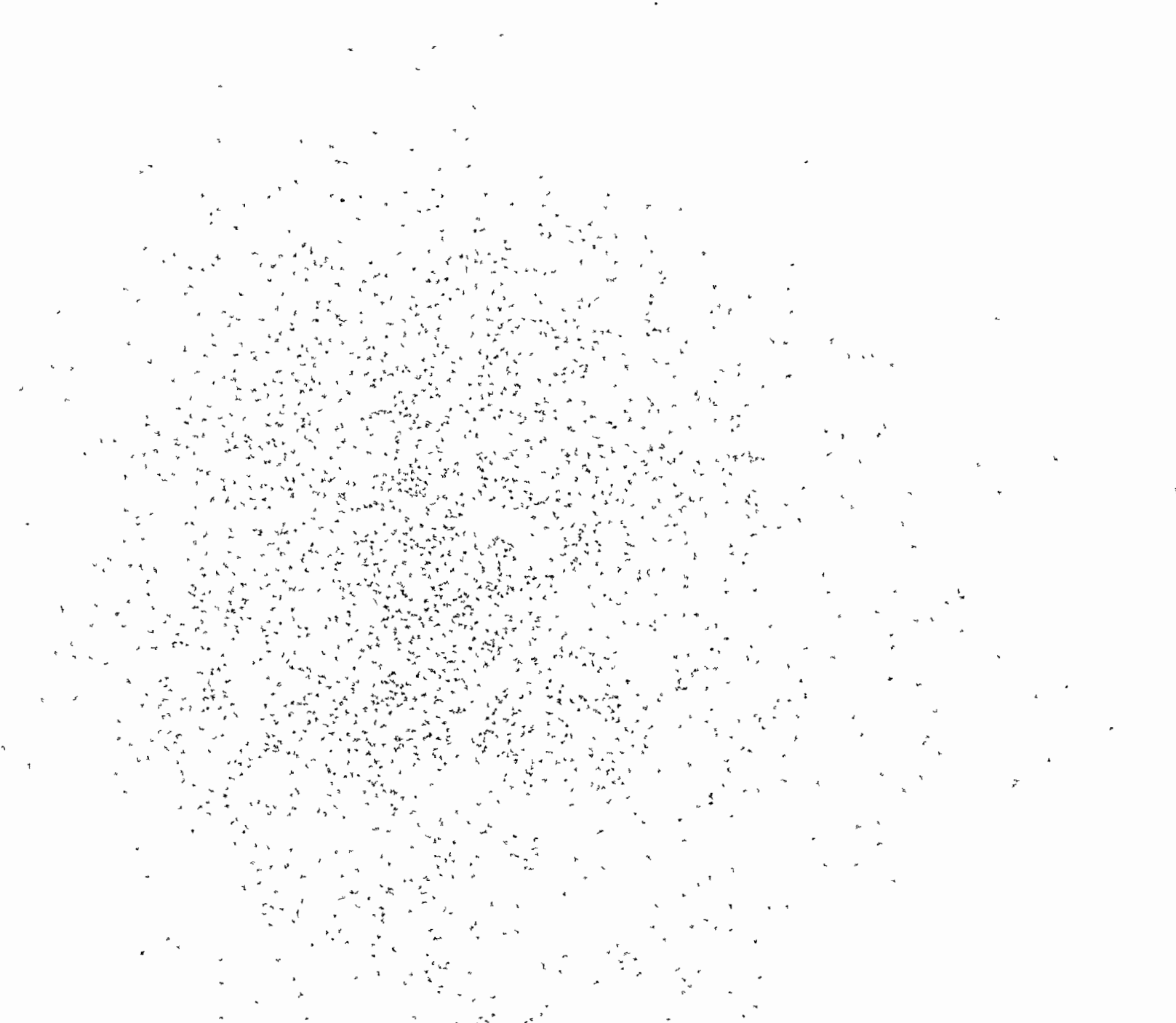


B1

A^M:



IDENTIFICATION

PRODUCT CODE: AC S914D MC
PRODUCT NAME: CXDUBDO UDASOA/KDASO Q DECX MOD
PRODUCT DATE: 20-SEP-1984
MAINTAINER: ROGER OAKY
AUTHOR: JOHN MERTZ

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT.

NO RESPONSIBILITY IS ASSUMED FOR THE USE OR RELIABILITY OF SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL OR ITS AFFILIATED COMPANIES.

COPYRIGHT (C) 1984 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DEC	DIBOL	RSX
DEC/CMS	EduSystem	UNIBUS
DECnet	IAS	VAX
DECsystem-10	MASSBUS	VMS
DECSYSTEM-20	PDP	VT
DECUS	PDT	Digital Logo
DECwriter	RSTS	

.ENABL LC

.REM E

TABLE OF CONTENTS

1.0	ABSTRACT
2.0	REQUIREMENTS
3.0	START UP
4.0	PASS DEFINITION
5.0	EXECUTION TIME
6.0	CONFIGURATION REQUIREMENTS
7.0	DEVICE/OPTION SETUP
8.0	MODULE OPERATION
9.0	OPERATION OPTIONS
10.0	PRINTOUTS
11.0	DUAL PORT OPERATION
12.0	GLOSSARY
13.0	BIBLIOGRAPHY

1.0 ABSTRACT

The exerciser will be similar to that of other disk subsystem exercisers. Writes will be performed to the disks followed by read and compare of the data read. The controller will do all error retrying. Errors will be reported on the console TTY.

All desired disk drives on the controller will be exercised simultaneously. If disk accessing is not required, then data written will go only as far as the controller's RAM memory.

If the results of the exerciser requires more information, two other PDP-11 diagnostic programs are available. They are:

CZUDHA0 - UDA50A/KDA50-Q and Disk Drive Diagnostic
 CZUDIA0 - UDA50A/KDA50-Q Disk Drive Exerciser
 CZUDKA0 UDA50A/KDA50 Q Disk Formatter.

2.0 REQUIREMENTS

Hardware for all cases:

One DEC/X11 module configures for one UDA50A or KDA50-Q

controller.

Hardware for disk accessing:

One controller with at least one drive is the minimum amount or one controller with four drives is the maximum amount.

Hardware for no disk accessing:

One controller is the only requirement.

Storage: DUBD requires

Decimal words -- 4096 MAX

3.0 START-UP

On the initial start, the program will clear bit1 of 'SR1' and type the following messages.

DUBD0 PA:0060162 APC: 000674 PASS #00000

'IF YOU WISH TO DESTROY CUSTOMER DATA, SET BIT1 (NOT BIT0)
IN SWITCH REGISTER 1(SR1) OF DUBD? EQUAL TO 1.'

DUBD0 PA:0060210 APC: 000722 PASS #00000
'! OPERATING WITH NO DISK ACCESSING !'

This will occur regardless of the condition of SR1 (bit1) at configure time.

If the operator wishes to exercise the drive, SR1 (bit1) must be modified at location 16 of CXDUBD0 module (see section 9). This can be accomplished by using the 'MOD' command supplied by the DECX11 run time system. Unless the program is reloaded or the operator modifies the location again, the contents of SR1 will remain the same on all subsequent starts.

On all subsequent starts, the condition of SR1 (bit1) will type to terminal in the following manner.

If bit1 of SR1 is equal to 0 (zero), the following warning will be typed.

DUBD0 PA:0060210 APC: 000722 PASS #00000
'! OPERATING WITH NO DISK ACCESSING !'

If bit1 of SR1 is equal to 1 (one), the following warning will be typed.

DUBD0 PA:0060210 APC: 000722 PASS #00000
'! CUSTOMER DATA WILL BE DESTROYED !'

FI

<<<< NOTE >>>>

When this DEC/X11 module runs in diskless mode, its data rate exceeds all other devices. This may cause erroneous data lates from other devices.

4.0 PASS DEFINITION

One pass of the DUBD module consists of 512 iterations of the basic test sequence (write, read, data-check). The test sequence writes a user defined number of words (default is 256) words, reads 256 words, and data-compare same.

5.0 EXECUTION TIME

The default execution time of one pass of DUBD running alone on a PDP-11/44 under sequential disk accessing mode will be approximately 20 seconds. Under random accessing mode, the time is 40 seconds. For no disk accessing, the time is five seconds

6.0 CONFIGURATION REQUIREMENTS

Default Parameters:

DEVADR: 172150, VECTOR: 154, BR1: 4, DEVCNT: 1, SR1: 0, SR2: 0

REQUIRED PARAMETERS:

Additional controller module(s) configured must have different Unibus address(es) and vector(s).

7.0 DEVICE/OPTION SETUP

For disk mode, make certain that all units are powered up, write enabled, connected to a controller via the SDI and ready.

For diskless mode, make certain the controller is powered up.

8.0 MODULE OPERATION

TEST SEQUENCE DISK MODE:

- A. Setup device register addresses and module variables.
Set controller characteristic.
- B. Reset all units on line and drop all that are not.

- C. Get a unit address.
- D. Get a disk address and a fresh block of data.
- E. Do a write -- if errors, report.
- F. Do a read -- if errors, report.
- G. Do a data-check -- if errors, report and continue.
- H. Make unit available.
- I. Wait for available attention message.
- J. If end of pass, report and go to D.
- K. If end of testing unit, go to C; else go to D.

Blocks determined defective won't be replaced by the exerciser during this sequence. The exerciser makes full use of the controller which does revectoring on its own.

TEST SEQUENCE DISKLESS MODE:

- A. Get a fresh block of data.
- B. Do a write to controller RAM buffer -- if errors, report.
- C. Do a read from controller RAM buffer -- if errors, report.
- D. Do a data-check -- if errors, report and continue.
- E. If end of pass, report.
- F. Go to A.

9.0 OPERATION OPTIONS

One or more software switch registers can be used by the module program general purpose switches. These words are used to define or specify a unique device option or to point to a specific routine in the module. Any option must be specified by the operator before the module is run. Switch Register 1 has the following characteristics.

- SR1 Bit 1 set (1): Allow disk transfers.
 <<<< NOTE >>>> IF SET, CUSTOMER DATA WILL BE DESTROYED!
 reset (0): No disk transfers.
- SR1 Bit 2 set (1): Do not report errors as they occur.
 reset (0): Report errors as they occur.
- SR1 Bit 3 set (1): Do not print error summary at end of pass.
 reset (0): Print error summary at end of pass.
- SR1 Bit 9 set (1): Run Dual port mode (only valid if SR1 Bit 1 is set)
 reset (0): Do not run Dual port mode
- SR1 Bit 10 set (1): Select random block addressing.(only valid if SR1 Bit 1 is set)
 reset (0): Select sequential block

addressing.

SR1 Bit 11 set (1): Bypass data compare.
reset (0): Do data compare.

Switch register 2 has the following characteristics.

SR2 Bits 0 to 5: Burst rate.

A burst rate to speed up NPR transfers by the controller can be used. This value is 6 bits maximum and set up in SR2 at configure time.

<<<< NOTE >>>>

The DVID1 mask reflects the number of units chosen for testing and which units on the system are to be tested. Example: If DVID1 contains a 1, only the first unit found on the system will be tested. A unit's order on the system is judged by its unit number. The lowest unit number zero (0). Unit 0 would be the first tested on the system.

If DVID1 contains a 10, the fourth unit on the system will be tested. If the first two units are chosen, DVID1 is 3. Four consecutive units means DVID1 is 17. Six units, DVID1 is 77.

If there is not a unit corresponding to the DVID1 bit setting, the bit set in DVID1 gets cleared. The exerciser will readjust the mask and drop the nonexistent units if more units are chosen than actually are present. The module is dropped if all DVID1 bits are cleared.

If the number of units chosen is less than the actual number of units present, only the desired units will be used during the exercise.

<<<< ANOTHER NOTE >>>>

Make sure all subunited drives are accounted for. Destroying customer data is not desirable.

<<<< ONE MORE NOTE >>>>

If SR1 Bit 3 is reset, a summary status is printed every 15 passes. This status is formatted as follows:

DUBDO PA: 00060470 ACP: 001210 PASS #00000

SOFT ERROR COUNT #00000 *** HARD ERROR COUNT #00000
CHECK DATA ERROR COUNT #00000

A. Most printouts have the standard formats described in the DEC/X11 document.

B. Non-standard printouts include error messages which dump the following:

- 1) Summary status
- 2) Flags and endcode
- 3) Unit number
- 4) Byte count
- 5) Hi 16-bit LBN value
- 6) Lo 16-bit LBN value
- 7) Extended address
- 8) Physical address

All values except for PASS, RUNTIME and ERRCNT are printed in octal. PASS, RUNTIME and ERRCNT are printed in decimal.
Example:

DUBDO PA: 00064116 APC: 004630 PASS: 00000 ERRCNT: 00001
CSRA: 172150 CSRC: 000000 ASTAT: 000006 ERRTP: 000006
RUNTIME: 000:00:22

DUBDO PA: 00064052 APC: 004564 PASS: 00000

STATUS ENDCOD UNITNU BYTECO HI LBN LO LBN EXTADR PHYADR
000006 000242 000005 000000 000003 116321 000001 062100

STATUS - response of the command sent to the controller.
This is contained in the last five bits of the word. Here is a list of status codes.

- 0 - success
- 1 - invalid command
- 2 - command aborted
- 3 - unit offline
- 4 - unit available
- 5 - media error
- 6 - write protected
- 7 - compare error
- 10 - data error
- 11 - host buffer access error
- 12 - controller error
- 13 - drive error

ENDCOD ending code of the command sent. This shows what command was sent to the UDA. Here is a list of all possible endcodes this module uses.

- 100 - AVAILABLE ATTENTION MESSAGE (not a command but a message sent to the host from the UDA)
- 200 - INVALID COMMAND
- 203 - GET UNIT STATUS
- 204 - SET CONTROLLER CHARACTERISTICS
- 210 - AVAILABLE
- 211 - ONLINE

230 - MAINTENANCE READ
231 - MAINTENANCE WRITE
241 - READ
242 - WRITE

UNITNU - unit number of the drive that is being accessed.
This is not relevant if the user is running diskless
mode.

BYTECO - size of the buffer in bytes.

HI LBN - high logical block number (upper 16 bits) which tells
the user where on the disk the data is going. This
is only valid for disk mode.

LO LBN - low logical block number (lower 16 bits).

EXTADR - extended address of the read/write buffer.

PHYADR - physical address of the read/write buffer.

C. If the controller failed to pass its internal diagnostic, one
of the following messages will be printed.

If the diagnostic found a fault:

DUBDO PA: 00062052 APC: 002564 PASS: 00000
CONTROLLER INIT ERROR, FOUND BY DIAGNOSTIC
SA REGISTER = xxxxxx IN STEP yyyy
ADDR = zzzzzz

If a step bit was not set as expected during the initialization
sequence of the controller:

DUBDO PA: 00062152 APC: 002664 PASS: 00000
CONTROLLER INIT ERROR, STEP NOT SET
SA REGISTER = xxxxxx IN STEP yyyy
ADDR = zzzzzz

If data passed back from the controller was not equal to the
expected value:

DUBDO PA: 00062252 APC: 002764 PASS: 00000
CONTROLLER INIT ERROR, EXPECTED DATA WAS INCORRECT
SA REGISTER = xxxxxx IN STEP yyyy
ADDR = zzzzzz

Where xxxxxx can have any of the following values and meanings:

104000 - Fatal sequencer error
104040 - D processor ALU error
104041 - D proc ROM parity error
105102 - D proc with no Board #2 or RAM parity error
105105 - D proc RAM buffer error
105152 - D proc SDI error
105153 - D proc write mode wrap SERDES error

105154 - D proc read mode SERDES, RSGEN, and ECC error
 106040 - U proc ALU error
 106041 - U proc Control Register error
 106042 - U proc DFMAIL/ROM parity error/Board #1 test count is wrong
 106047 - U proc Constant ROM error with D proc running SDI test
 106055 - Unexpected trap found, aborted diagnostic
 106071 - U proc ROM error
 106072 - U proc ROM parity error
 106200 - Step 1 data error (MSB not set)
 107103 - U proc RAM parity error
 107107 - U proc RAM buffer error
 107115 - Board #2 test count was wrong
 112300 - Step 2 error
 122240 - NPR error
 122300 - Step 3 error
 142300 - Step 4 error

Where yyyyy is the step in which the error was found.

Where zzzzzz is the address of the UDA.

If the maximum number of retries has been exceeded, the following message will be printed.

DUBDO PA: 00061414 APC:002126 PASS #00000

RETRY COUNT EXCEEDED, ABORT

This means the controller did not successfully complete the initialization in four passes. The module is then dropped.

D. If the UDA did not successfully clear the ring buffer in the host area, the following message will be printed.

DUBDO PA: 00061414 APC:002126 PASS #00000

RING AREA NOT CLEARED

This is a fatal error. It means that the controller did not access host memory that the controller would use to communicate with the host. The module is then dropped.

E. If the SA register displays a non-zero value after the initialization sequence is done, the following message will be printed.

DUBDO PA: 00064252 APC: 004764 PASS: 00000
 SA REGISTER IS NOT ZERO, = xxxxxx
 CONTROLLER IS GOING THROUGH INITIALIZATION

Where xxxxxx can have the following values and meanings.

004400 - controller has been initied by either a bus init or by writing into the IP register.
 100001 - bus envelope/packet read error (parity or timeout)

LI

100002 - bus envelopepacket write error (parity or timeout)
100003 - controller ROM and RAM parity error
100004 - controller RAM parity error
100005 - controller ROM parity error
100006 - bus ring read error
100007 - bus ring write error
100010 - bus interrupt master failure
100011 - Host access timeout error
100012 - Host exceeded credit limit
100013 - controller SDI hardware fatal error
100014 - DM XFC fatal error
100015 - Hardware timeout of instruction loop
100016 - Invalid virtual circuit identifier
100017 - Interrupt write error on bus

E. If a drive is dropped by the exerciser, one of the following messages will be printed.

If the drive had an error it could not handle properly after an iteration, the following message will be printed:

DUBD0 PA: 00063012 APC: 003524 PASS #00000

DRIVE 00000 DROPPED.
DEVICE ID BIT = 000001
ERRORS CAUSED DRIVE TO BE DROPPED

If the drive was not found by the exerciser, the following message will be printed:

DUBD0 PA: 00063012 APC: 003524 PASS #00000

DRIVE 00000 DROPPED.
DEVICE ID BIT = 000001
UNIT WAS NOT FOUND BY THE EXERCISER

If there were more device count bits set than the actual number of drives found, the following message will be printed:

DUBD0 PA: 00063012 APC: 003524 PASS #00000

DRIVE 00000 DROPPED.
DEVICE ID BIT = 000001
DVID: BIT SET HIGHER THAN ACTUAL # OF DRIVES FOUND

Solution: try a lesser number of units in DVID1 (loc 14)

11.0 DUAL PORT OPERATION

To run a dual port operation, set bit9 of SR1. The exerciser will check the unit to see if it is offline or available.

The controller will retain control of a unit until the MSCP Available command is entered by the host. During this time, the other controller is not allowed access to the unit through the other port between the write and read. The other controller senses when the unit becomes available and takes it. The MSCP Available command is only executed if SR1 bit 9 and SR1 bit 1 are set. This allows dual porting and disk accessing respectively.

DEC/X11 will only dual port a drive with another DEC/X11 exerciser.

12.0 GLOSSARY

DUBD follows the module name format described in the DEC/X11 Programmer's Guide.

- DU-- Identifies the hardware and thus the module.
- B- Distiguishes between two or more different modules for the same generic device. The sequence A, B, C, ETC. must be used for each additional example.
- D Specifies the module revision.

IOMODX is a type of module in an extended input/output mode. These modules are interrupt driven and are capable of input/output operation. Some added capabilities provided include:

- @ Use of monitor supplied write buffers.
- @ Ability to change the size of the write buffers.
- @ Access to the monitor's check data utility.
- @ Conversion routines to get 18 bit addresses from 16 bit addresses.

13.0 BIBLIOGRAPHY

- CXQUAAO 'DEC/X11 USER'S MANUAL' Sept 1978
- CXQAFDO 'DEC/X11 PROGRAMMERS'S GUIDE' Sept 1978
- CXQUBAO 'DEC/X11 CROSS-REFERENCE MANUAL' Sept 1978

2
3

.DSABL LC

B2

1
3

```

000000      .SBTTL  MODULE HEADER BLOCK
000000      IOMODX <DUBD >,172150,154,4,0,0,1000,104,RBUF,256.,256.
000000      MODULE 150000,DUBD ,172150,154,4,0,0,1000,104,RBUF,256.,256.
000000      .TITLE  DUBD DEC/X11 SYSTEM EXERCISER MODULE
;          .DDXCOM VERSION 6      23-MAY-78
000000      .LIST  BIN
;*****
000000      BEGIN:
000000      104      125      102  MOONAM: .ASCII /DUBD / ;MODULE NAME.
000003      104      040
000005      000
000006      172150
000010      000154
000012      200
000013      000
000014      000001
000016      000000
000020      000000
000022      000000
000024      000000
;*****
000026      150000
000030      000660
000032      000252
000034      000000
000036      001000
000040      000000
000042      000000
000044      000000
000046      000000
000050      000000
000052      000000
000054      000000
000056      000000
000060      000000
000062      000000
000064      000000
000066      000000
000070      000000
000072      000000
000074      000000
000076      000000
000100      000000
000102      000000
000102      000000
000104      000000
000104      000000
000106      000000
000106      000000
000110      000000
000112      001036
000114      000000
000116      000000
000120      000000
000122      000104
;*****
;STAT: 150000 ;STATUS WORD.
;INIT: START ;MODULE START ADDR.
;SPOINT: MODSP ;MODULE STACK POINTER.
;PASCNT: 0 ;PASS COUNTER.
;ICONT: 1000 ;# OF ITERATIONS PER PASS=1000
;ICOUNT: 0 ;LOC TO COUNT ITERATIONS
;SOF CNT: 0 ;LOC TO SAVE TOTAL SOFT ERRORS
;HRDCNT: 0 ;LOC TO SAVE TOTAL HARD ERRORS
;SOPAS: 0 ;LOC TO SAVE SOFT ERRORS PER PASS
;HRDPAS: 0 ;LOC TO SAVE HARD ERRORS PER PASS
;SYSCNT: 0 ;# OF SYS ERRORS ACCUMULATED
;RANNUM: 0 ;HOLDS RANDOM # WHEN RAND MACRO IS CALLED
;CONFIG: ;RESERVED FOR MONITOR USE
;RES1: 0 ;RESERVED FOR MONITOR USE
;RES2: 0 ;RESERVED FOR MONITOR USE
;SVR0: OPEN ;LOC TO SAVE R0.
;SVR1: OPEN ;LOC TO SAVE R1.
;SVR2: OPEN ;LOC TO SAVE R2.
;SVR3: OPEN ;LOC TO SAVE R3.
;SVR4: OPEN ;LOC TO SAVE R4.
;SVR5: OPEN ;LOC TO SAVE R5.
;SVR6: OPEN ;LOC TO SAVE R6.
;CSRA: OPEN ;ADDR OF CURRENT CSR.
;SBADR: ;ADDR OF GOOD DATA, OR
;ACSR: OPEN ;CONTENTS OF CSR.
;WASADR: ;ADDR OF BAD DATA, OR
;ASTAT: OPEN ;STATUS REG CONTENTS.
;ERRTYP: ;TYPE OF ERROR
;ASB: OPEN ;EXPECTED DATA.
;AWAS: OPEN ;ACTUAL DATA.
;RSTRT: RESTR ;RESTART ADDRESS AFTER END OF PASS
;WOTO: OPEN ;WORDS TO MEMORY PER ITERATION
;WOTP: OPEN ;WORDS FROM MEMORY PER ITERATION
;INT: OPEN ;# OF INTERRUPTS PER ITERATION
;ID: 104 ;MODULE IDENTIFICATION NUMBER=104

```

```

000124 007126      RBUFVA: RBUF      ;READ BUFFER VIRTUAL ADDRESS
000126 000000      RBUFPA: OPEN      ;READ BUFFER PHYSICAL ADDRESS
000130 000000      RBUFEA: OPEN     ;READ BUFFER EA BITS
000132 000400      RBUFSZ: 256.    ;SIZE OF THE READ BUFFER
000134 000000      WBUFPA: OPEN     ;WRITE BUFFER PHYSICAL ADDRESS
000136 000000      WBUFEA: OPEN     ;WRITE BUFFER EA BITS
000140 000400      WBUFRQ: 256.    ;WRITE BUFFER SIZE REQUESTED
000142 000000      WBUFSZ: OPEN     ;WRITE BUFFER SIZE AVAILABLE
000144 000000      CDRECT: OPEN     ;CDATA/DATCK ERROR COUNT
000146 000000      CDWDCT: OPEN     ;CDATA/DATCK WORD COUNT
000150 000000      FREE: OPEN      ;RESERVED FOR FUTURE USE
                                000040      ;MODULE STACK STARTS HERE.
                                .REPT   SPSIZ
                                .NLIST
                                .WORD   0
                                .LIST
                                .ENDR

000252      MODSP:
;*****
;*****
.SBTL  MODULE STORAGE AREA
;      VERSION 1.0   FOR RELEASE
;      VERSION 1.1   NO LONGER TEST AFTER STEP 4
;      VERSION 2.0   NO LONGER WAIT FOR INTERRUPT AFTER SENDING MSCP AVAILABLE COMMAND
;                   USE BIT 9 IN SR1 FOR DUAL PORTING. (DON'T SEND MSCP AVAILABLE
;                   COMMAND IF WE WANT JUST SEQUENTIAL OR RANDOM ACCESS MODE --
;                   IN OTHER WORDS, ONLY SEND ONLINE COMMAND ONCE DURING PASS UNLESS
;                   DUAL PORT MODE)
;      VERSION 3.0   KDA50-Q SUPPORT ADDED

15          000002      SR.XFR = BIT01 ;NO DISK TRANSFER 0 = NO DISK TRANSFER, 1 = DO DISK TRANSFER
16          000004      SR.REP = BIT02 ;REPORT ERROR AS THEY OCCUR 0 = REPORT, 1 = DON'T REPORT
17          000010      SR.SUM = BIT03 ;REPORT ERRORS ON END OF PASS 0 = REPORT, 1 = DON'T REPORT
18          001000      SR.DUA = BIT09 ;DUAL PORT 0 = NO DUAL PORT, 1 = DUAL PORT
19          002000      SR.SEQ = BIT10 ;RANDOM (NOT SEQUENTIAL) DISK ADDRESSING 0 = SEQUENTIAL, 1 = RANDOM
20          004000      SR.CMP = BIT11 ;NO DATA COMPARE 0 = DO DATA COMPARE, 1 = DON'T DO DATA COMPARE

22          ;IPREG: .WORD 0      ; CONTROLLER POLLING REG
23 000252 000000      SAREG: .WORD 0      ; CONTROLLER STATUS REG

25 000254 000000      CINTR: .WORD 0      ;COMMAND INTERRUPT INDICATOR
26 000256 000000      RINTR: .WORD 0      ;RESPONCE INTERRUPT INDICATOR

28 000260      RSPONC: .BLKW 2.    ;MESSAGE RING
29 000264      COMIND: .BLKW 2.    ;COMMAND RING

31 000270 000000      CMDREF: .WORD 0      ;COMMAND REFERENCE NUMBER

33 000272 000000      RSPLN: .WORD 0      ;RESPONCE PACKET LENGTH
34 000274 000000      RSPVIR: .WORD 0      ;RESPONCE PACKET VIRTUAL CIRCUIT
35 000276      RSPACK: .BLKW 24.    ;RESPONCE PACKET

37 000356 000000      CMPLN: .WORD 0      ;COMMAND PACKET LENGTH
38 000360 000000      CMPVIR: .WORD 0      ;COMMAND PACKET VIRTUAL CIRCUIT
39 000362      CMPACK: .BLKW 24.    ;COMMAND PACKET

41 000442 000264      VA: .WORD 0      ;GENERIC VIRTUAL ADDRESS FOR GETPA
42 000444 000000      PA: .WORD 0      ;GENERIC PHYSICAL ADDRESS
    
```

MODULE STORAGE AREA

43	000446	000000	EA:	.WORD	CPFN		;GENERIC EXTENDED ADDRESS
44							
45	000450	000000	RBFFEA:	.WORD	0		;READ BUFFER EXTENDED ADDRESS SAVE AREA
46	000452	000000	WBFFEA:	.WORD	0		;WRITE BUFFER EXTENDED ADDRESS SAVE AREA
47							
48	000454	000000	NUF:	.WORD	0		;ADDRESS USED IN OTOA
49	000456	000000	OLDPA:	.WORD	0		;THE OLD PHYSICAL ADDRESS
50	000460	000000	OLDEA:	.WORD	0		;THE OLD EXTENDED ADDRESS TO CHECK IF
51							; CONTROLLER WILL BE REINITED
52							
53		000017	PRTNUM = 15.				;PRINT MESSAGE EVERY 15TH TIME
54	000462	000017	PRNMSG:	.WORD	PRTNUM		;PRINT WORD SAVES THE VALUE TO CHECK FOR WHEN
55							; THE NEXT TIME AN END OF PASS MESSAGE IS WRITTEN
56		002260	TIMER = 1200.				;TIMER VALUE TO WAIT 2-3 SECONDS AFTER DAP COMMAND
57							
58	000464	177777	EXPAV:	.WORD	177777		;EXPECTING AN AVAILABLE ATTENTION MESSAGE = 0
59							;NOT EXPECTING AN AVAILABLE ATTENTION MESSAGE = 177777
60							
61	000466		ADR1:	.BLKB	6		
62	000474	000		.BYTE	0		
63	000475		ADR2:	.BLKB	6		
64	000503	000		.BYTE	0		
65	000504		ADR3:	.BLKB	6		
66	000512	000		.BYTE	0		
67	000513		ADR4:	.BLKB	6		
68	000521	000		.BYTE	0		
69	000522		ADR5:	.BLKB	6		
70	000530	000		.BYTE	0		
71	000531		ADR6:	.BLKB	6		
72	000537	000		.BYTE	0		
73	000540		ADR7:	.BLKB	6		
74	000546	000		.BYTE	0		
75	000547		ADR8:	.BLKB	6		
76	000555	000		.BYTE	0		
77				.EVEN			

E2

```

1          .SBTTL MORE MODULE STORAGE
2
3          ;
4          ;DO NOT CHANGE THE ORDER OF THE NEXT 4 LOCATIONS
5          ;NEEDED FOR MAP 22 ROUTINE
6          PA18: .WORD 0
7          XMEM: .WORD 0
8          PA22: .WORD 0
9          EA22: .WORD 0
10         SECL: .WORD 0          ;CURRENT SECTOR LO ORDER ADDRESS
11         SECH: .WORD 0          ;CURRENT SECTOR HI ORDER ADDRESS
12
13         UNSZL: .WORD 0         ;UNIT SIZE LO ORDER LIMIT FROM ONLINE CMND
14         UNSZH: .WORD 0         ;UNIT SIZE HI ORDER LIMIT
15
16         LIMIT: .WORD 3300      ;4K - 1200 = MOST WORDS MAITW CAN TAKE
17
18         DVICE: .WORD 1         ;DEVICE TO TEST
19         UNITNO: .WORD 0        ;UNIT NUMBER
20         TRY: .WORD 0           ;NUMBER OF TRIES
21         PORTID: .WORD 1        ;BIT POSITION SELECTS THE PORT
22         UNITFL: .WORD 0        ;SAVE UNIT FLAGS
23         WORK: .WORD 0          ;TEMPORARY WORK AREA
24
25         TIMEOUT = 3000.        ;TIME OUT GADGE
26         RLIM = 4               ;RETRY LIMIT
27
28         TABLEW: .WORD 0,1     ;TABLE ENTRY UNITNO,PORTID
29         .WORD -1,-1            ;CURRENT LAST TABLE ENTRY
30         .BLKW 12               ;REST OF TABLE
31         TEND: .WORD -1,-1     ;END MARKER
32
33         ;S: .WORD 0,0,0,0,0,0,0,0 ;FOR HARD AND SOFT ERRORS
34         ; .WORD 177777
35
36         ;TABLE: .WORD S        ;EACH ENTRY OF THE TABLE POINTS TO
37         ; .WORD S+2            ;THE CORRESPONDING ENTRY OF S.
38         ; .WORD S+4            ;THIS IS USED IN HRDER & SOFER
39         ; .WORD S+6
40         ; .WORD S+10
41         ; .WORD S+12
42         ; .WORD S+14
43         ; .WORD S+16
44         ; .WORD 177777

```

```
1          .SBTTL  MODULE PRIVATE DATA
2
3          000001      BIT00 = 1
4          000002      BIT01 = 2
5          000004      BIT02 = 4
6          000010      BIT03 = 10
7          000020      BIT04 = 20
8          000040      BIT05 = 40
9          000100      BIT06 = 100
10         000200      BIT07 = 200
11         000400      BIT08 = 400
12         001000      BIT09 = 1000
13         002000      BIT10 = 2000
14         004000      BIT11 = 4000
15         010000      BIT12 = 10000
16         020000      BIT13 = 20000
17         040000      BIT14 = 40000
18         100000      BIT15 = 100000
19
20         ;
21         ;          ERROR BITS
22         ;
23         000000      ERR.0 = 0          ;NOT DEFINED
24         000001      ERR.1 = 1          ;DATA ERROR
25         000003      ERR.3 = 3          ;CONTROLLER NOT READY
26         000006      ERR.6 = 6          ;DRIVE NOT READY, OFF LINE OR NON EXESTENT
27         000032      ERR.32 = 32       ;NPR ERROR
28
```

```

1          .SBTTL CONTROLLER BIT DEFINITIONS
2
3          ; SA REGISTER UNIVERSAL READ BITS
4
5          004000      SA.S1= 004000      ;STEP 1 STATUS BIT
6          010000      SA.S2= 010000      ;STEP 2 STATUS BIT
7          020000      SA.S3= 020000      ;STEP 3 STATUS BIT
8          040000      SA.S4= 040000      ;STEP 4 STATUS BIT
9          100000      SA.ERR= 100000     ;ERROR INDICATOR
10
11         ; SA REGISTER ERROR STATUS BITS
12
13         003777      SA.ERC= 003777     ;ERROR CODE
14
15         ; SA REGISTER STEP ONE READ BITS
16
17         ; SA.CTP=          003400      ;CONTROLLER TYPE
18         002000      SA.NSI= 002000     ; NON SETTABLE INTERRUPT
19         001000      SA.Q22= 001000    ; 22 BIT ADDRESS BUS
20         000400      SA.DIA= 000400    ; DIAG BIT IN SA REGISTER
21         000100      SA.MAP= 000100    ; MAPPING BIT
22         000040      SA.SM = 000040    ; SPECIAL MODE BIT FOR KDA50-Q
23
24         ; SA REGISTER STEP ONE WRITE BITS
25
26         000177      SA.VEC= 000177    ; INTERRUPT VECTOR (DIVIDED BY 4)
27         000200      SA.INT= 000200    ; INTERRUPT ENABLE DURING INITIALIZATION
28         003400      SA.RSP= 003400    ; MESSAGE RING LENGTH
29         034000      SA.CMD= 034000    ; COMMAND RING LENGTH
30
31         ; SA REGISTER STEP TWO READ BITS
32
33         000177      SA.VCE= 000177    ; INTERRUPT VECTOR ECHO
34         000200      SA.INE= 000200    ; INTERRUPT ENABLE ECHO
35
36         ; SA REGISTER STEP TWO WRITE BITS
37
38
39         000001      SA.PRG= 000001    ;LOW ORDER MESSAGE RING BYTE ADDRESS
40
41         ; SA REGISTER STEP THREE READ BITS
42
43         000017      SA.RSE= 000017    ;RESPONCE RING LENGTH ECHO
44         000360      SA.CME= 000360    ;COMMAND RING LENGTH ECHO
45
46         ; SA REGISTER STEP THREE WRITE BITS
47
48
49         040000      SA.LFC= 040000    ;HIGH ORDER MESSAGE RING BYTE ADDRESS
50
51         ; SA REGISTER STEP FOUR READ BITS
52
53         000377      SA.MCV= 000377    ; CONTROLLER MICROCODE VERSION
54
55         ; SA REGISTER STEP FOUR WRITE BITS
56
57         000001      SA.GO= BITO       ;GO BIT TO START CONTROLLER FIRMWARE

```

```

1          .SBTTL COMMAND/MESSAGE DESCRIPTOR BIT DEFINITIONS
2
3          100000      RG.OWN= BIT15          ;SET WHEN CONTROLLER OWNS RING
4          040000      RG.FLG= BIT14          ;FLAG BIT
5
6          ;OFFSETS INTO HOST COMMUNICATIONS AREA WITH ONE DESCRIPTOR TO EACH RING
7
8          000010      HC.SIZ= 8.             ;SIZE OF HOST COMM AREA IN BYTES
9          000060      PKTSIZ= 48.           ;SIZE OF PACKETS IN BYTES
10
11         000000      HC.RES= 0.             ;RESPONCE RING START
12         000002      HC.RCT= 2.             ;RESPONCE RING CONTROL WORD
13         000004      HC.CMD= 4.             ;COMMAND RING START
14         000006      HC.CCT= 6.             ;CONTROL RING CONTROL WORD
15         000276      HC.RPK= RSPACK         ;START OF RESPONCE PACKET BUFFER
16         000356      HC.CPK= HC.RPK+PKTSIZ ;START OF COMMAND PACKET BUFFER

```

		.SBTTL COMMAND PACKET OPCODES	
1			
2			
3	000001	OP.ABO= 01	;ABORT COMMAND
4	000020	OP.ACC= 20	;ACCESS COMMAND
5	000010	OP.AVL= 10	;AVAILABLE COMMAND
6	000021	OP.CCD= 21	;COMPARE CONTROLLER DATA COMMAND
7	000040	OP.CMP= 40	;COMPARE HOST DATA COMMAND
8	000013	OP.DAP= 13	;DETERMINE ACCESS PATHS COMMAND
9	000022	OP.ERS= 22	;ERASE COMMAND
10	000023	OP.FLU= 23	;FLUSH COMMAND
11	000002	OP.GCS= 02	;GET COMMAND STATUS COMMAND
12	000003	OP.GUS= 03	;GET UNIT STATUS COMMAND
13	000011	OP.ONL= 11	;ONLINE COMMAND
14	000041	OP.RD= 41	;READ COMMAND
15	000024	OP.RPL= 24	;REPLACE COMMAND
16	000004	OP.SCC= 04	;SET CONTROLLER CHARACTERISTICS COMMAND
17	000012	OP.SUC= 12	;SET UNIT CHARACTERISTICS COMMAND
18	000042	OP.WR= 42	;WRITE COMMAND
19	000030	OP.MRD= 30	;MAINTENANCE READ COMMAND
20	000031	OP.MWR= 31	;MAINTENANCE WRITE COMMAND
21	000200	OP.END= 200	;END PACKET FLAG
22	000100	OP.AVA= 100	;AVAILABLE ATTENTION MESSAGE
23	000101	OP.ERL= 101	;ERROR LOG ATTENTION MESSAGE
24	000102	OP.SHC= 102	;SHADOW COPY COMPLETE ATTENTION MESSAGE
25	000102	OP.ACP= 102	;ACCESS PATH ATTENTION MESSAGE
26			
27			
28			
29			

;NOTE: END PACKET OPCODES (ALSO CALLED ENDCODES) ARE FORMED BY ADDING THE END
;PACKET FLAG TO THE COMMAND OPCODE. THE UNKNOWN COMMAND END PACKET CONTAINS
;JUST THE END PACKET FLAG IN ITS OPCODE FIELD.

```

1          .SBTTL COMMAND MODIFIERS
2
3          040000      MD.CMP= 040000      ;COMPARE
4          100000      MD.EXP= 100000      ;EXPRESS REQUEST
5          010000      MD.ERR= 010000      ;FORCE ERROR
6          004000      MD.SCH= 004000      ;SUPPRESS CACHING (HIGH SPEED)
7          002000      MD.SCL= 002000      ;SUPPRESS CACHING (LOW SPEED)
8          001000      MD.SEC= 001000      ;SUPPRESS ERROR CORRECTION
9          000400      MD.SER= 000400      ;SUPPRESS ERROR RECOVERY
10         000200      MD.SSH= 000200      ;SUPPRESS SHADOWING
11         000100      MD.WBN= 000100      ;WRITE-BACK (NON-VOLATILE)
12         000040      MD.WBV= 000040      ;WRITE BACK (VOLATILE)
13         000001      MD.SPD= 000001      ;SPIN-DOWN
14         000001      MD.FEU= 000001      ;FLUSH ENTIRE UNIT
15         000002      MD.VOL= 000002      ;VOLATILE ONLY
16         000001      MD.NXU= 000001      ;NEXT UNIT
17
18         .SBTTL END PACKET FLAGS
19
20         000200      EF.BBR= 000200      ;BAD BLOCK REPORTED
21         000100      EF.BBU= 000100      ;BAD BLOCK UNREPORTED
22         000040      EF.LOG= 000040      ;ERROR LOG GENERATED
23         000020      EF.SEX= 000020      ;SERIOUS EXCEPTION
24
25         .SBTTL UNIT FLAGS
26
27
28         000001      UF.CMR= 000001      ;COMPARE READS
29         000002      UF.CMW= 000002      ;COMPARE WRITES
30         010000      UF.RPL= 010000      ;HOST INITIATED BAD BLOCK REPLACEMENT
31         040000      UF.INA= 040000      ;INACTIVE SHADOW SET UNIT
32         000200      UF.RMV= 000200      ;REMOVEABLE MEDIA
33         004000      UF.SCH= 004000      ;SUPPRESS CACHING (HIGH SPEED)
34         002000      UF.SCL= 002000      ;SUPPRESS CACHING (LOW SPEED)
35         000040      UF.WBN= 000040      ;WRITE-BACK (NON-VOLATILE)
36         020000      UF.WPH= 020000      ;WRITE PROTECT(HARDWARE)
37         010000      UF.WPS= 010000      ;WRITE PROTECT(SOFTWARE OR VOLUME)
38         000004      UF.576= 000004      ;576 BYTE SECTORS

```

```

1          .SBTTL CONTROLLER FLAGS
2
3          000200      CF.AVL= 000200      ;ENABLE AVAILABLE ATTENTION MESSAGES
4          000100      CF.MSC= 000100      ;ENABLE MISCELLANEOUS ERROR LOG MESSAGES
5          000040      CF.OTH= 000040      ;ENABLE OTHER HOST'S ERROR LOG MESSAGES
6          000020      CF.THS= 000020      ;ENABLE THIS HOST'S ERROR LOG MESSAGES
7          000002      CF.SHD= 000002      ;SHADOWING
8          000001      CF.576= 000001      ;576 BYTE SECTORS
9
10         .SBTTL COMMAND PACKET OFFSETS
11
12         ;          GENERIC COMMAND PACKET OFFSETS:
13         000000      P.CRF= 0.          ;COMMAND REFERENCE NUMBER
14         000004      P.UNIT= 4.         ;UNIT NUMBER
15         000010      P.OPCD= 8.         ;OPCODE
16         000012      P.MOD= 10.        ;MODIFIERS
17         000014      P.BCNT= 12.       ;BYTE COUNT
18         000020      P.BUFF= 16.       ;BUFFER DESCRIPTOR
19         000020      P.ADPA= 16.       ;BUFFER'S PHYSICAL ADDRESS (P.BUFF)
20         000022      P.ADEA= 18.       ;BUFFER'S EXTENDED ADDRESS (P.BUFF*2)
21         000034      P.LBN= 28.        ;LOGICAL BLOCK NUMBER
22         000040      P.SFTW= 32.       ;SOFTWARE WORDS
23
24         ;          ABORT AND GET COMMAND STATUS COMMAND PACKET OFFSETS:
25         000014      P.OTRF= 12.       ;OUTSTANDING REFERENCE NUMBER
26
27         ;          ONLINE AND SET UNIT CHARACTERISTICS COMMAND PACKET OFFSETS:
28         000016      P.UNFL= 14.       ;UNIT FLAGS
29         000020      P.HSTI= 16.       ;HOST IDENTIFIER
30         000024      P.UNTI= 20.       ;UNIT IDENTIFIER
31         000034      P.ELGF= 28.       ;ERROR LOG FLAGS
32         000040      P.SHUN= 32.       ;SHADOW UNIT
33         000042      P.CPSP= 34.       ;COPY SPEED
34
35         ;          REPLACE COMMAND PACKET OFFSETS:
36         000014      P.RBN= 12.        ;REPLACEMENT BLOCK NUMBER
37
38         ;          SET CONTROLLER CHARACTERISTICS COMMAND PACKET OFFSETS:
39         000014      P.VRSN= 12.       ;MSCP VERSION
40         000016      P.CNTF= 14.       ;CONTROLLER FLAGS
41         000020      P.HTMO= 16.       ;HOST TIMEOUT
42         000022      P.USEF= 18.       ;USE FRACTION
43         000024      P.TIME= 20.       ;QUAD-WORD TIME AND DATE
44
45         ;          MAINTENANCE READ AND MAINTENANCE WRITE COMMAND PACKET OFFSETS:
46         000034      P.RGID= 28.       ;REGION ID
47         000040      P.RGOF= 32.       ;REGION OFFSET

```

END PACKET OFFSETS

```

1
2
3
4          000000      ;
5          000004      P.CRF= 0.          ;COMMAND REFERENCE NUMBER
6          000010      P.UNIT= 4.         ;UNIT NUMBER
7          000011      P.OPCD= 8.         ;OPCODE (ALSO CALLED ENDCODE)
8          000012      P.FLGS= 9.         ;END PACKET FLAGS
9          000014      P.STS= 10.        ;MODIFIERS
10         000034      P.BCNT= 12.       ;BYTE COUNT
11         000040      P.FBBK= 28.      ;FIRST BAD BLOCK
12
13         ;
14         000014      P.OTRF= 12.      ;GET COMMAND STATUS END PACKET OFFSETS:
15         000020      P.CMST= 16.     ;OUTSTANDING REFERENCE NUMBER
16
17         ;
18         000014      P.MLUN= 12.      ;GET UNIT STATUS END PACKET OFFSETS:
19         000016      P.UNFL= 14.     ;MULTI-UNIT CODE
20         000020      P.HSTI= 16.     ;UNIT FLAGS
21         000024      P.UNTI= 20.     ;HOST IDENTIFIER
22         000040      P.SHUN= 32.     ;UNIT IDENTIFIER
23         000042      P.SHST= 34.     ;SHADOW UNIT
24         000044      P.TRCK= 36.     ;SHADOW STATUS
25         000046      P.GRP= 38.      ;TRACK SIZE
26         000050      P.CYL= 40.      ;GROUP SIZE
27         000054      P.RCTS= 44.     ;CYLINDER SIZE
28         000056      P.RBNS= 46.     ;RCT TABLE SIZE
29         000057      P.RCTC= 47.     ;RBN / TRACK
30
31         ;
32         000014      P.MLUN= 12.      ;ONLINE AND SET UNIT CHARACTERISTICS
33         000016      P.UNFL= 14.     ;MULTI-UNIT CODE
34         000020      P.HSTI= 16.     ;UNIT FLAGS
35         000024      P.UNTI= 20.     ;HOST IDENTIFIER
36         000040      P.SHUN= 32.     ;UNIT IDENTIFIER
37         000044      P.UNSZ= 36.     ;SHADOW UNIT
38         000050      P.VSER= 40.     ;UNIT SIZE
39
40         ;
41         000014      P.VRSN= 12.     ;SET CONTROLLER CHARACTERISTICS END PACKET OFFSETS:
42         000016      P.CNTF= 14.     ;MSCP VERSION
43         000020      P.CTMO= 16.     ;CONTROLLER FLAGS
44         000022      P.CNCL= 18.     ;CONTROLLER TIMEOUT
45         000024      P.CNTI= 20.     ;CONTROLLER COMMAND LIMIT
46         000034      P.MEDI= 28.     ;CONTROLLER ID
47         000042      P.SHST= 34.     ;MEDIA TYPE
48
49         ;ERROR LOG ATTENTION MESSAGE PACKET OFFSETS
50
51         000000      P.CRF= 0.          ;COMMAND REFERENCE NUMBER
52         000004      P.UNIT= 4.         ;UNIT NUMBER
53         000006      P.CNT= 6.         ;COUNT
54         000010      P.OPCD= 8.         ;OPCODE
55         000011      P.FLGS= 9.         ;ERROR LOG FLAGS
56         000012      P.SZOF= 10.      ;SIZE OR OFFSET
57         000014      P.LGDT= 12.      ;START OF ERROR LOG DATA

```



```

1          .SBTTL ERROR LOG FLAGS
2
3          000200      EF.FRS= 000200      ;FIRST PACKET
4          000100      EF.LST= 000100      ;LAST PACKET
5          000001      EF.MIS= 000001      ;MESSAGE MISSING
6
7          ;ERROR LOG MESSAGE OFFSETS
8
9          000000      L.EVNT= 0.          ;EVENT CODE
10         000002      L.SLOT= 2.         ;SLOT NUMBER
11         000004      L.CNTI= 4.         ;CONTROLLER IDENTIFIER
12         000014      L.CNTI= 12.        ;CONTROLLER SOFTWARE REVISION
13         000015      L.CHVR= 13.        ;CONTROLLER HARDWARE REVISION
14         000016      L.UNTI= 14.        ;UNIT IDENTIFIER
15         000026      L.USVR= 22.        ;UNIT SOFTWARE REVISION
16         000027      L.UHVR= 23.        ;UNIT HARDWARE REVISION
17         000030      L.ERLC= 24.        ;ERROR LOCATION
18         000034      L.CYL= 28.         ;CYLINDER
19         000040      L.GRP= 32.         ;GROUP
20         000041      L.TRCK= 33.        ;TRACK
21         000042      L.SCTR= 34.        ;SECTOR
22         000044      L.VSER= 36.        ;VOLUME SERIAL NUMBER
23         000050      L.DATA= 40.        ;EVENT DEPENDENT DATA
24
25         ;STATUS AND EVENT COE DEFINITIONS
26
27         000037      ST.MSK= 37          ;STATUS / EVENT CODE MASK
28         000040      ST.SUB= 40          ;SUB-CODE MULTIPLIER
29         000000      ST.SUC= 0           ;SUCCESS
30         000001      ST.CMD= 1           ;INVALID COMMAND
31         000002      ST.ABO= 2           ;COMMAND ABORTED
32         000003      ST.OFL= 3           ;UNIT-OFFLINE
33         000004      ST.AVL= 4           ;UNIT-AVAILABLE
34         000005      ST.MFE= 5           ;MEDIA ERROR
35         000006      ST.WPR= 6           ;WRITE PROTECTED
36         000007      ST.CMP= 7           ;COMPARE ERROR
37         000010      ST.DAT= 10          ;DATA ERROR
38         000011      ST.HST= 11          ;HOST BUFFER ACCESS ERROR
39         000012      ST.CNT= 12          ;CONTROLLER ERROR
40         000013      ST.DRV= 13          ;DRIVE ERROR
41         000037      ST.DIA= 37          ;MESSAGE FROM AN INTERNAL DIAGNOSTIC
42
43         ;
44         ;          SUBCODES FOR ST.OFL
45         ;
46         000040      SC.NVL = 40          ;NO VOLUME MOUNTED
47         ;          ; OR DRIVE DISAVLED VIA RUN/STOP SWITCH
48         000100      SC.IOP = 100        ;UNIT INOPERATIVE
49         000400      SC.DIS = 400        ;UNIT DISABLED BY FIELD SERVICE
50         ;          ; OR INTERNAL DIAGNOSTIC
51         000200      SC.DUP = 200        ;DUPLICATE UNIT NUMBER
52         ;
53         ;          SUBCODES FOR ST.DRV
54         ;
55         000040      SC.STO = 40          ;SDI RESPONCE TIME OUT
56         000100      SC.INV = 100        ;INVALID SDI RESPONCE

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57

```

.SBTTL MODULE CODE
*****
INIT VALUES
INIT CONTROLLER
XFER TO DISK?
  F FOR J = 1,CYCLE LIMIT
  MAITENANCE WRITE
  MAITENANCE READ
  CHECK DATA?
  T CHECK
NEXT J
  T FOR J = 1,CYCLE LIMIT
  GET UNIT STATUS
  IF DRIVE IS NOT AVAILABLE, WAIT UNTIL IT IS
  DRIVE THERE?
  F DROP
  ALL DRIVES DROPPED?
  T DROP MODULE
  F ---
  T ONLINE
  ONLINE?
  T PICK @LGCK - IF RANDOM, GET RAND @ MOD X
  ELSE INCREMENT
  IF LBN > LIMIT THEN LBN < 0
  WRITE
  READ
  CHECK DATA ?
  T CHECK
  AVAILABLE DRIVE(1)
  F TRY TO BRING ONLINE AGAIN
NEXT J
*****

```

```

*****
START CODE
IF THE CODE IS RESTARTED, CLEAR THE OLD ADDRESSES SO THE
THE CONTROLLER WILL GET REINITED.
*****

```

```

START:
INC @-1 ;FIRST TIME THRU HERE?
BNE 1$ ;BR IF NO
BIC @SR.XFR,SR1 ;DO NOT ALLOW DISK TRANSFERS
MSGN$,BEGIN,WARN1 ;ASCII MESSAGE CALL WITH COMMON HEADER
1$: BIT @SR.XFR,SR1 ;WILL CUSTOMER DATA BE OVERWRITTEN?
BEQ 2$ ;BR IF NO
MSGN$,BEGIN,WARN2 ;ASCII MESSAGE CALL WITH COMMON HEADER
54 000720 000403 BR 3$ ;
55 000722 2$:
56 000722 104403 000000' 005362' MSGN$,BEGIN,WARN3 ;ASCII MESSAGE CALL WITH COMMON HEADER
57 000730 3$:

```

MODULE CODE

58	000730	005067	177210	CLR	CDERCT	; CLEAR DATA CHECK ERROR COUNT
59	000734	012767	177777 177522	MOV	#177777,EXPV	; NOT EXPECTING AN INTERRUPT
60	000742	012767	000017 177512	MOV	#PRTNUM,PRNMSG	; INITIALIZE PRINT WORD
61	000750	016767	177040 177622	MOV	DVID1,DVICE	; DVICE HAS DESIRED BITS SET
62	000756	005067	177632	CLR	TABLEW	; SET TABLE FOR UNIT 0
63	000762	012767	000001 177626	MOV	#1,TABLEW*2	; SET TABLE FOR PORTID FOR UNIT 0
64	000770	005067	177274	CLR	CMDREF	; COMMAND REF # = 0
65	000774	104417	000000	RAND\$,BEGIN		
66	001000	016767	177050 177560	MOV	RANUM,SECL	
67	001006	005067	177556	CLR	SECH	; FOR RESTARTING (INITIAL SECTOR ADDR)
68	001012	016767	176770 177232	MOV	ADDR,SAREG	; STORE IN SA REG
69	001020	062767	000002 177224	ADD	#2,SAREG	; SA REGISTER HAS PROPER ADDRESS
70	001026	005067	177424	CLR	OLDPA	; OLD PHYSICAL ADDRESS CLEARED
71	001032	005067	177422	CLR	OLDEA	; OLD EXTENDED ADDRESS CLEARED
72						; FOR RESTARTING. THIS WILL FORCE A
73						; CONTROLLER REINIT TO TAKE PLACE

```

1
2
3
4
5
6
7
8
9
10
11
12
13 001036
14 001036 012767 000260' 177376
15 001044 104415 000000' 000442'
16 001052 026767 177366 177376
17 001060 001004
18 001062 026767 177360 177370
19 001070 001412
20 001072 016767 177346 177356
21 001100 016767 177342 177352
22 001106 004767 000332
23 001112 005067 177466
24 001116
25 001116 032767 000010 176672
26 001124 001034
27 001126 026767 177330 176700
28 001134 001030
29 001136 062767 000017 177316
30
    001144 104421 000000' 000042'
    001152 000475'
31 001154 105067 177322
32
    001160 104421 000000' 000044'
    001166 000504'
33 001170 105067 177315
34
    001174 104421 000000' 000144'
    001202 000466'
35 001204 105067 177263
36 001210 104403 000000' 005314'
37 001216 012777 004334' 176564 1$:
38
39 001224 104415 000000' 000124'
40 001232 016700 176672
41 001236 004767 000572
42 001242 010067 177202

```

```

;.....
;
; RESTART SEQUENCE
;
; CHECK THE ADDRESS OF THE RINGS TO SEE IF THEY WERE RELOCATED
; IF THEY WERE, REINIT THE CONTROLLER.
;
; GET THE NEW ADDRESSES. IF THE DISKLESS OPERATION IS DESIRED
; THEN DO THE MAINTENANCE WRITE AND READ. ELSE DO THE WRITE
; AND READ WITH A DRIVE.
;.....
RESTR:
MOV @RSPONC,VA ;DID THE RINGS RELOCATE?
GETPA$,BEGIN,VA ;GET PHYSICAL ADDRESS FROM 16 BIT VA
CMP PA,OLDPA ;IS THE OLD PHYS ADDR = NEW ONE?
BNE RESTR2 ;IF SO, REINIT
CMP EA,OLDEA ;IS THE OLD EXTN ADDR = NEW ONE?
BEQ RESTR1 ;IF NOT, DON'T REINIT
RESTR2: MOV PA,OLDPA ;ELSE SET THE OLD RING ADDR
MOV EA,OLDEA ;AND THE OLD EXTENDED ADDR
JSR PC,INITUD ;AND INIT THE CONTROLLER
CLR TRY ;CLEAR RETRY COUNT
RESTR1: BIT @SR.SUM,SR1 ;DO WE WANT THE REPORT?
BNE 1$ ;IF NOT, SKIP THE REPORT
CMP PRNMSG,PASCNT ;DO WE PRINT?
BNE 1$ ;IF PASS COUNT IS NOT = PRINT WORD, SKIP
ADD @PRTNUM,PRNMSG ;PRINT WORD IS INCREMENT
;.....
;CONVERT SOFCNT TO ASCII AND
;STORE AT ADR2
BTOD$,BEGIN,SOFcnt,ADR2
;.....
CLRB ADR2+5
;.....
;CONVERT HRCNT TO ASCII AND
;STORE AT ADR3
BTOD$,BEGIN,HRCNT,ADR3
;.....
CLRB ADR3+5
;.....
;CONVERT CDERTCT TO ASCII AND
;STORE AT ADR1
BTOD$,BEGIN,CDERTCT,ADR1
;.....
CLRB ADR1+5
MSGN$,BEGIN,ERRPAS ;ASCII MESSAGE CALL WITH COMMON HEADER
MOV @NTRUPT,@VECTOR ;GET VECTOR ADDRESS
;SET POINTER
GETPA$,BEGIN,RBUFVA ;GET PHYSICAL ADDRESS FROM 16-BIT RBUFVA
MOV RBUFEA,RO ;GET EA TO ADJUST
JSR PC,ASRO4 ;GO ADJUST IT
MOV RO,RBFFEA ;PUT ADJUSTED VALUE IN A SAVE AREA

```

```

43 001246 005067 177330 CLR UNITNO ;PRESET UNIT #
44 001252 032767 000002 176536 BIT #SR,XFR,SR1 ;DISK XFER???
45 001260 001454 BEQ MA10NC ;NO! DO MAINTENANCE (DISKLESS) ROUTINES
46 ;*****
47 ; DO THE DISK OPERATIONS ;
48 ; CHECK TO SEE WHICH PORTS ARE AVAILABLE ;
49 ;*****
50
51 001262 004767 001032 JSR PC,SETUP ;FIND DRIVES/SET UP TABLE
52 001266 005767 177306 TST DVICE ;ELSE, TEST FOR ANY MORE DRIVES
53 001272 001002 BNE LOOP1 ;IF TRUE, DO A CYCLE
54
55 001274 104410 000000' END$,BEGIN ;
56
57 001300 LOOP1: ;
58 001300 104414 000000' GMBUF$, BEGIN ;GET WRITE BUFFER INFORMATION
59 001304 016700 176626 MOV WBUFEA,RO ;GET WRITE BUFFER INFORMATION
60 001310 004767 000520 JSR PC,ASR04 ;ADJUST IT
61 001314 010067 177132 MOV RO,WBFFEA ;STORE EA IN SAVE AREA
62 001320 012704 000614' MOV #TABLEW,R4 ;R4 -> TABLE OF UNITNO AND PORTID
63 001324 012703 000001 MOV #1,R3 ;R3 IS AN INDEX TO DVICE
64 001330 LOOP2: ;
65 001330 030367 177244 BIT R3,DVICE ;HAS THE DRIVE BEEN DROPPED
66 001334 001412 BEQ 9$ ;IF SO, SKIP THIS DRIVE
67 001336 016467 000002 177242 MOV 2(R4),PORTID ;SET UP PORTID
68 001344 011467 177232 MOV (R4),UNITNO ;SET UP UNITNO
69 ; *** DO A DISK CYCLE
70 001350 004767 001250 JSR PC,CYCLED ;DO A CYCLE FOR DISK OPERATION
71 001354 103002 BCC 9$ ;IF SUCCESSFUL, CONTINUE
72 001362 001362 9$: JSR PC,DROP1 ;IF NOT, DROP DRIVE
73 001362 062704 000004 ADD #4,R4 ;POINT TO NEXT ENTRY OF THE TABLE
74 001366 006303 ASL R3 ;R3 POINTS TO NEXT BIT
75 001370 022704 000654' CMP #TEND,R4 ; POINT BEYOND LAST ENTRY?
76 001374 001403 BEQ 12$ ; IF NOT, THEN TRY AGAIN.
77 001376 020367 177176 CMP R3,DVICE ;IF R3 > DVICE THEN DONE WITH ITERATION
78 001402 003752 BLE LOOP2 ;IF < OR =, LOOP
79 001404 12$:
80 001404 104413 000000' ENDIT$,BEGIN ;SIGNAL END OF ITERATION.
81 001410 000733 BR LOOP1 ;MONITOR SHALL TEST END OF PASS
82 ;AND DO AGAIN
83
84 ;*****
85 ; MAINTENANCE ROUTINE, DO THE DISKLESS CODE ;
86 ;*****
87
88 001412 MA10NC: ;
89 001412 104414 000000' GMBUF$, BEGIN ;GET WRITE BUFFER INFORMATION
90 001416 016700 176514 MOV WBUFEA,RO ;GET EA TO ADJUST
91 001422 004767 000406 JSR PC,ASR04 ;ADJUST IT
92 001426 010067 177020 MOV RO,WBFFEA ;STORE EA IN SAVE AREA
93 001432 004767 001464 JSR PC,CYCLED
94 001436 104413 000000' ENDIT$,BEGIN ;SIGNAL END OF ITERATION.
;MONITOR SHALL TEST END OF PASS

```

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22 001444 012767 000260' 176770 INITUD: MOV #RSPONC,VA ;VA -> RSPONC
23 001452 104415 000000' 000442' GETPA$,BEGIN, VA ;GET PHYSICAL ADDRESS FROM 16-BIT VA
24 001460 005004 CLR R4 ;R4 IS USED IF AN ERROR IS DETECTED
25 001462 012702 000001 MOV #1,R2 ;R2 = STEP INDICATOR REG FOR MSG'S
26 001466 005077 176314 CLR @ADDR ;WRITE TO IP REGISTER TO INIT CONTROLLER
27 001472 012701 002260 MOV #TIMER,R1 ;SET TIME OUT LIMIT
28 001476 017700 176550 1$: MOV @SAREG,RO ;RO HAS SA REGISTER DATA
29 001502 032700 100000 BIT @<SA.ERR>,RO ;CHECK FOR ERROR
30 001506 001007 BNE 2$ ;IF FOUND, GET OUT OF LOOP
31 001510 104407 000000' BREAK$,BEGIN ;TEMPORARY RETURN TO MONITOR....
   001514 104407 000000' BREAK$,BEGIN ;THEN CONTINUE AT NEXT INSTRUCTION.
32 001520 005301 DEC R1 ;TIME OUT?
33 001522 001365 BNE 1$ ;IF NOT, LOOP
34 001524 000404 BR 4$ ;IF DONE, CONTINUE
35 001526 012703 004000 2$: MOV @SA.S1,R3 ;R3 = STEP 1 BIT
36 001532 000167 000412 JMP ERROR1 ;IF HERE, ERROR
37 001536 042700 001140 4$: BIC @<SA.Q22+SA.MAP+SA.SM>,RO ;CLEAR KDA50-Q DEPENDENT BITS
38 001542 022700 004400 CMP @<SA.S1+SA.DIA>,RO ;DID DATA COMPARE PROPERLY?
39 001546 001402 BEQ 5$ ;IF SO, CONTINUE
40 001550 000167 000370 JMP ERROR3 ;REPORT ERROR
41 ; STEP 2
42 001554 016705 176230 5$: MOV VECTOR,R5 ;VECTOR GIVEN
43 001560 006205 ASR R5 ;SET TO APPROPRIATE VALUE
44 001562 006205 ASR R5 ; = VECTOR/4
45 001564 052705 100200 BIS @<SA.INT+BIT15>,R5 ;ACTIVATE INTERRUPTS & SET MSB FOR STEP 1
46 ;LEN'S ARE 0
47 001570 010500 MOV R5,RO ;STORE R5 IN RO FOR SUBROUTINE
48 001572 012703 004000 MOV @SA.S1,R3 ;R3 HAS STEP BIT FOR SUBROUTINE
49 001576 004767 000244 JSR PC,SNDS1P ;SEND STEP DATA
50 001602 042705 100000 BIC @BIT15,R5 ;CLEAR MSB FOR COMPARE DATA
51 001606 042700 000200 BIC @BIT07,RO ;WAS BIT07 ONLY BIT SET?. SHOULD BE
52 001612 001404 BEQ 6$
53 001614 052700 010200 BIS @<SA.S2+BIT07>,RO ;SET RO TO REPORT THE ERROR
54 001620 000167 000320 JMP ERROR3 ;REPORT ERROR
55 001624 016700 176614 6$: MOV PA,RO ;RO GETS PHYSICAL ADDRESS
56 001630 004767 000212 JSR PC,SNDS1P ;SEND STEP DATA

```

MODULE CODE

```

57 001634 042705 177400      BIC    #177400,R5      ;HIGH BYTE CLEARED
58 001640 020500              CMP    R5,R0           ;CHECK ECHO DATA
59 001642 001402              BEQ    7$              ;IF OK, SKIP
60 001644 000167 000274      JMP    ERROR3          ;IF NOT, REPORT ERROR
61 001650                      7$:
62                          ;
63 001650 016700 176572      MOV    EA,R0           ;ADJUST THE EXTENDED ADDRESS BITS
64 001654 004767 000154      JSR    PC,ASR04        ;SHIFT EXTENDED ADDRESS BITS FOR CONTROLLER
65 001660 004767 000162      JSR    PC,SNSTP        ;SEND STEP DATA
66 001664 012700 000254      MOV    #RSPONC-4,R0   ;RO -> RING ENVELOP
67                          ;
68 001670 005720              8$:
69 001672 001117              TST    (R0).           ;IS THE RING ENTRY = 0?
70 001674 022700 000270      BNE    ERROR5         ;IF NOT, ERROR
71 001700 001373              CMP    #CMDFREF,R0    ;IS RO POIN; PAST THE RINGS?
72 001702 016700 176112      BNE    8$             ;IF NOT, LOOP
73 001706 000241              MOV    SR2,R0         ;RO = BURST VALUE
74 001710 006300              CLC                    ;CLEAR CARRY
75 001712 006300              ASL    RO              ;ALIGN BURST FOR STEP 4
76 001714 052700 000001      ASL    RO              ;
77 001720 010077 176326      BIS    #SA.GO,R0      ;SET GO BET
78 001724 012767 000362' 176510  MOV    RO,#SAREG      ;SEND DATA TO CONTROLLER/INIT DONE
79 001732 104415 000000' 000442'  MOV    #CMPACK,VA     ;GET COMMAND PACKET PA AND EA
80 001740 016767 176500 176316  GETPA$,BEGIN,VA      ;GET PHYSICAL ADDRESS FROM 16-BIT VA
81 001746 016700 176474      MOV    PA,COMMAND     ;STORE ADDRESS IN THE RING
82 001752 004767 000056      MOV    EA,R0           ;SAVE IN RO
83 001756 010067 176304      JSR    PC,ASR04        ;SHIFT EXTENDED ADDRESS BITS FOR CONTROLLER
84                          MOV    RO,COMMAND*2     ;MOVE ADJUSTED EA INTO RING
85 001762 012767 000276' 176452  MOV    #RSPACK,VA     ;GET RESPONCE PACKET PA AND EA
86 001770 104415 000000' 000442'  GETPA$,BEGIN,VA      ;GET PHYSICAL ADDRESS FROM 16-BIT VA
87 001776 016767 176442 176254  MOV    PA,RSPONC      ;STORE ADDRESS IN THE RING
88 002004 016700 176436      MOV    EA,R0           ;SAVE IN RO
89 002010 004767 000020      JSR    PC,ASR04        ;SHIFT EXTENDED ADDRESS BITS FOR CONTROLLER
90 002014 010067 176242      MOV    RO,RSPONC*2     ;MOVE ADJUSTED EA INTO RING
91 002020 012777 004334' 175762  MOV    #INTRUPT,@VECTOR ;STORE INTERRUPT ADDRESS IN VECTOR
92 002026 005067 176552      CLR    TRY             ;CLEAR TRY SO DRIVE WILL
93                          ;GO BACK ONLINE IF NECESSARY
94 002032 000207              RTS    PC
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109 002034
110 002034 006200
111 002036 006200
112 002040 006200
113 002042 006200

```

```

*****
;
; ASR04
; ARITHMETIC SHIFT RIGHT REG 0 FOUR TIMES
;
; EXTENDED ADDRESS BITS (16 & 17) ARE SET IN BIT POSITION 4 & 5
; RESPECTIVELY. SHIFT RIGHT FOUR TIMES TO REPOSTION THE VALUE
;
; INPUT RO = UNADJUSTED EXTENDED ADDRESS BITS
;
; OUTPUT RO = ADJUSTED EXTENDED ADDRESS BITS
;
*****
ASR04:
ASR    RO      ;SHIFT 10
ASR    RO      ;SHIFT 4
ASR    RO      ;SHIFT 2
ASR    RO      ;SHIFT 1

```

```

114 002044 000207          RTS      PC          ;RETURN
115
116          ;*****
117          ;
118          ; SEND STEP DATA
119          ;
120          ; INPUT:  R0 HAS DATA TO BE SENT TO CONTROLLER FOR STEP
121          ;          R3 HAS PREVIOUS STEP FLAG SET
122          ;
123          ; OUTPUT: R0 HAS DATA SENT FROM CONTROLLER TO HOST FOR ECHO AND NEXT STEP
124          ;          R3 HAS CURRENT STEP FLAG SET
125          ;
126          ;*****
127 002046 016701 175736  SNOSTP: MOV     VECTOR,R1          ;
128 002052 012721 002072'  MOV     @INTA,(R1)+        ;SET UP INTERRUPT HANDLER ADDRESS
129 002056 116711 175730  MOV     BR1,(R1)          ;SET PRIORITY LEVEL
130 002062 010077 176164  MOV     R0,@SAREG         ;SEND STEP1 WRITE FORMMATED DATA
131
132 002066 104400 000000'  EXIT$,BEGIN              ;EXIT TO MONITOR. MODULE WAIT FOR INTERRUPT.
133
134 002072          INTA:
135          002072 000004 000000' 002100'  PIRQ$,BEGIN,3$          ; QUEUE UP TO CONTINUE AT 3$ AND RTI
          ;-----
136 002100          3$:
137 002100 017700 176146  MOV     @SAREG,R0          ;GET STEP N FORMATTED DATA
138 002104 032700 100000  BIT     @SA.ERR,R0        ;TEST FOR ERROR
139 002110 001017          BNE     ERROR1            ;IF NOT OK, REPORT
140 002112 005202          INC     R2                ;SET STEP REGISTER
141 002114 006303          ASL     R3                ;R3 HAS STEP BIT PROPERLY SET
142 002116 030300          BIT     R3,R0            ;WAS STEP N SET?
143 002120 001002          BNE     4$                ;IF SO, CONTINUE
144 002122 000167 000020  JMP     ERROR2            ;IF NOT CORRECT STEP, ERROR
145 002126 040300          4$:  BIC     R3,R0            ;CLEAR THE STEP BIT, FOR COMPARE
146 002130 000207          RTS      PC          ;RETURN

```


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32

33
34

35
36
37
38
39
40
41
42
43
44
45
46

```

;*****
;
; ERROR 1
; PRINT AN ERROR REPORTED BY THE CONTROLLER DIAGNOSTICS
;
; ERROR2
; PRINT THE VALUE OF THE SA REGISTER WHEN THE STEP BIT WAS NOT SET
;
; ERROR3
; PRINT A THE VALUE OF THE SA REGISTER WHEN THE ECHO WAS NOT SET
;   CORRECTLY
;
; INPUT  R0 -> SA REGISTER
;        R2 = STEP COUNT
;
; OUTPUT THE RETRY COUNT IS INCREMENTED
;        IF THE RETRY COUNT > RETRY LIMIT, END MODULE
;
; ERRORS
; RING WASN'T ALL ZERO -> ERROR
; DROP UDBAO
;*****

```

```

ERRORS:
002132 002132 104403 000000' 005372' MSGN$,BEGIN,ZERO ;ASCII MESSAGE CALL WITH COMMON HEADER
002140 002140 104410 000000' END$,BEGIN ;
;
ERROR3: INC R4 ;R4 = 3 FOR ERROR3
ERROR2: INC R4 ;R4 = 2 FOR ERROR2
ERROR1: INC R4 ;R4 = 1 FOR ERROR1
002152 010267 176276 MOV R2,NUM ;STORE STEP REG IN A NUMBER FOR CONVRT
;*****
;CONVERT NUM TO ASCII AND
;STORE AT ADR2
002156 104420 000000' 000454' OTOA$,BEGIN,NUM,ADR2
002164 000475'
;*****
002166 017767 176060 176260 MOV BSAREG,NUM ;STORE VALUE IN A NUMBER
34 ;*****
;CONVERT NUM TO ASCII AND
;STORE AT ADR1
002174 104420 000000' 000454' OTOA$,BEGIN,NUM,ADR1
002202 000466'
;*****
35
36 002204 005304 DEC R4 ;ERROR 1?
37 002206 001003 BNE 1$ ;IF NOT, CHECK IF IT IS THE NEXT ERROR
38 002210 104403 000000' 005202' MSGN$,BEGIN,INITE1 ;ASCII MESSAGE CALL WITH COMMON HEADER
39 002216 1$:
40 002216 005304 DEC R4 ;ERROR 2?
41 002220 001003 BNE 2$ ;IF NOT, CHECK IF IT IS THE NEXT ERROR
42 002222 104403 000000' 005226' MSGN$,BEGIN,INITE2 ;ASCII MESSAGE CALL WITH COMMON HEADER
43 002230 2$:
44 002230 005304 DEC R4 ;ERROR 3?
45 002232 001003 BNE 3$ ;IF NOT, CHECK IF IT IS THE NEXT ERROR
46 002234 104403 000000' 005234' MSGN$,BEGIN,INITE3 ;ASCII MESSAGE CALL WITH COMMON HEADER

```

```

47 002242          3$:
48                ;*****
                ;CONVERT ADDR TO ASCII AND
                ;STORE AT ADR3
                OTOA$,BEGIN,ADDR,ADR3
                ;*****
                ;*****
49                ;*****
002242 104420 000000' 000006' 002250 000504'
                HRDR$,BEGIN,NULL
                ;*****
50 002260 104403 000000' 005210' MSGN$,BEGIN,INITER ;ASCII MESSAGE CALL WITH COMMON HEADER
51 002266 005267 176312          INC TRY ;INCREMENT RETRY COUNT
52 002272 022767 000004 176304  CMP #RLIM,TRY ;IS THE RETRY COUNT EXCEEDED?
53 002300 001402          BEQ 6$ ;IF SO, END IT
54 002302 000167 176352          JMP START ;IF NOT, TRY AGAIN
55 002306          6$:
002306 104403 000000' 005366' MSGN$,BEGIN,ABORT ;ASCII MESSAGE CALL WITH COMMON HEADER
56 002314 104410 000000'          END$,BEGIN
57

```

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16 002320
17
18 002320 004767 001650
19 002324 005367 175740
20 002330 001110
21 002332 012703 000001
22 002336 012704 000614'
23 002342 011467 176234
24 002346 016714 176230 1$:
25 002352 010367 176230
26 002356 010364 000002
27 002362 012764 177777 000004
28 002370 016464 000004 000006
29 002376 012767 002400 176206
30 002404 004767 001526 3$:
31 002410 103006
32 002412 005367 176174
33 002416 001372
34 002420 004767 000774 5$:
35 002424 000437
36 002426 016767 175650 176146 7$:
37
38
39 002434 012702 000614'
40 002440 012705 000001
41 002444 020227 000654' 9$:
42 002450 001420
43 002452 020305
44 002454 001416
45 002456 026712 176120
46 002462 001404
47 002464 062702 000004 11$:
48 002470 006305
49 002472 000764
50 002474 011467 176102 13$:
51 002500 010367 176102
52 002504 004767 000720
53 002510 000405
54 002512 15$:
55
56 002512 026714 176064
57 002516 001402

```

```

;*****
;
;   SET UP
;
;   GO FIND OUT WHAT DRIVES ARE OUT THERE.
;   A TABLE IS FILLED WITH UNIT NUMBERS(MAX IS 16)
;
;   THIS SHOULD ONLY BE DONE AT THE VERY BEGINNING OF RUNNING
;   THIS DECX MODULE; THEN NOT RUN AGAIN.
;
;   INPUT:  DVICE HAS APPROPRIATE BITS SET.  THE # OF BITS =
;           # OF DRIVES WANTED TO TEST.
;           POSITION OF BITS = WHICH DRIVE IN THE SYSTEM IS DESIRED.
;*****
SETUP:
; *** SET CONTRL CHAR AND WAIT FOR THE ATTENTION MESSAGES
; JSR PC,SCC ;SET CONTROLLER CHARACTERISTICS
; DEC CMDREF ;ONLY SET UP AT BEGINNING OF MODULE
; BNE 19$ ; (USE DRIVES FOUND AT BEGINNING)
; MOV #1,R3 ;INITIAL PORTID VALUE
; MOV #TABLEW,R4 ;R4 -> TABLEW
; MOV (R4),UNITNO ;INITIAL UNITNO IN TABLEW
; MOV UNITNO,(R4) ;UNIT NO SET IN TABLEW;READY TO TEST
; MOV R3,PORTID ;PORT ID SET
; MOV R3,2(R4) ;PORTID SET IN TABLEW
; MOV #17777,4(R4) ;INSERT NEW -1,-1 FOR LAST ENTRY
; MOV 4(R4),6(R4) ; OF THE TABLEW
; MOV #2400,WORK ;WORK = RETRY LIMIT
; JSR PC,GTSTAT ;GET STATUS, GET NEXT UNIT NUMBER
; BCC 7$ ;OK, CONTINUE
; DEC WORK ;ELSE IF OFFLINE, DECR COUNT
; BNE 3$ ;IF COUNT > 0, TRY AGAIN.
; JSR PC,DROP2 ;DROP THE DRIVE
; BR 17$ ;TRY NEXT UNIT
; MOV P,UNIT+RSPACK,UNITNO ;UNIT NUMBER FROM RESPONCE PACKET IN UNITNO
; *** CHECK FOR CASE WHERE THE MORE UNITS THEN DRIVES HAVE BEEN SPECIFIED.
; *** NEXT UNIT MODIFIER WILL GIVE A DUPLICATE UNIT NUMBER.
; MOV #TABLEW,R2 ;R2 -> TABLE TO FIND DUPLICATE
; MOV #1,R5 ;R5 IS TEMP PORTID
; CMP R2,#TEND ;REACHED THE BOTTOM?
; BEQ 15$ ;IF SO, EXIT
; CMP R3,R5 ;REACHED THE LATEST ENTRY?
; BEQ 15$ ;IF SO, EXIT
; CMP UNITNO,(R2) ;DO WE HAVE A DUPLICATE UNIT NUMBER?
; BEQ 13$ ;IF SO, ERROR
; ADD #4,R2 ;IF NOT, POINT TO NEXT POINTER
; ASL R5
; BR 9$ ;AND CONTINUE
; MOV (R4),UNITNO ;DROP DRIVE FROM TABLE
; MOV R3,PORTID
; JSR PC,DROP3 ;AND DROP IT
; BR 17$
; ***
; CMP UNITNO,(R4) ;IS THE UNITNO CORRECT?
; BEQ 17$ ;IF SO, CHECK FOR NEXT UNIT

```

```

58 002520 016714 176056          MOV      UNITNO,(R4)          ;ELSE, CORRECT THE UNIT NUMBER IN TABLE
59 002524                          17$:  ASL      R3                ;NEXT PORTID SET
60 002524 006303                          CMP      DVICE,R3          ;DONE?
61 002526 026703 176046          BMI      19$              ;IF R3 > DVICE, ALL DESIRED DRIVES ARE FOUND
62 002532 100407                          INC      UNITNO           ;NEXT UNITNO SET
63 002534 005267 176042          ADD      @4,R4            ;POINT TO NEXT ENTRY TO TEST DRIVE
64 002540 062704 000004          CMP      @TEND,R4        ;POINT TO END? IF SO, TABLE FULL
65 002544 022704 000654          BHI      1$              ;IF R4 NOT REACHED END, GO TEST
66 002550 101276          19$:  RTS      PC
67 002552
68 002552 000207
69
70 ;*****
71 ;
72 ;       TSTOFL
73 ;       TEST TO SEE WHAT KIND OF AN OFFLINE CONDITION HAS OCCURED.
74 ;
75 ;*****
76 002554 022700 000003  TSTOFL:  CMP      @ST.OFL,R0      ;WAS THE DRIVE FOUND OFFLINE?
77 002560 001403          BEQ      10$              ;CHECK WHAT KIND OF OFFLINE
78 002562 022700 000013          CMP      @ST.DRV,R0      ;WAS IT A DRIVE ERROR? -> SDI?
79 002566 001012          BNE      13$              ;IF IT WAS NOT, ERROR (DROP DRIVE)
80 002570 032767 000740 175512 10$:  BIT      @<SC.NVL+SC.DIS+SC.DUP+SC.IOP>,P.STS+RSPACK ;WERE ANY OF THESE BITS SET?
81 ;       ; = NO VOLUME MOUNTED, UNIT DISABLED BY FIELD SREVICE
82 ;       ; OR DUPLICATE UNIT NUMBER OR UNIT INOPERATIVE
83 002576 001004          BNE      12$              ;IF SO, EXIT
84 002600 032767 177000 175502  BIT      @+C<SC.NVL+SC.DIS+SC.DUP+SC.IOP+ST.MSK>,P.STS+RSPACK ; ANY OTHER DATA?
85 002606 001002          BNE      13$              ;IF SO, DROP
86 002610 000241          12$:  CLC
87 002612 000207          RTS      PC              ;CLEAR CARRY
88 002614 000261          13$:  SEC                ;RETURN
89 ;       ;SET CARRY, DRIVE WAS FOUND TO BE OFFLINE
90 002616 004767 002124          JSR      PC,ERRORM      ; OR ANOTHER ERROR
91 002622 000207          RTS      PC              ;REPORT ERROR
                               ;RETURN

```

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18 002624
19 002624 032767 001000 175164
20 002632 001004
21
22 002634 005767 175744
23 002640 100443
24 002642 000422
25
26
27
28
29 002644 012701 000010
30 002650 004767 001262
31 002654 103013
32 002656 004767 177672
33 002662 103507
34
35 002664 005067 175574
36 002670 052767 140000 175364
37 002676 004767 001426
38
39 002702 000402
40 002704 005301
41 002706 001360
42 002710 004767 001316
43 002714 103753
44 002716 016767 175422 175650
45 002724 016767 175412 175640
46 002732 001006
47 002734 005767 175634
48 002740 001731
49
50 002742 012767 100000 175634
51
52
53
54
55
56
57

```

```

*****
;
; CYCLE DISK
;
; DO THE DISK CYCLE
; DO GET STATUS COMMANDS TO ASSURE THAT THE DRIVE
; IS AVAILABLE (FOR DUAL PORTING)
; CHECK DRIVE TO BE ONLINE
; IF TRUE
; PICK THE BLOCK
; WRITE
; READ
; DATA CHECK
; MAKE THE DRIVE AVAILABLE
; ELSE DROP DRIVE
;
*****
CYCLED:
; BIT #SR.DUA,SHL ;DUAL PORT?
; BNE 2# ;IF NOT, CONTINUE
; *** CHECK IF WE DO ONLINE FOR THE FIRST TIME.
; TST TRY ;IF TRY HAS SET MSB, DON'T DO ONLINE
; BMI 16# ; DON'T DO ONLINE
; BR 10# ; ELSE DO ONLINE (1ST TIME THROUGH IN THIS PASS)
;
; ***
; *** DO GET STATUS COMMANDS TO ASSURE THE DRIVE IS AVAILABLE TO THE CONTROLLER
; *** FOR DUAL PORTING.
; ***
2# : MOV #10,R1 ;R1 = # OF GET STATUS TO DO
4# : JSR PC,GTSTAT ;IS THE DRIVE OFFLINE?
; BCC 6# ;IF ALL OK, DO THE CYCLE
; JSR PC,TSTOFL ;ELSE, CHECK IF OFFLINE
; BCS 24# ;IF IT ERRED, DROP THE DRIVE
; *** HANDLE OFF LINE DRIVE, WAIT FOR AVAILABLE ATTENTION MESSAGE
; CLR EXPAV ;EXPECT AN AVAILABLE ATTENTION MESSAGE
; BIS #<RG.OWN+RG.FLG>,RSPONC+2 ;SET RING FOR ATTN MESSAGE
; JSR PC,INTERP ;WAIT FOR MESSAGE
; ; 2ND ATTENTION MESSAGE
;
; BR 10#
6# : DEC R1 ;DONE?
; BNE 4# ;IF NOT DONE, TRY AGAIN
10# : JSR PC,ONLINE ;DO AND ONLINE COMMAND
; BCS 2# ;IF CARRY WAS SET, TRY AGAIN
14# : MOV P,UNSZ+2+RSPACK,UNSZH ;IS THE UNIT SIZE HI ADDRESS
; MOV P,UNSZ+RSPACK,UNSZL ;GET UNIT SIZE/IS IT = 0?
; BNE 16# ;IF NOT ZERO, CONTINUE WITH ITERATION
; TST UNSZH ;IS UNSZH ALSO 0?
; BEQ CYCLED ;IF 0, TRY TO BRING ONLINE AGAIN
; *** SET MSB OF TRY TO SHOW THAT INITIAL ONLINE IS DONE
; MOV #100000,TRY
;
*****
;
; THE FOLLOWING SEGMENT SETS THE LIMIT FOR THE UNIT SIZE.
; THE VALUE (UNIT SIZE - (WRITE BUFFER SIZE/NORMAL BLOCK SIZE))
; IS THE LAST SECTOR POSSIBLE TO RIGHT TO.
;

```

```

58 ;*****
59 002750 16$: MOV WBUF$Z,RO ;WBUF$Z IN RO AS A LIMIT
60 002750 016700 175166 CLR R1 ;R1 = # OF BLOCKS
61 002754 005001 18$: INC R1 ;INCREMENT THE # OF BLOCKS
62 002756 005201 SUB #~J,RO ;DECREMENT A BLOCK
63 002760 162700 000400 BPL 18$ ;BR IF > 0
64 002764 100374 SUB R1,UN$ZL ;ADJUST THE UNIT SIZE
65 002766 160167 175600 ; *** NOW PICK WHICH BLOCK TO WRITE TO
66 ; *** JSR PC,PICKBK ;ELSE SELECT A SECTOR TO TEST
67 002772 004767 000156 JSR PC,GT$TAT ;DID WE NOT GET THE DRIVE ONLINE?
68 002776 004767 001134 BCS 2$ ;IF WE DID NOT, GO BACK TO TOP AND TRY AGAIN
69 003002 103720 CMP #ST.AVL,RO ;IS IT AVAILABLE?
70 003004 022700 000004 BEQ 2$ ;IF SO, GO BACK TO TOP AND TRY AGAIN
71 003010 001715 ; *** WRITE TO THE BLOCK SELECTED
72 ; *** JSR PC,WRITE ;WRITE THE DATA FOR USER DEFINED # OF WORDS
73 003012 004767 000720 BCC 19$ ;IF OK, CONTINUE
74 003016 103007 BIT #SR.DUA,SR1 ;ARE WE DOING DUAL PORT?
75 003020 032767 001000 174770 BNE 2$ ;IF YES, RETRY
76 003026 001306 JSR PC,ERR$RH ;ELSE, HARD ERROR
77 003030 004767 001712 BR 22$ ;AND EXIT; BCS 22$ ;IF ERROR, EXIT
78 003034 000421 ; *** READ IT BACK
79 ; *** JSR PC,READ ;READ A BLOCK
80 003036 004767 000730 19$: BCS 22$ ;IF ERROR, EXIT
81 003042 103416 BIT #SR.CMP,SR1 ;DO A DATA COMPARE?
82 003044 032767 004000 174744 BNE 20$ ;IF NOT, SKIP THE COMPARE
83 003052 001004 ; *** COMPARE DATA
84 ; *** CDATA$,BEGIN,RB$FPA ; REQUEST FOR MONITOR TO CHECK DATA
85 003054 104412 000000' 000126' .+2 ; IF ERROR, CONTINUE
86 003062 003064' BIT #SR.DUA,SR1 ;DO WE DO AN AVAILABLE?
87 003064 032767 001000 174724 20$: BEQ 22$ ;IF NOT(BIT NOT SET) SKIP AVAILABLE
88 003072 001402 ; *** MAKE THE DRIVE AVAILABLE
89 003074 004767 001014 JSR PC,AVAILB ;RELEASE THE DRIVE
90 003100 000241 22$: CLC ;EVERY THING WAS OK
91 ; WASTE A LITTLE TIME SO OTHER
92 ; CONTROLLER CAN GRAB DRIVE
93 003102 000207 24$: RTS PC ;RETURN
94
95 ; *** SUBROUTINE TO WAIT FOR AN INTERRUPT
96 ; *** RETURNS AFTER THE INTERRUPT OCCURS
97 003104 DOINTR:
98 003104 005067 175354 CLR EXP$AV ;EXPECT AN AVAILABLE ATTENTION MESSAGE
99 003110 052767 140000 175144 BIS #<RG.OWN+RG.FLG>,R$PONC+2 ; SET OWN AND FLAG FOR RESPONCE RING
100 003116 000167 001206 JMP INTERP ;WAIT FOR ATTENTION MESSAGE & RETURN
101
102 ;*****
103 ;
104 ; DISKLESS CYCLE ;
105 ; DO A MAITENENCE WRITE ;
106 ; AND A MAITENENCE READ ;
107 ; AND CHECK THE DATA ;
108 ;
109 ;*****
110 CYCLEL:
111 003122 JSR PC,MAITW ;DO A MAINTENENCE WRITE
112 003126 004767 000430 JSR PC,MAITR ;DO A MAINTENENCE READ
113 003132 032767 004000 174656 BIT #SR.CMP,SR1 ;DO A DATA COMPARE?

```

```
114 003140 001004          BNE      21$          ;IF NOT, SKIP THE COMPARE
115 003142 104412 000000' 000126' CDATA$,BEGIN,RBUFPA ; REQUEST FOR MONITOR TO CHECK DATA
      003150 003152'          .+2          ; IF ERROR, CONTINUE
116 003152          21$:          RTS      PC
```

MODULE CODE

```

1
2
3
4
5
6
7
8
9
10
11 003154
12 003154 032767 002000 174634
13 003162 001467
14 003164
15 003164 104417 000000
16 003170 016746 174660
17 003174 104417 000000
18 003200 016746 174650
19
20
21
22 003204 000241
23 003206 042716 100000
24 003212 012667 175352
25 003216 005767 175352
26 003222 001430
27
28 003224 016700 175344
29 003230 005100
30 003232 012701 100000
31 003236 030100
32 003240 001403
33 003242 000241
34 003244 006001
35 003246 000773
36 003250 040100
37 003252 000241
38 003254 006001
39 003256 001374
40 003260 040067 175304
41 003264 026767 175300 175302
42 003272 002420
43 003274 001405
44 003276 006267 175266
45 003302 000414
46
47
48
49 003304 005067 175260
50 003310 005767 175256
51 003314 001406
52 003316 166716 175250
53 003322 103375
54 003324 066716 175242
55 003330 000401
56 003332 005016
57 003334 012667 175226

```

```

;*****
;
;   PICK A BLOCK TO WRITE TO.
;
;   EITHER PICK THE NEXT SEQUENTIAL BLOCK (DEFAULT) OR TAKE ONE AT
;   RANDOM.
;
;   OUTPUT:  FILL SECH & SECL (CURRENT SECTOR ADDR)
;*****
PICKBK:
12      BIT      @SP,SEQ,SR1      ;CHECK SR1 FOR RANDOM ACCESS MODE
13      BEQ      SEQACC           ;BR IF SEQUENTIAL ACCESS
RANACC:
15      RAND$,BEGIN
16      MOV      RANNUM,-(SP)      ;GENERATE THE SECTOR ADDRESS
17      RAND$,BEGIN
18      MOV      RANNUM,-(SP)      ;GENERATE THE SECTOR ADDRESS
;
;   ADJUST HI ADDRESS FIRST
;
22      CLC
23      BIC      @100000,(SP)      ;CLEAR CARRY FOR ROTATE
24      MOV      (SP)+,SECH        ;CLEAR UPPER BIT MAKES SURE VALUE'S
25      TST      UNSZH             ;STORE IN SECTOR HI ADDRESS
26      BEQ      3$               ;IS THE MAX SIZE 0?
27      ; *** UNSZH > 0 IF CODE FALLS THROUGH HERE
28      MOV      UNSZH,RO          ;IF 0, GET LOW SECTOR ADDRESS
29      COM      RO                ;RO = MAX VALUE
30      MOV      @100000,R1        ;RO COMPLEMENT, NOW FIND MS ZERO
31      BIT      R1,RO            ;R1 IS INDEX INTO MAX VALUE
32      BEQ      2$               ;HAVE 0 YET?
33      CLC
34      ROR      R1                ;IF 1ST 0 REACHED, CLEAR REST OF THE BITS
35      BR      1$                ;CLEAR CARRY FOR ROR
36      BIC      R1,RO            ;POINT TO NEXT BIT
37      CLC
38      ROR      R1                ;BRANCH TO TEST AGAIN
39      BNE      2$               ;CLEAR REST OF THE BITS
40      BIC      RO,SECH          ;CLEAR CARRY FOR ROR
41      CMP      SECH,UNSH        ;IF R1 ROTATES INTO CARRY, R1 = 0
42      BLT      7$               ;IF R1 NOT 0, MORE BITS TO CLEAR
43      BEQ      4$               ;CLEAR UPPER BITS OF HIGH SECTOR VALUE
44      ASR      SECH             ;IF THE HIGH SECTOR VALUE > MAX VALUE?
45      BR      7$               ;IF <, EXIT
46
47      ; GET LOW SECTOR ADDRESS
48
49      CLR      SECH             ;IF =, TEST LOW ORDER VALUE
50      TST      UNSZL            ;SECH = SECH/2  CAN'T BE > MAX NOW
51      BEQ      6$               ;EXIT
52      SUB      UNSZL,(SP)       ;CLEAR HI SECTOR SIZE
53      BCC      5$               ;IS THE HIGHEST POSSIBLE = 0?
54      ADD      UNSZL,(SP)       ;IF TRUE, DON'T DO LOOP
55      BR      7$               ;ELSE, SECL = SECL - UNSZL (ADJUST)
56      CLR      (SP)            ;IF UNSZL > SECL, LOOP
57      MOV      (SP)+,SECL       ;ELSE SUBTRACTD ONCF TOO OFTEN
; AND EXIT
;CLEAR LO SECTOR ADDRESS (IF HIGHEST POSSIBLE = 0)
;SAVE LO SECTOR ADDRESS

```



```

58 003340 000207          RTS    PC          ; RETURN
59
60
61          ;GENERATE DISK ADDRESS BY SEQUENTIAL ADDRESSING
62
63 003342          SEQACC:
64 003342 005267 175220          INC    SECL          ; INCREMENT THE SECTOR ADDRESS
65 003346 001405          BEQ    16:          ; BR IF ZERO
66 003350 026767 175212 175214  CMP    SECL,UNSZL    ; OVER LIMIT?
67 003356 103413          BLO   18:          ; BR IF LOWER
68 003360 000402          BR    17:          ; SKIP THE INCREMENT
69 003362          16:
70 003362 005267 175202          INC    SECH          ; INCREMENT SECTOR HIGH ADDRESS
71 003366          17:
72 003366 026767 175176 175200  CMP    SECH,UNSZH    ; OVER LIMIT?
73 003374 103404          BLO   18:          ; BR IF LOWER
74 003376 005067 175164          CLR    SECL          ; RESET THE STARTING SECTOR ADDRESS
75 003402 005067 175162          CLR    SECH          ;
76
77 003406          18:
78 003406 000207          RTS    PC
  
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31

32
33

34
35
36
37
38
39
40
41
42
43
44
45
46

```

;*****
;
; DROP A DRIVE
;
; A DRIVE WOULDN'T RESPOND, DROP IT. SET THIS UP IN DVICE.
;
; INPUT UNITNO = UNIT NUMBER OF DRIVE TO DROP
;        PORTID = BIT SET TO DROP DRIVE
;
; OUTPUT DVICE HAS A BIT CLEARED. THE BIT POSITION
;         REPRESENTS THE DRIVE
;*****

```

```

DROP1:
17 003410 012767 000001 175036 MOV #1,NUM
18 003416 000407 BR DROP4

DROP2:
19 003420 012767 000002 175026 MOV #2,NUM
20 003420 012767 000002 175026 BR DROP4

DROP3:
21 003426 000403
22 003430
23 003430 012767 000003 175016 MOV #3,NUM
24 003436 036767 175144 175134 DROP4: BIT PORTID,DVICE ;HAS THE DRIVE BEEN DROPPED, DON'T DROP AGAIN
25 003444 001445 BEQ 10# ;
26 003446 022767 177777 175132 CMP #177777,PORTID ;IF DRIVE HAS BEEN DROPPED, DON'T DROP AGAIN
; (WILL ZERO DVICE PRENTURE)
; IF =, DRIVE HAS BEEN DROPPED > EXIT ROUTINE
27
28 003454 001441 BEQ 10#
29
30 003456 046767 175124 175114 BIC PORTID,DVICE ;DROP THE DRIVE
31 ;*****
;CONVERT UNITNO TO ASCII AND
;STORE AT ADR2
003464 104421 000000' 000602' BTOD#,BEGIN,UNITNO,ADR2
003472 000475'

;*****
32 003474 105067 175002 CLR# ADR2+5
33 ;*****
;CONVERT PORTID TO ASCII AND
;STORE AT ADR1
003500 104420 000000' 000606' OTOA#,BEGIN,PORTID,ADR1
003506 000466'

;*****
34 003510 012764 177777 000002 MOV #177777,2(R4) ;DESELECT DRIVE SO IT WON'T BE USED AGAIN.
35 003516 005367 174732 DEC NUM ;DROPPED FOR WHICH ERROR?
36 003522 001004 BNE 1# ;IF NOT FOR ERRORS, CONTINUE
37 003524 104403 000000' 005242' MSGN#,BEGIN,DRP1 ;ASCII MESSAGE CALL WITH COMMON HEADER
38 003532 000412 BR 10#
39 003534 005367 174714 1# : DEC NUM ;WAS UNIT NOT FOUND?(NON EXISTENT UNIT)
40 003540 001004 BNE 2# ;IF NOT, CONTINUE
41 003542 104403 000000' 005260' MSGN#,BEGIN,DRP2 ;ASCII MESSAGE CALL WITH COMMON HEADER
42 003550 000403 BR 10#
43 003552 104403 000000' 005276' 2# : MSGN#,BEGIN,DRP3 ;ASCII MESSAGE CALL WITH COMMON HEADER
; ACTUAL UNITS FOUND
44
45 003560 000207 10# : RTS PC
46

```


MODULE CODE

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50

```

;*****
;
;      MAITENANCE READ
;
;      SET UP A PACKET WITH:
;          OPCODE & MODIFIER
;          REGION ID & REGION OFFSET
;          READ BUFFER DESCRIPTOR
;          BYTE COUNT
;      THEN SEND THE PACKET
;*****

```

```

003562 004767 001066
003566 012767 000030 174576
003574 016767 174650 174602
003602 016767 174320 174572
003610 016700 174316
003614 000424

```

```

MAITR: JSR    PC,CLRPAK          ;CLEAR THE PACKETS
        MOV    #OP.MRD,P.OPCD.CMPACK ;SET THE OPCODE
        MOV    RBFFEA,P.ADEA.CMPACK ;SET THE BUFFER DESCRIPTOR
        MOV    RBUFPA,P.ADPA.CMPACK ;
        MOV    RBUFSZ,RO          ;STORE THE BUFFER SIZE IN WORDS
        BR     MAITP             ;SET UP THE REST OF THE PACKET

```

```

;*****
;
;      MAITENANCE WRITE
;
;      SET UP A PACKET WITH:
;          OPCODE & MODIFIER
;          REGION ID & REGION OFFSET
;          WRITE BUFFER DESCRIPTOR
;          BYTE COUNT (EITHER WBUFSZ OR LIMIT IF WBUFSZ > LIMIT)
;      THEN SEND THE PACKET
;*****

```

```

003616 004767 001032
003622 012767 000031 174542
003630 016767 174616 174546
003636 016767 174272 174536
003644 026767 174272 174724
003652 100403
003654 016700 174716
003660 000402
003662 016700 174254
003666 006300
003670 010067 174502
003674 012767 000020 174370
003702 012767 000044 174446
003710 012767 000001 174500
003716 012767 177777 174434
003724 012767 177777 174342
003732 000167 00032?

```

```

MAITW: JSR    PC,CLRPAK          ;CLEAR THE PACKETS
        MOV    #OP.MWR,P.OPCD.CMPACK ;SET THE OPCODE
        MOV    WBFFEA,P.ADEA.CMPACK ;SET THE BUFFER DESCRIPTOR
        MOV    WBUFPA,P.ADPA.CMPACK ;
        CMP    WBUFSZ,LIMIT       ;IS THE BUFFER SIZE > LIMIT?
        BMI    1$                ;IF NOT, WBUFSZ IS OK
        MOV    LIMIT,RO          ;STORE THE BUFFER SIZE IN WORDS
        BR     MAITP             ;AND SKIP
1$:     MOV    WBUFSZ,RO          ;STORE THE BUFFER SIZE IN WORDS
MAITP: ASL    RO                 ;MAKE IT NUMBER OF BYTES
        MOV    RO,P.BCNT.CMPACK   ;SET WRITE BUFFER SIZE
        MOV    #16.,RSPLN        ;SET RESPONSE PACKET LENGTH
        MOV    #36.,CMPLEN       ;SET COMMAND PACKET LENGTH
        MOV    #1,P.RGID.CMPACK  ;SET REGION ID = 1
        MOV    #177777,CMPVIR    ;SET COMMAND VIRTUAL CIRCUIT ( 1 FOR DM)
        MOV    #177777,RSPVIR    ;SET COMMAND VIRTUAL CIRCUIT
        JMP    SEND              ;SEND THE PACKET

```

MODULE CODE

```

1
2
3
4
5
6
7
8
9
10
11
12 003736 004767 000712
13 003742 012767 000042 174422
14 003750 016700 174166
15 003754 016767 174154 174420
16 003762 016767 174464 174414
17 003770 000415
18
19
20
21
22
23
24
25
26
27
28
29
30 003772 004767 000656
31 003776 012767 000041 174366
32 004004 016700 174122
33 004010 016767 174434 174366
34 004016 016767 174104 174356
35 004024 012767 000040 174240
36 004032 012767 000040 174316
37 004040 006300
38 004042 010067 174330
39 004046 016767 174514 174342
40 004054 016767 174510 174336
41 004062 000476

```

```

;*****
;
; WRITE
;
; SET UP OP CODE, MODIFIERS,BUFFER SIZE (BYTE COUNT),
; BUFFER DESCRIPTOR (PHYSICAL AND EXTENDED ADDRESS)
; LET READ SET SIMILAR DATA IN THE PACKET;
; DISK ADDRESS AND CYLINDER ID (LOGICAL BLOCK NUMBER),
; THEN SEND THE PACKET.
;*****
WRITE: JSR PC,CLRPAK ;CLEAR PACKETS
MOV #OP.WR,P.OPCD.CMPACK ;SET THE OPCODE
WRITEA: MOV WBUFZ,RO ;STORE THE BUFFER SIZE IN WORDS
MOV WBUFPA,P.ADPA.CMPACK ;SET THE BUFFER DESCRIPTOR(PA)
MOV WBFEEA,P.ADEA.CMPACK ;SET THE BUFFER DESCRIPTOR(EA)
BR READA ;
;*****
;
; READ
;
; SET UP OP CODE, MODIFIERS,BUFFER SIZE (BYTE COUNT),
; BUFFER DESCRIPTOR (PHYSICAL AND EXTENDED ADDRESS),
; DISK ADDRESS AND CYLINDER ID (LOGICAL BLOCK NUMBER),
; THEN SEND THE PACKET.
;*****
READ: JSR PC,CLRPAK ;CLEAR PACKETS
MOV #OP.RD,P.OPCD.CMPACK ;SET THE OPCODE
MOV RBUFZ,RO ;STORE THE BUFFER SIZE IN WORDS
MOV RBFFEA,P.ADEA.CMPACK ;SET THE BUFFER DESCRIPTOR
MOV RBUFPA,P.ADPA.CMPACK ;
READA: MOV #32.,RSPLN ;SET RESPONSE PACKET LENGTH
MOV #32.,CMPLEN ;SET COMMAND PACKET LENGTH
ASL RO ;MAKE IT NUMBER OF BYTES
MOV RO,P.BCNT.CMPACK ;SET READ BUFFER SIZE
MOV SECL,P.LBN.CMPACK ;SET LOGICAL BLOCK NUMBER
MOV SECH,P.LBN*2.CMPACK ;
BR SEND ;SEND THE PACKET
;*****

```

MODULE CODE

```

1      ;*****
2      ;
3      ;   DETERMINE ACCESS PATHS
4      ;
5      ;   SET UP CODE, GO SEND PACKET
6      ;
7      ;*****
8 004064 004767 000564 DAP: JSR PC,CLRPAK ;CLEAR PACKETS
9 004070 012767 000013 174274 MOV #OP.DAP,P.OPCD+CMACK ;SET OPCODE
10 004076 012767 000074 174166 MOV #60.,RSPLN ;SET LENGTHS
11 004104 012767 000074 174244 MOV #60.,CMPLN ;
12 004112 000462 BR SEND ;SEND THE PACKET
13
14 ;*****
15 ;
16 ;   AVAILABLE PACKET
17 ;
18 ;   SET OP CODE AND MODIFIERS THEN SEND THE PACKET
19 ;
20 ;*****
21 004114 004767 000534 AVAILB: JSR PC,CLRPAK ;CLEAR PACKETS
22 004120 012767 000010 174244 MOV #OP.AVL,P.OPCD+CMACK ;SET THE OPCODE
23 004126 012767 000014 174136 MOV #12.,RSPLN ;SET RESPONSE PACKET LENGTH
24 004134 000413 BR GTSTAA ;SEND THE PACKET
25
26 ;*****
27 ;
28 ;   GET UNIT STATUS
29 ;
30 ;   SET OPCODE AND MODIFIER (FOR THEN NEXT UNIT
31 ;   THEN SEND THE PACKET
32 ;
33 ;*****
34 004136 004767 000512 GTSTAT: JSR PC,CLRPAK ;CLEAR PACKETS
35 004142 012767 000003 174222 MOV #OP.GUS,P.OPCD+CMACK ;SET THE OPCODE
36 004150 012767 000001 174216 MOV #MD.NXU,P.MOD+CMACK ;CLEAR MODIFIERS
37 004156 012767 000060 174106 MOV #48.,RSPLN ;SET RESPONSE PACKET LENGTH
38 004164 012767 000014 174164 GTSTAA: MOV #12.,CMPLN ;SET COMMAND PACKET LENGTH
39 004172 000432 BR SEND ;SEND THE PACKET

```

```

1
2
3
4
5
6
7
8
9
10
11 004174
12 004174 004767 000454
13 004200 012767 000040 174150
14 004206 012767 000034 174056
15 004214 012767 000004 174150
16 004222 012767 000200 174150
17
18 004230 000413
19
20
21
22
23
24
25
26
27
28
29 004232 004767 000416
30 004236 012767 000040 174026
31 004244 012767 000044 174104
32 004252 012767 000011 174112

```

```

;*****
;
;   SET CONTROLLER CHARACTERISTICS
;
;   SET OP CODE AND CONTROLLER FLAG (ENABLE ATTENTION MSGS)
;   CLEAR MSCP VERSION, HOST TIMEOUT, USE FRACTION,
;   AND ALL OF QUAD WORD TIME AND DATE.
;   THEN SEND PACKET
;*****
SCC:
      JSR    PC,CLRPAK           ;GO CLEAR THE COMMAND PACKET
      MOV    #32.,CMPLN         ;SET UP COMMAND PACKET LENGTH
      MOV    #28.,RSPLN         ;SET UP RESPONSE PACKET LENGTH
      MOV    #OP.SCC,P.OPCD+CMPACK ;SET THE OPCODE
      MOV    #CF.AVL,P.CNTF+CMPACK ;SET THE CONTROLLER FLAGS
; TO ENABLE ATTENTION MSGS
      BR     SEND               ;SEND THE PACKET
;*****
;
;   ONLINE
;
;   SET OPCODE, MODIFIERS, UNIT ID, HOST ID
;   SHADOW UNIT, ERROR FLAGS
;   THEN SEND PACKET
;*****
ONLINE: JSR    PC,CLRPAK           ;CLEAR PACKETS
      MOV    #32.,RSPLN         ;SET RESPONSE PACKET LENGTH
      MOV    #36.,CMPLN         ;SET COMMAND PACKET LENGTH
      MOV    #OP.ONL,P.OPCD+CMPACK ;SET THE OPCODE

```

```

1
2
3      SEND - SEND A PACKET
4      INTERP - WAIT FOR AN INTERRUPT
5
6      SET UP THE COMMAND REFERENCE NUMBER AND UNITNO IN THE PACKET
7      SET OWN, CLEAR FLAG IN THE COMMAND RING (FOR CONTROLLER)
8      SET OWN & FLAG IN MESSAGE RING (FOR INTERRUPTS BY CONTROLLER)
9      AFTER INTERRUPT, MAKE SURE THE PACKET WAS PROCESSED (NO HARD
10     OR SOFT ERRORS) THEN RETURN TO CYCLED.
11
12     INPUT:  CMPACK IS FILLED EXCEPT FOR CMDREF & UNITNO
13             INTERRUPT VECTOR AND BR LEVEL ARE ESTABLISHED
14
15     OUTPUT: MSPACK IS FILLED
16             CLEAR CARRY IF COMMAND PACKET WAS OK
17             ELSE GO DO A HARD/SOFT ERROR
18
19 *****
20 004260 005267 174004 SEND: INC CMDREF ;NEW COMMAND REFERENCE NUMBER
21 004264 001775 BEQ SEND ;COMMAND REF # CANNOT = 0
22 004266 016767 173776 174066 MOV CMDREF,P.CRF+CMPPACK ;SET COMMAND REF NUMBER
23 004274 016767 174302 174064 MOV UNITNO,P.UNIT+CMPPACK ;SET UNIT NUMBER
24 004302 042767 040000 173756 BIC @RG.FLG,COMMAND+2 ;CLEAR FLAG
25 004310 052767 100000 173750 BIS @RG.OWN,COMMAND+2 ;SET OWN FOR COMMAND RING
26 004316 052767 140000 173736 BIS @<RG.OWN+RG.FLG>,RSPONC+2 ;SET OWN AND FLAG FOR MESSAGE RING
27 004324 005777 173456 TST @ADDR ;FORCE POLLING TO PACKET
28 004330 INTERP:
29 004330 104400 000000' EXIT$,BEGIN ;EXIT TO MONITOR. MODULE WAIT FOR INTERRUPT.
30
31 004334 ;
32 004334 000004 000000' 004342' PIRQ$,BEGIN,1$ ; QUEUE UP TO CONTINUE AT 1$ AND RTI
33 -----
34 1$:
35 004342 005067 173710 CLR RINTR ;CLEAR INTERRUPT FLAG
36 004346 022767 000100 173732 CMP @OP.AVA,P.OPCD+RSPACK ;WAS AN AVAILABLE ATTENTION RECIEVED?
37 004354 001524 BEQ 15$ ;IF IT WAS, EXIT
38 004356 022767 000102 173722 CMP @OP.ACP,P.OPCD+RSPACK ;WAS THE ACCESS PATH ATTENTION RECIEVED?
39 004364 001527 BEQ 16$ ;IF IT WAS, GO PROCESS
40 ; ELSE CHECK SUCCESS
41 2$:
42 004366 016700 173716 MOV P.STS+RSPACK,RO ; SUCCESS?
43 004372 001513 BEQ 14$ ;IF YES, EXIT
44 004374 042700 177740 BIC @177740,RO ;CLEAR UPPER 11 BITS OF SUB-STATUS
45 004400 001510 BEQ 14$ ;IF SUCCESS = 0, EXIT OK
46 004402 005067 173500 CLR ERRTP ;IF GOT HERE, ERROR
47 004406 122700 000013 CMPB @ST.DRV,RO ; DRIVE ERROR?
48 004412 001015 BNE 3$ ;IF NOT NEXT TEST
49 004414 032767 001000 173374 BIT @SR.DUA,SR1 ;ARE WE DUAL PORTING?
50 004422 001472 BEQ 12$ ;IF NOT, GO REPORT ERROR/ELSE EXPECTED
51 004424 022767 000053 173656 CMP @<ST.DRV+SC.STO>,P.STS+RSPACK ;IS IT AN SDI RESPONCE TIMEOUT?
52 004432 001464 BEQ 10$ ;IF TRUE, DRIVE IS NOT ONLINE, EXIT
53 004434 022767 000113 173646 CMP @<ST.DRV+SC.INV>,P.STS+RSPACK ;IS IT THE INVALID SDI RESPONCE?
54 004442 001460 BEQ 10$ ;IF TRUE, DRIVE IS NOT ONLINE, EXIT
55 004444 000461 BR 12$ ;ELSE HARD ERROR

```



```

56 004446          3$:
57 004446 122700 000012      CMPB  #ST.CNT,RO      ; CONTROLLER ERROR?
58 004452 001004          BNE   4$             ; IF NOT NEXT TEST
59 004454 012767 000003 173424  MOV   #ERR.3,ERRTYP ; ELSE, SET ERROR TYPE
60 004462 000531          BR    ERRORH        ; AND HARD ERROR
61 004464          4$:
62 004464 122700 000011      CMPB  #ST.HST,RO      ; HOST BUFFER ACCESS ERROR?
63 004470 001004          BNE   5$             ; IF NOT NEXT TEST
64 004472 012767 000032 173406  MOV   #ERR.32,ERPTYP ; ELSE, SET ERROR TYPE
65 004500 000522          BR    ERRORH        ; AND HARD ERROR
66 004502          5$:
67 004502 122700 000010      CMPB  #ST.DAT,RO      ; DATA ERROR?
68 004506 001004          BNE   6$             ; IF NOT NEXT TEST
69 004510 012767 000001 173370  MOV   #ERR.1,ERRTYP ; ELSE, SET ERROR TYPE
70 004516 000533          BR    ERRORS        ; AND SOFT ERROR
71 004520          6$:
72 004520 122700 000006      CMPB  #ST.WPR,RO      ; WRITE PROTECTED?
73 004524 001431          BEQ   12$            ; ELSE HARD ERROR
74 004526          8$:
75 004526 122700 000004      CMPB  #ST.AVL,RO      ; STILL AVAILABLE?
76 004532 001005          BNE   9$             ; IF NOT NEXT TEST
77 004534 022767 000003 173630  CMP   #OP.GUS,P.OPCD+CMPACK ; ELSE, IF COMMAND WAS
78                                     ; GET UNIT STATUS
79 004542 001427          BEQ   14$            ; THEN EXPECTED & LEAVE ROUTINE
80 004544 000421          BR    12$            ; ELSE HARD ERROR
81 004546          9$:
82 004546 122700 000003      CMPB  #ST.OFL,RO      ; UNIT OFFLINE?
83 004552 001022          BNE   13$            ; IF NOT NEXT TEST
84                                     ; *** OFFLINE WHEN TRIED ONLINE OR GET UNIT STATUS
85 004554 022767 000011 173610  CMP   #OP.ONL,P.OPCD+CMPACK ; WAS IT AN ONLINE COMMAND?
86 004562 001410          BEQ   10$            ; IF SO, SET CARRY/EXIT
87 004564 022767 000003 173600  CMP   #OP.GUS,P.OPCD+CMPACK ; IS IT GET UNIT STATUS COMMAND?
88 004572 001404          BEQ   10$            ; IF SO, SET CARRY/EXIT
89 004574 022767 000042 173570  CMP   #OP.WR,P.OPCD+CMPACK ; IS IT WRITE COMMAND?
90 004602 001002          BNE   12$            ; IF NOT, REPORT HARD ERROR
91 004604 000261          SEC                     ; ELSE, SET CARRY TO
92 004606 000207          RTS    PC                ; AND RETURN TO DROP DRIVE/AWAIT AVAILABLE DRIVE
93                                     ; *** HARD ERROR EXIT WITH ERROR TYPE = 6
94 004610          12$:
95 004610 012767 000006 173270  MOV   #ERR.6,ERRTYP ; ELSE, SET ERROR TYPE
96 004616 000453          BR    ERRORH        ; AND HARD ERROR
97                                     ; *** SOFT ERROR EXIT WITH ERROR TYPE = 0
98 004620          13$:
99 004620 000472          BR    ERRORS        ; ERROR WITH ERRTP = 0 & IS A SOFT ERROR
100                                     ; ST.CMP,ST.MFE,..ST.ABO,ST.CMD
101                                     ; *** SUCCESSFUL EXIT
102 004622          14$:
103 004622 000241          CLC                     ; CLEAR CARRY 'CAUSE PACKET IS OK
104 004624 000207          RTS    PC                ; ELSE, OK, SO FAR.
105 004626          15$:
106                                     ; *** WAIT FOR ATTENTION INTERRUPT
107                                     ; *** DID WE GET AN AVAILABLE ATTENTION MESSAGE THAT WE EXPECTED?
108 004626 005767 173632      TST   EXPAV          ;
109 004632 001004          BNE   16$            ; IF EXPAV IS NOT 0, WE GOT ONE WE DIDN'T EXPECT
110 004634 012767 177777 173622  MOV   #177777,EXPVAV ; CLEAR EXPECTED AVAILABLE ATTENTION MESSAGE WORD
111 004642 000767          BR    14$             ; AND RETURN
112 004644          16$:

```

113 004644 052767 140000 173410
114 004652 000626
115

BIS @<RG.OWN*RG.FLG>,RSPONC*2 ;
BR INTERP ;WAIT FOR RESPONCE OF LAST PACKET SENT

MODULE CODE

```

1
2
3
4
5
6
7
8
9
10
11
12 004654
13 004654 017767 173372 173572
14 004662 001421
15
;*****
;
; CLEAR PACKETS
;
; ASSUMPTION: 1) RESPONSE BUFFER PRECEDES THE COMMAND BUFFER
;              2) TWO WORDS BEFORE EACH BUFFER IS FOR LENGTH
;              OF PACKET AND VIRTUAL CIRCUIT
;
; OUTPUT: R2 = 0 WHEN DONE
;         R5 = END OF COMMAND PACKET WHEN DONE
;*****
CLRPAK:
MOV    @SAREG,NUM          ;IF SA REG NOT ZERO, STORE IN NUM
BEQ    5$                  ;IF SA REG IS ZERO, CLEAR PACKETS
;*****
;CONVERT NUM TO ASCII AND
;STORE AT ADR1
004664 104420 000000' 000454'   OTOA$,BEGIN,NUM,ADR1
004672 000466'
;*****
16 004674 104403 000000' 005334' MSGN$,BEGIN,SANOTO ;ASCII MESSAGE CALL WITH COMMON HEADER
17 004702 010346          MOV    R3,-(SP)      ;SAVE R3
18 004704 010446          MOV    R4,-(SP)      ;SAVE R4
19 004706 004767 174532    JSR   PC,INITUD     ;RE INIT SA REGISTER
20 004712 012603          MOV    (SP)+,R3     ;RESTORE R3
21 004714 012604          MOV    (SP)+,R4     ;RESTORE R4
22 004716 004767 177252    JSR   PC,SCC        ;SET CONTROLLER CHARS AGAIN
23 004722 005267 173116    INC   HRDCNT        ;INCREMENT HARD ERROR COUNT
24
25 004726 012702 000064    5$:   MOV    @52.,R2   ;DOING THIS WILL CAUSE ANOTHER CALL TO CLRPAK
26 004732 012705 000272'   MOV    @RSPLEN,R5   ;R2 = # OF WORDS TO CLEAR
27 004736 005025          6$:   CLR    (R5)+      ;R5 -> RSPLEN, 1ST WORD TO CLEAR
28 004740 005302          DEC   R2            ;CLEAR WORD
29 004742 001375          BNE   6$            ;R2 = ZERO? (DONE CONDITION)
30 004744 000207          RTS   PC            ;IF NOT ZERO, LOOP
31
32
33
34
35
36
37 004746
38 004746 032767 000004 173042 ERRORH: BIT    @SR.REP,SR1   ;DO WE REPORT THE ERROR?
39 004754 001403          BEQ    7$            ;IF SO, REPORT
40 004756 005267 173062    INC   HRDCNT        ;ELSE, INCREMENT THE HARD ERROR
41
42 004762 000407          BR    8$            ;COUNT IF NOT REPORTED
43 004764 004767 000056    7$:   JSR   PC,SETTAB    ;SKIP REPORT
44
45 004770 104405 000000' 000000   HRDR$,BEGIN,NULL   ;SET UP TABLE
46 004776 004767 000070    JSR   PC,PRINTE
47 005002 000261          8$:   SEC
48 005004 000207          RTS   PC            ;RETURN TO CYCLED
49
50
;*****
;

```

```

51          :      SOFT ERROR      CARRY WILL BE SET      :
52          :      :      :      :      :      :      :      :      :
53          :      :      :      :      :      :      :      :      :
54 005006   :      :      :      :      :      :      :      :      :
55 005006   032767 000004 173002 :      :      :      :      :      :      :      :      :
56 005014   001403 :      :      :      :      :      :      :      :      :
57 005016   005267 173020 :      :      :      :      :      :      :      :      :
58          :      :      :      :      :      :      :      :      :
59 005022   000407 :      :      :      :      :      :      :      :      :
60 005024   004767 000016 :      :      :      :      :      :      :      :      :
61          :      :      :      :      :      :      :      :      :
   005030   104406 000000' 000000 :      :      :      :      :      :      :      :      :
62 005036   004767 000030 :      :      :      :      :      :      :      :      :
63 005042   000261 :      :      :      :      :      :      :      :      :
64 005044   000207 :      :      :      :      :      :      :      :      :
65          :      :      :      :      :      :      :      :      :
66          :      :      :      :      :      :      :      :      :
67          :      :      :      :      :      :      :      :      :
68          :      :      :      :      :      :      :      :      :
69          :      :      :      :      :      :      :      :      :
70          :      :      :      :      :      :      :      :      :
71          :      :      :      :      :      :      :      :      :
72          :      :      :      :      :      :      :      :      :
73 005046   :      :      :      :      :      :      :      :      :
74 005046   016767 172734 173024 :      :      :      :      :      :      :      :      :
75 005054   016767 173230 173022 :      :      :      :      :      :      :      :      :
76 005062   017767 173164 173012 :      :      :      :      :      :      :      :      :
77 005070   000207 :      :      :      :      :      :      :      :      :
78          :      :      :      :      :      :      :      :      :
79          :      :      :      :      :      :      :      :      :
80          :      :      :      :      :      :      :      :      :
81          :      :      :      :      :      :      :      :      :
82          :      :      :      :      :      :      :      :      :
83          :      :      :      :      :      :      :      :      :
84          :      :      :      :      :      :      :      :      :
85 005072   :      :      :      :      :      :      :      :      :
86          :      :      :      :      :      :      :      :      :
   005072   104420 000000' 000310' :      :      :      :      :      :      :      :      :
   005100   000466' :      :      :      :      :      :      :      :      :
87          :      :      :      :      :      :      :      :      :
   005102   104420 000000' 000306' :      :      :      :      :      :      :      :      :
   005110   000475' :      :      :      :      :      :      :      :      :
88          :      :      :      :      :      :      :      :      :
   005112   104420 000000' 000302' :      :      :      :      :      :      :      :      :
   005120   000504' :      :      :      :      :      :      :      :      :
89          :      :      :      :      :      :      :      :      :

```

```

005122 104420 000000' 000312'      ;STORE AT ADR4
005130 000513'      OTOA$,BEGIN,P.BCNT*RSRACK,ADR4
;.....
;.....
90      ;CONVERT P.LBN*2*CHPACK TO ASCII AND
;STORE AT ADR5
005132 104420 000000' 000420'      OTOA$,BEGIN,P.LBN*2*CHPACK,ADR5
005140 000522'      ;.....
;.....
91      ;CONVERT P.LBN*CHPACK TO ASCII AND
;STORE AT ADR6
005142 104420 000000' 000416'      OTOA$,BEGIN,P.LBN*CHPACK,ADR6
005150 000531'      ;.....
;.....
92      ;CONVERT P.ADEA*CHPACK TO ASCII AND
;STORE AT ADR7
005152 104420 000000' 000404'      OTOA$,BEGIN,P.ADEA*CHPACK,ADR7
005160 000540'      ;.....
;.....
93      ;CONVERT P.ADPA*CHPACK TO ASCII AND
;STORE AT ADR8
005162 104420 000000' 000402'      OTOA$,BEGIN,P.ADPA*CHPACK,ADR8
005170 000547'      ;.....
;.....
94 005172 104403 000000' 005376'      MSGN$,BEGIN,BANNER      ;ASCII MESSAGE CALL WITH COMMON HEADER
95 005200 000207      RTS      PC
96

```

1			.SBTTL	MODULE MESSAGES
2	005202	005444	INITE1:	MSG2
3	005204	005515		MSG4
4	005206	177777		177777
5				
6	005210	005475	INITER:	MSG3
7	005212	000466		ADR1
8	005214	005675		MSG10
9	005216	000475		ADR2
10	005220	006023		MSG14
11	005222	000504		ADR3
12	005224	177777		177777
13				
14	005226	005444	INITE2:	MSG2
15	005230	005542		MSG5
16	005232	177777		177777
17				
18	005234	005444	INITE3:	MSG2
19	005236	005560		MSG6
20	005240	177777		177777
21				
22	005242	005652	DRP1:	MSG8
23	005244	000475		ADR2
24	005246	005662		MSG9
25	005250	006245		MSG20
26	005252	000466		ADR1
27	005254	006732		MSGD1
28	005256	177777		177777
29				
30	005260	005652	DRP2:	MSG8
31	005262	000475		ADR2
32	005264	005662		MSG9
33	005266	006245		MSG20
34	005270	000466		ADR1
35	005272	006776		MSGD2
36	005274	177777		177777
37				
38	005276	005652	DRP3:	MSG8
39	005300	000475		ADR2
40	005302	005662		MSG9
41	005304	006245		MSG20
42	005306	000466		ADR1
43	005310	007040		MSGD3
44	005312	177777		177777
45				
46	005314	005707	ERRPAS:	MSG11
47	005316	000475		ADR2
48	005320	005733		MSG12
49	005322	000504		ADR3
50	005324	005771		MSG13
51	005326	000466		ADR1
52	005330	005442		MSG1
53	005332	177777		177777
54				
55	005334	006105	SANOTO:	MSG17
56	005336	000466		ADR1
57	005340	006142		MSG18

58	005342	177777	177777
59			
60	005344	006034'	UNIOFF: MSG16
61	005346	000466'	ADR1
62	005350	177777	177777
63			
64	005352	006556'	WARN1: MSG40
65	005354	177777	177777
66			
67	005356	006433'	WARN2: MSG37
68	005360	177777	177777
69			
70	005362	006364'	WARN3: MSG36
71	005364	177777	177777
72			
73	005366	005615'	ABORT: MSG7
74	005370	177777	177777
75			
76	005372	006215'	ZERO: MSG19
77	005374	177777	177777
78			
79	005376	006270'	BANNER: MSG21
80	005400	000466'	ADR1
81	005402	006362'	MSG23
82	005404	000475'	ADR2
83	005406	006362'	MSG23
84	005410	000504'	ADR3
85	005412	006362'	MSG23
86	005414	000513'	ADR4
87	005416	006362'	MSG23
88	005420	000522'	ADR5
89	005422	006362'	MSG23
90	005424	000531'	ADR6
91	005426	006362'	MSG23
92	005430	000540'	ADR7
93	005432	006362'	MSG23
94	005434	000547'	ADR8
95	005436	005442'	MSG1
96	005440	177777	177777

DUBD DEC/X11 SYSTEM EXERCISER M MACRO V05.01b Thursday 20 Sep-84 10:17 Page 29
 MORE MODULE MESSAGES

```

1          .SBTTL  MORE MODULE MESSAGES
2          .NLIST  BEX
3
4 005442    045    000    MSG1:  .ASCIZ  '#
5 005444    045    103    117 MSG2:  .ASCIZ  '#CONTROLLER INIT ERROR. '
6 005475    045    123    101 MSG3:  .ASCIZ  '#SA REGISTER = '
7 005515    106    117    125 MSG4:  .ASCIZ  'FOUND BY DIAGNOSTIC '
8 005542    123    124    105 MSG5:  .ASCIZ  'STEP NOT SET.'
9 005560    105    130    120 MSG6:  .ASCIZ  'EXPECTED DATA WAS INCORRECT '
10 005615   045    122    105 MSG7:  .ASCIZ  '#RETRY COUNT EXCEEDED, ABORT'
11 005652   045    104    122 MSG8:  .ASCIZ  '#DRIVE '
12 005662   040    104    122 MSG9:  .ASCIZ  ' DROPPED. '
13 005675   040    111    116 MSG10: .ASCIZ  ' IN STEP '
14 005707   045    123    117 MSG11: .ASCIZ  '#SOFT ERROR COUNT #'
15 005733   040    040    040 MSG12: .ASCIZ  ' *** HARD ERROR COUNT #'
16 005771   045    103    110 MSG13: .ASCIZ  '#CHECK DATA ERROR COUNT #'
17 006023   045    101    104 MSG14: .ASCIZ  '#ADDR = '
18 006034   045    125    116 MSG16: .ASCIZ  '#UNIT WAS FOUND OFFLINE. UNIT NUMBER = '
19 006105   045    123    101 MSG17: .ASCIZ  '#SA REGISTER IS NOT ZERO, = '
20 006142   045    103    117 MSG18: .ASCIZ  '#CONTROLLER IS GOING THROUGH INITIALIZATION'
21 006215   045    122    111 MSG19: .ASCIZ  '#RING AREA NOT CLEARED#'
22 006245   045    104    105 MSG20: .ASCIZ  '#DEVICE ID BIT = '
23 006270   045    123    124 MSG21: .ASCIZ  '#STATUS ENCOD UNITNU BYTECO HI LBN LO LBN EXTADR PHYADR#'
24 006362   040    000    MSG23: .ASCIZ  '
25 006364   040    041    040 MSG36: .ASCIZ  ' ! OPERATING WITH NO DISK ACCESSING !#'
26 006433   007    007    040 MSG37: .ASCII  '<07><07>' ! CUSTOMER DATA WILL BE OVERWRITTEN !#'
27 006504   040    055    055   .ASCIZ  '-----#<07><07>'
28 006556   040    111    106 MSG40: .ASCII  ' IF YOU WISH TO DESTROY CUSTOMER DATA, SET BIT1 (NOT BIT0)#'
29 006651   040    111    116   .ASCIZ  ' IN SWITCH REGISTER 1(SR1) OF DUBD? EQUAL TO 1.#'
30 006732   045    105    122 MSGD1: .ASCIZ  '#ERRORS CAUSED DRIVE TO BE DROPPED#'
31 006776   045    125    116 MSGD2: .ASCIZ  '#UNIT WAS NOT FOUND BY EXERCISER#'
32 007040   045    104    126 MSGD3: .ASCIZ  '#OVID1 BIT SET HIGHER THAN ACTUAL # OF DRIVES FOUND#'
33   .EVEN
34 007126   .BLKW  256.          ;THE READ BUFFER
35   .END
000001

```


Symbol table

ABORT	005366R	CF.576	000001	HC.CMD	000004	MD.WBV	000040	OP.ONL	000011
ACSR	000102R	CINTR	000254R	HC.CPK	000356R	MODNAM	000000R	OP.RD	000041
ADDR	000006R	CLRPAK	004654R	HC.RCT	000002	MODSP	000252R	OP.RPL	000024
ADDR22	001000	CMDFEF	000270R	HC.RES	000000	MSGD1	006732R	OP.SCC	000004
ADR1	000466R	CHMPACK	000362R	HC.RPK	000276R	MSGD2	006776R	OP.SMC	000102
ADR2	000475R	CHMPLEN	000356R	HC.SIZ	000010	MSGD3	007040R	OP.SUC	000012
ADR3	000504R	CHMPVIR	000360R	HRDCNT	000044R	MSGN1	104403	OP.WR	000042
ADR4	000513R	CHMPIND	000264R	HRDR1	104405	MSG51	104402	OTOA1	104420
ADR5	000522R	CHMFIG	000056R	HRDPAS	000050R	MSG1	104401	PA	000444R
ADR6	000531R	CSRA	000100R	ICONT	000036R	MSG10	005442R	PASCNT	000034R
ADR7	000540R	CYCLED	002624R	ICOUNT	000040R	MSG11	005675R	PA18	000556R
ADR8	000547R	CYCLEL	003122R	IDNUM	000122R	MSG12	005707R	PA22	000562R
ASB	000106R	DAP	004064R	IMODX	000000	MSG13	005733R	PICKBK	003154R
ASR04	002034R	DATCK1	104411	INIT	000030R	MSG14	005771R	PIRQ1	000004
ASTAT	000104R	DATER1	104404	INITER	005210R	MSG16	006023R	PKTSIZ	000060
AVAILB	004114R	DOINTR	003104R	INITE1	005202R	MSG17	006034R	POPSP	005726
AWAS	000110R	DROP1	003410R	INITE2	005226R	MSG18	006105R	POPSP2	022626
BANNER	005376R	DROP2	003420R	INITE3	005234R	MSG19	006142R	PORTID	000606R
BEGIN	000000R	DROP3	003430R	INITUD	001444R	MSG20	006215R	PRINTE	005072R
BIT0	000001	DROP4	003436R	INTA	002072R	MSG2	005444R	PRMSG	000462R
BIT00	000001	DRP1	005242R	INTERP	004330R	MSG21	006245R	PRNUM	000017
BIT01	000002	DRP2	005260R	INTR	000120R	MSG22	006270R	PRTY	000000
BIT02	000004	DRP3	005276R	LIMIT	000576R	MSG23	006362R	PRTY0	000000
BIT03	000010	DVICE	000600R	LOOP1	001300R	MSG3	005475R	PRTY1	000040
BIT04	000020	DVID1	000014R	LOOP2	001330R	MSG36	006364R	PRTY2	000100
BIT05	000040	EA	000446R	L.CHVR	000015	MSG37	006433R	PRTY3	000140
BIT06	000100	EA22	000564R	L.CNTI	000014	MSG4	005515R	PRTY4	000200
BIT07	000200	EF.BBR	000200	L.CYL	000034	MSG40	006556R	PRTY5	000240
BIT08	000400	EF.BBU	000100	L.DATA	000050	MSG5	005542R	PRTY6	000300
BIT09	001000	EF.FRS	000200	L.ERLC	000030	MSG6	005560R	PRTY7	000340
BIT1	000002	EF.LOG	000040	L.EVNT	000000	MSG7	005615R	PS	177776
BIT10	002000	EF.LST	000100	L.GRP	000040	MSG8	005652R	PSW	177776
BIT11	004000	EF.MIS	000001	L.SCTR	000042	MSG9	005662R	PUSH	005746
BIT12	010000	EF.SEX	000020	L.SLOT	000002	NTRUPT	004334R	PUSH2	024646
BIT13	020000	ENDIT1	104413	L.TRCK	000041	NULL	000000	PWRFLG	000002
BIT14	040000	END1	104410	L.UHVR	000027	NUM	000454R	P.ADEA	000022
BIT15	100000	ERRORH	004746R	L.UNTI	000016	OLDEA	000460R	P.ADPA	000020
BIT2	000004	ERRORS	005006R	L.USVR	000026	OLDPA	000456R	P.BCNT	000014
BIT3	000010	ERROR1	002150R	L.VSER	000044	ONEFIL	000001	P.BUFF	000020
BIT4	000020	ERROR2	002146R	MAITP	003666R	ONLINE	004232R	P.CMST	000020
BIT5	000040	ERROR3	002144R	MAITR	003562R	OPEN	000000	P.CNCL	000022
BIT6	000100	ERROR5	002132R	MAITW	003616R	OP.ABO	000001	P.CNT	000006
BIT7	000200	ERRPAS	005314R	MAP221	104416	OP.ACC	000020	P.CNTF	000016
BIT8	000400	ERRTYP	000106R	MA10NC	001412R	OP.ACP	000102	P.CNTI	000024
BIT9	001000	ERR.0	000000	MD.CMP	040000	OP.AVA	000100	P.CPSP	000042
BREAK1	104407	ERR.1	000001	MD.ERR	010000	OP.AVL	000010	P.CRF	000000
BR1	000012R	ERR.3	000003	MD.EXP	100000	OP.CCD	000021	P.CTMO	000020
BR2	000013R	ERR.32	000032	MD.FEU	000001	OP.CMP	000040	P.CYL	000050
BTOD1	104421	ERR.6	000006	MD.NXU	000001	OP.DAP	000013	P.ELGF	000034
CDATA1	104412	EXIT1	104400	MD.SCH	004000	OP.END	000200	P.FBBK	000034
CDERCT	000144R	EXPAV	000464R	MD.SCL	002000	OP.ERL	000101	P.FLGS	000011
CDMDCT	000146R	FREE	000150R	MD.SEC	001000	OP.ERS	000022	P.GRP	000046
CF.AVL	000200	GETPA1	104415	MD.SER	000400	OP.FLU	000023	P.HSTI	000020
CF.MSC	000100	GTSTAA	004164R	MD.SPD	000001	OP.GCS	000002	P.HTMO	000020
CF.OTH	000040	GTSTAT	004136R	MD.SSH	000200	OP.GUS	000003	P.LBN	000034
CF.SMD	000002	GMBUF1	104414	MD.VOL	000002	OP.HRD	000030	P.LGDT	000014
CF.THS	000020	HC.CCT	000006	MD.WBN	000100	OP.MWR	000031	P.MEDI	000034

Symbol table

P.MLUN= 000014	RESTR1 001036R	SA.S1 = 004000	SR4 000024R	UF.CMW= 000002
P.MOD = 000012	RESTR2 001116R	SA.S2 = 010000	START 000660R	UF.INA= 040000
P.OPCD= 000010	RESTR2 001072R	SA.S3 = 020000	STAT 000026R	UF.RMV= 000200
P.OTRF= 000014	RES1 000056R	SA.S4 = 040000	ST.ABO= 000002	UF.RPL= 010000
P.RBN = 000014	RES2 000060R	SA.VCE= 000177	ST.AVL= 000004	UF.SCH= 004000
P.RBNS= 000056	RG.FLG= 040000	SA.VEC= 000177	ST.CMD= 000001	UF.SCL= 002000
P.RCTC= 000057	RG.OVN= 100000	SBADR 000102R	ST.CMP= 000007	UF.WBN= 000040
P.RCTS= 000054	RINTR 000256R	SCC 004174R	ST.CNT= 000012	UF.WPH= 020000
P.RGID= 000034	RLIM = 000004	SC.DIS= 000400	ST.DAT= 000010	UF.WPS= 010000
P.RGOF= 000040	RSPACK 000276R	SC.DUP= 000200	ST.DIA= 000037	UF.576= 000004
P.SFTW= 000040	RSPLN 000272R	SC.INV= 000100	ST.DRV= 000013	UNIOFF 005344R
P.SHST= 000042	RSPONC 000260R	SC.IOP= 000100	ST.HST= 000011	UNITFL 000610R
P.SHUN= 000040	RSPVIR 000274R	SC.NVL= 000040	ST.MFE= 000005	UNITNO 000602R
P.STS = 000012	RSTR1 000112R	SC.STO= 000040	ST.MSK= 000037	UNSZH 000574R
P.SZOF= 000012	R6 =#000006	SECH 000570R	ST.OFL= 000003	UNSZL 000572R
P.TIME= 000024	R7 =#000007	SECL 000566R	ST.SUB= 000040	VA 000442R
P.TRCK= 000044	SANDTO 005334R	SEND 004260R	ST.SUC= 000000	VECTOR 000010R
P.UNFL= 000016	SAREG 000252R	SEQACC 003342R	ST.WPR= 000006	WARN1 005352R
P.UNIT= 000004	SA.CMD= 034000	SETTAB 005046R	SVR0 000062R	WARN2 005356R
P.UNSZ= 000044	SA.CME= 000360	SETUP 002320R	SVR1 000064R	WARN3 005362R
P.UNTI= 000024	SA.DIA= 000400	SNDSTP 002046R	SVR2 000066R	WASADR 000104R
P.USEF= 000022	SA.ERC= 003777	SOFcnt 000042R	SVR3 000070R	WBFFEA 000452R
P.VRSN= 000014	SA.ERR= 100000	SOFER# = 104406	SVR4 000072R	WBUFEA 000136R
P.VSER= 000050	SA.GO = 000001	SOFPAS 000046R	SVR5 000074R	WBUFPA 000134R
RANACC 003164R	SA.INE= 000200	SPOINT 000032R	SVR6 000076R	WBUFRO 000140R
RAND# = 104417	SA.INT= 000200	SPSIZ = 000040	SYSCNT 000052R	WBUFSZ 000142R
RANNUM 000054R	SA.LFC= 040000	SR.CMP= 004000	TABLEW 000614R	WDFR 000116R
RBFFCA 000450R	SA.MAP= 000100	SR.DUA= 001000	TEND 000654R	WDT0 000114R
RBUF 007126R	SA.MCV= 000377	SR.REP= 000004	TIMER = 002260	WORK 000612R
RBUFEA 000130R	SA.NSI= 002000	SR.SEQ= 002000	TIMOUT= 005670	WRITE 003736R
RBUFPA 000126R	SA.PRG= 000001	SR.SUM= 000010	TRPDFD= 000022	WRITEA 003750R
RBUFSZ 000132R	SA.Q22= 001000	SR.XFR= 000002	TRY 000604R	XFLAG 000005R
RBUFVA 000124R	SA.RSE= 000017	SR1 000016R	TSTOFL 002554R	XMEM 000560R
READ 003772R	SA.RSP= 003400	SR2 000020R	UF.CMR= 000001	ZERO 005372R
READA 004024R	SA.SM = 000040	SR3 000022R		

. ABS. 000000 000 (RW,I,GBL,ABS,OVR)
 010126 001 (RW,I,LCL,REL,CON)

Errors detected: 0

*** Assembler statistics

Work file reads: 0
 Work file writes: 0
 Size of work file: 12704 Words (50 Pages)
 Size of core pool: 14336 Words (56 Pages)
 Operating system: RT-11 (Under RTEM 11)

Elapsed time: 00:00:54.00
 XDUBDO,XDUBDO/C=XDUBDO.DOC,DDXCOM.MAC,XDUBDO.MAC

DUBD DEC/X11 SYSTEM EXERCISER M MACRO V05.01b Thursday 20-Sep-84 10:17 Page 5-3
 Cross reference table (CREF V05.01)

ERRTYP	4-30	26-46*	26-59*	26-64*	26-69*	26-95*
EXIT\$	4-30	16-132	26-29			
EXPAV	4-580	14-59*	19-35*	19-98*	26-108	26-110*
FREE	4-30					
GETPA\$	4-30	15-15	15-39	16-23	16-79	16-86
GTSTAA	24-24	24-380				
GTSTAT	18-30	19-30	19-68	24-340		
GWBUF\$	4-30	15-57	15-88			
HC.CCT	8-140					
HC.CMD	8-130					
HC.CPK	8-160					
HC.RCT	8-120					
HC.RES	8-110					
HC.RPK	8-150	8-16				
HC.SIZ	8-80					
HRDCNT	4-30	15-32	27-23*	27-40*		
HRDER\$	4-30	17-49	27-44			
HRDPAS	4-30					
ICONT	4-30					
ICOUNT	4-30					
IDNUM	4-30					
IMOOD.	4-30	15-57	15-88			
INIT	4-30					
INITE1	17-38	28-20				
INITE2	17-42	28-140				
INITE3	17-46	28-180				
INITER	17-50	28-60				
INITUD	15-22	16-220	27-19			
INTA	16-120	16-1340				
INTERP	19-37	19-100	26-280	26-114		
INTR	4-30					
L.CHVR	13-130					
L.CNTI	13-110	13-120				
L.CYL	13-180					
L.DATA	13-230					
L.ERLC	13-170					
L.EVNT	13-90					
L.GRP	13-190					
L.SCTR	13-210					
L.SLOT	13-100					
L.TRCK	13-200					
L.UHVR	13-160					
L.UNTI	13-140					
L.USVR	13-150					
L.VSER	13-220					
LIMIT	5-160	22-37	22-39			
LOOP1	15-53	15-570	15-81			
LOOP2	15-630	15-78				
MA10NC	15-45	15-880	15-94			
MAITP	22-18	22-40	22-420			
MAITR	19-112	22-130				
MAITW	19-111	22-330				
MAP22\$	4-30					
MD.CMP	10-30					
MD.ERR	10-50					
MD EXP	10-40					

P.SHST	12-23#	12-47#						
P.SHUN	11-32#	12-22#	12-36#					
P.STS	12-8#	18-80	18-84	26-42	26-51	26-53	27-75	27-86
P.SZOF	12-56#							
P.TIME	11-43#							
P.TRCK	12-24#							
P.UNFL	11-28#	12-19#	12-33#					
P.UNIT	11-14#	12-5#	12-52#	18-36	26-23*	27-88		
P.UNSZ	12-37#	19-44	19-45					
P.UNTI	11-30#	12-21#	12-35#					
P.USEF	11-42#							
P.VRSN	11-39#	12-41#						
P.VSER	12-38#							
PA	4-42#	15-16	15-20	16-55	16-80	16-87		
PA18	5-5#							
PA22	5-7#							
PASCNT	4-3#	15-27						
PICKBK	19-67	20-11#						
PIRQ#	4-3#	16-135	26-32					
PKTSIZ	8-9#	8-16						
POPSP	4-3#							
POPSP2	4-3#							
PORTID	5-21#	15-66*	18-25*	18-51*	21-24	21-26	21-30	21-33
PRINTE	27-45	27-62	27-85#					
PRMSG	4-54#	14-60*	15-27	15-29*				
PRTNUM	4-53#	4-54	14-60	15-29				
PRTY	4-3#							
PRTY0	4-3	4-3#						
PRTY1	4-3#							
PRTY2	4-3#							
PRTY3	4-3#							
PRTY4	4-3	4-3#						
PRTY5	4-3#							
PRTY6	4-3#							
PRTY7	4-3#							
PS	4-3#							
PSW	4-3#							
PUSH	4-3#							
PUSH2	4-3#							
PURFLG	4-3#							
R6	4-3#							
R7	4-3#							
RANACC	20-14#							
RAND1	4-3#	14-65	20-15	20-17				
RANUM	4-3#	14-66	20-16	20-18				
RBFEEA	4-45#	15-42*	22-15	23-33				
RBUF	4-3	29-34#						
RBUFEA	4-3#	15-40						
RBUFPA	4-3#	19-85	19-115	22-16	23-34			
RBUFSZ	4-3#	22-17	23-32					
RBUFVA	4-3#	15-39						
READ	19-80	23-30#						
READA	23-17	23-35#						
RES1	4-3#							
RES2	4-3#							
RESTR1	15-19	15-24#						

