3-5
Connect 3-5
Disconnect 3-5
WRITE-READ 3-5
Cursor Positioning 3-5
MSOS 5 1843-2 Terminal Driver 3-5
Request Format 3-5
MOTION 3-6
READ/FREAD 3-6
WRITE 3-6
FWRITE 3-6
Connect 3-6
Disconnect 3-6
WRITE-READ 3-6

4. DRIVER DESCRIPTIONS AND DATA FORMATS FOR MASS MEMORY DRIVERS 4-1

1866-14 Cartridge Disk Driver (CDD) 4-1
Data Transfer Request Formats 4-1
Disk Driver Requests 4-2
READ/WRITE 4-2
FREAD/FWRITE 4-2
MOTION 4-2
Status and Error Handling 4-2
1867-10/20/40 and 1868-1 Storage Module Drive Driver (SMD) 4-3
Request Format 4-3
Driver Description 4-3
Error Recovery 4-4
Control Unit Connection Error 4-4
Drive Connection/Seek Error 4-4
Error Correction Code Error 4-5
Data Transfer Error (Read) 4-5
Error Codes 4-6
Disk Pack Initialization Format 4-7
Diagnostic Features (Optional) 4-7
ITOS 2 Flexible Disk Driver 4-7
Data Formats 4-8
READ/WRITE 4-8
FREAD/FWRITE 4-8
MOTION 4-8
Error Recovery 4-8
Accessing Flexible Disk 4-9
MSOS 5 Flexible Disk Driver 4-9
Data Formats 4-10
READ/WRITE 4-10
FREAD/FWRITE 4-10
MOTION 4-11
Error Recovery 4-11
Special Features 4-13
Flexible Disk Utility 4-13

5. DRIVER DESCRIPTIONS AND DATA FORMATS FOR LINE PRINTER AND CARD READER DRIVERS 5-1

Line Printer Drivers 5-1
WRITE/FWRITE 5-1
MOTION 5-1
Character Editing 5-1
Error Conditions 5-2
1827-30/60/90 Line Printer Driver 5-2
WRITE/FWRITE/MOTION 5-2
Character Editing 5-2
Status and Error Handling 5-2

ITOS 2 1827-7 Line Printer Driver 5-3
WRITE/FWRITE/MOTION 5-3
Character Editing 5-3
Status and Error Handling 5-3
1829-30/60 Card Reader Driver 5-3
Data Formats 5-3
Read Binary 5-3
Read ASCII 5-4
FREAD Binary 5-4
FREAD ASCII 5-4
EOF Processing and Motion Requests 5-4
Status and Error Handling 5-4

6. DRIVER DESCRIPTIONS AND DATA FORMATS FOR MAGNETIC TAPE DRIVERS 6-1

Magnetic Tape Drivers 6-1
Data Formatting 6-1
Formatted Requests 6-1
Unformatted Requests 6-1
Motion Requests 6-1
Error Recovery 6-1
Software Requirement 6-1
Switched Mode 6-2
Successful Recovery 6-2
Unsuccessful Recovery 6-2
1860-3/4 Low-Cost Tape Transport 6-2
Data and Record Format 6-2
Data Format 6-2
Record Constraints 6-3
READ/WRITE/FREAD/FWRITE 6-3
MOTION 6-4
Error Recovery 6-4
1860-5/6 Low-Cost Tape Transport/ Formatter Driver (Dual Mode) 6-5
Data Format 6-5
Record Constraints 6-5
READ/WRITE/FREAD/FWRITE 6-6
MOTION 6-6
Error Recovery 6-6

7. OPERATING SYSTEM DRIVERS 7-1

Dummy Driver 7-1
Pseudo-Tape Driver 7-1
Cosy Driver 7-1
Software Buffer Driver 7-1

8. SYSTEM INITIALIZER DRIVERS 8-1

Driver Operation 8-1
Input Drivers 8-1
Mass-Memory Drivers 8-2
Driver Errors 8-2

9. DISK-TO-TAPE UTILITY DRIVERS 9-1

10. SYSTEM CHECKOUT DRIVERS 10-1

B18331 (SMD) and B18334 (CDD) Constraints 10-1

APPENDIXES

A Glossary A-1
B Standard Equipment/Interrupt Assignments for CYBER 18 Series Equipment B-1
C Logical Unit and Physical Device Tables C-1
D ASCII Conversion Table D-1
E Driver Coding Structure E-1
F MSOS FDUTIL Formatting Flexible Disk F-1

INDEX

FIGURES

2-1 ADT Table For Multiple A/Q Devices 2-5
2-2 ADT Table for Single or Multiple MO5 Devices 2-6
4-1 Module Drive Driver Maximum Hardware Configuration 4-4
4-2 Head Positioning 4-5
5-1 Binary Format Record Cards 5-5

TABLES

4-1 Data Formats 4-8
4-2 Flexible Disk Commands 4-12
6-1 Use of Buffers for READ, FREAD, WRITE, FWRITE Requests 6-3
8-1 Hardware Device Drivers 8-1

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), the timeout 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 processpr returns to the caller.

Motion requests are handled in a similar way.

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 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 memory. 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:

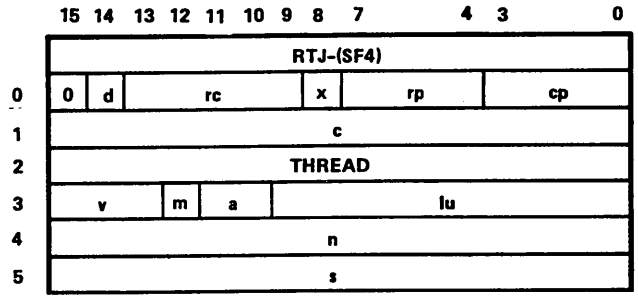
```
READ,rc=1
FREAD,rc=4
WRITE,rc=2
FWRITE,rc=6
        } lu,c,s,n,m,rp,cp,a,x,d
```

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 calling sequence generated by the macro is as follows:



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 of 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 memory 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 locaton 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.

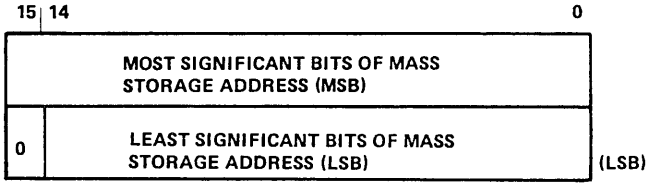
x is the relative/indirect indicator; this parameter affects parameters c, s, and n as shown here. Because of the wrap-around feature, computed addresses may be before or after the parameter list.

(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 memory location that contains the starting address. If the request is on mass memory, the mass memory address follows the memory 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) the memory 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	
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:



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

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

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. 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. Available motion request code meanings vary with the device.

dy is the density parameter.

- 0 No change
- 1 800 bpi
- 2 556 bpi
- 3 200 bpi

External rejects result when an illegal density selection is attempted.†

- 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.

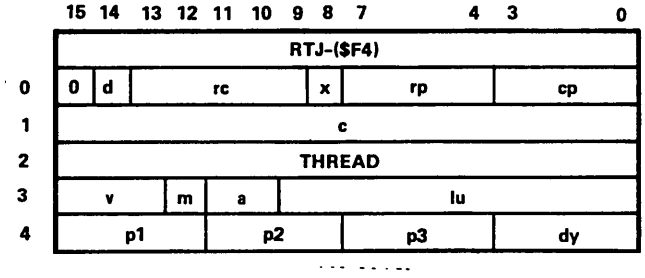
d is set to

- 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:



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.

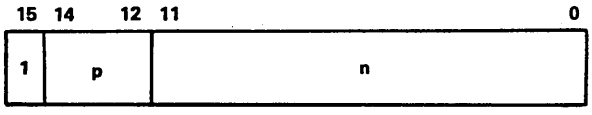
p is the motion code.

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:



Where 1 indicates that the request can be repeated.

† In addition to the attempt to set a density that is not legal for a unit (for example, 200/556 bpi on an 1860), the drivers do not allow a density change if the unit is not at load point.

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

BSR* lu,a,n,c,p (Code 1)

EOF* lu,a,n,c,p (Code 2)

REW* lu,a,n,c,p (Code 3)

UNL* lu,a,n,c,p (Code 4)

ADF* lu,a,n,c,p (Code 5)

BSF* lu,a,n,c,p (Code 6)

ADR* lu,a,n,c,p (Code 7)

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 memory location.

```
EOF     FA16,I,,,0
```

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) routine 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 memory address of the physical device table entry in the I register.

DEVICE SHARED

The FNR routine 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; that is, the device priority completely supersedes the scheduling priority. Scheduling priority becomes operational only within the thread of requests when the logical unit is activiated. In other words, 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) routine is entered by an indirect return jump through $B6_{16}$ 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 routine 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 one second).

An input/output hang-up error occurs when a driver fails to get a completion interrrupt 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.

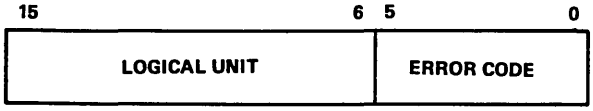
When a hang-up occurs, the diagnostic timer accesses the physical device table entry for that device to obtain the driver memory location to be executed in case of a hang-up. This location is executed by a SCHDLE request at the same priority level as the driver. Q contains the memory address of the physical 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 device, 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.
2. The controller hardware is cleared and an error word is set in the Q register.



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 alternate 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 is 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 only the first time the device is accessed after CD has been entered.

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.

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

MASS-STORAGE RESIDENT DRIVERS

Most drivers are capable of functioning as either main- or mass-memory resident drivers. The following device types are not considered to be good candidates for mass-storage residency.

- Teletypewriters/conversational display terminal keyboard drivers
- Mass storage devices
- 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 main memory-resident executive routine, MMEXEC. It is possible to load either one or two drivers in the buffer, depending on its size.

The main memory 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 memory 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 memory at the same time to complete the request for the COSY driver. This is also true when using print spoolers, batch input/output drivers, and (in some cases) magnetic tape simulator devices.

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 (that is, 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 memory and passes control to it. If the driver is not in memory, it is determined if there is sufficient memory 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 memory 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 memory 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 memory 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 controller. These routines usually 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 controller, 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 Q register (see alternate device handler). The call is:

```
RTJ+    LOG
```

A/Q CHANNEL ALLOCATION

The CYBER 18 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.

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.

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.

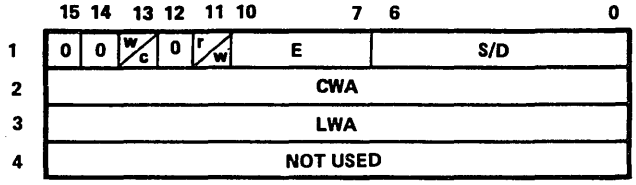
While the ADT operation is in progress, the emulator is executing macro instructions. However, after each macro instruction is executed, interrupts are checked. If the particular ADT micro interrupt has become the highest active interrupt, the next data word is input or output. After the interruption,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



The ADT table for a single A/Q device 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-FWA+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.)

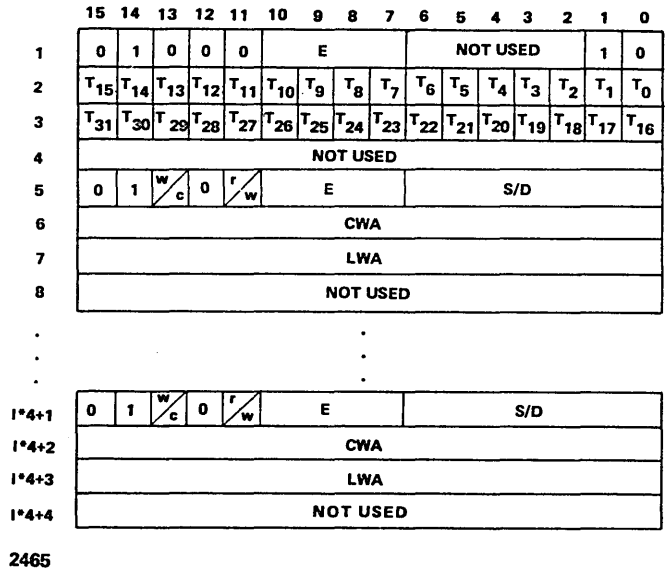
S/D is station/director bits of the device. Director bits should specify a data (not a status/function) transfer.

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 shown in figure 2-1:



2465

Figure 2-1. ADT Table For Multiple A/Q Devices

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 the micro interrupt.

Where:

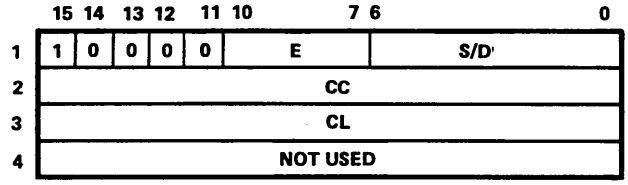
E is equipment number of the device. (It cannot conflict with any MO5 I/O port numbers.)

T_n 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 (for example, a CLF instruction).

Words 5 through 8 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 clock is as follows:



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 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 is shown in figure 2-2.

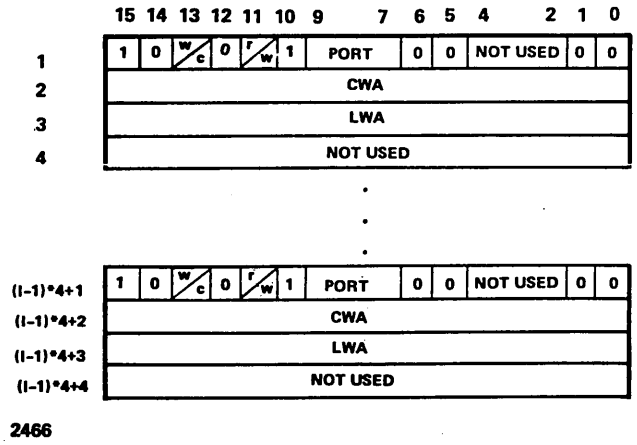


Figure 2-2. ADT Table for Single or Multiple MO5 Devices

Where:

w/c is same as for 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 (that is, 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+4, (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.

DRIVER DESCRIPTION AND DATA FORMATS FOR KEYBOARD AND TERMINAL DRIVERS

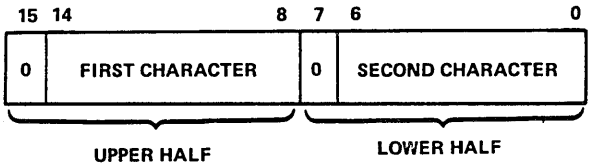
MSOS 5 1811/722 CONSOLE DISPLAY

Four types of requests, READ, WRITE, FREAD, and FWRITE, are honored from the keyboard. Each request specifies the memory 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 memory 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 memory location. The lower character is filled with a hexadecimal FF.



FREAD

Words in memory are filled, starting at the specified memory 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 memory 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 memory 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 (code 2) to the keyboard is honored by executing a top-of-form function. All other MOTION requests cause no action with normal completion of the request.

ITOS 2/COMM 18 1843-1 DUAL CHANNEL COMMUNICATION LINE ADAPTER

The driver supports simultaneous transmission/ reception on one or both of the channels of a DC CLA and multiple DC CLAs. Data transmission and reception is performed using the ADT mode of data transfer. Being a generalized synchronous driver, the transmission odd or even parity, data bit length, synchronization character code, and so forth, is specified by issuing an initialization function prior to performing data communications on the channel.

Each channel of each DC CLA is assigned a logical unit number.

Once the channel has been initialized, data blocks can be transmitted by issuing standard monitor WRITE/FWRITE requests to the appropriate logical unit. READ/FREAD monitor requests are illegal and are rejected by the driver.

MOTION requests are used to initialize and/or control channel functions.

All detected hardware errors are logged in the system engineering file without any messages on the console CRT (does not use the alternate device handler). Error status is passed to the applications program in request completion address in the V-field of the Q-register and in the monitor request for WRITE/FWRITE and magnetic tape motion requests only.

WRITE/FWRITE

Standard monitor WRITE/FWRITE requests are used to transmit data blocks. Both types of requests are handled identically. The mode bit is ignored.

NOTE

All leading sync characters, control characters, CRC, and PADs must be supplied in the user's data buffer. To ensure transmission of the complete data block, two extra PAD characters should be placed at the end of the user's data buffer. The data buffer is to be packed two characters (bytes) per word, the leftmost byte being transmitted first. The buffer must contain an even number of bytes since the length of the transmission block is given in number of words.

Once the transmission has been initiated, a timeout period of 10 seconds is set. This timeout period is cleared upon completion of the transmission. If an error or output timeout occurs during data transmission, a return is made to the completion address of the applications program with the error V-field set as follows:

Bit 15 = 1	Indicates an error has occurred
Bit 14 = 1	Indicates a short transfer
Bit 13 = 1	If carrier on during transmission
Bit 13 = 0	If carrier turned off during transmission

READ/FREAD

All READ/FREAD requests are rejected and return is made to the user's program completion address with the error V-field set to 100 binary.

MOTION

Standard MOTION requests are accepted by the driver. Channel initialization or control functions are govered by the mode bit (word 3, bit 12) of the request.

Mode Bit=1

This request is used to initialize the channel prior to data communications.

Word 4 of the request contains the absolute address of a six-word initialization table in the calling program of the following format:

Word	Contents
0	Transmitter control word
1	Receiver control word
2	Transmitter fill/receiver synchronous character
3	PAD character format
4	ADDR of user PGM for input character handling
5	Priority for schedule of user PGM address

NOTE

The transmitter fill character and receiver sync character are assumed by the driver to have the same format.

This function will cause the channel to be initialized as specified, data terminal ready will be set, and the channel will be placed in search sync mode waiting for data reception.

Mode Bit=0

These types of requests are used to control certain functions of the data channel. The type of control function performed is governed by the value of bits 14 to 12 of the motion type word (word 4) in the request. The other bits in this word are ignored. Only one channel function is handled per request.

Word	Action
0	Request is rejected
1	Request is rejected
2	Set data terminal READY/RESYNC - Data terminal ready status is set, all error conditions are cleared, and receiver is placed in search sync mode. This control function is used to reconnect the line after type 3 control (clear data terminal ready) or type 5 control (stop intput of characters).

Word	Action
3	Clear data terminal ready - I/O interrupts and data terminal ready status are cleared. This control function is used to disconnect a line on a switched network.
4	Release communications line - The channel is cleared, data terminal not ready, and user program address in PHYSTB is cleared. This frees the channel for use by another applications program.
5	Stop input of characters - The user program address in the PHYSTB is cleared, and search sync bit in channel function word is cleared stopping the reception of any further characters. The input of characters is automatically restored after a WRITE/FWRITE request or MOTION request type 2.
6	Request is rejected
7	Request is rejected

DATA RECEPTION

Once initialized, the channel is ready to receive input data. After transmitting a data block, the channel is automatically placed back into receive mode.

The driver waits until synchronization has been established and until all leading sync characters have been received before passing data to the user program.

Once synchronization has been established, the first non-sync character sets a timeout period of 10 seconds. This timeout period is cleared by the reception of a PAD character. Alternatively, any monitor request to the driver will clear the timeout condition. If an input timeout occurs, the channel is placed back into search sync mode to attempt to resynchronize on the next data block received. Bit 10 of PHYSTB word 9 (ESTAT1) is used as a flag to indicate an input timeout is in progress.

The first non-sync character and all following characters (including control, sync, CRC, and PADS) are passed to the user's program by scheduling the address at the priority specified in the initialization table. Passed to the user's program is the Q-register set as follows:

Bits 0 to 7 = Data character

Bits 8 to 14 = Logical unit of channel

NOTE

If the priority for the schedule of the user's address during channel initialization is zero, the user's program is entered by a jump at driver priority instead of the schedule. It is essential that the user's PGM give up control (jump to the dispatcher) as quickly as possible or lost data conditions will occur.

DC CLA Channel Status Bit Definitions

Obtained by input with Q-register, D = 001

Bit	Definition
0	Ready
1	Busy
2	Input data interrupt/data ready
3	Output data interrupt/data request
4	Receive error interrupt
5	Auto data transfer on input
6	Auto data transfer on output
7	SYNC1 match
8	Carrier on
9	Clear to send
10	Ring indicator
11	Data set ready
12	Data not available
13	Auto data transfer EOP (both channels)
14	Program protect
15	Test mode

DC CLA Channel Function Bit Definitions

Output to channel with Q-register, D = 001

Bit	Definition
0	Clear channel
1	Clear input interrupt
2	Clear output interrupt
3	Enable input interrupt
4	Enable output interrupt
5	Enable receive error interrupt
6	Set receive auto data transfer
7	Set transmit auto data transfer
8	Set data terminal ready
9	Set request to send
10	Not used
11	Clear auto data transfer EOP
12	Reset data not available

Bit	Definition
13	Reset receive error
14	Set search sync mode
15	Set test mode

DC CLA Input Data Transfer Bit Definitions

Obtained by input with Q-register, D = 000

Bits	Definition
0-7	Data character
8	Not used
9	Parity error
10-12	Not used
13	Lost data
14	Character request
15	Not used

DC CLA Output Data Transfer Bit Definitions

Output to channel with Q-register, D = 000

Bits	Definition
0-7	Data character
8-15	Not used

ITOS 2 1842-2 TERMINAL DRIVER

The ITOS executive provides an interface between the user program and the system communications terminal I/O driver.

All non-MSOS features of this interface are invisible to the user program, since the ITOS executive performs all reformatting prior to making the request.

The executive operates all terminals in an unsolicited input mode. This mode of operation requires that the executive establish a logical connection with the communications terminals prior to any driver requests. Once this connection is made, all input received on the port is placed in the specified input buffer, and the specified completion address scheduled by the driver when the input termination character is entered. The ITOS executive does not perform READ or FREAD requests. When this type of request is received by the protect processor from a user program, it is translated into a WRITE OR FWRITE request with no completion address specified. Following the entry of data by the terminal user, the unsolicited input location is scheduled by the driver, which forms the completion of the user program READ request.

REQUEST FORMAT

The MSOS monitor requests specified by the executive are standard, except for the following:

- Request mode bit A specifies the data element of the request. All requests are ASCII, either character- or word-oriented:

  A = 0 indicates that the request length specifies characters.

  A = 1 indicates that the request length specifies words.

- All communications terminals are defined as a single logical unit in MSOS. The individual communications line is specified by a logical port number contained in the message header.

- The Q-register contains the V-field and the logical port number at the completion of an I/O request.

- The monitor request is nine words in length: word 6 contains a control point indicator (bit 15 = 1), word 7 contains the address of a seven-word request buffer, and word 8 contains the control point of the user's request. The format of the seven-word request buffer is as follows:

Word	Bits	Definition
0	15-13	Reserved for future use
	12	Completion type 0 Request complete 1 Message moved from user program
	11	Port-active indicator
	10-0	Contain the logical port number of the communication line to be used. Set by the ITOS executive on WRITE or FWRITE; set by the driver on unsolicited input.
1	15-5	Contain the status set by the driver only if bit 15 of the request V-field is equal to one.
	15	Communications subsystem down
	14	Transmission error
	13	Spare
	12	Time-out error
	11	Illegal request
	10	Reserved
	9	Reserved
	8	Spare
	7	Spare

Word	Bits	Definition
	6	Spare
	5	Spare
	4-0	Contain the subrequest code 0 Normal mode 1 Logical connect 2 Logical disconnect 3 Write-read operation 4-31 Reserved for further use
2	15-0	Contain the secondary device status for use by the communications driver
3	15-0	Contain the termination code. Set by the driver on unsolicited input.
4	15-8	Cursor X position prior to data output. Set by the ITOS executive.
	7-0	Cursor Y position prior to data output. Set by the ITOS executive.
5	15-8	Cursor X position prior to data input. Set by the ITOS executive.
	7-0	Cursor Y position prior to data input. Set by the ITOS executive.
6	15-0	Contain the number of characters requested. The driver returns the actual number of entered characters upon read completion. Set by the ITOS executive.

MOTION

All MOTION requests are ignored.

READ/FREAD

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

FWRITE

The number of characters/words 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

The 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 resets the port from 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 722/752 CRT. Two methods are provided:

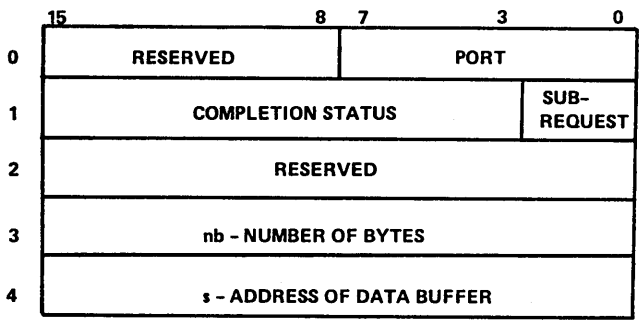
- A cursor positioning sequence ($1B_{16}$,31_{16},x,y) may be embedded in any write buffer.
- For a WRITE-READ request, word 2 of the extended request parameter block requests the cursor position that is to be used prior to receiving input data. A -1 ($FFFE_{16}$) in word 2 disables this feature.

MSOS 5 1843-2 TERMINAL DRIVER

The MSOS 5.0 Terminal Driver provides communication with up to 16 CRT terminals. Each terminal functions as a separate logical unit within the MSOS system.

REQUEST FORMAT

The request format is standard with the addition of a five-word extended request parameter block as shown below:



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/ Subrequest — Status as at the completion of the request

Bit	Definition
15	Subsystem down
14	Line disconnect
13	Line down
12	Terminal not connected
11	Spare
10	Request timeout
9	Illegal request
8	Parity error
7	Break received
6-4	Spare

Subrequest — 3-0 code as follows:

Bit	Definition
3	WRITE-READ
2	Logical disconnect
1	Logical connect
0	Normal mode

nb — Number of bytes requested to be read for WRITE-READ requests set by driver to actual number of characters input on completion of READ and unsolicited input 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 ignored.

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

The 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 resets the port from 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.

DRIVER DESCRIPTIONS AND DATA FORMATS FOR MASS MEMORY DRIVERS 4

1866-14 CARTRIDGE DISK DRIVER (CDD)

Data is transferred to and from disk mass-storage devices. The device driver performs formatted FREAD/FWRITE (sector addressing) and unformatted READ/WRITE (word address simulation) requests and processes motion requests that result in no operation. The mass-memory driver also detects 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 executed.

The hardware compare feature is optional for the CDD disk driver. 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 systems having more than one drive, overlapping seek operations are utilized for maximum data transfer efficiency. The software driver for the CDD initiates 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.

NOTE

Cartridge disk drives have one or two switches (depending on model) labeled WRITE PROTECT. When a switch is set, no data can be written on the selected (fixed or removable) disk platter. It is the operator's responsibility to place the switches 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 memory 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 memory and mass storage locations contain identical 16-bit images.

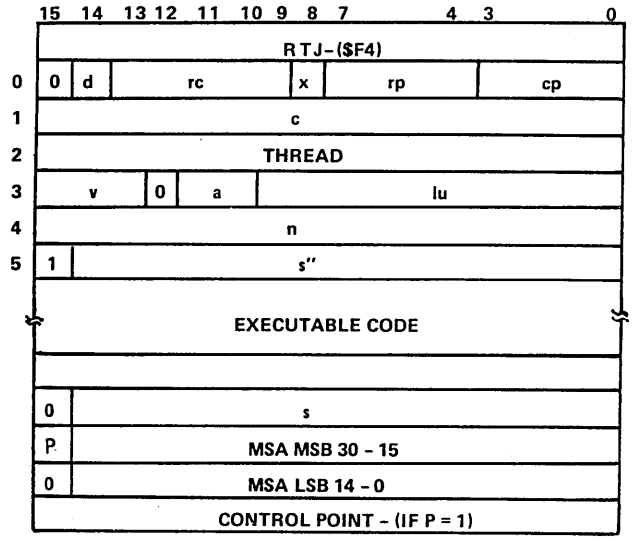
The first request format is consistent with normal requests (the x parameter is not set) by providing a seven-word format.

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 five after the request is made.

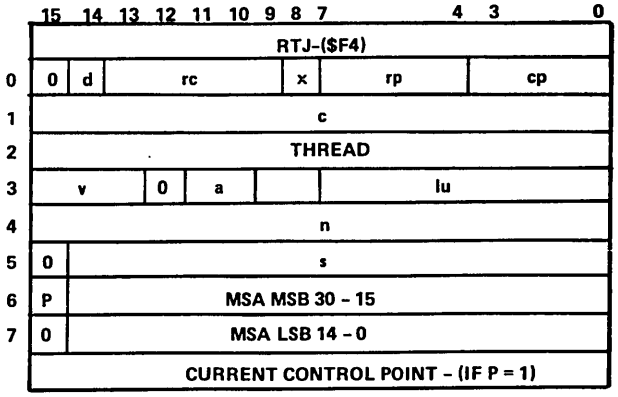
If P (bit 15 of the MSB word) is =1, current control point is part of the request; otherwise it is ignored.

The first request format is as follows.



The second request format adds two 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.



DISK DRIVER REQUESTS

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. The driver converts the word address to sector and word in the sector by dividing by 96.

READ - The READ request fills memory 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.

WRITE - The WRITE request transfers the requested number of words from memory 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

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

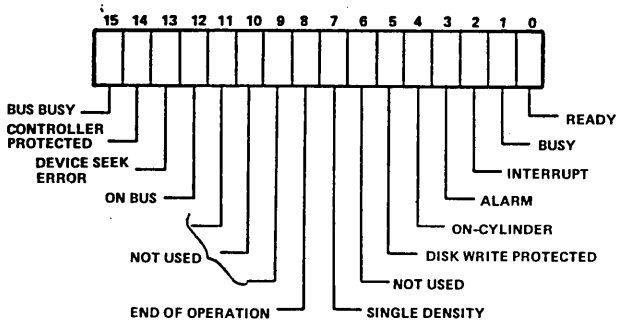
FWRITE - This request transfers the specified number of words from memory to disk. The starting disk address is interpreted by the driver as a sector address; the most significant bits must be zero. If a zero number of words is requested, one word is transferred. The remainder of a partially updated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

MOTION

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

STATUS AND ERROR HANDLING

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



The following errors are detected by the driver:

00 Timeout

01 Lost data

02 Incorrect device transfer or sector format - alarm

03 DMA device parity error

05 Internal reject

06 External reject

07 Compare

13 Write protected

14 Not ready

16 Controller seek error

17 Seek error

18 End of medium

19 DMA protect fault

20 Checkword error

49 Address error

50 Missing index sector pulse

84 Bus relinquished

85 Errors in current word address, bank status, true cylinder, and true sector

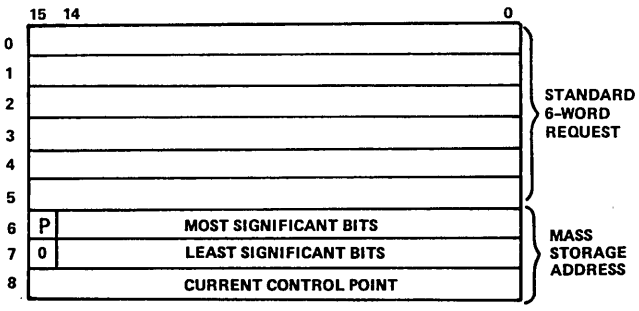
88 Data address error

1867-10/20/40 AND 1868-1 STORAGE MODULE DRIVE (SMD) DRIVER

This driver handles CYBER 18 data transfers to and from any mix of 25, 80, and 300 megabyte SMD and MMD hardware (up to 8 drives).

REQUEST FORMAT

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.

Current control point (CCP) indicates the segment of memory 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.

P = 1 indicates that current control point is part of the calling sequence.

P = 0 Word 8 is omitted from calling sequence.

The maximum buffer size for input and output disk transfers is 65K. If a write transfer extends across a cylinder boundary, the disk handler divides the request into two sections, but this operation is transparent to the using program.

DRIVER DESCRIPTION

The driver controls the module drive adapter and controller. That unit in turn controls up to eight modular 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 primarly 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 4-1.

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, the drivers of both CPUs use the table to gain and release control of the control unit.

A safeguard is provided to prevent one CPU from locking out the other CPU. 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.

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, that is, a MOTION command results in a no-operation. The driver itself develops a seek command 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.

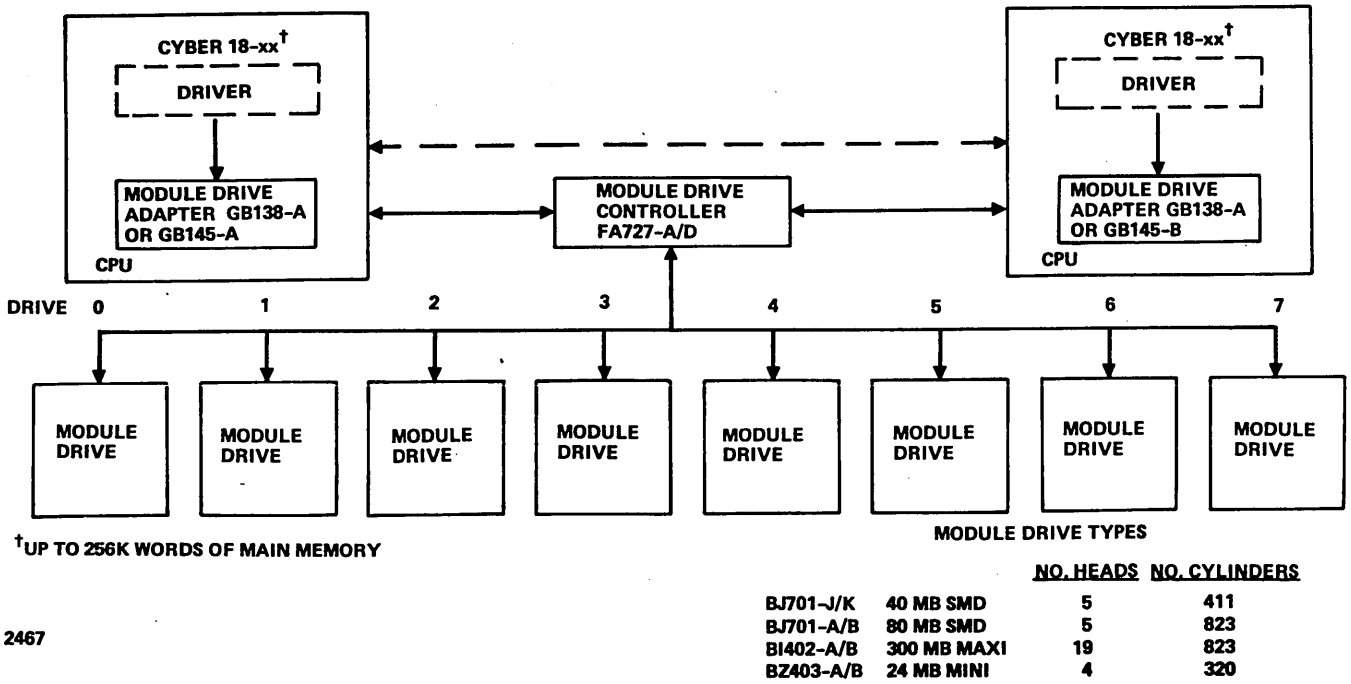


Figure 4-1. Module Drive Driver Maximum Hardware Configuration

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 irrecoverable 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

On all write operations a special code is generated in line with the data within the sector. The code can detect a head 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.

The pack formatter (SMD2F) contained in ODS level II, which is used to format packs for ITOS only, maps out bad spots on the media if they exist. By definition a bad spot is an area within a sector that exhibits a data error greater than 11 bits in length. The encounter of a bad spot during a READ/WRITE request will be transparent to the user. SMD2F will have set the bad sector bit in the address tag of the sector containing the bad spot. In addition to the setting of the bad sector bit, the address of a replacement sector is placed in the flag bytes of the address tag. Replacement sectors are made available on the last available cylinder of the media. For each bad sector two replacement sectors are used. The first replacement sector is used to retain the data that would normally reside at the bad sector. The second replacement sector is used to vector back to the bad sector +1 in the same manner as the bad sector vectored to the replacement sector. At each encounter of a sector that is marked with the bad sector bit, an alarm error interrupt will inform the driver that this sector is to be replaced.

CAUTION

The last two tracks of the disk pack are used for bad sector replacement and must not be used if bad sector replacement occurred during pack formatting.

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 4-2.

To counter this type of error, the driver automatically retries reading the data using numerous combinations of offset positions and strobing variations, 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.

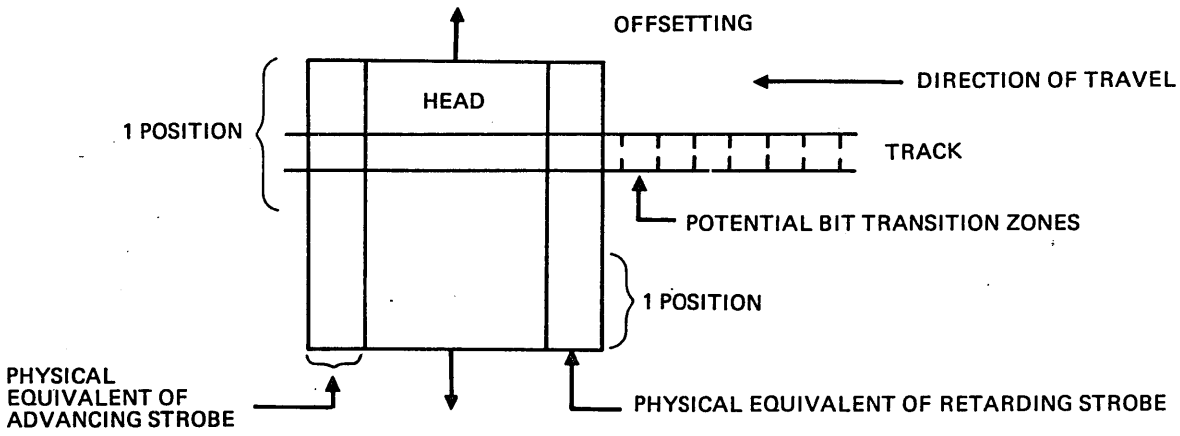


Figure 4-2. Head Positioning

Error Codes

The error codes for all errors are:

Code	Definition
0	I/O hang up (one-second data transfer timeout)
2	Alarm error (hardware)
5	Internal reject
6	External reject
10	Mis-seek
14	Unit not ready (disk drive)
17	Seek error
41	Unsuccessful request (data transfer error)
70	Connect error (control unit or disk drive)
71	Uncorrectable by error correction code (read or write)
82	Control unit error (includes write protect errors due to write protect switch on)
83	Mass storage address error (disk adapter addresses nonexistent main memory)

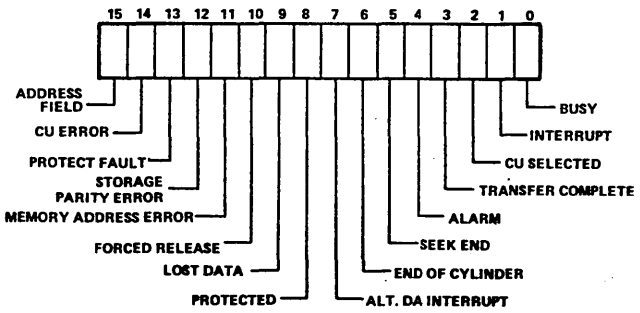
An error message of the following form is displayed when mass memory errors prevent further processing.

MM ERR ee LU=lu T=hhmm:ss S=ssss

Where:

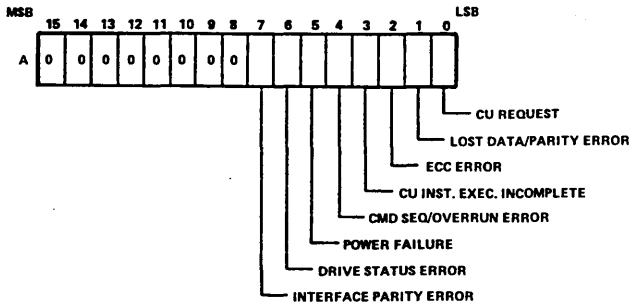
ee	is	error code
lu	is	logical unit
hhmm	is	hour and minute
ss	is	second
ssss	is	DA status of drive. (This status does not always reflect the status that caused the failure.)

DA status is shown below:

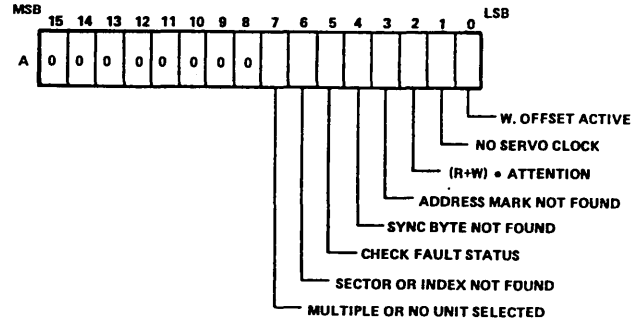


The following hardware statuses can be found in the physical device table assigned to a failed device. Note that the values in the table do not always reflect current status.

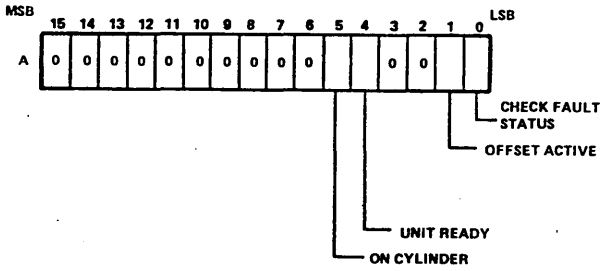
CU status is shown below:



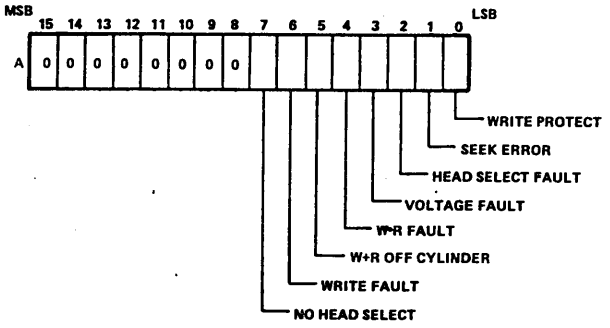
Drive status 1 is shown below:



Drive status 2 is shown below:



Drive fault status is shown below:



DISK PACK INITIALIZATION FORMAT

The disk packs for the disk driver consists of several platters: an inner and outer protective platter, and inner platters for magnetic information. One full surface is preformatted for head positioning (die bits). 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 information. This was discussed in the error recovery section.

The media residing on any module drive is partitioned into cylinders, tracks and sectors. A cylinder is defined as the set of tracks that can be accessed on a single disk pack without head movement. A track is defined as the set of sectors that can be accessed by a single read/write head without head movement. For example, the 80 megabyte, BJ701-AB storage module drive has five tracks per cylinder, and up to 64 sectors per track. If 64 sectors exist per track, each sector contains 96 words.

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).

Disk media must be formatted prior to being used. A formatting module is provided as a part of the systems materials. The user is referred to the appropriate system installation handbook for details.

DIAGNOSTIC FEATURES (OPTIONAL)

Some diagnostic features included in the driver are optional. These diagnostic features must be defined during the system ordering/configuration phase. The diagnostic buffer in which the parameters are defined is provided within the user's request area. The physical device table for the disk is included in appendix C.

The additional requirements specified in the physical device table of the test unit 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 physical device table of the requested unit.
- 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, word 64 in the physical device table of the unit) 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, and so forth, is stored in the 40-word diagnostic status buffer (see the DILUSS module of the module driver DSMD).

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). It is the responsibility of the caller to examine these counters in conjunction with the status words to determine the proper result.

ITOS 2 FLEXIBLE DISK DRIVER

This driver handles data transfers to and from the flexible disk.

The driver performs four major functions:

- Disk transfers
- Error recovery and logging
- Diagnostic logical unit support
- 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.

Three control pushbuttons on the console alter flexible disk operations:

INIT ENABLE	Not functional on standard systems.
WRITE ENABLE	When lit, WRITE ENABLE allows writing to the flexible disk unit.
UNIT REVERSE	When illuminated, the UNIT REVERSE causes the controller to reverse the diskette identity from that set in the controller (that is, unit 0 expects to be addressed as unit 1).

DATA FORMATS

The ITOS driver and standard hardware configuration do not support the DMA transfers required to format (initialize) flexible diskettes. Two separate data formats (as shown in table 4-1) can be used and are determined at the time the diskette is formatted.

TABLE 4-1. DATA FORMATS

Format	Word/ Sector	Sectors/ Track	Tracks/ Drive	Number of Drives (Maximum)	Total Words
CDC	96	19	77†	2	281K
IBM	64	26	77†	2	256K

†74 only are available to user programs

Formatted diskettes for either format may be purchased from CDC.

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 data format is obtained from the newly loaded diskette.

The transfer of data is made using A/Q data transfers. 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).

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 calling program.

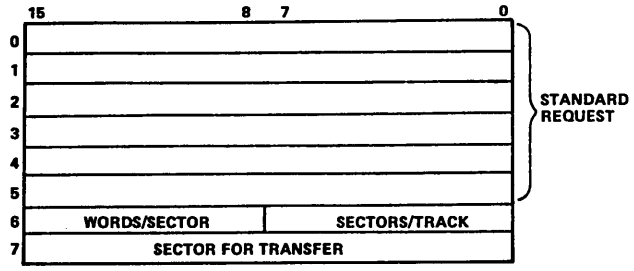
READ/WRITE

Unformatted I/O requests are treated as word-addressed requests. 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.

FREAD/FWRITE

These read and write commands are oriented to the sector/track formatting scheme and are as described below:



The n parameter determines the number of words to transfer. If n is not a divisor of a sector on an FWRITE request, the data is transferred to the following sector and the balance of that sector is zero filled. On a FREAD request, only the specified number of words is transferred; this in effect masks out the remainder of the last sector.

MOTION

All MOTION requests are ignored.

ERROR RECOVERY

The user has no control over the error recovery sequence. On I/O requests the driver attempts to complete the request up to ten times. If unsuccessful, 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 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:

Bit	Definition
0	Unit ready
1	Unit busy
2	Head loaded
3	Seeking
4	Reading/writing
5	Interrupt
6	Interrupt selected
7	Not used
8	Not used
9	Not used
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.

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

Code	Definition
0	Timeout fault
1	Initiator group status error
2	Continuator group status error
3	Time out group status error
4	Speed deviation error
5	Internal reject
6	External reject
7	Short/long transfer error
8	Track/sector fault
9	Status fault(s) after bad track read
10	Status fault after I/O transfer
11	Excessive error recoveries for media

ACCESSING FLEXIBLE DISK

The flexible disk is supported as a magnetic tape compatible device via the magnetic tape simulator and is accessed by the device name FLEXTAPE. If the diskette is to be written to in this mode, the control panel WRITE ENABLE switch must be on. In addition, the write ring for the magnetic tape simulator must be enabled by manually interrupting and typing: WRON,lu (where lu is the logical unit of the flexible disk magnetic tape simulator unit).

MSOS 5 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.

Three control pushbuttons on the console alter flexible disk operations:

INIT ENABLE	Not functional on standard systems.
WRITE ENABLE	When lit, WRITE ENABLE allows writing to the flexible disk unit. A diskette may be individually write protected, in which case writing cannot occur on that diskette, even if the switch is lit. (See motion requests for write enable.)
UNIT REVERSE	When illuminated, the UNIT REVERSE causes the controller to reverse the diskette identity from that set in the controller (that is, unit 0 expects to be addressed as unit 1).

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 mass-storage resident as described in section 2.

The driver performs six major functions:

- Data 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.

DATA FORMATS

The standard hardware configuration does not support the DMA transfers required to format (initialize) flexible diskettes. Two separate data formats (as shown in table 4-1) can be used and are determined at the time the diskette is formatted.

Formatted diskettes for either format may be purchased from CDC.

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 diskette format, bad track sectors, read/write, and write compare options are obtained from the newly loaded diskette. Because the two devices 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.

The transfer of data is made using A/Q data transfers. 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 data. Interlacing is done only at the time a diskette is initialized. It is transparent for all 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
19	10	10
.	.	
.	.	
.	.	
25	13	
26	26	

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 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 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 as described 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 for IBM), the remainder of the sector is transferred anyway on an FWRITE request. However, the remainder of the sector is filled with zeros. 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	Definition
0	Terminate request
1	Not used
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 4-2.

ERROR RECOVERY

The user 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 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:

Bit	Definition
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

TABLE 4-2. FLEXIBLE DISK COMMANDS

Definition	I/O[1]	Format[2]	Mode[3]	Count[4]	Print[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: Logical sector addressing starts with track 1
   Track 0: Logical sector addressing starts with track 0
9. DEL REC: Deleted record
10. Status Located in physical device table after request

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	Definition
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 (for example, 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.

FLEXIBLE DISK UTILITY

The flexible disk utility program described in appendix F is included in the program library for all MSOS systems containing flexible disk drives.

DRIVER DESCRIPTIONS AND DATA FORMATS FOR LINE PRINTER AND CARD READER DRIVERS 5

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 that each print line contains an embedded control character. 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 physical device table of the printer 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 FWRITE mode interprets the first character of the record as carriage control information. The following are carriage control characters.

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

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

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

MOTION

MOTION request codes 2 and 4 are honored by the drivers. All other MOTION requests cause a normal request completion with no action.

Code	Definition
2	Page eject
4	Reset line count

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}- $5E_{16}$ if using a 64-character set, or leave unchanged if using a 96-character set
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 buffer, select format tape level two, and continue printing from next print position
$0C_{16}$- Form feed	Select format tape level one (top of form) and continue printing from next print position
$0D_{16}$- Carriage return	Print buffer, upspace one line, and continue printing from beginning of line
$1B_{16}$- Escape	Used for direct function control of the line printer The next character is interpreted as follows: 00_{16}- $2F_{16}$ Ignore 30_{16} Print buffer, no upspace, next line starts at the beginning 31_{16} Print 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
$3F_{16}$	Not applicable
40_{16}	Clear controller and continue
41_{16}-$7F_{16}$	Ignore
80_{16}-BF_{16}	Print buffer, skip 0-63 lines, and continue printing from the next print position

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 the tab stops are at 20, 40, 60, 80, 100, and 120.

When the 12-channel format tape selections are used, the line count is always reset to zero. Lines are counted when the non-FORTRAN FWRITE and WRITE modes are used.

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.

1827-30/32/60/90 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. Only one line printer can be handled by the driver. The driver provides FORTRAN format conversion.

The driver communicates with the line printer exclusively on the A/Q channel. The driver will function in double-buffered mode provided bit 0 of word 36 of the PHYSTB is set. The driver normally functions in pre-print mode.

WRITE/FWRITE/MOTION

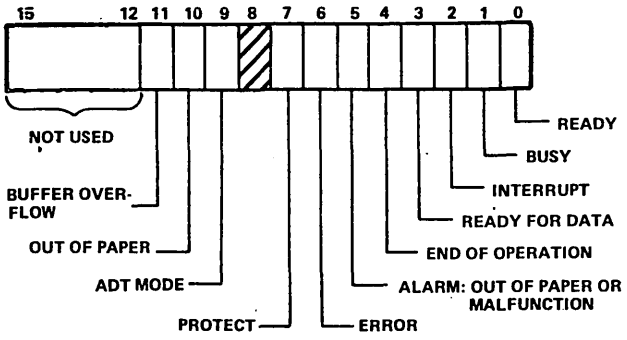
The characteristics of these requests are discussed above.

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	Definition
0	Timeout
2	Alarm
3	Parity
5	Internal reject
6	External reject
38	Paper out

NOTE

A 96-character band may replace the standard 64-character band. In order to take advantage of the extended character set, bit 15 of word 36 of the physical device table must be set to one.

ITOS 2 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 nonstandard MSOS 5 request to the 1843-2 CLA driver indicating the CLA channel attached to the printer.

WRITE/FWRITE/MOTION

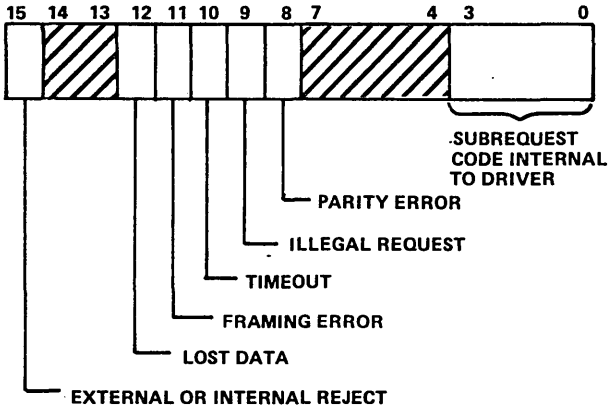
The characteristics of these requests are discussed above.

CHARACTER EDITING

Characters are edited as discussed above, except that the escape character ($1B_{16}$) when followed by 33_{16}-FF_{16} is ignored. The 96-character set option is not available on this printer.

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:



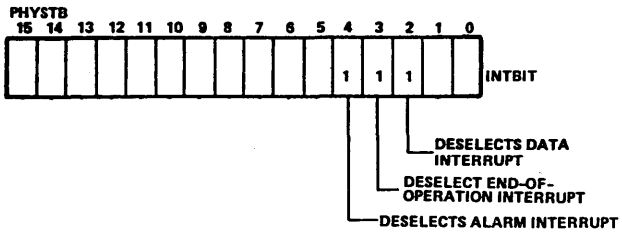
If an error occurs, it is reported by the alternate device handler, and logged in the engineering file. Fault code values are as follows:

Code	Definition
0	Timeout
2	Alarm/status
3	Parity

1829-30/60 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 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:



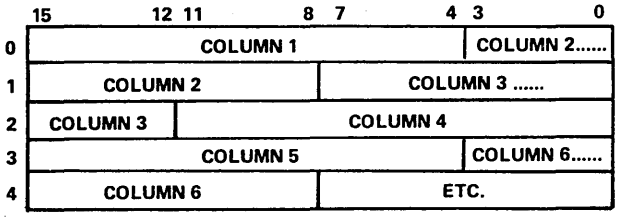
If any INTBIT is set, WAITAD (word 34) must contain the absolute address of the status routine that checks completion of the I/O event. The kernel driver executes the routine and receives control after execution at the continuator entrance of the driver. Transfers are made in ADT mode.

DATA FORMATS

The user may select either ASCII63 or ASCII68 conversion for ASCII reads. This is done by setting word 43 of the physical device table to 0 (ASCII63) or 1 (ASCII68). The manual interrupt processor accepts CARD26 or CARD29 as a command to perform this function.

READ BINARY

The calling sequence specifies the number of words in memory 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 memory 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 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 below 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 5-1 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 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.

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.

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).

EOF PROCESSING AND MOTION REQUESTS

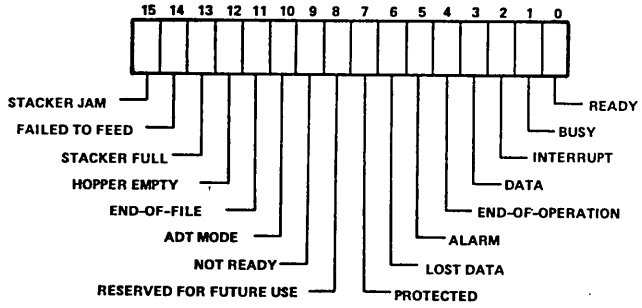
The card reader driver provides the capability of handling end-of-files. An end-of-file is a card with column 1 punched with the configuration set into word 42 of the physical device table. Normally it is a 6/7/8/9 punch, therefore, PHYSTB word 42 contains $000F_{16}$.

The MOTION request to skip file forward is honored by the driver. All other MOTION requests cause no action.

STATUS AND ERROR HANDLING

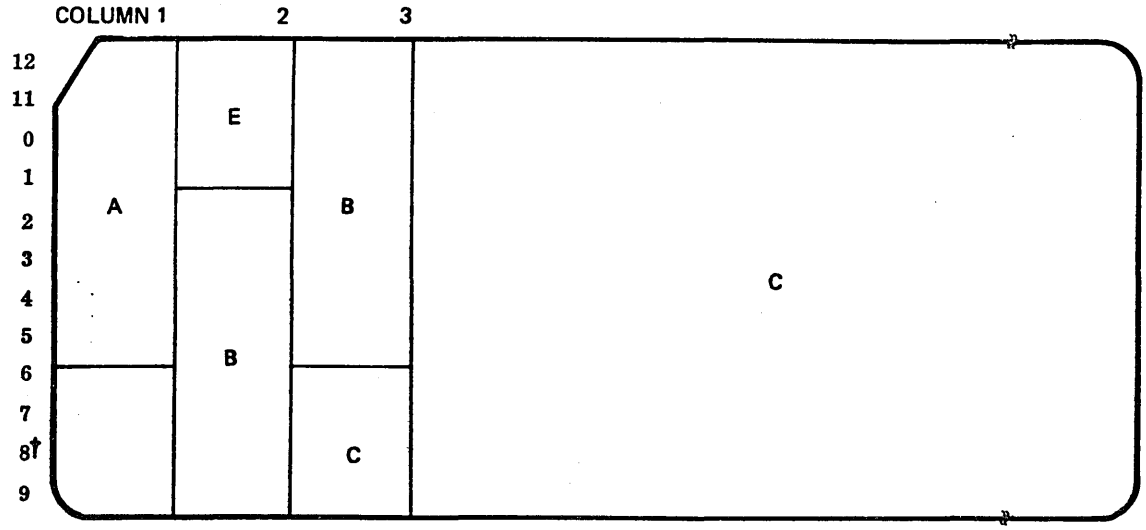
As discussed above, the operator has the option of driving the card reader from status information as well as from various interrupts.

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:

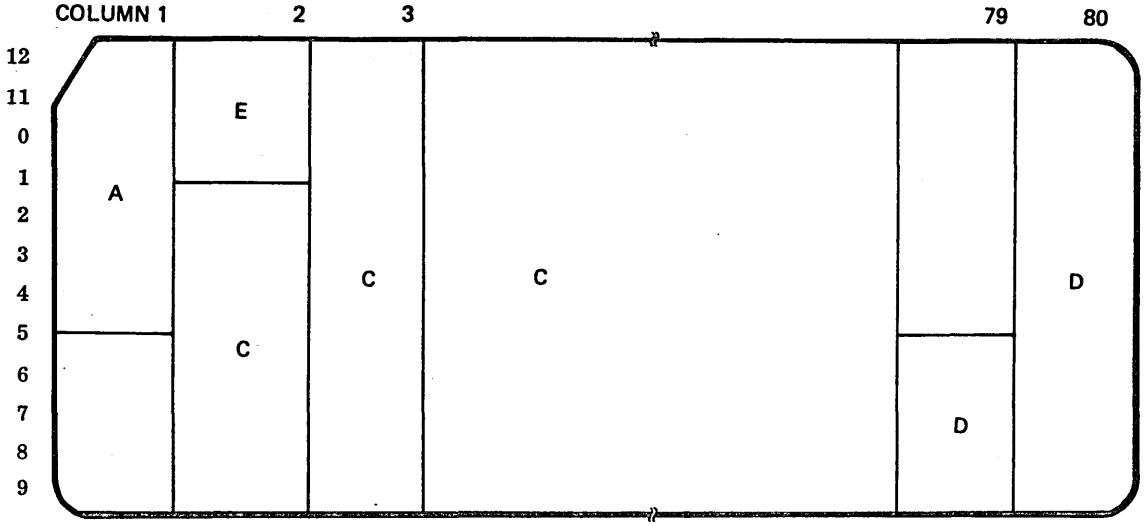


A diagnostic logical unit is supported by the driver. When using the diagnostic logical unit, a maximum of one card can be read even if the user's request specifies a word count greater than one card. A zero word request updates the two director status words in the physical device table but does not 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.

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.

2469

Figure 5-1. Binary Format Record Cards

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	Definition
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	Illegal Hollerith punch
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

DRIVER DESCRIPTIONS AND DATA FORMATS FOR MAGNETIC TAPE DRIVERS 6

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 transfers.

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 rquired. Two parity error checks, LRC (logitudinal redundancy check), and CRC (cyclic redundancy check) are used for verification. They are transparent except to the driver. 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.

For READ requests on either seven- or nine-track, one logical record may be one or more physical record. 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 RECOVERY

The recovery procedure provides a uniform recovery alogrithm for magnetic tape controllers.

Software Requirement

The magnetic tape drivers provide a capability for maximum recovery by utilizing noise records. A system noise record is defined in size and marks off a bad spot of up to two words 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.

Operator response:

RP Repeats the request

CU Allows the processing to continue

DU Suspends any further processing

1860-3/4 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.

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

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 be omitted.

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

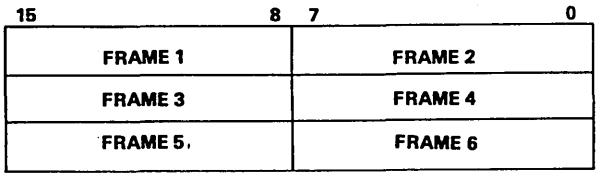
The LCTT driver operates in the buffered data mode; data transfers using ADT and A/Q are not supported. Tape recording density is 800 bpi. Neither reverse read nor nonstop read capability are present.

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.

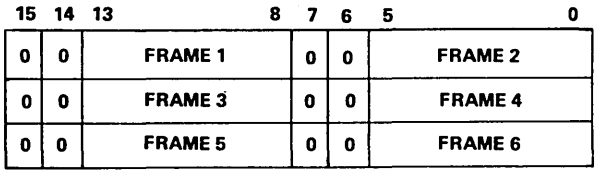
DATA AND RECORD FORMAT

Data Format

Nine-Track Data Transfers - The basic transfer (nine-track) does not need reformatting. All data transfers on a nine-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 memory corresponds to the data on tape and is packed as shown:



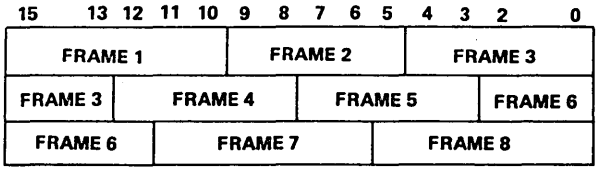
Seven-Track Data Transfers - This option requires the TK7 module. If there is no module TK7DAT in the system, data is transferred to and from memory without conversion. Since the hardware operates in assembly/disassembly mode only, the data appearance in main memory (both ASCII and binary) is as shown:



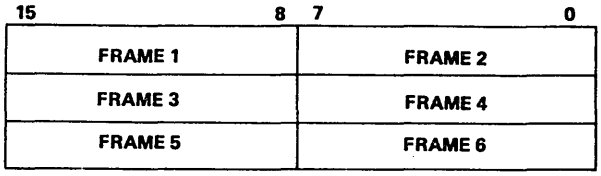
Bits 6, 7, 14, and 15 must be zero whether reading into or writing from memory. If any of these bits are set during a write-to-tape operation, a hardware program error occurs. This error appears in the V-field as a short record error. For raw data transfers, the logical record length is defined by the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

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

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



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



Record Constraints

READ/WRITE records - Record lengths for nine- and seven-track LCTTs 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 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 seven-track requests are oriented to the physical record, 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 stops at the first physical record gap encountered. If the physical record is larger than the number of words requested, only the number requested is placed in memory. If the number requested is greater than the physical record size, the short read indicator is set. The record may be a file mark, in which case file mark status is indicated.

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 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 6-1.

TABLE 6-1. USE OF BUFFERS FOR READ, FREAD, WRITE, FWRITE REQUESTS

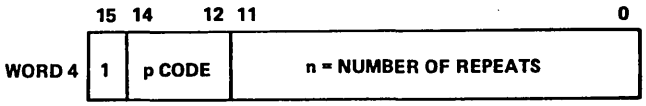
Request	Mode	7-Track with S18326 (LCTT/FORMATTER) 7-Track with TK7DAT (LCTT)	9-Track and 7-Track without TK7DAT (LCTT) or without S18326 (LCTT/FORMATTER)
READ or FREAD	Binary	LCTT → PHYSTB buffer → packed 4 to 3 into requestor's buffer	LCTT → requestor's buffer
	ASCII	LCTT → PHYSTB buffer; convert external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

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	Definition
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.

If a load point is detected during execution of a backspace request, an alarm is indicated and tape is ready at load point.

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.

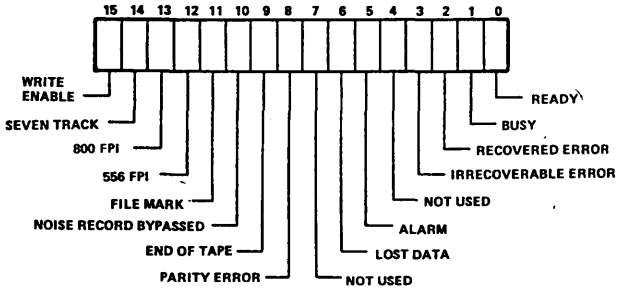
Write operations are as follows:

1. Back over the area just used, erasing and rewriting.
2. Verify the rewritten information.
3. Indicate an unrecoverable error because tape has run out, 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	Definition
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 requested (less than three words)
41	Incomplete error (unrecoverable)

1860-5/6 LOW-COST TAPE TRANSPORT/FORMATTER DRIVER (DUAL MODE)

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.

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 nine frames) compose a single CPU word. ASCII is transferred without conversion. The data in memory 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 memory without conversion. Since the hardware operates in assembly/disassembly mode only, the data appearance in memory (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 or writing from memory. 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 memory; ASCII data is converted to or from external BCD.

Seven-track binary mode assumes memory 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 memory to be converted to or from external BCD. The ASCII characters in memory correpond 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 stops at the first physical record gap encountered. If the record contains more words than requested, only the number requested is placed in memory. If the record contains fewer words, the short read indicator is set. The record may be file mark, in which case, file mark status is indicated.

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.

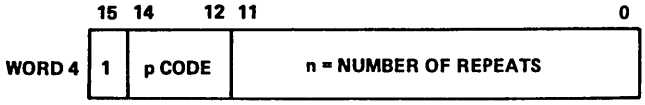
The use of buffers for the four acceptable transfer requests is shown above in table 6-1.

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:

Code	Definition
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:



If a load point is deleted during execution of a backspace request, an alarm is indicated.

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:

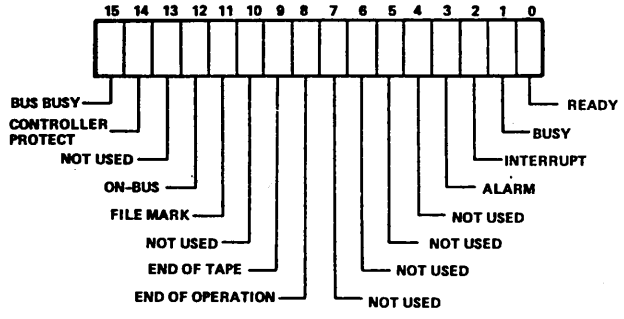
Write Operations:

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 requested (less than two words)
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)

OPERATING SYSTEM DRIVERS 7

The drivers specified in this section are part of the MSOS 5.0 package and are included in this manual as a convenience to the user. These drivers are not necessarily available or desirable in all systems. Refer to the appropriate system reference manual for additional information.

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.

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, and so forth, using:

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

These statements cause the necessary initialization of the pseudo-tape physical device table. After this set-up process, the devices may be accessed as normal magnetic tapes, with the exception that the first EOF terminates the pseudo tape and write is always enabled.

Foreground pseudo tapes function in the same way as background pseudo tapes, except that the user does not need to use *DEFINE, *OPEN, and so forth to set up the tape for use.

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.

SOFTWARE BUFFER DRIVER

The software buffer driver is capable of improved memory utilization in situations where programs require data output to system devices that are classified as having slow data rates (for example, I/O-TTY, paper tape punch, and so forth).

SYSTEM INITIALIZER DRIVERS 8

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 8-1 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

TABLE 8-1. HARDWARE DEVICE DRIVERS

Logical Unit	Driver Name	Hardware Devices	Equipment Code/WES	Dummy Name †	Other Requirements
2	QCARD	1829-30/60	11/0581	QCDDMY	
3	Q18326	1860-5/6	9/0480	QMTDMY	Unit 0 used
3	QMLCT7	1860-3/4	9/0480	QMTDMY	Unit 0 used
3	QMLCT9	1860-3/4	9/0480	QMTDMY	Unit 0 used
4	Q18331	1867-10/20/40	14/0700		Unit 0 used
4	Q18334	1866-14	14/0700		Unit 0 used
4	Q8331I	1867-10/20/40	14/0700		Unit 0 used
4	Q8334I	1866-14	14/0700		Unit 0 used
6	Q1810	1810-1	1/0091		
7	Q1827	1827-30/32/60/90	7/201	QPRDMY	
7	Q18277	1827-7	7/201		
8	DUMMY††				

†Used to allow linkage of unused driver entry points

††Used when no device action is desired

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 regular 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 9

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.

Input/output 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 to magnetic tape, 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 input/output routines.

MSOS uses the following DTLP drivers.

DSKCDR DECK-ID D11 DRIVERS 1.4C
(comment device)

MDRSMD DECK-ID C94 DRIVERS 1.4C
(module drive)

CDDRVM DECK-ID D63 DRIVERS 1.4C
(cartridge disk drive)

DTLCM7 DECK-ID C45 DRIVERS 1.4C
(1860-3/4 mag tape)

DTLCM9 DECK-ID C46 DRIVERS 1.4C
(1860-3/4 mag tape)

M18326 DECK-ID D79 DRIVERS 1.4C
(1860-5/6 mag tape)

ITOS uses the following DTLP drivers.

DSKCDR DECK-ID D11 DRIVERS 1.4C
(comment device)

SMDMDR DECK-ID P15 DRIVERS 1.4C
(module drive)

CDDRVE DECK-ID U40 DRIVERS 1.4C
(cartridge disk drive)

DTLCT9 DECK-ID U41 DRIVERS 1.4C
(1860-3/4 mag tape)

M8326I DECK-ID U42 DRIVERS 1.4C
(1860-5/6 mag tape)

B18331 (SMD) AND B18334 (CDD)

B18331 (SMD) and B18334 (CDD) are memory-resident drivers, which are used by the system checkout program to copy the memory image to a section of the disk defined in SYSDAT. They are entered via a direct jump to COBOP. At termination the Q register contains a zero for successful completion. $FFFF in the Q-register indicates an error.

CONSTRAINTS

- Memory image must be less than 32768 (sector).
- MSIZV4 must be less than $FFFF.
- Routines have no recovery capability.

GLOSSARY A

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, for example, 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, like that sent to an interactive terminal

Alternate device handler
A driver for an alternate hardware device. An alternate device is used for 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 data channel that can handle input/output only through the A register

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 no 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 data.

Block
Consecutive machine words or characters considered or transferred as a unit, particularly applicable to input/output

Bootstrap
A technique or device designed to bring itself into a desired state by means of its own action; for example, 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 infuencing 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 systems. Buffered data controllers transfer blocks of data. Micro interrupts are handled by hardware or firmware without driven intervention.

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

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
A series of CDC minicomputer systems.

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 unused bits, for example, 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 (for example, 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
An 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 drive

File mark
Mark imposed on tape (as a separate record) to signify end-of-file

Find next request
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 (for example, 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, 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 main memory 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, such as, a word transfer on the A/Q channel or a data block transfer on buffered channel 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, but 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, such as 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, that is, 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 or 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 (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 the 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 (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, 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 (nonformatted)

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

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 memory address; for example, 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, for example, busy or not busy. I/O devices normally send status of the operation just finished to the CPU.

Strobing
Timing of data, as 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 region
A region in low memory 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 seven-, or nine-track) and punched paper tape are used by various CYBER 18/1700 systems.

Thread
A list of entries each of which contains a pointer to the next entry; for example, logical unit thread. I/O requests (for exanple, parameter lists) that are a part of the requesting program are usually threaded in place, and are therefore scattered throughout memory. 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 automatically controlled and may or may not include a priority scheme by using multiprocessing. Time-sharing may reduce total processing time from that required to do batch processing.

Timer requests
The stack of requests awaiting timeout 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 system 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 (nonformatted)

Write enable
A command or physical device (for example, write-enable button on disk controller or 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 SERIES EQUIPMENT

B

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

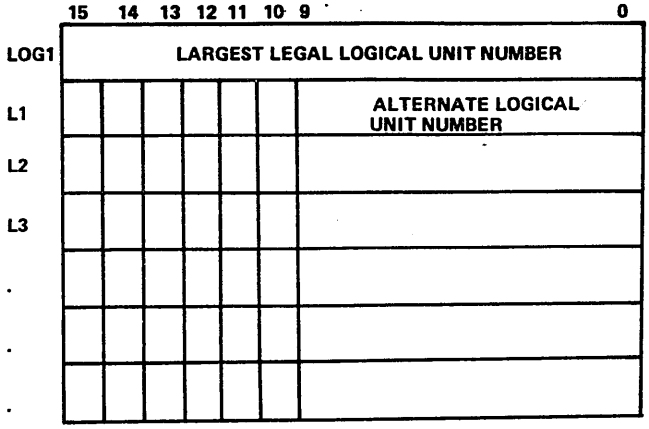
Peripheral	Equipment Code	Macro Interrupt	Micro Interrupt
Teletypewriter/conversational display terminal	1	1	1
Paper tape reader	2	2	2
Paper tape punch	2	2	2
Card punch	2	2	2
Dual-channel communication line adapter	2	2	2
None	3	3	3
Line printer	4	4	4
None	5	5	5
None	6	6	6
Flexible disk drive (FDD)	7	7	N/A
Clock	1	8	8
Magnetic tape transport (NRZI only)	9	9	9,0
Eight-channel communications line adapter	10	10	10
Card reader	11	11	11
Magnetic tape transport (NRZI and phase encoded)	12	12	N/A
Not assigned	13	13	N/A
Storage module disk (SMD)	14	14	N/A
Cartridge disk drive (CDD)	14	14	N/A
Not assigned	15	15	N/A

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.

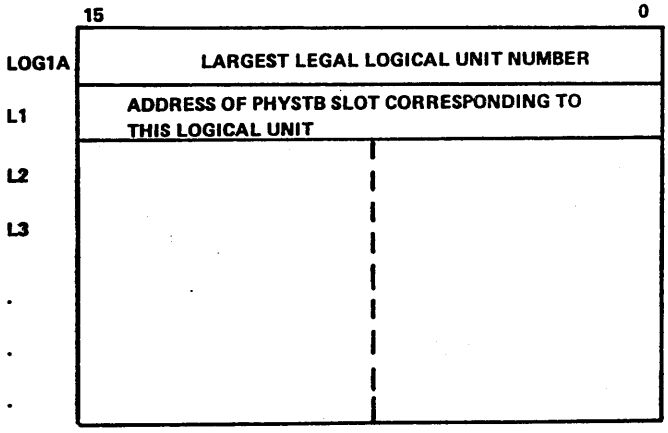


Where:

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.

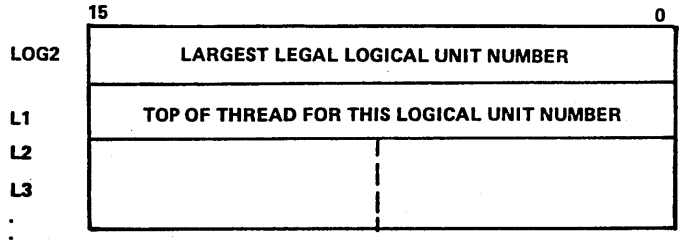
LOG1A TABLE - LOGICAL/PHYSICAL UNIT TABLE

This table contains pointers that relate each logical unit to its physical device table.



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 memory. At execution time, a request being processed is moved to protected core unless swapping is prohibited.



PHYSICAL DEVICE TABLES

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

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 devices the table contains at least 16 words. Words 0 through 15 have a standard function for all devices. Additional words are added for special use by drivers. For kernel drivers, words 16 through 23 are also predefined. A detailed description of these common driver words is given in figure C-1. Figure C-2 details equipment type constants.

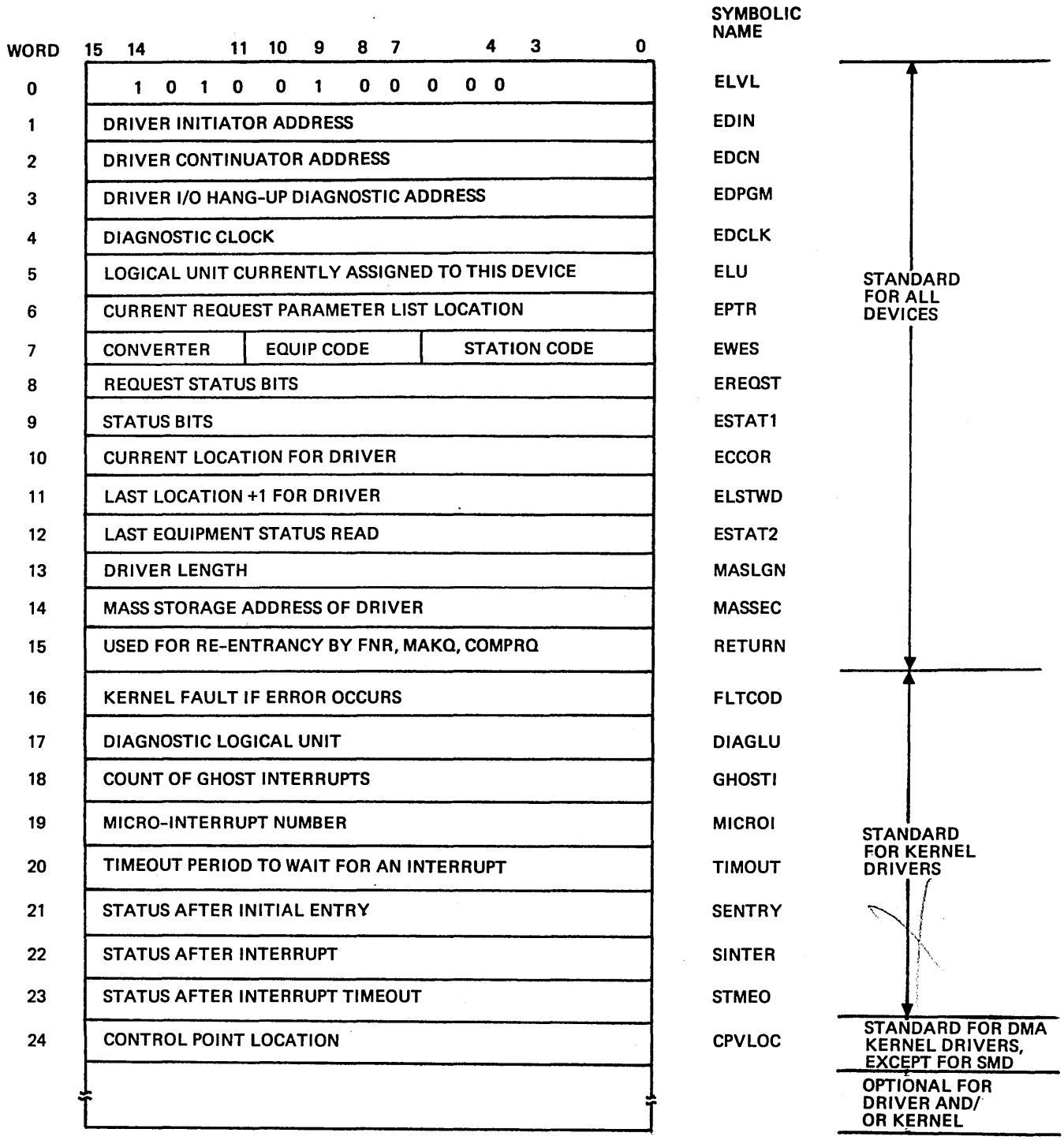


Figure C-1. Common Words in PHYSTB for All Kernel Drivers

The following is a detailed description of each of the words for all drivers (figure C-1).

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 B)
11 through 15	Converter

This 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	
15	1	If operation is in progress
	0	If operation is complete
14	1	If driver detects I/O hardware failure
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 thru 4	Equipment type constant (T)	

See figure C-2

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 1

Bits	Code	Device	
15	1	If error condition and/or end-of-file is detected	
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	

Bits	Code	Device	
12 11 10 9 8 7 6		Reserved for special use by individual drivers	
5	0	If this is a control character	
4	0	If this is the first character	Driver
3	1	If ASCII; 0 if binary mode	
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	

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 $7FFFF_{16}$.

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 micro-interrupt number, if any

Word 20 TIMOUT

The amount of time in seconds to wait for an expected interrupt

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 continuator entry)

Word 23 STIMEO

The status after the device interrupt has time 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 table together all using the same driver/kernel, or save multiple status words (the auxiliary status words should start at word 24).

In the case of multiple devices (for example LCTTs), a physical device table is required for each unit. The PHYSTB is located in SYSDAT.

Following the standard PHYSTB table, PHYSTBs for selected specific devices are shown in figures C-3 through C-10.

CODE	DEVICE
0	1711
1	1721/1722
2	1723/1724
3	1752
4	713-10/711-100 713-120
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
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	1728-30/60 CARD READER
64	1827-30 LINE PRINTER
65	1860-72 7-TRACK TAPE
66	1860-92/95 9-TRACK TAPE
67	1832-5 CASSETTE
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	SPOOLED PRINTER
79	1843-1 2-CHANNEL CLA
80	1860-5 MAGNETIC TAPE
81	1843-3 SYNCHRONOUS CLA
82	1843-5 BUFFERED CLA
83	1867-40 180 MB SMD
84-90	UNASSIGNED
91	915 OCR PAGE READER
92	929 DOCUMENT READER
93	936 DOCUMENT READER
94	UNASSIGNED
95	955 PAGE/DOCUMENT READER
96	UNASSIGNED
97	979 READER SORTER
98	UNASSIGNED
99	UNASSIGNED
100-127	FOR USER ASSIGNMENT

Figure C-2. Equipment Type Constants (Bits 10-4 of word 8)

WORD	DESCRIPTION	SYMBOLIC NAME
0	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
23		
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 (NONDIAGNOSTIC 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-3. CB 104 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	026/029 MODE FLAG	RMODE
44	ADT TABLE CONTROL WORD	ADTCW
45	ADT TABLE FWA - 1	ADTFW
46	ADT TABLE LWA	ADTLW
47	ADT TABLE DUMMY	ADTXX
48	SIM200 LOGICAL UNIT	SIM200
49 ≈ 128	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-3. CB 104 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
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 CHECKWORD - CHECK MOTIONS PARAMETER	COMCHC
49	RESERVED FOR OVERLAY ROUTINE	ECALL1
50	RESERVED FOR OVERLAY ROUTINE	ECALL2
51	RESERVED FOR OVERLAY ROUTINE	ECALL3

Figure C-4. Cartridge Disk Driver PHYSTB (Sheet 1 of 2)

WORD	DESCRIPTION	SYMBOLIC NAME
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-4. Cartridge Disk Driver PHYSTB (Sheet 2 of 2)

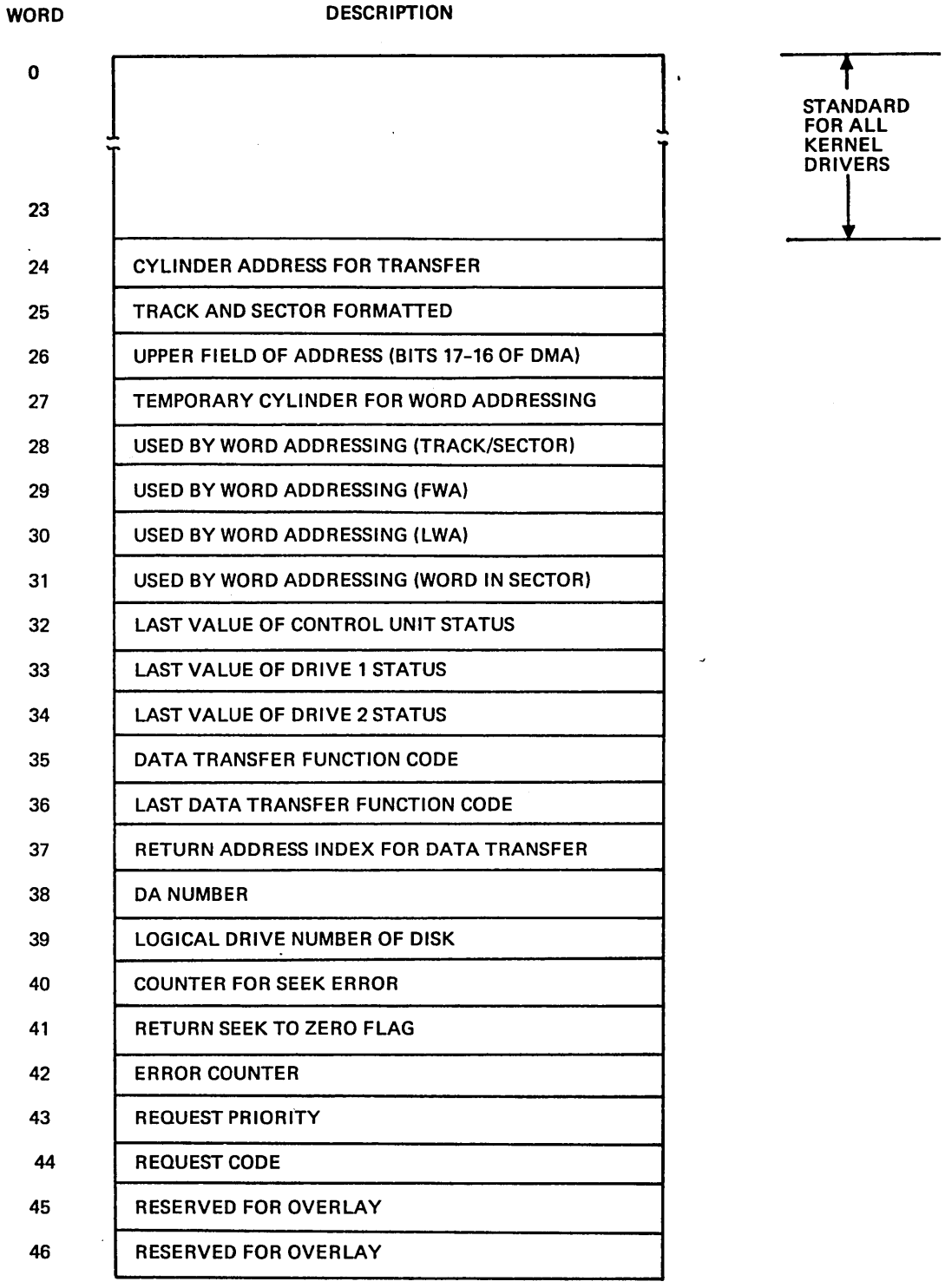


Figure C-5. Storage Module Driver PHYSTB (Sheet 1 of 2)

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 NUMBER 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
68	SELECT STATUS VALUE (15-8) AND MASK (7-0) SOMB SMD
69	DRIVE SIZE VALUE (15-8) AND MASK (7-0) SOMB SMD
70	MAXIMUM WORDS/SECTOR
71	MAXIMUM SECTORS/TRACK

Figure C-5. Storage Module Driver PHYSTB (Sheet 2 of 2)

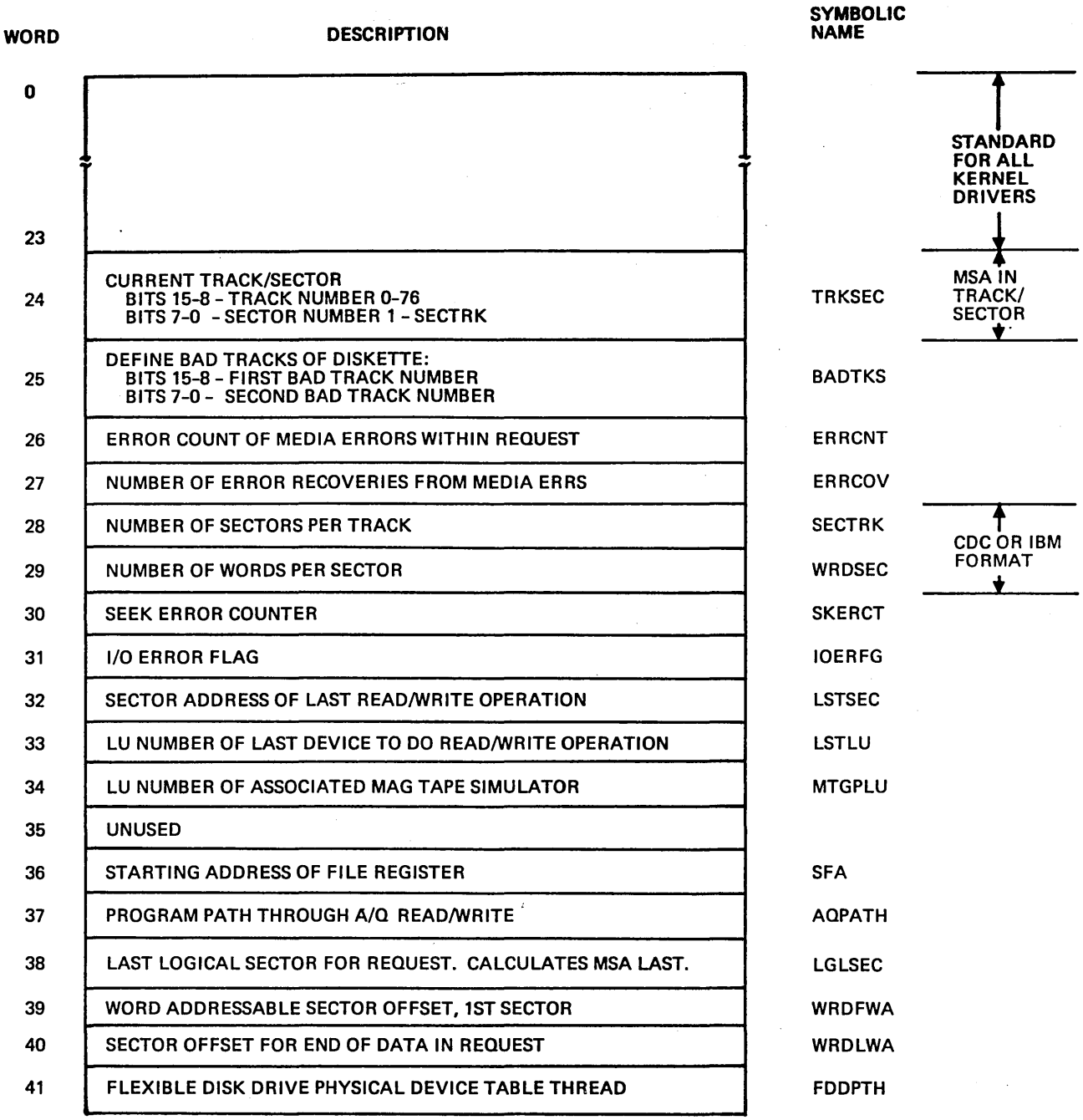


Figure C-6. ITOS 2 Flexible Disk Drive PHYSTB

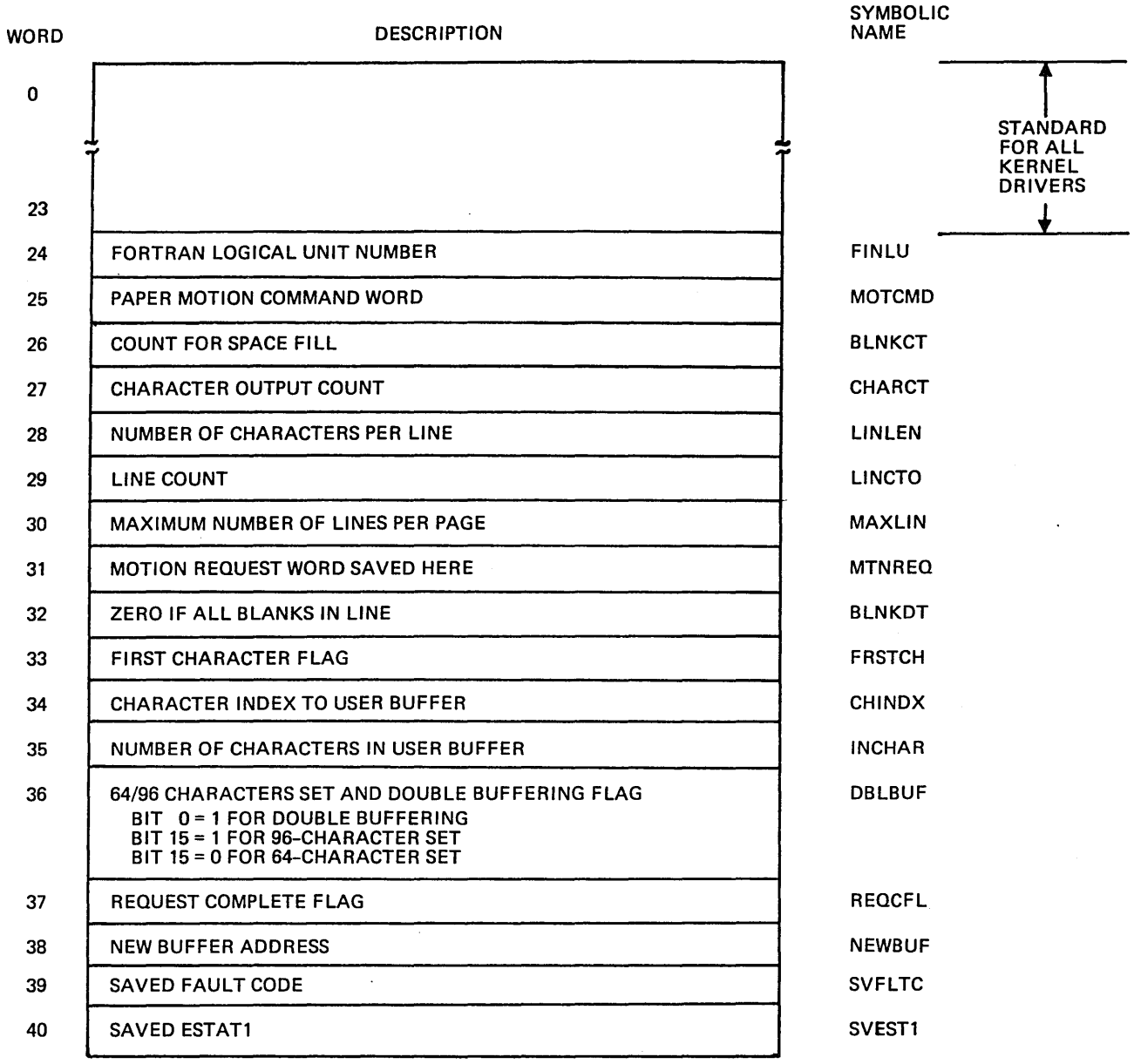


Figure C-7. 1827 Line Printer PHYSTB

WORD	DESCRIPTION	SYMBOLIC NAME
0		STANDARD FOR ALL LERNEL DRIVERS
23		
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 DATA BUFFER	NEWSDB
35	NEW CONTROL POINT INDICATOR	NEWCPI
36	NEW START ADDRESS REQUEST BUFFER/HEADER	NEWSHD
37	NUMBER OF BLANKS TO OUTPUT FOR TAB	TABCNT
38	N, SUCH THAT TAB STOPS EVERY N CHARACTERS	TABSET
39	CURRENT PAGE LINE COUNTER	LINCT
40	MAXIMUM LINES PER PAGE	MAXLIN
41	LINES TO ADD TO LINE COUNT FOR VERT TAB	VTABLF
42	PHYSTB THREAD	ELINK
43	PSEUDO COMPLETION FOR NEW REQUEST	NCOMP

Figure C-8. 1827-7/1843-2 Matrix Printer PHYSTB Pesudo Driver

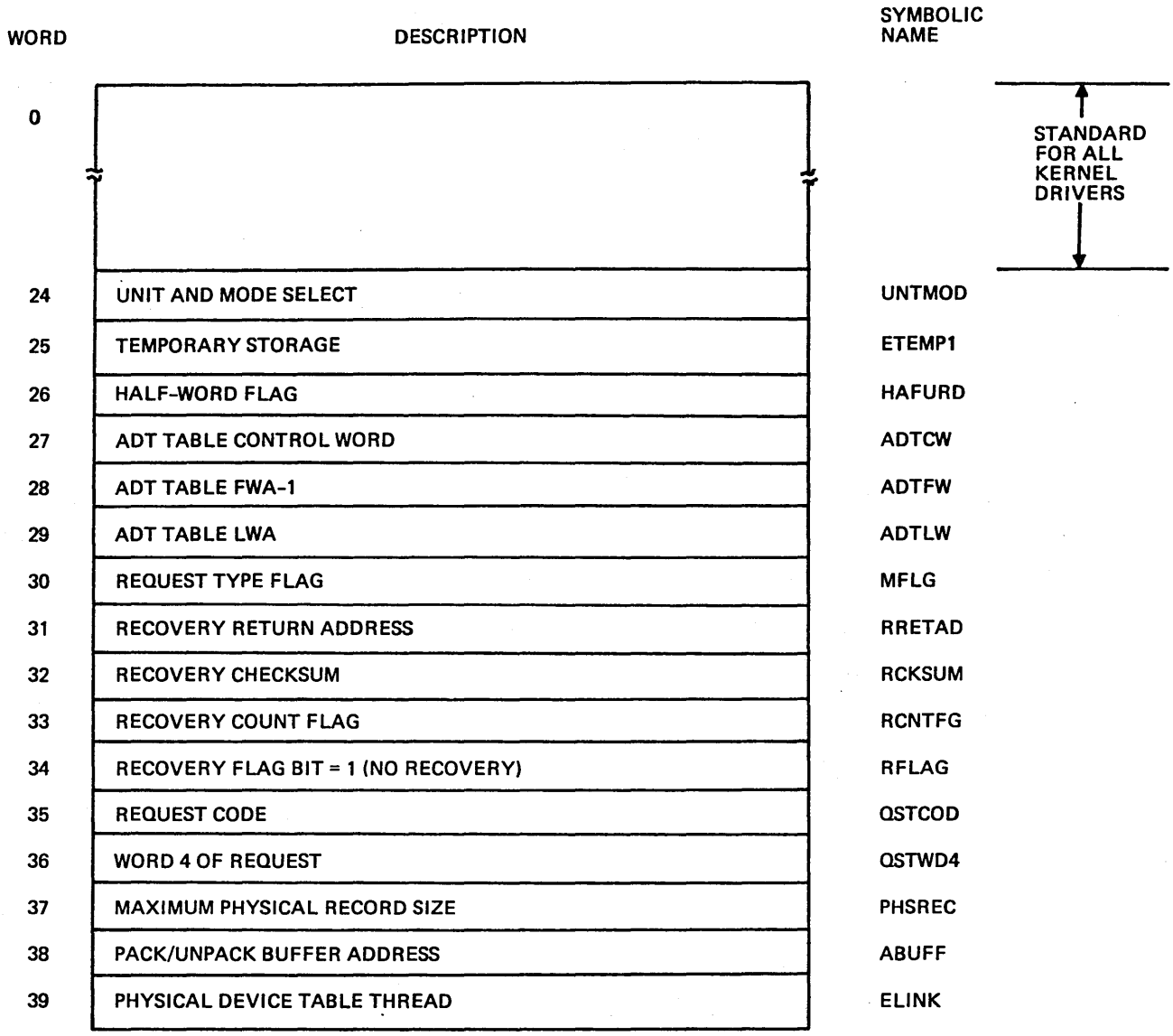


Figure C-9. LCTT PHYSTB

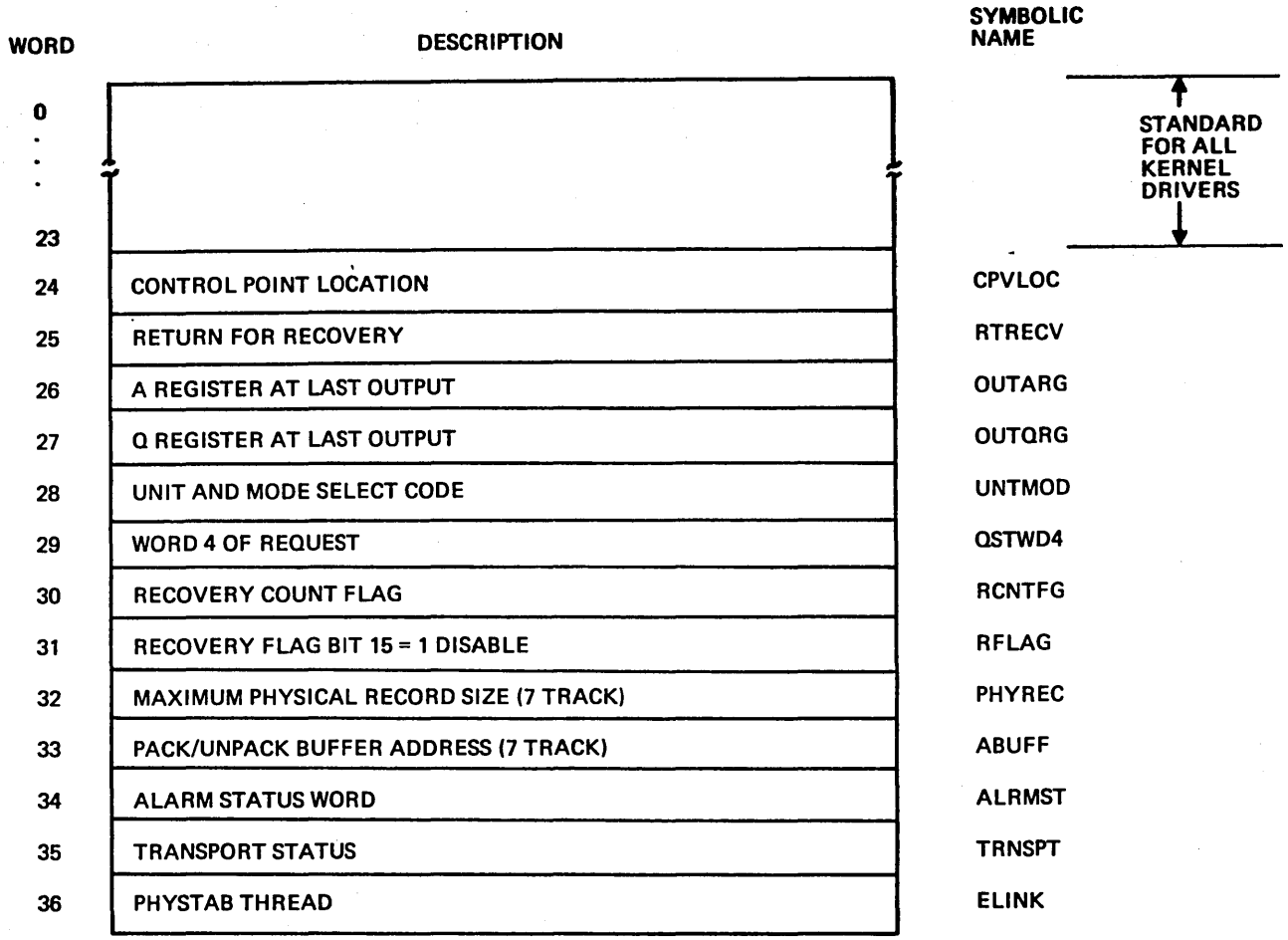


Figure C-10. LCTT/Formatter (1860-5/6) PHYSTB

ASCII CONVERSION TABLE

The 1963 American Standard Code for Information Interchange (ASCII) is used by 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 lists ASCII conversion symbols. Table D-2 lists ASCII and BCD character sets.

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† (Hex)	Character	026†† Punches	029†† Punches	6-Bit Ext. BCD Magnetic Tape (Octal)
20	Space	No punch	No punch	20
21	!	11-8-2	12-8-7	52
22	"	8-7	8-7	17
23	#	12-8-7	8-3	77
24	$	11-8-3	11-8-3	53
25	%	0-8-5	0-8-4	35
26	&	8-2	12	00 (35)†††
27	'	8-4	8-5	14
28	(	0-8-4	12-8-5	34
29	)	12-8-4	11-8-5	74
2A	*	11-8-4	11-8-4	54
2B	+	12	12-8-6	60
2C	,	0-8-3	0-8-3	33
2D	-	11	11	40
2E	.	12-8-3	12-8-3	73
2F	/	0-1	0-1	21
30	0	0	0	12
31	1	1	1	01
32	2	2	2	02
33	3	3	3	03
34	4	4	4	04
35	5	5	5	05
36	6	6	6	06
37	7	7	7	07
38	8	8	8	10
39	9	9	9	11
3A	:	8-5	8-2	15
3B	;	11-8-6	11-8-6	56
3C	<	12-8-6	12-8-4	76
3D	=	8-3	8-6	13
3E	>	8-6	0-8-6	16
3F	?	12-8-2	0-8-7	72
40	@	0-8-7	8-4	37
41	A	12-1	12-1	61
42	B	12-2	12-2	62
43	C	12-3	12-3	63
44	D	12-4	12-4	64
45	E	12-5	12-5	65
46	F	12-6	12-6	66
47	G	12-7	12-7	67
48	H	12-8	12-8	70
49	I	12-9	12-9	71

†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).

††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.

†††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 (%) and $26 (&) are written as BCD 35 (octal). On tape read operations, the BCD code 35 (octal) is always translated to an ASCII $25 (%).

TABLE D-2. ASCII AND BCD CHARACTER SET (Contd)

8-Bit ASCII Codes (Hex)	Character	026 Punches	029 Punches	6-Bit Ext. BCD Magnetic Tape (Octal)
4A	J	11-1	11-1	41
4B	K	11-2	11-2	42
4C	L	11-3	11-3	43
4D	M	11-4	11-4	44
4E	N	11-5	11-5	45
4F	O	11-6	11-6	46
50	P	11-7	11-7	47
51	Q	11-8	11-8	50
52	R	11-9	11-9	51
53	S	0-2	0-2	22
54	T	0-3	0-3	23
55	U	0-4	0-4	24
56	V	0-5	0-5	25
57	W	0-6	0-6	26
58	X	0-7	0-7	27
59	Y	0-8	0-8	30
5A	Z	0-9	0-9	31
5B	[	12-8-5	12-8-2	75
5C	\	0-8-2	0-8-2	36
5D	]	11-8-5	11-8-2	55
5E	^	11-8-7	11-8-7	57
5F	←	0-8-6	0-8-5	32
60	@		8-1	
61	a		12-0-1	
62	b		12-0-2	
63	c		12-0-3	
64	d		12-0-4	
65	e		12-0-5	
66	f		12-0-6	
67	g		12-0-7	
68	h		12-0-8	
69	i		12-0-9	
6A	j		12-11-1	
6B	k		12-11-2	
6C	l		12-11-3	
6D	m		12-11-4	
6E	n		12-11-5	
6F	o		12-11-6	
70	p		12-11-5	
71	q		12-11-8	
72	r		12-11-9	
73	s		11-0-2	
74	t		11-0-3	
75	u		11-0-4	
76	v		11-0-5	
77	w		11-0-6	
78	x		11-0-7	
79	y		11-0-8	
7A	z		11-0-9	
7B	{		12-0	
7C	|		12-11	
7D	}		11-0	
7E	(ALT mode)		11-0-1	
7F	DEL (Rub out)		12-9-7	

DRIVER CODING STRUCTURE E

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 main- 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 (memory-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-  ENDIN,I     SAVE INITIATOR ADDRESS
        TRQ   A
        ADD   =XC1799-START
        STA-  EDCGN,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-      1        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-      1        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 entries for 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 words, 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
                                current 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:

- Driver 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.

MSOS FDUTIL FORMATTING FLEXIBLE DISK F

This appendix describes the flexible disk drive (FDD) utility program FDUTIL.

The purpose of FDUTIL is to provide the user with a simple method of transferring data from external media to a diskette in the formats specified. Because FDUTIL accepts input only from the standard input device, FDUTIL control records must be embedded in the data to be transferred.

When writing to flexible disks, the write-enable switch on the console must be on. Occasionally when re-using diskettes, a read-only diskette will be encountered and an LUxxFAILED 13 will occur even though write enable is on. If the diskette is to be reused an *UNL,lu motion request from the job processor will generate the motion request to clear the protect bit.

The program executes as a job on a CYBER 18 computer using MSOS. 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 list 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:

- *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 format
- *S - Set for operator intervention
- *R - Reset for no operator intervention
- *Z - Terminate the FDUTIL program
- *D - Initialize directory
- *O - Copy binary and build directory

Each input request record contains an asterisk in the first character position and an alphabetic character (A, B, D, O, H, 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

```
...,^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^^^,...
```

*A, ABSOLUTIZE REQUEST

This causes 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 list device:

Position	Description
2	*, if deadstart output specified; , 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 list 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 a 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 list 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 list 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 from the standard input device to be 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 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 reocrds 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 list 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 list device:

Position	Description
1-2	*T
8-10	Logical next sector address (next sector following the end of the program just placed on diskette)

*C, 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,lul,lu2,ssl,esl,ss2,num

Where:

lul 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 lul. If blank or zero, logical unit lul is used.

ssl is the starting sector address of the data to be read from logical unit lul. It is represented as three hexadecimal digits (positions 12-14).

esl is the ending sector of the data to be read from logical unit lul. It is represented as three hexadecimal digits (positions 16-18). If blank or zero, starting sector address ssl 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 ssl is used. Note that the ending sector address where the data is to be written on logical unit lu2 is implied: (ss2 + esl - ssl).

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

Console 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,lul,lu2,ssl,esl,ss2,num

Where:

lul 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 lul. If blank or zero, logical unit lul is used.

ssl is the starting sector address of logical unit lul to be verified, represented as three hexadecimal digits (12-14).

esl is the ending sector address of logical unit lul to be verified, represented as three hexadecimal digits (positions 16-18). If blank or zero, starting sector address ssl 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 ssl is used. Note that the ending sector address of logical unit lu2 is implied: (ss2 + esl - ssl).

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 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 if the standard input device is the standard comment input device (teletypewriter or CRT).

*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

*D, ODS INITIALIZE DIRECTORY

This request specifies that the ODS directory area (sectors 9-10) is to be initialized to -1. A starting sector of 046_{16} is placed in the first directory. The request format is:

*D,LUN

Where:

LUN is the logical unit of the diskette to be initialized, represented as three decimal digits (positions 4-6).

*O, ODS COPY AND BUILD DIRECTORY REQUEST

This request specifies a copy of an absolutized ODS monitor or test and the building of an ODS directory. Each directory entry has the form:

Word 1 - ASCII-coded first two characters of program test name ($4141)

Word 2 - Second two characters of test name ($4141)

Word 3 - Last character of test name ($4120)

Word 4 - Sector address of test (hexadecimal)

Word 5 - Length of test binary (hexadecimal)

Word 6 - Not used ($FFFE)

The request format is:

```
* O, LUN
*, PGMNAM
   Program Binary
   .
   .
   .
*T
```

Where

LUN	is	Hexadecimal logical unit of diskette
*T	is	termination

Output information (standard list device)

AAAAA start=XXXX length=XXXX next available sector=XXXX

Where

AAAAA	is	MON or the name of the test loaded
XXXX	is	four hexadecimal digits.
Start	is	the hexadecimal diskette sector at which the new monitor or test is loaded
Length	is	the length of load in hexadecimal words next available.
Next Available Sector	is	the next diskette sector available for loading. (Last used +1)

When a monitor or test is loaded on a diskette that has a previous copy of that monitor or test, the new version will be loaded over the old if the old version occupied sufficient space to load the new. If there is not sufficient space, the new version will be loaded at the next available sector. Since the directory is changed to reflect the new version, the old version is inaccessible to the user, whether it was or was not destroyed. There is no other provision for allocating unused space on the diskette.

The error messages are:

0250 - Program name specified does not match binary input.

This message may imply that an attempt is being made to load a level I test on a level II diskette or to load a level II test on a diskette which does not have a level II monitor as its first directory entry.

1011 - Operator is probably attempting to use a diskette for which the ODS directory has not been initialized. This error may also indicate that the ODS directory is full (unlikely).

*Z, 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 list device:

FDD UTILITY PROGRAM, ERROR xxxx, RESTART OPERATION

Where:

xxxx is an error code. The error codes and their meanings are listed in table F-1.

After the error message is output, the FDUTIL is automatically restarted. If an *R is in effect, there will be no opportunity to correct the error and many more may occur before the utility finds a valid command.

Table F-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 an *S request is in effect, 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 FDD drive (that is, diskette inserted and drive door closed)
- Change the FDD switches
- Readjust the input records after an error

Whenever a pause occurs, FDUTIL outputs a message 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.

TABLE F-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 a *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 on an *T record.
0240	-	No binary program entered. A *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 binary programs. If using *O, see special instructions.
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, SFR, 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 memroy area.
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, when 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.
1101	*	Directory is FULL. See special instructions for *O.
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 are set to $E5E5_{16}$ (two EBCDIC alphabetic Vs) per IBM standard. Certain words in track 0 are overwritten with valid information during the initialization process.

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.

Unused 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 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.

INDEX

Alternate device handler 1-1; 2-2
Alternate device table, LOG1 2-1; C-1
A/Q channel allocation 2-4
A/Q data transfers 4-10
A register 2-14; 8-1, 2
ASCII conversion table D-1
Auto data transfer 2-4
  clock table 2-6
  multiple A/Q device table 2-5
  single A/Q device table 2-5
  single or multiple MO5 devices table 2-6

Binary/ASCII 5-1
BCD character set D-3

Carriage control information 5-1
Card reader driver 5-3
  binary format record cards 5-5
  data formats 5-3
  EOF processing and motion requests 5-4
  FREAD ASCII 5-4
  FREAD binary 5-4
  PHYSTB C-8, 9
  read ASCII 5-4
  read binary 5-4
  status and error handling 5-4
Cartridge disk 4-1
Cartridge disk driver 4-1
  data transfer request formats 4-1
  disk driver requests 4-2
  FREAD 4-2
  FREAD/FWRITE 4-2
  MOTION 4-2
  READ 4-2
  READ/WRITE 4-2
  Status and error handling 4-2
  WRITE 4-2
Character editing 5-1
Clock table for ADT 2-6
Code conversion table D-1
Code request 1-1, 2
Codes, equipment type C-5
Complete request routine 1-1, 2; 2-1
COMPRQ routine 2-1
Console display driver 3-1
  FREAD request 3-1
  FWRITE request 3-1
  MOTION request 3-1
  READ request 3-1
  WRITE request 3-1
Continuator 1-1
Control point 4-3
COSY driver 7-1
Current control point 4-3

Device priority 2-1
Data transfer 2-4
Diagnostic clock 1-1
Diagnostic clock counter 2-1; 4-4
Diagnostic status buffer 4-7
Diagnostic timer 1-1; 2-2
Disk, cartridge 4-1
Disk pack initialization format 4-7
Disk-to-tape utility drivers 9-1
Driver coding structure E-1
  continuation functions E-2
  diagnostic error functions E-2
  diagnostic unit handling E-2
  initiator functions E-1
  pre-initiation functions E-1
Dual channel CLA, ITOS2/COMM18 (1843-1) 3-1
  Data reception 3-3
  MOTION 3-2
  READ/FREAD 3-2
  WRITE/FWRITE 3-2
Dummy driver 7-1

End-of-file processing 1-4
Engineering file 2-3, 4; 4-7
Equipment interrupt assignments B-1
Equipment type codes C-5
Error flag 1-1
Error recovery 4-3; 6-1, 4
Exit processor request 1-1

Find-next-request routine 1-1; 2-1
Flag, error 1-1
Flexible disk driver, ITOS 2 4-7
  data formats 4-8
  error recovery 4-8
  FREAD/FWRITE 4-8
  MOTION 4-8
  READ/WRITE 4-8
Flexible disk driver, MSOS 5 4-9
  compare data logic 4-13
  data formats 4-9
  error recovery 4-10
  flexible disk commands 4-12
  flexible disk utility 4-13
  FREAD/FWRITE 4-10
  MOTION 4-11
  overlay requests 4-13
  PHYSTB C-12
  READ/WRITE 4-10
  unprotected requests 4-13

Flexible disk drive utility, (FDUTIL) F-1
absolutize request F-1
ASCII data F-7
ASCII request F-3
binary data F-7
binary request F-1
copy request F-3
deadstart data F-7
define format request F-4
FDD output data format F-7
FDUTIL error codes F-6
FDUTIL error processing F-5
Initialization data F-7
ODS copy and build directory request F-4
ODS initialize directory F-4
operator intervention procedure F-5
reset operator intervention request F-4
set operator intervention request F-4
terminate FDUTIL request F-5
verify request F-4
FNR routine 2-1
Fortran FWRITE 5-1
FREAD request 1-1
FWRITE request 1-1

Glossary A-1

Handler, alternate device 1-1; 2-2
Hang-up 1-1; E-3

Initiator 1-1
Interfaces, system 2-1
Interrupt
condition 1-1
lost 1-1
mask 1-1
response routine 1-1; 2-4
stack 1-1
Input/output, (I/O)
command 1-1
device 1-1
driver 1-1
operation 1-1
request 1-1
I register 2-1; 8-1, 2

Jump request 1-1

Kernel drivers C-2

Line printer driver
binary/ASCII 5-1
character editing 5-1
error conditions 5-2
MOTION 5-1
WRITE/FWRITE 5-1
Line printer driver (1827-30/60/90) 5-2
character editing 5-2
PHYSTB C-13
status and error handling 5-2
WRITE/FWRITE/MOTION 5-2
Line printer driver, ITOS 2 (1827-7) 5-3
character editing 5-3
PHYSTB C-14
status and error handling 5-3
WRITE/FWRITE/MOTION 5-3
Logical unit table 2-1; C-1
LOG1, alternate device table C-1
LOG1A, logical/physical unit table 1-2
LOG2, task threads table 1-1; C-1
Lost interrupt 1-1
Low-cost tape transport 6-2
data format 6-2
FREAD/FWRITE records 6-3
MOTION 6-3
nine-track data transfers 6-2
PHYSTB C-15
READ/WRITE/FREAD/FWRITE 6-3
READ/WRITE RECORDS 6-3
record constraints 6-3
seven-track data transfers 6-2
Low-cost tape transport/formatter driver, dual mode (1860-5/6) 6-5
data format 6-5
error recovery 6-6
FREAD/FWRITE records 6-6
MOTION 6-6
nine-track data transfers 6-5
PHYSTB C-16
READ/WRITE/FREAD/FWRITE 6-6
READ/WRITE records 6-5
record constraints 6-5
seven-track data transfers 6-5

Magnetic tape drivers 6-1
data formatting 6-1
error recovery 6-1
formatted requests 6-1
motion requests 6-1
unformatted requests 6-1
MAKQ routine 1-1; 2-1
Macro format 1-1, 2
FREAD request 1-1
FWRITE request 1-1
MOTION request 1-4
READ request 1-1
WRITE request 1-1
Malfunction, hardware 1-1
Mass memory
address format 1-3
drivers 8-2
sector address 1-3
word address 1-3
Mass storage address 7-3
Monitor 1-1
Monitor request 1-1; E-1
Motion processing 1-4
MOTION request 1-1, 4
Multiple A/Q device table for ADT 2-5
Multiple device handlers E-3
Multiple MO5 device table for ADT 2-6

Operating system drivers 7-1
COSY driver 7-1
dummy driver 7-1
pseudo-tape driver 7-1
software buffer driver 7-1
Overflow register 2-1

Parameter list 1-1, 2
Physical device table, PHYSTB 1-1; 2-1; C-1
Priority
device 2-1
scheduling 2-1
Processor request 1-1
Pseudo-tape driver 7-1

Q register 1-2; 2-1, 2, 4; 8-1, 2; E-1
Queue
beginning 1-1
end 1-1
request 1-1
Queueing of I/O requests 1-1

READ request 1-1
re-entrance E-3
Request
code 1-1, 2
exit processor 1-1
FREAD 1-1
FWRITE 1-1
I/O 1-1
jump 1-1
MOTION 1-1, 4
processor 1-1
queue 1-1
READ 1-1
scheduler 1-1
TIMER 2-2
waiting 2-1
WRITE 1-1
Routine
complete request 1-1, 2
diagnostic timer 1-1
find-next-request 1-1
interrupt response 1-1
MAKQ 1-1
Schedule request 1-1
Scheduling priority 2-1
Single A/Q device table for ADT 2-5
Single MO5 device table for ADT 2-6
Software buffer driver 7-1
System checkout drivers 10-1
constraints 10-1
B18331 (SMD) and B18334 (CDD) 10-1
System interfaces 2-1
System initializer drivers 8-1
driver errors 8-2
hardware device drivers 8-1
input drivers 8-1
Mass-memory drivers 8-2
operation 8-1
Storage module drive driver (1867-10/20/40, 1868-1) 4-3
Disk pack initialization format 4-7
Driver description 4-2
Driver diagnostics 4-7
Error recovery 4-4
PHYSTB C-10
Request format 4-3

Table of logical units 1-1
Terminal driver, ITOS 2 (1843-2) 3-4
Connect 3-5
Cursor positioning 3-5
Disconnect 3-5
FWRITE 3-5
MOTION 3-5
READ/FREAD 3-5
Request format 3-4
WRITE 3-5
WRITE-READ 3-5
Terminal driver, MSOS 5 (1843-2) 3-5
Connect 3-6
Disconnect 3-6
FWRITE 3-6
MOTION 3-6
READ/FREAD 3-6
Request format 3-5
WRITE 3-6
WRITE-READ 3-6
Timeout
entry 1-1
error 1-1
period 4-3
TIMER request 2-2
Thread 1-1, 2

Waiting request 2-1
WRITE request 1-1

COMMENT SHEET

MANUAL TITLE CDC® Software Peripheral Drivers Reference Manual

PUBLICATION NO. 96769390 REVISION F

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.

☐ Please reply ☐ No reply necessary

CUT ALONG LINE

STAPLE

STAPLE

FOLD

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 8241 MINNEAPOLIS, MINN.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION
PUBLICATIONS AND GRAPHICS DIVISION
4455 EASTGATE MALL
LA JOLLA, CALIFORNIA 92037

CUT ALONG LINE

FOLD

FOLD

CORPORATE HEADQUARTERS, P.O. BOX 0, MINNEAPOLIS, MINN. 55440
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD

LITHO IN U.S.A.

CONTROL DATA CORPORATION