

RP04/05/06

MULTI-DRIVE LOGIC EXER
MD-11-DZRJD-A

EP-DZRJD-A-DL-A

NOV 1976

COPYRIGHT © 1976

FICHE 1 OF 1 MADE IN US

DZRJD
500

.REM 2

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DZRJD-A-D
PRODUCT NAME: RPO4/5/6 MULTI-DRIVE EXERCISER PROGRAM
DATE CREATED: MAY 1976
MAINTAINER: DIAGNOSTIC ENGINEERING
AUTHOR: C. HESS

COPYRIGHT (C) 1976 DIGITAL EQUIPMENT CORP., MAYNARD, MASS.

THE INFORMATION IN THIS STATEMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.

ACTUAL DISTRIBUTION OF THE SOFTWARE DESCRIBED IN THIS DOCUMENT WILL BE SUBJECT TO TERMS AND CONDITIONS TO BE ANNOUNCED ON SOME FUTURE DATE BY DIGITAL EQUIPMENT CORPORATION.

DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.

THIS SOFTWARE IS FURNISHED TO PURCHASER UNDER A LICENSE TO USE ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH INCLUSION OF DEC'S COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY OTHERWISE BE PROVIDED IN WRITING BY DEC.

CONTENTS

1. ABSTRACT
2. REQUIREMENTS
 - 2.1 EQUIPMENT
 - 2.2 MEDIA
 - 2.3 PRELIMINARY PROGRAMS
3. OPERATING THE PROGRAM
 - 3.1 LOADING THE PROGRAM
 - 3.2 STARTING THE PROGRAM
 - 3.3 RESTARTING THE PROGRAM
 - 3.4 PROGRAM CONTROL
 - 3.5 PASS/TEST TERMINATION
 - 3.5.1 PASS TERMINATION
 - 3.5.2 TEST TERMINATION
 - 3.6 RUN TIME
 - 3.6.1 DATA TRANSFER MODE
 - 3.6.2 SEEK VERIFICATION MODE
 - 3.7 UNIBUS & VECTOR ADDRESSES
 - 3.8 DUAL PORT OPERATION
4. CONTROLLING THE PROGRAM
 - 4.1 DATE & OPERATOR IDENTIFICATION
 - 4.2 PARAMETERS
 - 4.2.1 PROGRAM CONTROL PARAMETERS
 - 4.2.2 PERIPHERAL DEVICE ADDRESSES
 - 4.2.3 PARAMETERS FOR THE FIRST OPERATION
 - 4.3 SWITCH REGISTER SETTINGS
 - 4.4 KEYBOARD COMMANDS
 - 4.4.1 'T' COMMAND
 - 4.4.2 'D' COMMAND
 - 4.4.3 'S' COMMAND
 - 4.4.4 'W' COMMAND
 - 4.4.5 'R' COMMAND
 - 4.4.6 GENERAL COMMAND INFORMATION
5. PERFORMANCE SUMMARY TYPEOUT
 - 5.1 PERFORMANCE SUMMARY TYPEOUT EXPLANATION
 - 5.2 HARD/SOFT ERROR DEFINITIONS
 - 5.2.1 HARD ERRORS
 - 5.2.2 SOFT ERRORS
6. DATA CHECKING & ERROR RECOVERY
 - 6.1 DATA BUFFER COMPARISON
 - 6.2 VERIFICATION OF DATA WRITTEN
 - 6.3 SECTOR REFORMATTING

6.4 BAD TRACK/SECTOR FLAGGING

7. ERROR MESSAGES

- 7.1 ERROR DESCRIPTION LINES
- 7.2 DETAIL ERROR LINES

8. PROGRAM DESCRIPTION

- 8.1 HOW THE PROGRAM OPERATES
- 8.2 DUAL PORT OPERATION
- 8.3 HOW VARIABLES ARE SELECTED FOR EACH OPERATION
- 8.4 DATA PATTERNS

9. PROGRAM LISTING

1. ABSTRACT

THE RPO4/5/6 MULTIDRIVE EXERCISER PROGRAM IS DESIGNED TO PERFORM AN INTERACTIVE TEST ON RPO4/5/6 DISK DRIVES CONNECTED TO A MASSBUS SUBSYSTEM. THE SUBSYSTEM MAY BE COMPOSED OF INTERMIXED RPO4, RPO5 OR RPO6 DISK DRIVES CONTROLLED BY EITHER AN RH11 OR AN RH70. IN ADDITION TO PERFORMING AN INTERACTIVE TEST OF THE DISK DRIVES ON THE SUBSYSTEM, THE PROGRAM IS INTENDED TO BE USED TO VERIFY THAT THE DRIVES UNDER TEST ARE PERFORMING TO THEIR DATA ERROR RATE AND SEEK ERROR RATE SPECIFICATIONS.

THE RPO4/5/6 MULTIDRIVE EXERCISER PROGRAM WILL EXERCISE DRIVES CONNECTED AS EITHER SINGLE OR DUAL PORT UNITS. DUAL PORT DRIVES ARE TESTED BY LOADING AND RUNNING THE PROGRAM FROM BOTH CONTROLLING SYSTEMS. THE PROGRAM WILL EXERCISE A MIXED SYSTEM OF DUAL PORT AND NON DUAL PORT DRIVES.

TO OBTAIN INTERACTIVE TESTING, OPERATIONS ON THE DRIVES ARE OVERLAPPED (OTHER DRIVES ARE PERFORMING SEEK/SEARCH OPERATIONS WHILE ONE DRIVE IS PERFORMING A DATA TRANSFER OR WRITE CHECK OPERATION). OPERATIONS AMONG THE DRIVES ARE OPTIMIZED SO THAT A HIGH SUBSYSTEM DATA TRANSFER RATE OR A HIGH POSITIONING OPERATION RATE IS MAINTAINED.

THE PERFORMANCE OF EACH DRIVE IS MONITORED BY THE PROGRAM. IF A DRIVE EXCEEDS A PRESET NUMBER OF ERRORS IN ANY OF SEVERAL CATEGORIES, THAT DRIVE IS AUTOMATICALLY DEASSIGNED. (THE OPERATOR MAY OVERRIDE THE AUTOMATIC DEASSIGNMENT FEATURE.) THE PROGRAM REPORTS PERFORMANCE STATISTICS FOR EACH DRIVE BEING EXERCISED ON REQUEST FROM THE OPERATOR OR AUTOMATICALLY AT AN INTERVAL DETERMINED BY THE OPERATOR.

ALL DATA TRANSFER COMMANDS ARE USED (I.E., WRITE DATA, WRITE HEADER & DATA, READ DATA, AND READ HEADER & DATA) AS WELL AS WRITE CHECK DATA AND WRITE CHECK HEADER & DATA COMMANDS. RECALIBRATE AND READ-IN PRESET COMMANDS ARE USED AT STARTUP AND DRIVE INITIALIZATION. RECALIBRATE, OFFSET, AND RETURN-TO-CENTERLINE COMMANDS ARE USED DURING ERROR PROCESSING.

THE DATA TRANSFER COMMANDS ARE SELECTED RANDOMLY EXCEPT FOR THE WRITE CHECK COMMANDS. THE WRITE CHECK COMMANDS ARE USED TO VERIFY A PREVIOUS WRITE OPERATION. THUS, WHEN A WRITE COMMAND IS SELECTED, THE DATA WRITTEN IS VERIFIED BY THE APPROPRIATE WRITE CHECK COMMAND.

PROGRAM OPERATOR COMMUNICATIONS ARE THROUGH THE KEYBOARD; DYNAMIC PROGRAM OPTIONS ARE SELECTED BY SWITCH REGISTER SETTINGS. ERRORS ARE REPORTED ON THE TELETYPE.

ALL COMMANDS, DATA PATTERNS, AND DATA BUFFER SIZES ARE SELECTED RANDOMLY BY THE PROGRAM. ADDITIONALLY THE ADDRESSES (EG, CYLINDER, TRACK, AND SECTOR) FOR EACH OPERATION ARE SELECTED RANDOMLY.

2. REQUIREMENTS

2.1 EQUIPMENTREQUIRED

PDP-11 PROCESSOR
16K MEMORY (20K IF THE PROGRAM IS INCLUDED IN AN 'XXDP' CHAIN
TELETYPE
PROGRAM LOADING DEVICE
KW11-L OR KW11-P CLOCK
RH11 OR RH70 WITH 1 RPO4, RPO5, OR RPO6 DISK DRIVE

OPTIONAL

ADDITIONAL MEMORY TO A MAXIMUM OF 28K
1 TO 7 ADDITIONAL RPO4/5/6'S ON THE SAME RH11 OR RH70

2.2 MEDIA

THE RPO4/5/6 MULTIDRIVE EXERCISER PROGRAM REQUIRES FORMATTED DISK PACKS WHICH CONTAIN RANDOM OR PATTERNED DATA RECOGNIZED BY THE EXERCISER. DISK PACKS USED BY THE PROGRAM MAY BE GENERATED BY THE RPO4/5/6 FORMATTER PROGRAM (MAINDEC-11-DZRJB) OR BY THE 'W' COMMAND OF THE RPO4/5/6 MULTIDRIVE EXERCISER (SEE SECTION 4.4). THE PACKS MUST BE FORMATTED IN 22 SECTOR (16 BIT) MODE; THE ALTERNATE (20 SECTOR - 18 BIT) MODE IS NOT SUPPORTED.

2.3 PRELIMINARY PROGRAMS

RPO4/5/6 DISKLESS CONTROLLER TEST
PART 1 (MAINDEC-11-DZRJG)
PART 2 (MAINDEC-11-DZRJH)

RPO4/5/6 FUNCTIONAL CONTROLLER TEST
PART 1 (MAINDEC-11-DZRJI)
PART 2 (MAINDEC-11-DZRJJ)

RPO4/5/6 DUAL CONTROLLER LOGIC TEST (FOR DUAL PORT DRIVE TESTING)
PART 1 (MAINDEC-11-DZRJE)
PART 2 (MAINDEC-11-DZRJF)

3. OPERATING THE PROGRAM

3.1 THE PROGRAM MAY BE LOADED FROM PAPER TAPE USING THE ABSOLUTE LOADER OR IT MAY BE LOADED FROM THE APPROPRIATE 'XXDP' MEDIA USING THE ASSOCIATED LOADER. THE PROGRAM MAY BE INCLUDED IN AN 'XXDP' CHAIN. IF THE

PROGRAM IS BEING RUN ON A PROCESSOR WITH 16K, THE 'XXDP' LOADER WILL NOT BE PRESERVED. THE PROGRAM MUST BE RUN ON A SYSTEM WITH 20K OR MORE TO PRESERVE THE 'XXDP' LOADER. THE 'ABSOLUTE' LOADER WILL BE PRESERVED IN A 16K SYSTEM, HOWEVER.

- 3.2 THE PROGRAM STARTS AT LOCATION 200(8). PARAMETERS NOT INCLUDED IN THE TELETYPE DIAGLOGUE GROUP MUST BE CHANGED BEFORE THE PROGRAM IS STARTED.
- 3.3 START THE PROGRAM AT LOCATION 204(8) IF THE RH11 OR THE RH70 IS NOT AT ADDRESS 176700.
- 3.4 ONCE THE PROGRAM IS LOADED AND STARTED, OPERATIONS ARE INITIATED AND CONTROLLED BY KEYBOARD COMMANDS AND SWITCH REGISTER SWITCH SETTINGS.

IF THIS IS THE PROGRAM'S FIRST START, THE STATUS OF THE DRIVES ON THE SELECTED MASSBUS SUBSYSTEM WILL BE TYPED OUT. ON SUBSEQUENT STARTS, THIS TYPEOUT MAY BE INHIBITED BY SETTING SW<02>.

3.5 PASS/TEST TERMINATION

A PASS IS DETERMINED BY EITHER BITS READ OR SEEKS PERFORMED. THE NUMBER OF BITS OR SEEKS REQUIRED FOR A PASS IS DERIVED FROM EITHER THE SOFT ERROR RATE SPECIFICATION OR THE SEEK ERROR RATE SPECIFICATION. THE SPECIFICATIONS FOR RPO4'S, RPO5'S, AND RPO6'S SPECIFY NO MORE THAN 1 SOFT ERROR (NON-PACK RELATED) IN 1×10^{19} BITS READ OR NO MORE THAN 1 SEEK ERROR IN 1×10^{16} SEEKS. THE NUMBER OF BITS OR SEEKS DETERMINING A PASS WERE SELECTED TO PROVIDE A 90% CONFIDENCE LEVEL THAT THE DRIVE IS PERFORMING TO THE APPLICABLE SPECIFICATION.

3.5.1 PASS TERMINATION

END OF PASS MAY BE DETERMINED BY EITHER OF THE FOLLOWING CONDITIONS. THE END OF PASS CONDITION USED IS DETERMINED BY PARAMETER 'ENDET'.

- A. IF PARAMETER 'ENDET' IS 1, END OF PASS OCCURS WHEN THE DRIVE HAS READ 1.875×10^{18} WORDS (3×10^{19} BITS).
- B. IF PARAMETER 'ENDET' IS 0, END OF PASS OCCURS WHEN THE DRIVE HAS PERFORMED 3×10^{16} SEEKS.

3.5.2 TEST TERMINATION

THE TEST FOR A DRIVE IS TERMINATED (SW<04> = 0) WHEN:

- A. THE DRIVE HAS COMPLETED THE NUMBER OF PASSES SPECIFIED IN PARAMETER 'PASCNT'.
- B. THE TOTAL ERRORS ACCUMULATED EXCEED 100.
- C. A FATAL ERROR OCCURS: EM12 OR EM14.

3.6 RUN TIME

THE EXERCISER PROGRAM MAY BE RUN IN TWO MODES. THE MODE IS DETERMINED BY THE VALUE IN PARAMETER 'MAXDL'. IF 'MAXDL' IS ONE SECTOR, THE PROGRAM RUNS IN A SEEK HEAVY MODE; IF 'MAXDL' APPROACHES

ONE TRACK IN SIZE (5720 DECIMAL) THE PROGRAM RUNS IN A DATA TRANSFER HEAVY MODE. THE PROGRAM RUN TIME VARIES GREATLY DEPENDING ON THE OPERATION MODE SELECTED, THE MEMORY AVAILABLE OVER 16K, THE READ/WRITE RATIO PARAMETER - 'RATIO', AND BY SWITCHES 0, 1, & 2.

3.6.1 DATA TRANSFER MODE

1 DRIVE - APPROXIMATELY 2.5 HRS (TO REACH 1.875×10^{18} WORDS)
TO
8 DRIVES - APPROXIMATELY 11 HRS (FOR ALL DRIVES TO REACH 1.875×10^{18} WORDS)

NOTE: IF SW<01> = 1 (NO SOFTWARE DATA COMPARSIONS), THE RUN TIMES ARE THE FOLLOWING VALUES, APPROXIMATELY:

1 DRIVE - 1.7 HRS (1.875×10^{18} WORDS READ)
ADD 1/2 HOUR FOR EACH ADDITIONAL DRIVE TESTED.

IF THE PROGRAM IS RUN WITH BOTH SW<00> AND SW<01> SET, THE RUN TIMES SHOULD BE ABOUT 20% FASTER.

3.6.2 SEEK VERIFICATION MODE

PARAMETER 'MAXDL' = 1 SECTOR (256 WORDS)
PARAMETER 'MAXTRK' = 'MINTRK'
PARAMETER 'MAXSEC' = 'MINSEC'
SW<00> = 1 (READ ONLY MODE)

1 DRIVE - APPROXIMATELY 25 HRS (3×10^{16} SEEKS)
TO
8 DRIVES - APPROXIMATELY 40 HRS (3×10^{16} SEEKS FOR ALL DRIVES)

3.7 UNIBUS & VECTOR ADDRESSES

THE PROGRAM ASSUMES THE FOLLOWING UNIBUS AND VECTOR ADDRESSES. (REFER TO SECTION 4.2.2 FOR THE LOCATIONS AT WHICH TO CHANGE THESE ADDRESSES.)

UNIT	UNIBUS ADDRESS	VECTOR ADDRESS
RH11 OR RH70	176700	254
TTY PRINTER	177564	NOT USED
TTY KEYBOARD	177560	60
KW11-L	177546	100
KW11-P	172542	104

3.8 DUAL PORT OPERATION

A. LOAD THE RPO4/5/6 MULTIDRIVE EXERCISER PROGRAM INTO BOTH PROCESSORS.

- B. SWITCH THE 'CONTROLLER SELECT' SWITCH TO 'A/B' ON EACH DRIVE WHICH IS TO BE TESTED AS A DUAL PORT DRIVE; CYCLE THE DRIVES UP.
- C. START THE PROGRAM IN EACH PROCESSOR. RUN THE PROGRAM AS THROUGH EACH PROCESSOR WERE RUNNING INDEPENDENTLY OF THE OTHER.

4. CONTROLLING THE PROGRAM

THE FOLLOWING KEYBOARD CONVENTIONS ARE USED BY THE KEYBOARD ENTRY ROUTINES IN THE PROGRAM:

- A. TO DELETE AN INCORRECT CHARACTER FROM AN ENTRY STRING, TYPE A 'RUBOUT' ('RO'). TYPING A 'RO' WILL DELETE SUCESSIVE CHARACTERS FROM THE INPUT.
- B. TO DELETE AN ENTIRE LINE, TYPE A 'CONTROL U' ('↑U').
- C. AN ENTRY MUST BE TERMINATED BY EITHER A 'CARRIAGE RETURN' OR A 'PERIOD'. THE 'PERIOD' TERMINATION IS RECOGNIZED BY THE PROGRAM AS A DEFAULT ENTRY REQUEST. WHEN A LINE IS TERMINATED BY A 'PERIOD' INSTEAD OF A 'CARRIAGE RETURN', THE PROGRAM WILL ACCEPT THE ENTERED VALUE AND WILL DEFAULT TO THE PRELOADED VALUES FOR ANY REMAINING ENTRIES.
- D. IF A 'CONTROL C' IS TYPED DURING KEYBOARD ENTRY, THE PROGRAM WILL RETURN TO THE BEGINNING OF THE GROUP BEING ENTERED.

4.1 DATE & OPERATOR IDENTIFICATION

WHEN THE PROGRAM IS INITIALLY STARTED, IT WILL ASK FOR DATE AND OPERATOR I.D. ENTRIES. (THE REQUEST FOR THESE ENTRIES OCCURS ONLY WHEN THE PROGRAM IS FIRST STARTED AND WILL NOT APPEAR WHEN THE PROGRAM IS RESTARTED.) THESE ENTRIES ARE OPTIONAL AND MAY BE BYPASSED BY ENTERING A 'CARRIAGE RETURN' IN RESPONSE TO THE REQUEST. THE PROGRAM DOES NOT EDIT OR CHECK EITHER ENTRY. UP TO 8 CHARACTERS OF DATE INFORMATION AND UP TO 6 CHARACTERS OF OPERATOR IDENTIFICATION MAY BE ENTERED. BOTH THE DATE AND THE OPERATOR I.D. WILL BE TYPED WHEN THE 'SA' COMMAND IS PERFORMED (SEE SECTION 4.4.3).

4.2 PARAMETERS

WHEN THE PROGRAM IS STARTED FROM LOCATION 200, THE OPERATOR WILL BE ASKED TO ENTER PARAMETERS. THE FOLLOWING MESSAGE WILL BE DISPLAYED:

ENTER PARAMETERS:

THE OPERATOR MUST ENTER A 'Y' TERMINATED BY A CARRIAGE RETURN <CR> IF PARAMETER ENTRIES ARE TO BE MADE. ANY OTHER CHARACTER IS ACCEPTED AS A 'NO' ENTRY. THE PROGRAM WILL IDENTIFY THE PARAMETER BY THE NAME

GIVEN BELOW, DISPLAY THE CURRENT VALUE OF THE PARAMETER AND WAIT FOR THE ENTRY. THE PROGRAM WILL TYPE 'INVALID ENTRY' IF THE ENTRY IS NOT CORRECT AND WAIT FOR A CORRECT ENTRY TO BE TYPED.

4.2.1 KEYBOARD ENTRY PARAMETERS

NAME	BASE	DEFAULT VALUE	VALUE RANGE	FUNCTION
MAXDL	10	(SEE NOTE)		CONTROLS THE MAXIMUM BUFFER SIZE USED FOR DATA TRANSFERS
PASCNT	10	1	1 - 999	NUMBER OF PASSES TO END OF TEST.
INTRVL	10	5	0 - 256	DETERMINES THE INTERVAL (IN MINUTES) BETWEEN AUTOMATIC PERFORMANCE SUMMARY TYPEOUTS
CMPPLMT	10	3	0 - 'MAXDL'	ERRORS PRINTED OUT IF SW<07>=0
WCSEL	8	000000	0 OR 1	THE NUMBER OF DATA COMPARSION IF PARAMETER = 0, THE DATA TRANSFER WORD COUNT IS RANDOMLY SELECTED BETWEEN 4 (10) AND THE VALUE IN 'MAXDL'
ENDET	8	000001	0 OR 1	IF PARAMETER = 1, THE DATA TRANSFER WORD COUNT WILL BE THE VALUE IN 'MAXDL'
FORMAT	8	000001	0 OR 1	IF PARAMETER = 1, END OF PASS DETERMINED BY THE 'WORDS READ' COUNT.
RATIO	8	000003	0 - 7	IF PARAMETER = 0, END OF PASS IS DETERMINED BY THE NUMBER OF SEEKS.
AUTOCK	8	000001	0 OR 1	IF PARAMETER = 0; DO NOT PERFORM WRITE HEADER & DATA ORDERS; IF PARAMETER > 0, PERFORM WRITE HEADER & DATA ORDERS

VALUE R/W RATIO

0	15/1
1	7/1
2	6/2
3	5/3
4	4/4
5	3/5
6	2/6
7	1/7

IF PARAMETER = 1, THE PROGRAM PERFORM WRITE CHECKS AFTER

NOTPRT 8 000001 0 OR 1

EACH WRITE COMMAND.
IF PARAMETER = 0, THE PROGRAM
WILL PERFORM WRITE CHECKS
RANDOMLY.
IF PARAMETER = 1, DO NOT PRINT
ERROR MESSAGES FOR DATA ERRORS
OCCURRING AT LOCATIONS DEFINED
BY THE OPERATOR AS BAD PACK
LOCATION.
IF PARAMETER = 0, PRINT ERROR
MESSAGES ASSOCIATED WITH BAD
PACK LOCATIONS.

NOTE: THE PROGRAM WILL SELECT A MAXIMUM BUFFER SIZE WHICH
IS DETERMINED BY THE MEMORY AVAILABLE. THE MAXIMUM BUFFER
SIZE ASSIGNED BY THE PROGRAM IS 5980 (10) WORDS. THE
OPERATOR MAY SPECIFY ANY OTHER MAXIMUM SIZE AS LONG AS THE
VALUE SPECIFIED IS AT LEAST 4 WORDS BUT NO LARGER THAN
THE INITIAL VALUE OF 'MAXDL' DETERMINED BY THE PROGRAM.

4.2.2 PERIPHERAL ADDRESSES AND OTHER LOCATIONS OF INTEREST

TO ALTER THESE LOCATIONS, THE OPERATOR MUST MAKE MANUAL ENTRIES
BEFORE THE PROGRAM IS STARTED. THE KEYBOARD ENTRY ROUTINE DOES
NOT PROVIDE ACCESS TO THESE LOCATIONS.

LOC	TAG	CONTENTS	FUNCTION
---	---	-----	-----
1144	\$TKS	177560	TTY KEYBOARD STATUS REGISTER
1146	\$TKB	177562	TTY KEYBOARD BUFFER REGISTER
1150	\$TPS	177564	TTY PRINTER STATUS REGISTER
1152	\$TPB	177566	TTY PRINTER BUFFER REGISTER
1174	\$LKCSR	172540	ADDRESS OF KW11-P STATUS REGISTER
1176	\$LKCSB	172542	ADDRESS OF KW11-P COUNTER BUFFER
1200	\$LPVEC	104	KW11-P VECTOR ADDRESS
1202	\$LKS	177546	ADDRESS OF KW11-L STATUS REGISTER
1204	\$LLVEC	100	KW11-L VECTOR ADDRESS
1212	HZ	74	74 (60 DECIMAL) IF SYSTEM IS 60 HZ; 62 (50 DECIMAL) IF SYSTEM IS 50 HZ.

THE RH11-RH70 ADDRESS AND VECTOR MAY BE CHANGED WHEN THE PROGRAM
IS STARTED FROM LOCATION 204(8) OR IF THE PROGRAM DOES NOT RECEIVE
A RESPONSE WHEN IT ACCESSES THE DEFAULT RH11-RH70 ADDRESS.

4.2.3 PARAMETERS FOR THE FIRST OPERATION

THE FOLLOWING PARAMETERS ARE USED FOR THE INITIAL OPERATION (IN
ADDITION TO THE 'MINIMUM' ADDRESS VALUES).

INITIAL VALUE

LOC	TAG	VALUE	RANGE	FUNCTION
1412	BEGPAT	10	1 - 15	THE CODE FOR THE STARTING PATTERN. (IF A WRITE ORDER OR A WRITE CHECK ORDER IS SPECIFIED IN 'BEGCOD')
1414	BEGCOD	5	0 - 5	THE INITIAL COMMAND FOR EACH DRIVE EXERCISED. 0 = WRITE CHECK DATA 1 = WRITE CHECK HEADER & DATA 2 = WRITE DATA 3 = WRITE HEADER & DATA 4 = READ DATA 5 = READ HEADER & DATA
1416	BEGSIZ	404	4 - MAXDL	THE BUFFER SIZE FOR THE FIRST DATA TRANSFER OPERATION.

4.3 SWITCH REGISTER SETTINGS

SW <15> = 1 HALT ON ERROR
 SW <13> = 1 INHIBIT ERROR TYPEOUT
 SW <10> = 1 RING THE TELETYPE BELL IF ERROR
 SW <7> = 1 DISPLAY ALL DATA COMPARE ERRORS
 SW <6> = 1 DO NOT ALTER THE CURRENT OPERATION PARAMETERS
 SW <5> = 1 PARTIAL REGISTER DISPLAY IF ERROR; DO NOT DISPLAY ECC CORRECTION RESULTS
 SW <4> = 1 INHIBIT MAXIMUM ERROR COUNT CHECK; DO NOT DEASSIGN DRIVES WHEN NORMAL END OF TEST REACHED.
 SW <3> = 1 DISPLAY THE SECTOR IN ERROR (BEFORE RETRY ATTEMPTS) IF 'DCK', 'DTE', OR 'WCF' ERRORS OR AFTER THE 28TH RETRY IF 'UNCORRECTABLE' 'DCK' ERROR.
 IF DATA COMPARE ERRORS & SW<7> SET, DISPLAY REST OF BUFFER
 SW <2> = 1 INHIBIT SUBSYSTEM STATUS TYPEOUT DURING STARTUP.
 INHIBIT PERFORMANCE SUMMARY TYPEOUTS.
 SW <1> = 1 INHIBIT DATA COMPARISON AFTER READ ORDERS
 SW <0> = 1 READ ONLY MODE

IF THE PROGRAM IS BEING RUN ON A SWITCHLESS PROCESSOR (I.E. AN 11/34) THE PROGRAM WILL DETERMINE THAT THE HARDWARE SWITCH REGISTER IS NOT PRESENT AND WILL USE A 'SOFTWARE' SWITCH REGISTER. THE 'SOFTWARE' SWITCH REGISTER IS LOCATED AT LOCATION 176 (8). THE SETTINGS OF THE 'SOFTWARE' SWITCHES ARE CONTROLLED THROUGH A KEYBOARD ROUTINE WHICH IS CALLED BY TYPING A 'CONTROL G'. THE PROGRAM WILL RECOGNIZE THE 'CONTROL G' AT ANY TIME EXCEPT WHEN THE PROGRAM IS IN KEYBOARD ENTRY MODE, OR IS AT A HIGHER PRIORITY PROCESSING AN RPO4/5/6 INTERRUPT. THE 'SOFTWARE' SWITCH VALUES ARE ENTERED AS AN OCTAL NUMBER IN RESPONSE TO THE PROMPT FROM THE SWITCH ENTRY ROUTINE:

'SWR = NNNNNN NEW ='

EACH TIME SWITCH SETTING ARE ENTERED, THE ENTIRE SWITCH REGISTER

IMAGE MUST BE ENTERED. LEADING ZEROS ARE NOT REQUIRED. 'RUBOUT' AND 'CONTROL U' FUNCTIONS MAY BE USED TO CORRECT TYPING ERRORS DURING SWITCH ENTRY.

ON PROCESSORS WITH HARDWARE SWITCH REGISTERS, THE 'SOFTWARE' SWITCH REGISTER MAY BE USED. IF THE PROGRAM FINDS ALL 16 SWITCHES IN THE 'UP' POSITION, ALL SWITCH REGISTER REFERENCES WILL BE TO THE 'SOFTWARE' REGISTER AND THE PROCEDURES DESCRIBED ABOVE MUST BE FOLLOWED.

4.4 KEYBOARD COMMANDS

THROUGH THE KEYBOARD COMMANDS, THE OPERATOR MAY ASSIGN DRIVES FOR TEST ('T' COMMAND), WRITE AND CHECK DATA PACKS ('W' COMMAND), PERFORM A SEQUENTIAL READ OF A PACK ('R' COMMAND), REQUEST A DRIVE PERFORMANCE SUMMARY ('S' COMMAND), OR DEASSIGN A DRIVE WHICH IS BEING TESTED, READING, OR WRITING ('D' COMMAND).

AFTER THE PROGRAM HAS BEEN INITIALIZED, THE FOLLOWING MESSAGE WILL BE TYPED:

'PROGRAM INITIALIZATION COMPLETE
'TYPE A CONTROL C TO ENTER COMMANDS'

KEYBOARD ENTRIES WILL NOT BE RECOGNIZED UNTIL THE OPERATOR TYPES A 'CONTROL C'. WHEN THE PROGRAM SEES A 'CONTROL C' ENTRY, IT WILL SUSPEND THE SCHEDULING OF FURTHER DEVICE OPERATIONS AND WAIT UNTIL ALL OUTSTANDING ORDERS HAVE TERMINATED. THE PROGRAM WILL ENTER COMMAND ENTRY MODE AND TYPE THE FOLLOWING PROMPTING MESSAGE:

'HH:MM:SS
'ENTER COMMANDS:'

THE PROGRAM WILL THEN ACCEPT ANY OF THE VALID COMMANDS. AT THE COMPLETION OF A COMMAND, THE PROGRAM WILL EXIT COMMAND MODE; THE OPERATOR MUST TYPE ANOTHER 'CONTROL C' TO RETURN THE PROGRAM TO COMMAND MODE. IF THE COMMAND ENTERED SPECIFIED AN 'A' DRIVE NUMBER, THE PROGRAM WILL REMAIN IN COMMAND MODE UNTIL ALL AVAILABLE DRIVES HAVE BEEN PROCESSED.

THE 'T', 'W', AND 'R' COMMANDS REQUIRE ADDRESS LIMITS, DRIVE I.D, AND BAD LOCATION ADDRESS ENTRIES FOR THE DRIVE BEING REFERENCED.

THE PROGRAM WILL FIRST ASK FOR ADDRESS LIMITS WITH THE FOLLOWING TYPEOUT:

'ENTER ADDRESS LIMITS FOR DRV #N / RP04' (OR RP05 OR RP06)

THE PROGRAM WILL REQUEST VALUES FOR THE FOLLOWING ADDRESS LIMIT PARAMETERS.

NAME	BASE	DEFAULT VALUE	VALUE RANGE	FUNCTION
------	------	---------------	-------------	----------

MINCYL	10		(SEE NOTE)	THE MINIMUM CYLINDER ADDRESS
MAXCYL	10		(SEE NOTE)	THE MAXIMUM CYLINDER ADDRESS
MINTRK	10	0	0 - 18	THE MINIMUM TRACK ADDRESS
MAXTRK	10	18	0 - 18	THE MAXIMUM TRACK ADDRESS
MINSEC	10	0	0 - 21	THE MINIMUM SECTOR ADDRESS
MAXSEC	10	21	0 - 21	THE MAXIMUM SECTOR ADDRESS

- NOTE: 1. THE ADDRESS LIMITS ARE IN DECIMAL
2. THE MAXIMUM VALUES OF 'MINCYL' AND 'MAXCYL' WILL BE EITHER 410 (10) OR 814 (10) DEPENDING ON THE TYPE OF DRIVE.
3. THE MINIMUM CYLINDER, TRACK, OR SECTOR ADDRESS MAY BE SPECIFIED AS BEING LARGER THAN THE MAXIMUM ADDRESS. WHEN THESE VALUES ARE INVERTED, THE PROGRAM WILL SELECT ADDRESSES BETWEEN THE 'MIN' ADDRESS AND THE UPPER PHYSICAL LIMIT FOR THAT ADDRESS AND BETWEEN '0' AND THE VALUE IN 'MAX'.

EACH COMMAND, EXCEPT THE 'S' AND THE 'D' COMMANDS, WILL ASK FOR A DRIVE IDENTIFICATION ENTRY. THE DRIVE IDENTIFICATION ENTRY REQUEST IS MADE BY THE FOLLOWING MESSAGE:

'ENTER I.D. FOR DRV #N:'

THE OPERATOR MAY ENTER AN I.D. NUMBER FOR THE DRIVE OF UP TO 6 CHARACTERS IN LENGTH. THIS I.D. WILL BE DISPLAYED, ALONG WITH THE DATE AND OPERATOR I.D. ENTRIES (SEE SECTION 4.1), WHEN THE 'SA' COMMAND IS EXECUTED. THE OPERATOR MAY ENTER ANY CHARACTER STRING, TERMINATED BY A 'CARRIAGE RETURN', OR A 'PERIOD' ONLY (NULL ENTRY) IN RESPONSE TO THE I.D. REQUEST.

THE PROGRAM WILL THEN ASK FOR THE ADDRESSES OF KNOWN BAD SPOTS ON THE DISK PACK USED, UP TO 16 ADDRESSES MAY BE ENTERED:

'BAD TRK/SEC ADRS FOR DRV #N ?'

THE OPERATOR MAY DECLARE UP TO 16 BAD LOCATIONS ON THE PACK BEING USED. THE LOCATION MAY BE AN ENTIRE CYLINDER, A BAD TRACK, OR A SINGLE SECTOR. THE FORMATS USED ARE AS FELLOW:

FORMAT 1: C,T,S<CR>

- A. THE PROGRAM WILL INHIBIT DATA ERROR MESSAGES OR WILL IDENTIFY DATA ERRORS WHICH OCCUR AT THE SPECIFIED ADDRESS, DEPENDING ON THE VALUE OF PARAMETER 'NOTPRT'.
- B. LEADING ZEROS ARE NOT REQUIRED.

FORMAT 2: C,T<CR>

- A. WHEN THIS FORMAT IS USED, THE ENTIRE TRACK WILL BE CONSIDERED BAD; DATA ERRORS WILL BE HANDLED AS IN 'FORMAT 1', ABOVE.

B. LEADING ZEROS ARE NOT REQUIRED.

FORMAT 3: C<CR>

- A. WHEN THIS FORMAT IS USED, THE ENTIRE CYLINDER WILL BE CONSIDERED BAD; DATA ERRORS WILL BE HANDLED AS IN 'FORMAT 1' ABOVE.
- B. LEADING ZEROS ARE NOT REQUIRED.

NOTE: CYLINDER, TRACK, AND SECTOR ENTRIES ARE IN DECIMAL.

THE OPERATOR MAY TERMINATE THE BAD ADDRESS ENTRY BY ENTERING A 'PERIOD' IN RESPONSE TO THE ENTRY REQUEST OR BY TERMINATING AN ENTRY WITH A 'PERIOD' INSTEAD OF A 'CARRIAGE RETURN'.

4.4.1 'T' COMMAND

USED TO ASSIGN A DRIVE(S) FOR TEST. THIS COMMAND IS REQUIRED TO PERFORM THE TEST OF THE DRIVE(S). THE OTHER COMMANDS ARE CONVIENCE COMMANDS OR SUPPORT COMMANDS.

FORMAT: TN<CR>

N = DRIVE NUMBER. MAY BE 0 TO 7 OR 'A'. ENTRY MUST BE TERMINIATED BY A CARRIAGE RETURN <CR>.

EXAMPLE: T0<CR> - ASSIGN DRIVE 0 FOR TEST
TA<CR> - ASSIGN ALL AVAILABLE DRIVES FOR TEST

NOTE: DRIVE OPERATION BEGINS IMMEDIATELY AFTER COMMAND IS ENTERED.

4.4.2 'D' COMMAND

USED TO DEASSIGN A DRIVE(S) BEING EXERCISED.

FORMAT: DN<CR>

N = DRIVE NUMBER. MAY BE 0 TO 7 OR 'A'. ENTRY MUST BE TERMINIATED BY A CARRIAGE RETURN <CR>.

EXAMPLE: D0<CR> - DEASSIGN DRIVE 0
DA<CR> - DEASSIGN ALL DRIVES BEING TESTED.

- NOTES:
1. IF THE 'D' COMMAND REFERENCES A DRIVE NOT ASSIGNED THE PROGRAM WILL TYPEOUT '?DRIVE NOT ASSIGNED'
 2. THE DRIVES WILL BE DEASSIGNED AS THEIR OPERATIONS COMPLETE.
 3. IF 'DA' IS USED, ONLY DRIVES BEING

TESTED WILL BE DEASSIGNED - THE
ERROR MESSAGE IN (1) ABOVE WILL NOT
BE DISPLAYED.

4.4.3 'S' COMMAND

USED TO REQUEST A PERFORMANCE SUMMARY TYPEOUT FOR THE REFERENCED
DRIVE(S).

FORMAT: SN<CR>

N = DRIVE NUMBER. MAY BE 0 TO 7 OR 'A'. ENTRY MUST BE
TERMINATED BY A CARRIAGE RETURN <CR>.

EXAMPLE: SO<CR> - TYPEOUT PERFORMANCE SUMMARY FOR DRIVE 0
SA<CR> - TYPEOUT PERFORMANCE SUMMARY
FOR ALL DRIVES BEING TESTED.

- NOTES:
1. IF 'SA' IS USED ONLY DRIVES BEING TESTED WILL
BE DISPLAYED.
 2. IF PARAMETER 'INTRVL' IS NOT ZERO, THE PROGRAM
WILL AUTOMATICALLY DISPLAY A PERFORMANCE SUMMARY
FOR EACH DRIVE BEING TESTED AT A RATE DETERMINED BY
'INTRVL'.
 3. IF THE 'SA' COMMAND IS USED, THE PROGRAM WILL TYPEOUT
THE OPERATOR ENTERED DATE, OPERATOR I.D., AND THE DRIVE
I.D. FOR EACH DRIVE BEING TESTED. THE DATE AND OPERATOR
I.D. WILL NOT BE TYPED OUT IF NO DRIVES ARE BEING TESTED.

4.4.4 'W' COMMAND

USED TO WRITE A DATA PACK WITH DATA ACCEPTABLE TO THE RPO4/5/6 MULTI-
DRIVE EXERCISER PROGRAM.

FORMAT: WN<CR>

N = DRIVE NUMBER. MAY BE 0 TO 7 OR 'A'. ENTRY MUST BE
TERMINATED BY A CARRIAGE RETURN <CR>.

EXAMPLE: WA<CR> - WRITE DATA PACKS ON ALL AVAILABLE DRIVES.
WD<CR> - GENERATE A DATA PACK ON DRIVE 0.

- NOTES:
1. DATA PACKS GENERATED BY THE RPO4/5/6 FORMATTER PROGRAM
(MD-11-DZRJB) OR BY THE RPO4/5/6 MECHANICAL & READ/WRITE
PROGRAM (MD-11-DZRJA) TEST 20, ARE ACCEPTABLE. (PACKS
WRITTEN BY TESTS 16, 17 OR 21 OF 'DZRJA' CANNOT BE USED
AND MUST BE REWRITTEN.)
 2. THE 'W' COMMAND SHOULD NOT BE USED UNLESS 'MAXDL'
PARAMETER IS APPROXIMATELY 1 TRACK IN SIZE - 5000 (10).
IF THE BUFFER SIZE IS MUCH LESS THAN 1 TRACK, THE

TIME REQUIRED TO WRITE A DATA PACK IS TOO GREAT TO BE PRACTICAL.

3. THE 'W' COMMAND PERFORMS A SEQUENTIAL WRITE OF THE PACK USING A 'WRITE DATA' COMMAND. THE DATA PATTERN USED FOR EACH WRITE IS SELECTED RANDOMLY. HOWEVER, THE OPERATION OF THE COMMAND IS SEQUENTIAL, BEGINNING AT 'MINCYL', 'MINTRK' AND CONTINUING TO 'MAXCYL', 'MAXTRK'.
4. THE 'W' COMMAND DOES NOT WRITE HEADERS AND ASSUMES THAT THE FORMAT OF THE PACK IS GOOD.
5. THE 'W' COMMAND CANNOT BE STARTED IF SWITCH 0 (READ ONLY MODE) IS SET. IF SWITCH 0 SET DURING THE OPERATION OF THE 'W' COMMAND, THE DRIVE PERFORMING THE 'W' COMMAND WILL IGNORE THE SWITCH.
6. THE DATA WRITTEN IS VERIFIED BY A 'WRITE CHECK DATA' COMMAND.

4.4.5 'R' COMMAND

USED TO PERFORM A SEQUENTIAL READ OF THE PACK.

FORMAT: RN<CR>

N = DRIVE NUMBER. MAY BE 0 TO 7 OR 'A'. ENTRY MUST BE TERMINATED BY A CARRIAGE RETURN <CR>.

EXAMPLE: RA<CR> - READ THE PACKS ON ALL OF THE ONLINE DRIVES.
RO<CR> - READ THE PACK ON DRIVE 0.

- NOTES:
1. THE PROGRAM WILL PERFORM A NORMAL CHECK OF ALL DATA READ. HOWEVER, ALL OPERATIONS WILL BE SEQUENTIAL.
 2. THE PROGRAM WILL READ THE PACK STARTING AT THE ADDRESS SPECIFIED BY 'MINCYL', 'MINTRK' TO THE ADDRESS SPECIFIED BY 'MAXCYL', 'MAXTRK'. THE READ WILL BE SEQUENTIAL.

4.4.6 GENERAL COMMAND INFORMATION

- A. WHEN A COMMAND IS ENTERED, THE PROGRAM WILL TYPE OUT THE TIME
- B. IF THE COMMAND ENTERED IS NOT VALID, THE PROGRAM WILL TYPE 'INVALID COMMAND'.
- C. DRIVES ASSIGNED (WITH THE 'T' COMMAND) OR DEASSIGNED (WITH THE 'D' COMMAND) MAY BE ENTERED IN ANY SEQUENCE.
- D. THE ERROR RESPONSES FROM THE PROGRAM ARE AS FOLLOWS

<u>RESPONSE</u>	<u>COMMAND(S)</u>
?UNIT N OFFLINE	T, W, R
?UNIT N NOT ASSIGNED	D, S
?UNIT N ALREADY ASSIGNED	T, W, R
?UNIT N NOT PRESENT	T, W, RR
?UNIT N UNSAFE	T, W, R
?UNIT N NOT AN RP04/5/6	T, W, R

5. PERFORMANCE SUMMARY TYPEOUT

5.1 THE PROGRAM WILL DISPLAY A PERFORMANCE SUMMARY FOR THE DRIVES BEING EXERCISED. THIS SUMMARY WILL BE DISPLAYED AUTOMATICALLY IF THE PARAMETER 'INTRVL' IS NOT ZERO OR CAN BE DISPLAYED ON REQUEST BY THE OPERATOR THROUGH THE USE OF THE 'S' COMMAND. THE PERFORMANCE SUMMARY TYPEOUT CONTAINS THE FOLLOWING FIELDS:

'DRV'	THE DRIVE NUMBER
'PASS'	THE PRESENT PASS COUNT FOR THE DRIVE
'ORDERS'	THE NUMBER OF ORDERS PERFORMED BY THE DRIVE
'SEEKS'	THE NUMBER OF SEEK OPERATIONS THE DRIVE PERFORMED
'WRDS XFER'	THE TOTAL NUMBER OF WORDS WRITTEN AND READ BY THE DRIVE
'WRDS READ'	THE TOTAL NUMBER OF WORDS READ BY THE DRIVE
'SOFT'	THE NUMBER OF SOFT DATA ERRORS
'HARD'	THE NUMBER OF HARD DATA ERRORS
'SKI'	THE NUMBER OF 'SKI' OR 'OCYL' ERRORS
'MISP'	THE NUMBER OF POSITIONING ERRORS
'OTHER'	THE TOTAL ERRORS OF OTHER TYPES

NOTE: ERRORS EM1, EM2, EM3, EM4, EM5, & EM10 ARE NOT INCLUDED IN THE 'OTHER' ERROR TOTAL.

5.2 SOFT/HARD ERROR DEFINITIONS

5.2.1 HARD ERRORS

A. A 'DTE' (DRIVE TIMING ERROR) OR A 'DCK' (DATA CHECK ERROR) WHICH OCCURS DURING A READ DATA OR A READ HEADER & DATA OPERATION AND IS NOT CORRECTED OR BECOMES CORRECTABLE AFTER THE PROGRAM HAS PERFORMED THE COMPLETE RETRY SEQUENCE ON THE BAD SECTOR. THE RETRY SEQUENCE IS 16 RE-READS AT TRACK CENTER AND 2 ATTEMPTS AT EACH OF THE FOLLOWING OFFSETS:

<u>RP04/5</u>	<u>RP06</u>
+400 MICRO-INCHES	+200 MICRO-INCHES
-400 MICRO-INCHES	-200 MICRO-INCHES
+800 MICRO-INCHES	+400 MICRO-INCHES
-800 MICRO-INCHES	-400 MICRO-INCHES

+1200 MICRO-INCHES
-1200 MICRO-INCHES+800 MICRO-INCHES
-800 MICRO-INCHES

5.2.2 SOFT ERRORS

- A. ECC CORRECTABLE 'DCK' ERRORS.
- B. 'DCK' & 'ECH' ERRORS WHICH BECOME ECC CORRECTABLE DURING RETRY OR WHICH ARE READ CORRECTLY DURING RETRY.
- C. HEADER READ ERRORS - READ DATA, READ HEADER & DATA, OR WRITE DATA ORDERS.
- D. 'DTE' ERRORS WHICH ARE CORRECTED OR WHICH BECOME ECC CORRECTABLE 'DCK' ERROR DURING THE RETRY SEQUENCE.

6. DATA CHECKING & ERROR RECOVERY

6.1 DATA COMPARISON

DATA COMPARISON OCCURS AFTER EACH 'RDDAT' (READ DATA) OR 'RDHD' (READ HEADER AND DATA) OPERATION UNDER THE FOLLOWING CONDITIONS:

- A. THE ORDER TERMINATED WITH NO ERROR.
- B. THE OPERATION TERMINATED WITH 'DCK' SET AND THE ERROR IS ECC CORRECTABLE OR THE SECTOR IN ERROR IS READ CORRECTLY AFTER RETRY ATTEMPTS.

6.2 VERIFICATION OF DATA WRITTEN

AFTER EACH WRITE OPERATION, THE DATA WRITTEN IS CHECKED WITH THE APPROPRIATE WRITE CHECK COMMAND - IF PARAMETER 'AUTOCK' IS 1. (IF 'AUTOCK' IS 0, WRITE CHECKS WILL BE PERFORMED RANDOMLY.)

6.3 SECTOR REFORMATTING

THE PROGRAM WILL REFORMAT AN UNCORRECTABLE ERROR SECTOR IN THE FOLLOWING CASES (PARAMETER 'FORMAT' MUST BE SET AND SW<00> = 0). THIS PREVENTS THE SAME ERROR FROM BEING CONTINUOUSLY REPORTED.

- A. DATA CHECK ERRORS - EM21
- B. HEADER READ ERRORS - EM20, EM24, EM25, EM26, EM27
- C. DRIVE TIMING ERRORS - EM31
- D. OPERATION INCOMPLETE ERRORS - EM32
- E. WRITE CHECK ERRORS - EM22, EM23

6.4 BAD TRACK/SECTOR FLAGGING

SINCE THE RPO4 SUBSYSTEM DOES NOT HAVE AN AUTOMATIC BAD TRACK HANDLING CAPABILITY, THE MULTIDRIVE EXERCISER ALLOWS THE OPERATOR TO IDENTIFY UP TO 8 BAD TRACK/SECTOR LOCATIONS WHEN THE DRIVE IS ASSIGNED FOR TEST. (SEE SECTION 4.1 FOR ADDITIONAL INFORMATION.)

IF ONE OF THE FOLLOWING ERRORS OCCURS AT A LOCATION IDENTIFIED BY THE OPERATOR, THE PROGRAM WILL INHIBIT THE ERROR REPORT FOR THAT ERROR.

DATA CHECK ERRORS ('DCK')
WRITE CHECK ERRORS ('WCE')
OPERATION INCOMPLETE ERRORS ('OPI')
DRIVE TIMING ERRORS ('DTE')
HEADER READ ERRORS ('FER' & 'HCRC', 'HCE' & 'HCRC', 'HCRC')

OPTIONALLY, THE OPERATOR MAY REQUEST AN ERROR REPORT FOR FLAGGED AREAS. (PARAMETER 'NOTPRT' MUST BE SET TO 0 AT STARTUP.) THE PROGRAM WILL IDENTIFY ERROR MESSAGES ASSOCIATED WITH THE PACK; THESE ERRORS WILL NOT BE ADDED TO THE ERROR TOTALS MAINTAINED BY THE PROGRAM.

7. ERROR MESSAGES

DRIVE ERRORS ARE REPORTED ON THE TELETYPE OR (IF AVAILABLE) A LINE PRINTER. ALL ERROR CONDITIONS ARE REPORTED IN ERROR MESSAGES; THE PROGRAM CONTAINS NO CODED ERROR HALTS. IF THE PROGRAM HALTS (ASSUMING, OF COURSE, THAT SW<15> IS NOT SET), AN UNRECOVERABLE PROGRAM CONDITION HAS OCCURRED OR A CENTRAL PROCESSOR FAILURE HAS OCCURRED.

ERROR MESSAGES ARE MADE UP OF SEVERAL LINES. EACH TYPE OF ERROR HAS SEVERAL OPTIONAL LINES WHICH MAY APPEAR WITH IT. ALL OF THE POSSIBLE ERROR MESSAGE LINES WHICH MAY APPEAR ARE GIVEN IN THE SECTION DESCRIBING THE PARTICULAR ERROR HEADER.

7.1 ERROR DESCRIPTION LINES

MESSAGES EM1, EM2, EM3, EM4, EM5, EM10, EM11, & EM12 ARE ALWAYS DISPLAYED ON THE TTY. THE OTHER MESSAGES ARE DISPLAYED ON EITHER THE LINE PRINTER (IF AVAILABLE) OR THE TTY.

(THE MESSAGE TAGS ARE GIVEN FOR REFERENCE.)

MESSAGE
TAG

TEXT

EM1	RH11 INTERRUPT OCCURRED (RPAS=0)
	THE RH11 INTERRUPTED AND THE ATTENTION SUMMARY REGISTER (RPAS) WAS CLEARED.
EM2	UNEXPECTED ATTENTION OCCURRED
	THE INDICATED DRIVE INTERRUPTED BUT THE DRIVE WAS NOT

PERFORMING AN OPERATION.

- EM3 MASSBUS PARITY ERROR (MCPE=1)
THE RH11 DETECTED A CONTROL BUS PARITY ERROR WHEN READING THE INDICATED REGISTER FROM THE INDICATED DRIVE.
- EM4 MASSBUS PARITY ERROR (PAR=1)
THE INDICATED RPO4 DETECTED A CONTROL BUS PARITY ERROR WHEN THE RH11 LOADED THE SPECIFIED REGISTER.
- EM5 ADDRESS PLUG CHANGE BIT SET
THE 'OPE' BIT WAS SET WHEN THE INDICATED DRIVE INTERRUPTED.
- EM6 RH11 DIDN'T RESPOND TO ADDRESSING
WHEN THE PROGRAM ADDRESSED THE RH11, NO RESPONSE WAS RECEIVED FROM THE INDICATED ADDRESS.
- EM10 UNCORRECTABLE MASSBUS PARITY ERROR
THE PROGRAM HAS TRIED 3 TIMES TO READ OR WRITE THE INDICATED REGISTER.
- EM11 FATAL MASSBUS PARITY ERROR
A CONTROL BUS PARITY ERROR OCCURRED WHEN THE RH11 ATTEMPTED TO PROCESS A PREVIOUS, DIFFERENT PARITY ERROR.
- EM12 PERSISTENT DEVICE UNSAFE
THE DRIVE BECAME UNSAFE; DRIVE CLEAR TO THE DRIVE DID NOT CLEAR THE UNSAFE CONDITION. THE PROGRAM WILL AUTOMATICALLY DEASSIGN THE DRIVE. THE DRIVE CANNOT BE EXERCISED UNTIL THE UNSAFE CONDITION HAS BEEN CLEARED.
- EM13 OPERATION NOT COMPLETED WITHIN TIME LIMIT
THE DRIVE DID NOT COMPLETE THE OPERATION WITHIN 1 SECOND AFTER THE OPERATION WAS INITIATED.
- EM14 UNIT WENT OFFLINE
THE DRIVE WENT OFFLINE DURING THE INDICATED OPERATION. (THE 'MOL' BIT BECAME ZERO.) THE PROGRAM WILL AUTOMATICALLY DEASSIGN THE DRIVE. THE OPERATOR MUST REASSIGN THE DRIVE WITH THE 'T' COMMAND TO RE-INITIATE TESTING.
- EM15 NO RESPONSE TO PORT REQUEST
THE PROGRAM IS TESTING A DUAL PORT DRIVE WHICH HAS NOT SWITCHED

TO THE REQUESTING PORT WITHIN 10 SECONDS AFTER PORT REQUEST
TO THE DRIVE FROM THE REPORTING PORT.

EM20 HEADER CRC ERROR

A HEADER CRC ERROR WAS DETECTED AT THE INDICATED DISK ADDRESS.
THE CONTENTS OF THE HEADER ARE DISPLAYED. THE OPERATION WILL
BE RETRIED 3 TIMES.

EM21 DATA CHECK ('DCK') ERROR

A DATA CHECK ERROR WAS DETECTED AT THE INDICATED SECTOR.
THE FULL RETRY SEQUENCE (INCLUDING OFFSET) WILL BE INITIATED
FOR THE SECTOR IN ERROR IF THE ECC HARD ERROR ('ECH) BIT
IS SET.

EM22 WRITE CHECK ERROR - DATA CHECK ('DCK') SET

A WRITE CHECK ERROR OCCURRED AND THE DATA CHECK ('DCK') BIT
WAS SET. IF 'ECH' IS NOT SET, THE OPERATION WILL BE RETRIED
UP TO 3 TIMES; IF THE 'ECH' BIT IS SET, THE OPERATION WILL
BE RETRIED UP TO 16 TIMES.

EM23 WRITE CHECK ERROR - DATA CHECK ('DCK') NOT SET

A WRITE CHECK ERROR OCCURRED AND 'DCK' WAS NOT SET. THE
WORDS WHICH CAUSED THE ERROR ARE DISPLAYED IN THE ERROR
MESSAGE. THE OPERATION WILL BE RETRIED 3 TIMES.

EM24 HEADER READ ERROR - 'FMT' BIT DROPPED

A WRITE DATA, WRITE CHECK DATA, OR A READ DATA WAS BEING
PERFORMED AND A 'FMT' ERROR OCCURRED. THE PROGRAM RE-READ THE
HEADER OF THE ERROR SECTOR AND THE 'HCRC' BIT WAS SET. THE
CONTENTS OF THE HEADER ARE DISPLAYED. THE OPERATION WILL
BE RETRIED 3 TIMES.

EM25 HEADER READ ERROR - HEADER COMPARE ('HCE') ERROR

SIMILAR TO EM24, EXCEPT THAT THE 'HCE' ERROR BIT WAS
SET INITIALLY. THE OPERATION WILL BE RETRIED 3 TIMES.

EM26 FORMAT ERROR ('FER')

FORMAT ERROR OCCURRED. WHEN THE HEADER WAS RE-READ, THE
'HCRC' BIT WAS NOT SET. THE CONTENTS OF THE HEADER ARE
DISPLAYED. THE OPERATION WILL BE RETRIED 3 TIMES.

EM27 HEADER COMPARE ('HCE') ERROR

SIMILAR TO EM26 EXCEPT THAT THE 'HCE' BIT WAS SET INITIALLY.
THE OPERATION WILL BE RETRIED 3 TIMES.

EM30 MISCELLANEOUS DRIVE ERROR

THIS MESSAGE IS GIVEN FOR THE FOLLOWING ERROR BITS:
'IXE', 'AOE', 'RMR', 'ILF', OR 'ILR'

- EM31 OPERATION INCOMPLETE ('OPI') ERROR
AN OPERATION INCOMPLETE ERROR OCCURRED AT THE INDICATED SECTOR.
- EM32 DRIVE TIMING ('DTE') ERROR
DRIVE TIMING ERROR OCCURRED ON THE INDICATED SECTOR. THE OPERATION WILL BE RETRIED 3 TIMES.
- EM33 PARITY ('PAR') ERROR AFTER OPERATION STARTED
THE 'PAR' BIT WAS SET WHEN THE OPERATION WAS COMPLETED. THE OPERATION WILL BE RETRIED 3 TIMES.
- EM34 WRITE CLOCK FAILURE ('WCF')
A WRITE CLOCK FAILURE OCCURRED DURING THE OPERATION. THE OPERATION WILL BE RETRIED 3 TIMES.
- EM35 INVALID ADDRESS ('IAE') ERROR
AN INVALID ADDRESS ERROR OCCURRED DURING THE OPERATION.
- EM36 WRITE LOCK ('WLE') ERROR
A WRITE OPERATION WAS ATTEMPTED BUT THE DRIVE WAS WRITE LOCKED.
- EM40 RH11 OR UNIBUS TRANSFER ERROR
'TRE' IS SET IN THE RH11 CONTROL REGISTER AND NO DRIVE ERROR HAS OCCURRED. THE OPERATION WILL BE RETRIED 3 TIMES IF THE ERROR WAS CAUSED BY 'DLT', 'LPE', 'MXF', OR 'MDPE'.
- EM41 BUS ADDRESS OR WORD COUNT INCORRECT
NO DRIVE ERROR OCCURRED BUT EITHER THE BUS ADDRESS INDICATES THAT AN INCORRECT NUMBER OF WORDS WERE TRANSFERED OR THE WORD COUNT REGISTER IS NOT ZERO.
- EM42 DATA COMPARE ERRORS - NO DRIVE ERROR DETECTED
NO SUBSYSTEM ERROR WAS SIGNALLED; HOWEVER, THE DATA DOES NOT COMPARE.
- EM43 CAN'T MATCH DATA READ WITH A PATTERN
THE DATA IN THE BUFFER DOES NOT MATCH ANY OF THE STANDARD

- PATTERNS.
- EM44 ERROR BIT(S) SET, BUT NO ERROR SIGNALLED BY THE RH11
THE OPERATION COMPLETED NORMALLY; HOWEVER, THE PROGRAM FOUND EITHER ERROR BITS IN THE RPO4 SET OR ERROR BITS IN THE RH11 SET.
- EM45 ECC LOGIC FAILURE
THE CONTENTS OF EITHER THE ECC POSITION REGISTER (RPEC1) OR THE CONTENTS OF ECC PATTERN REGISTER (RPEC2) ARE NOT VALID. THE POSITION REGISTER IS EITHER '0' OR > 10041 (8) OR THE PATTERN REGISTER CONTAINS ZEROS.
- EM46 BUS ADDRESS OR WORD COUNT NOT CONSISTENT
THE PROGRAM WAS PROCESSING AN ERROR AND FOUND THAT THE NUMBER OF WORDS TRANSFERED AS INDICATED BY THE BUS ADDRESS REGISTER DOES NOT AGREE WITH THE TRANSFER COUNT FROM THE WORD COUNT REGISTER.
- EM50 SEEK INCOMPLETE OR OFF CYLINDER ERROR
THE DRIVE SIGNALLED EITHER 'SKI' OR 'OCYL' ERROR BITS.
- EM51 PROGRAM DETECTED POSITIONING ERROR
A HEADER COMPARE ERROR OCCURRED ('HCE'); HOWEVER, WHEN THE PROGRAM EXAMINED THE HEADER OF THE SECTOR IN ERROR, IT FOUND THAT THE CYLINDER FIELD DID NOT AGREE WITH THE CONTENTS OF 'RPCC' OF THE DRIVE. THE DRIVE WILL BE RECALIBRATED.
- EM60 DEVICE UNSAFE
THE INDICATED DRIVE UNSAFE ERROR OCCURRED; THE ERROR WAS CLEARED BY A 'DRIVE CLEAR' INSTRUCTION.

7.2 DETAIL ERROR LINES

THE LINE NUMBERS GIVEN BELOW ARE FOR REFERENCE ONLY.

LINE 1

TT:TT:TT (DESCRIPTION OF ERROR)

'TT:TT:TT' IS THE TIME SINCE THE PROGRAM WAS STARTED. TT:TT:TT IS GIVEN IN HOURS: MINUTES: SECONDS.

LINE 2

'PRESENT ORDER = XXXX PREVIOUS ORDER = YYYY'

MNEMONICS USED FOR THE ORDERS ARE DEFINED BELOW:

UNLOAD - UNLOAD (OCTAL 3)
 SEEK - SEEK (OCTAL 5)
 RECAL - RECALIBRATE (OCTAL 7)
 DRVCLR - DRIVE CLEAR (OCTAL 11)
 RELSE - RELEASE (OCTAL 13)
 OFFSET - OFFSET (OCTAL 15)
 RTC - RETURN TO CENTERLINE (OCTAL 17)
 READIN - READIN PRESET (OCTAL 21)
 PACK - PACK ACKNOWLEDGE (OCTAL 23)
 SEARCH - SEARCH (OCTAL 31)
 WCKD - WRITE CHECK DATA (OCTAL 51)
 WCKHD - WRITE CHECK HEADER & DATA (OCTAL 53)
 WRDAT - WRITE DATA (OCTAL 61)
 WRTHD - WRITE CHECK HEADER & DATA (OCTAL 63)
 RDDAT - READ DATA (OCTAL 71)
 RDHD - READ HEADER & DATA (OCTAL 73)

(DISPLAY OF THE RP04/5/6 REGISTERS IN TWO GROUPS:
 RPCS1, RPCS2, RPDS1, RPER1, RPER2, RPER3, RPEC1, & RPEC2 FORM THE FIRST
 GROUP; ALL THE OTHER REGISTERS ARE IN THE SECOND GROUP.
 IF SW<05> IS SET, ONLY THE REGISTERS IN THE FIRST GROUP WILL BE
 DISPLAYED.)

THE ABOVE LINE WILL BE TYPED IF THE ERROR OCCURRED DURING
 THE NON-DATA TRANSFER PART OF THE OPERATION.

'* ERROR AT BAD TRACK/SECTOR'

THE ABOVE LINE WILL BE PRINTED IF A DATA ERROR OCCURES AT AN ADDRESS
 ON THE PACK WHICH THE OPERATOR HAS IDENTIFIED AS BEING BAD. PARAMETER
 'NOTPRT' MUST BE 0 FOR THE ERROR TO BE REPORTED.

A WORD CALLED 'STATUS' IS DISPLAYED WITH THE RP04/5/6 REGISTERS. THE
 CONTENTS OF THIS WORD IDENTIFY HOW THE ERROR WAS PROCESSED BY THE
 RP04/5/6 DRIVE HANDLER ROUTINE. THE BITS IN THIS WORD ARE ENCODED
 AS FOLLOWS:

BIT #	MEANING IF BIT IS '1'
----	-----
15	ERROR OCCURRED DONE (BIT07=0), BITS 14-9, 2, 1 SPECIFY TYPE DONE (BIT07=1), BITS 6-3 SPECIFY TYPE
14	DRIVE IS OFFLINE
12	PERSISTENT UNSAFE CONDITION EXISTS

- 11 UNCORRECTABLE PARITY ERROR OCCURRED
- 10 FATAL PARITY ERROR OCCURRED. MASSBUS CLEAR WAS PERFORMED
- 9 OPERATION NOT COMPLETED WITHIN 1 SECOND MASSBUS CLEAR PERFORMED. ALL OTHER OUTSTANDING OPERATIONS WERE RESTARTED.
- 7 DONE - OPERATION COMPLETED
- 6 DATA ERROR OCCURRED DURING THE TRANSFER
- 5 ERROR OCCURRED WHILE SEARCHING FOR THE 'TRANSFER' SECTOR OR DURING RECALIBRATE OR OFFSET COMMANDS
- 4 CORRECTABLE UNSAFE CONDITION OCCURRED
- 3 DRIVE ERROR OCCURRED THAT CAUSED AN AUTOMATIC RECALIBRATE SEQUENCE
- 2 PORT REQUEST TIMEOUT
- 1 NON-EXISTENT DRIVE REQUESTED

LINE 3

ERROR AT CXXX TYY SZZ PREV ADDR = CUUU TVV SWW

THE ACTUAL ADDRESS OF THE ERROR SECTOR AND THE PREVIOUS DISK ADDRESS ARE GIVEN IN THIS LINE. CYLINDER, TRACK, & SECTOR ADDRESSES ARE IN DECIMAL.

LINE 4

PRESENT ADDR = CXXX TYY SZZ PREV ADDR = CUUU TVV SWW

THIS LINE IDENTIFIES THE ADDRESS WHEN THE ERROR WAS DETECTED; THE PREVIOUS ADDRESS IS ALSO GIVEN. CYLINDER, TRACK, & SECTOR ADDRESSES ARE GIVEN IN DECIMAL.

LINE 5

START CYL = XXX END CYL = YYY

THIS LINE IDENTIFIES THE STARTING CYLINDER OR A SEEK (IMPLIED) AND THE DESTINATION CYLINDER. CYLINDER ADDRESSES ARE IN DECIMAL.

LINE 6

START CYL = XXX END CYL = YYY ACTUAL CYL = ZZZ

THIS LINE IDENTIFIES THE STARTING CYLINDER OF AN IMPLIED SEEK, THE DESTINATION CYLINDER, AND THE CYLINDER THE DISK ACTUALLY STOPPED AT. CYLINDER ADDRESSES ARE IN DECIMAL.

LINE 7

RPBA = XXXX RPWC = YYYY

THIS LINE GIVES THE CONTENTS OF THE RH11 BUFFER ADDRESS REGISTER AND THE RH11 WORD COUNT REGISTER. THIS LINE IS NOT PRINTED IF SW<05> IS NOT SET.

LINE 8

START CYL = XXX START TRK = YY START SECTOR = ZZ

THIS LINE IDENTIFIES THE STARTING DISK ADDRESS OF THE PRESENT OPERATION. CYLINDER, TRACK, AND SECTOR VALUES ARE DECIMAL.

LINE 9

RPDA = XXXX RPCA = YYYY

THIS LINE GIVES THE CONTENTS OF THE RPO4 TRACK AND SECTOR ADDRESS REGISTER AND THE CONTENTS OF THE DESIRED CYLINDER ADDRESS REGISTER. THIS LINE IS NOT PRINTED IF SW<05> IS NOT SET.

LINE 10

BUFFER ADDR = XXXX SIZE = YYYY ACTUAL NUMBR WRDS XFRD = ZZZZ

THIS LINE GIVES THE STARTING ADDRESS OF THE BUFFER USED FOR THE CURRENT DATA TRANSFER OPERATION, ITS SIZE, AND THE ACTUAL NUMBER OF WORD TRANSFERED. THE STARTING ADDRESS OF THE BUFFER IS IN OCTAL, THE SIZE AND WORD TRANSFERED VALUE ARE IN DECIMAL.

LINE 11

GOOD DATA = XXXX BAD DATA = YYYY SECT POS = ZZZ

THIS LINE GIVES THE GOOD DATA, THE ACTUAL DATA FROM THE DISK, AND THE LOCATION IN THE SECTOR OF THE ACTUAL DATA. THE SECTOR POSITION IS IN DECIMAL.

LINE 12

HEADER CONTENTS OF ERROR SECTOR = XXXX XXXX XXXX XXXX

THIS LINE GIVES THE CONTENTS OF THE HEADER OF THE SECTOR WHICH
GAVE THE ERROR.LINE 13

RPEC1 = XXXX RPEC2 = YYYY

THIS LINE WILL BE PRINTED AFTER A SUCCESSFUL RETRY OF A SECTOR
WHICH BECAME ECC CORRECTABLE DURING RETRY.LINE 14

ECC CORRECTABLE WITHOUT OFFSET

THE SECTOR IN ERROR IS ECC CORRECTABLE; NO RETRY ATTEMPTS ARE
NECESSARY.LINE 15

READ CORRECTLY AT OFFSET X MICRO-INCHES

THE SECTOR IN ERROR WAS READ WITHOUT ERROR AT THE INDICATED
OFFSET VALUE.LINE 16

ECC CORRECTABLE AT OFFSET X MICRO-INCHES

THE SECTOR IN ERROR BECAME ECC CORRECTABLE AT THE INDICATED
OFFSET.LINE 17

CORRECTED ON X RETRY

THE OPERATION WAS PERFORMED ERROR FREE ON THE INDICATED RETRY
ATTEMPT.LINE 18

UNCORRECTABLE AFTER X RETRIES

THE OPERATION CANNOT BE PERFORMED CORRECTLY AFTER THE
INDICATED NUMBER OF RETRY ATTEMPTS.

LINE 19

DIFFERENT ERROR DURING RETRY

WHILE THE PROGRAM WAS RETRYING THE ERROR, A DIFFERENT OCCURRED.
IF THIS LINE IS PRINTED, THE RH11/RP04 REGISTERS WILL ALSO BE
PRINTED (SEE LINE 2).

LINE 20

DATA COMPARISON ERRORS

A PRINTOUT OF THE DATA COMPARISON ERRORS FOLLOW THIS LINE.

LINE 21

TOTAL COMPARE ERRORS = XXXX

THIS LINE GIVES THE TOTAL DATA COMPARISON ERROR COUNT. THE
VALUE GIVEN IS IN DECIMAL.

LINE 22

THE DATA COMPARED OK

THIS LINE INDICATES THE RESULTS OF THE DATA COMPARISON FOLLOWING
ECC CORRECTION.

LINE 23

ECC CORRECTION RESULTS

THE PROGRAM PERFORMED ECC CORRECTION AND THE RESULTS ARE REPORTED.
THE ADDRESS IN MEMORY OF THE WORD(S) IN ERROR ARE GIVEN, THE WORD(S)
BEFORE CORRECTION AND THE WORD(S) AFTER CORRECTION ARE PRINTED.

LINE 24

ERROR BURST BEGINS AT WORD XXX IN DATA FIELD OF ERROR SECTOR

THIS IS AN INFORMATIONAL LINE WHICH WILL BE PRINTED FOR 'DCK' ERRORS
WHICH ARE ECC CORRECTABLE OR WHICH BECOME ECC CORRECTABLE DURING
RETRY. 'XXX' IS THE WORD OFFSET VALUE FROM 'RPEC1' AND IS IN
DECIMAL.

LINE 25

ERROR WAS NOT IN THE DATA READ -
ECC CORRECTION CAN'T BE PERFORMED

THE DATA ERROR WAS NOT IN DATA TRANSFERED TO MEMORY.

LINE 26

CONTENTS OF THE ERROR SECTOR (REPORTED ABOVE)

IF SW<03> IS SET, THE SECTOR WHICH GAVE THE 'DCK', 'DTE' OR,
'WCF' ERROR OR 'HARD' DATA CHECK ERROR IS PRINTED. THE
CONTENTS OF THE SECTOR FOLLOW THIS LINE.

LINE 27

ORDERS: WWW ERRORS: X WRDS XFR: YYYY WRDS READ: ZZZZ

THIS IS THE LAST LINE PRINTED FOR ALL NON-POSITIONING
TYPE ERRORS.

'ORDERS' IS THE TOTAL NUMBER OF COMMANDS GIVEN TO THE DRIVE
WHICH REPORTED THE ERROR.

'ERRORS' IS THE TOTAL ERROR COUNT FOR THE DRIVE AND INCLUDES
EVERY ERROR DETECTED, REGARDLESS OF TYPE.

'WRDS XFR' IS THE TOTAL NUMBER OF WORDS WRITTEN AND READ BY
THE DRIVE.

'WRDS READ' IS THE TOTAL NUMBER OF WORD READ BY THE DRIVE.

LINE 28

ORDERS: WWWW TOTAL SEEKS: XXX TOTAL POS ERR = YYY TOTAL SKI, OCYL ERR = Z

THIS IS THE LAST LINE PRINTED FOR ALL POSITIONING TYPE ERRORS.

'ORDERS' IS THE TOTAL NUMBER OF ORDERS GIVEN TO THE DRIVE WHICH
REPORTED THE ERROR.

'TOTAL SEEKS' IS THE TOTAL NUMBER OF SEEK OPERATIONS PERFORMED
BY THE DRIVE.

'TOTAL POS ERR' IS THE TOTAL NUMBER OF POSITIONING ERRORS WHICH
THE PROGRAM DETECTED FOR THE DRIVE.

'TOTAL SKI, OCYL ERR' IS THE TOTAL NUMBER OF 'SKI' OR 'OCYL' ERRORS
SIGNALLED BY THE DRIVE.

1
2
3

8. PROGRAM DESCRIPTION

8.1 PROGRAM OPERATION

WHEN THE PROGRAM IS STARTED, ALL TABLES AND PARAMETERS ARE CLEARED OR INITIALIZED. THE PARAMETERS WHICH ARE UNDER OPERATOR TTY ENTRY CONTROL ARE CHECKED FOR VALIDITY AND CONSISTENCY. RH11 INTERRUPT ENABLE ('IE') IS SET, TTY KEYBOARD INTERRUPT ENABLE IS SET, AND THE KW11-L OR KW11-P CLOCK IS STARTED. WHEN THESE ACTIONS HAVE BEEN COMPLETED, THE PROGRAM TYPES OUT 'PROGRAM INITIALIZE COMPLETE'. COMMAND ENTRIES WILL NOW BE ACCEPTED BY THE PROGRAM

THE PROGRAM SCANS ITS INTERNAL ASSIGNMENT TABLES, LOOKING FOR:

- 1) DRIVES TO ASSIGN/DEASSIGN
- 2) PERFORMANCE SUMMARY TYPEOUT REQUESTS
- 3) DRIVES REQUIRING COMMAND INITIATION, BUFFER ASSIGNMENT, OR PARAMETER SELECTION.
- 4) DRIVES COMPLETING CURRENT OPERATIONS.

THE PROGRAM CONTINUES SCANNING ITS TABLES UNTIL AN ENTRY IS FOUND. IN THE CASE OF THE PROGRAM AT INITIAL START, THE FIRST ENTRY WILL BE MADE BY THE OPERATOR WHEN A DRIVE IS ASSIGNED ('T' COMMAND).

WHEN A DRIVE IS ASSIGNED, THE KEYBOARD ENTRY ROUTINE VERIFIES THAT THE DRIVE IS PRESENT, IS AN RPO4/5/6, AND IS ONLINE. THE ASSIGNMENT ROUTINE THEN ISSUES A 'READIN PRESET' INSTRUCTION, SETS 'FMT22', AND ISSUES A 'RECALIBRATE' INSTRUCTION.

PARAMETERS FOR THE OPERATION ARE SELECTED AND A BUFFER IS ASSIGNED. IF THE OPERATION IS A WRITE OR WRITE CHECK ORDER, THE ASSIGNED BUFFER WILL BE FILLED WITH THE SELECTED PATTERN. (WRITE CHECK ORDERS ARE ISSUED AFTER EACH WRITE ORDER. THE WRITE CHECK ORDER USES THE PARAMETERS SELECTED FOR THE PRECEDING WRITE ORDER.) CONTROL IS THEN PASSED TO THE COMMAND INITIATION ROUTINE.

THE COMMAND INITIATION ROUTINE FIRST LOOKS AT THE CYLINDER ADDRESS OF THE REQUESTED OPERATION. IF THE DRIVE MUST SEEK TO ANOTHER CYLINDER TO PERFORM THE OPERATION, THE PROGRAM ISSUES A SEARCH INSTRUCTION TO THE DRIVE WITH A 'TARGET' SECTOR WHICH IS 8 SECTORS EARLIER THAN THE 'TRANSFER' SECTOR. (THIS ALLOWS THE PROGRAM TO INITIATE OPERATIONS ON ANOTHER DRIVE WHILE THE PRESENT DRIVE, OR OTHER DRIVES, ARE SEARCHING FOR 'TARGET' SECTORS. ALL SEEKS ISSUED BY THE PROGRAM ARE IMPLIED SEEK SEARCH OPERATIONS.) WHEN A SEARCHING DRIVE FINDS THE 'TARGET' SECTOR AND INTERRUPTS, THE PROGRAM READS THE LOOK AHEAD REGISTER (RPLA) OF THE INTERRUPTING DRIVE AND COMPARES THE POSITION OF THE DISK WITH THAT OF THE DESIRED SECTOR.

IF OTHER DRIVES ARE WAITING ON CYLINDER, THEY ARE ALSO CHECKED. THE PROGRAM THEN ISSUES THE REQUESTED ORDER TO THE DRIVE NEAREST ITS TRANSFER SECTOR. THE DRIVES NOT SELECTED WILL HAVE ANOTHER SEARCH INITIATED. IF A DRIVE IS NOT SELECTED FOR TRANSFER AFTER THREE

REVOLUTIONS OF ITS DISK, IT IS GIVEN PRIORITY OVER DRIVES WHICH HAVE NOTE BEEN ON CYLINDER AS LONG.

WHEN THE DATA TRANSFER OPERATION IS COMPLETE, THE DRIVE REGISTERS ARE STORED AND A DATA TRANSFER IS INITIATED FOR A WAITING DRIVE.

IF THE OPERATION HAS BEEN COMPLETED NORMALLY, THE SAVED DRIVE REGISTERS ARE CHECKED TO VERIFY THAT NO ERROR BITS ARE SET; THE RH11 BUS ADDRESS AND WORD COUNT ADDRESS REGISTERS ARE CHECKED TO VERIFY THAT THE CORRECT NUMBER OF WORDS HAVE BEEN TRANSFERED AND THAT THE TWO REGISTERS ARE CONSISTENT WITH EACH OTHER; AND IF THE ORDER WAS A READ ORDER, THE DATA BUFFER IS COMPARED. WHEN THIS SEQUENCE IS COMPLETED, THE DRIVE IS RETURNED TO THE ASSIGNED, INACTIVE LIST. THE PROGRAM THEN INITIATES A DATA TRANSFER ON A WAITING DRIVE AND RESELECTS AND REINITIATES ANOTHER OPERATION ON THE RELEASED DRIVE.

ERRORS WHICH OCCUR ARE PROCESSED IN THE FOLLOWING ORDER. MULTIPLE ERRORS WILL BE REPORTED AS THE FIRST ERROR TYPE CHECKED.

A. ERRORS REPORTED FOR OPERATIONS WHICH HAVE NOT COMPLETED NORMALLY.

PERSISTENT UNSAFE CONDITION - EM12
UNCORRECTABLE MASSBUS PARITY ERROR - EM10
FATAL MASSBUS PARITY ERROR - EM11
OPERATION NOT COMPLETED WITHIN TIME LIMIT - EM13
UNIT WENT OFFLINE - EM14

B. ERRORS REPORTED FOR OPERATIONS WHICH COMPLETE NORMALLY.

CORRECTABLE UNSAFE - EM60
DRIVE TIMING ERROR - EM32
DATA CHECK ERROR - EM21
WRITE CHECK WITH DCK SET - EM22
HEADER CRC ERRORS - EM20
FORMAT ERRORS - EM24, EM26
HEADER COMPARE ERRORS - EM25, EM27
PROGRAM DETECTED POSITIONING ERROR - EM51
SEEK INCOMPLETE OR OFF CYLINDER ERROR - EM50
WRITE CHECK WITHOUT 'DCK' SET - EM23
RH11 OR UNIBUS TRANSFER ERROR - EM40
'OPI' ERROR - EM31
'PAR' ERROR - EM33
'WCF' ERROR - EM34
'IAE' ERROR - EM35
'WLE' ERROR - EM36
MISCELLANEOUS DRIVE ERROR - EM30

C. ERRORS NOT FLAGGED BY THE HARDWARE ERROR DETECTION LOGIC.

BUS ADDRESS OR WORD COUNT INCORRECT - EM41
DATA COMPARE ERRORS - NO DRIVE ERROR DETECTED - EM42
CAN'T MATCH DATA READ WITH A PATTERN - EM43
ERROR BIT(S) SET, BUT NO ERROR SIGNALLED BY THE RH11 - EM44
ECC LOGIC FAILURE - EM45

BUS ADDRESS OR WORD COUNT NOT CONSISTENT - EM46

8.2 DUAL PORT OPERATION

DUAL PORT OPERATION IS NEARLY IDENTICAL TO THE OPERATION DESCRIBED IN SECTION 8.1. THE DIFFERENCES ARE IN COMMAND SEQUENCE INITIATION AND ORDER TERMINATION.

WHEN THE DUAL PORT HANDLER ROUTINE IN THE MULTIDRIVE PROGRAM RECEIVES A REQUEST FOR A DRIVE, THE PROGRAM VERIFIES THAT THE DRIVE IS ONLINE. THE DRIVE IS SELECTED AND 0'S ARE WRITTEN INTO 'RPDS1': IF THE DRIVE IS IN NEUTRAL, THIS WILL SEIZE THE DRIVE. IF THE DRIVE IS SEIZED BY THE OTHER PORT, WRITING INTO 'RPDS1' WILL SET 'PORT REQUEST'. THE PROGRAM CHECKS 'DVA' IN 'RPCS1'. IF THE DRIVE IS AVAILABLE AS INDICATED BY THE 'DVA' BIT, THE COMMAND SEQUENCE WILL BE INITIATED IN THE NORMAL MANNER (SEE SECTION 8.1 ABOVE). IF 'DVA' WAS NOT SET, THE PROGRAM MAKES AN ENTRY FOR THE DRIVE IN AN INTERNAL 'PORT REQUEST PENDING' TABLE AND STARTS A 20 SECOND TIMER FOR THE DRIVE. IF THE DRIVE HAS NOT SWITCHED TO THE REQUESTING SYSTEM WITHIN THE 20 SECOND INTERVAL, THE PROGRAM REPORTS A 'NO RESPONSE TO PORT REQUEST' ERROR. NORMALLY THIS ERROR MESSAGE INDICATES A FAILURE IN THE DUAL PORT CONTROL LOGIC IN THE DRIVE BEING TESTED; HOWEVER, UNDER CERTAIN CONDITIONS (E.G. MASSBUS PARITY ERRORS BEING REPORTED ON THE OTHER SYSTEM ON A MOD33 TTY), THE OTHER PROCESSOR WAS UNABLE TO PROCESS THE DRIVE AFTER IT HAD REQUESTED THE DRIVE. THE OPERATOR MUST BE AWARE OF WHAT THE OTHER SYSTEM IS DOING AT ALL TIMES TO INTERPRET THE PORT RELATED ERROR MESSAGES PROPERLY.

AFTER A DRIVE HAS COMPLETED AN OPERATION, THE PROGRAM WILL STORE THE REGISTERS AND ISSUE A 'RELEASE' TO THE DRIVE. IF THE OPERATION TERMINATED WITH AN ERROR, THE DRIVE WILL NOT BE RELEASED UNTIL ERROR PROCESSING HAS BEEN COMPLETED.

SINGLE PORT DRIVES, DRIVES WHICH ARE IN NEUTRAL BUT NOT BEING EXERCISED BY THE OPPOSITE PORT ARE STILL TREATED AS DUAL PORT DRIVES IN THAT A RELEASE COMMAND IS ISSUED AT THE END OF NORMAL ORDER PROCESSING OR AT THE END OF ERROR PROCESSING. A RELEASE COMMAND ISSUED UNDER THESE CONDITIONS HAS NO FUNCTIONAL EFFECT ON THE OPERATION OF THE DRIVE.

8.3 SELECTION OF OPERATION VARIABLES

A. SECTOR ADDRESS SELECTION IS RANDOM BETWEEN THE VALUES IN 'MINSEC' AND 'MAXSEC'. TRACK ADDRESS SELECTION IS RANDOM BETWEEN THE VALUES IN 'MINTRK' AND 'MAXTRK'. CYLINDER ADDRESS SELECTION IS RANDOM BETWEEN 'MINCYL' AND 'MAXCYL'. IF A MINIMUM ADDRESS IS GREATER THAN THE CORRESPONDING MAXIMUM ADDRESS, THE PROGRAM WILL EXCLUDE ALL ADDRESSES BETWEEN 'MAX' AND 'MIN' FROM THE SELECTION. FOR EXAMPLE: IF 'MINTRK' IS 18 AND 'MAXTRK' IS 5, THEN TRACK ADDRESS SELECTION WILL EXCLUDE TRACKS 6 - 17 FROM THE SELECTION AND SELECT AN ADDRESS FROM AMONG ADDRESSES 18, 0, 1, 2, 3, 4, 5.

- B. THE BUFFER SIZE IS RANDOM SELECTED BETWEEN 4 (10) - AND THE VALUE IN 'MAXDL'. THE SIZE SELECTED IS WEIGHTED TO ENSURE THAT AT LEAST 4 WORDS ARE WRITTEN IN THE DATA AREA OF THE LAST SECTOR. THIS IS NECESSARY AS THE PROGRAM REQUIRES 4 LOCATIONS IN THE DATA PORTION OF THE SECTOR TO BE ABLE TO MATCH THE DATA TO A PATTERN FOR DATA COMPARISON PURPOSES.
- C. THE DATA WRITTEN IS RANDOMLY SELECTED AMONG THE 15 STANDARD PATTERNS. THE KEYWORDS IN THE HEADER (WHEN PERFORMING A WRITE HEADER & DATA ORDER) ARE ZERO FILLED. THE PROGRAM EXPECTS TO FIND THAT THE KEYWORDS ARE ZERO.
- D. THE ORDERS ARE SELECTED RANDOMLY. WRITE CHECK DATA AND WRITE CHECK HEADER & DATA ORDERS ARE PERFORMED ONLY IF THE PREVIOUS ORDER WAS THE APPROPRIATE DATA ORDER. IF THE 'FORMAT' PARAMETER IS ZERO, THE PROGRAM WILL NOT SELECT WRITE HEADER & DATA (AND WRITE CHECK HEADER & DATA) ORDERS. WHEN THE PROGRAM SELECTS A WRITE HEADER & DATA ORDER, THE BUFFER SIZE IS FORCED TO 260 (10); THE PROGRAM WILL NOT PERFORM A MULTI-SECTOR FORMAT WRITE OPERATION.
- E. THE FIRST ORDER PERFORMED AFTER A UNIT IS ASSIGNED WITH A 'T', 'W', OR 'R' COMMAND IS NOT RANDOMLY SELECTED. THE PARAMETERS FOR THE FIRST OPERATION ARE THE MINIMUM OR STARTING VALUES OF THE VARIABLES.

8.4 DATA PATTERNS

THE PROGRAM SELECTS ONE OF THE FOLLOWING DATA PATTERNS TO WRITE WHEN A WRITE ORDER IS SELECTED. THE ENTIRE BUFFER IS FILLED WITH THE SELECTED PATTERN. WHEN DATA IS READ FROM THE DISK, THE PROGRAM COMPARES DATA ON A SECTOR BASIS: FROM THE FIRST 4 DATA WORDS OF EACH SECTOR, THE PROGRAM MATCHES THE DATA TO ONE OF THE FOLLOWING PATTERNS. TO MAINTAIN COMPATIBILITY WITH PACKS WRITTEN BY THE FORMAT PROGRAM (MAINDEC-11-DZRJB), THE PROGRAM WILL ACCEPT ALL ZERO'S AND ALL ONE'S PATTERNS; HOWEVER, ALL ZERO'S AND ALL ONE'S PATTERNS ARE NOT WRITTEN BY THE EXERCISER PROGRAM.

PATTERN '8' IS DEFINED AS THE 'WORST CASE' PATTERN.

PAT 1	PAT 2	PAT 3	PAT 4	PAT 5	PAT 6	PAT 7	PAT 8
000001	177776	000000	000000	052525	007417	026455	165555
000003	177774	000000	010421	052525	007417	026455	133333
000007	177770	000000	021042	052525	007417	026455	165555
000017	177760	177777	031463	125252	170360	151322	133333
000037	177740	177777	042104	125252	170360	151322	165555
000077	177700	177777	052525	125252	170360	151322	133333
000177	177600	000000	063146	052525	007417	026455	165555
000377	177400	000000	073567	052525	007417	026455	133333
000777	177000	177777	104210	125252	170360	151322	165555
001777	176000	177777	114631	125252	170360	151322	133333
003777	174000	000000	125252	052525	007417	026455	165555
007777	170000	177777	135673	125252	170360	151322	133333

017777	160000	000000	146314	052525	007417	026455	165555
037777	140000	177777	156735	125252	170360	151322	133333
077777	100000	000000	167356	052525	007417	026455	165555
177777	000000	177777	177777	125252	170360	151322	133333

PAT 9	PAT 10	PAT 11	PAT 12	PAT 13	PAT 14	PAT 15
000001	177776	172666	077777	153333	000000	177777
000002	177775	155555	137777	066667	177777	000000
000004	177773	172666	157777	153333	177777	000000
000010	177767	155555	167777	066667	177777	000000
000020	177757	172666	173777	153333	177777	000000
000040	177737	155555	175777	066667	177777	000000
000100	177677	172666	176777	153333	177777	000000
000200	177577	155555	177377	066667	177777	000000
000400	177377	172666	177577	153333	177777	000000
001000	176777	155555	177677	066667	177777	000000
002000	175777	172666	177737	153333	177777	000000
004000	173777	155555	177757	066667	177777	000000
010000	167777	172666	177767	153333	177777	000000
020000	157777	155555	177773	066667	177777	000000
040000	137777	172666	177775	153333	177777	000000
100000	077777	155555	177776	066667	177777	000000

9. PROGRAM LISTING

Handwritten scribbles

Handwritten scribbles

K03

.MAIN. MACY11 27(655) 28-APR-76 17:08 PAGE 35
DZRJDA.DOC

SEQ 0035

3
.END

ERRORS DETECTED: 0

*.DZRJDA/SOL=DZRJDA.DOC
RUN-TIME: 6 11 0 SECONDS
CORE USED: 3K

15	OPERATIONAL SWITCH SETTINGS
37	BASIC DEFINITIONS
150	RH11 REGISTERS
196	RPO4/5/6 REGISTERS
391	RPO4/5/6 DRIVER COMMANDS
416	TRAP CATCHER
425	STARTING ADDRESS(ES)
430	ACT11 HOOKS
440	COMMON TAGS
546	COMMON PARAMETERS
595	VALUES FOR FIRST OPERATION
618	TABLES, CONSTANTS, AND VARIABLE LOCATIONS
831	DATA PATTERNS
1126	ERROR POINTER TABLE
1187	SETUP AND INITIALIZATION ROUTINE
1198	INITIALIZE THE COMMON TAGS
1239	GET VALUE FOR SOFTWARE SWITCH REGISTER
1401	MAIN PROGRAM
3250	ERROR MESSAGE GENERATION ROUTINES
3692	GENERAL SUPPORT SUBROUTINES
5009	MACRO ROUTINES
5011	ERROR HANDLER ROUTINE
5053	ERROR MESSAGE TIMEOUT ROUTINE
5101	TYPE ROUTINE
5172	BINARY TO OCTAL (ASCII) AND TYPE
5250	CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
5318	TTY INPUT ROUTINE
5427	RANDOM NUMBER GENERATOR ROUTINE
5464	SAVE AND RESTORE RD-R5 ROUTINES
5510	DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE
5573	DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
5613	TRAP DECODER
5636	TRAP TABLE
5664	SINGLE/DUAL PORT RH11/RPO4/5/6 DRIVER (REV 1.0)
7083	DATA, CONTROL, & STATUS BLOCKS
7247	ERROR MESSAGES
7806	TELETYPE MESSAGES
8122	ROUTINE TO SIZE MEMORY
8153	BUSADR - GET BUS ADDRESS AND VECTOR ADDRESS FOR RH11
8217	CK.NUM - CHECK NUMBER (OCTAL)

M03

MD-11-DZRJD-A, RPO4/5/6 MULTI-DRIVE EXERCISER MACY11 27(655) 28-APR-76 17:04 PAGE 1
RPO456.010

SEQ 0037

1

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

.TITLE MD-11-DZRJD-A, RP04/5/6 MULTI-DRIVE EXERCISER
.*COPYRIGHT (C) 1975
.*DIGITAL EQUIPMENT CORP.
.*MAYNARD, MASS. 01754
.*
.*PROGRAM BY C. HESS
.*
.*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
.*PACKAGE (MAINDEC-11-DZQAC-C1),MAR 24, 1976.

.SBTTL OPERATIONAL SWITCH SETTINGS

SWITCH	USE
15	HALT ON ERROR
13	INHIBIT ERROR TYPEOUTS
10	BELL ON ERROR
7	DISPLAY ALL DATA COMPARE ERRORS
6	DON'T CHANGE PARAMETERS (LOOP ON PRESENT VALUES)
5	A. PARTIAL REGISTER DISPLAY IF ERROR B. NO ECC CORRECTION RESULTS DISPLAYED IF ERROR
4	A. DO NOT CHECK FOR MAXIMUM ERROR COUNTS B. DO NOT DROP DRIVE WHEN NORMAL END OF TEST REACHED
3	A. DISPLAY ERROR SECTOR IF 'DCK', 'DTE', OR 'WCF' ERROR B. DISPLAY SECTOR IF 'DCK' ERR UNCORRECTABLE AFTER 28TH RETRY C. IF DATA COMPARE ERROR & SW7 SET, DISPLAY REMAINDER OF BUFFER
2	A. DON'T TYPE SUBSYSTEM STATUS WHEN PROGRAM STARTED B. DON'T TYPE PERFORMANCE SUMMARY
1	INHIBIT DATA COMPARSION AFTER READ ORDERS
0	READ ONLY MODE

.SBTTL BASIC DEFINITIONS

001100 .*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
 STACK= 1100
 .EQUIV EMT,ERROR ;:BASIC DEFINITION OF ERROR CALL
 .EQUIV IOT,SCOPE ;:BASIC DEFINITION OF SCOPE CALL

.*MISCELLANEOUS DEFINITIONS
 HT= 11 ;:CODE FOR HORIZONTAL TAB
 LF= 12 ;:CODE FOR LINE FEED
 CR= 15 ;:CODE FOR CARRIAGE RETURN
 CRLF= 200 ;:CODE FOR CARRIAGE RETURN-LINE FEED
 PS= 177776 ;:PROCESSOR STATUS WORD
 .EQUIV PS,PSW
 STKLMT= 177774 ;:STACK LIMIT REGISTER
 PIRQ= 177772 ;:PROGRAM INTERRUPT REQUEST REGISTER
 DSWR= 177570 ;:HARDWARE SWITCH REGISTER
 DDISP= 177570 ;:HARDWARE DISPLAY REGISTER

.*GENERAL PURPOSE REGISTER DEFINITIONS

AS

110 004000
111 002000
112 001000
113 000400
114 000200
115 000100
116 000040
117 000020
118 000010
119 000004
120 000002
121 000001
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163

BIT11= 4000
BIT10= 2000
BIT09= 1000
BIT08= 400
BIT07= 200
BIT06= 100
BIT05= 40
BIT04= 20
BIT03= 10
BIT02= 4
BIT01= 2
BIT00= 1
.EQUIV BIT09,BIT9
.EQUIV BIT08,BIT8
.EQUIV BIT07,BIT7
.EQUIV BIT06,BIT6
.EQUIV BIT05,BIT5
.EQUIV BIT04,BIT4
.EQUIV BIT03,BIT3
.EQUIV BIT02,BIT2
.EQUIV BIT01,BIT1
.EQUIV BIT00,BIT0

.*BASIC "CPU" TRAP VECTOR ADDRESSES
ERRVEC= 4 : TIME OUT AND OTHER ERRORS
RESVEC= 10 : RESERVED AND ILLEGAL INSTRUCTIONS
TBITVEC= 14 : "T" BIT
TRTVEC= 14 : TRACE TRAP
BPTVEC= 14 : BREAKPOINT TRAP (BPT)
IOTVEC= 20 : INPUT/OUTPUT TRAP (IOT) **SCOPE**
PWRVEC= 24 : POWER FAIL
EMTVEC= 30 : EMULATOR TRAP (EMT) **ERROR**
TRAPVEC= 34 : "TRAP" TRAP
TKVEC= 60 : TTY KEYBOARD VECTOR
TPVEC= 64 : TTY PRINTER VECTOR
PIRQVEC= 240 : PROGRAM INTERRUPT REQUEST VECTOR

;;*****

.SBTTL RH11 REGISTERS

;;*****

;CONTROL AND STATUS REGISTER 1 (RPCS1)

IE= 100 ; INTERRUPT ENABLE (BIT #6)
RDY= 200 ; READY (BIT #7)
A16= 400 ; HIGH ORDER BUS ADDRESS BIT (BIT #8)
A17= 1000 ; HIGH ORDER BUS ADDRESS BIT (BIT #9)
PSEL= 2000 ; PORT SELECT (BIT #10)
MCPE= 20000 ; MASSBUSS PARITY ERROR (BIT #13)
TRE= 40000 ; TRANSFER ERROR (BIT #14)
;SC= 100000 ; SPECIAL CONDITION (BIT #15)

218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271

000200
000400
001000
002000
004000
010000
020000
040000
100000

DRY= 200
DPR= 400
PGM= 1000
LST= 2000
WRL= 4000
MOL= 10000
PIP= 20000
ERR= 40000
ATA= 100000

;DRIVE READY (BIT #7)
;DRIVE PRESENT (BIT #8)
;PROGRAMABLE (BIT #9)
;LAST SECTOR TRANSFERRED (BIT #10)
;WRITE LOCK (BIT #11)
;MEDIUM ON-LINE (BIT #12)
;POSITIONING OPERATION IN PROGRESS (BIT #13)
;COMPOSITE ERROR (BIT #14)
;ATTENTION ACTIVE (BIT #15)

;ERROR REGISTER #01 (RPER1) (#02)

000001
000002
000004
000010
000020
000040
000100
000200
000400
001000
002000
004000
010000
020000
040000
100000

ILF= 1
ILR= 2
RMR= 4
PAR= 10
FER= 20
WCF= 40
ECH= 100
HCE= 200
HCRC= 400
AOE= 1000
IAE= 2000
WLE= 4000
DTE= 10000
OPI= 20000
UNS= 40000
DCK= 100000

;ILLEGAL FUNCTION (BIT #0)
;ILLEGAL REGISTER (BIT #1)
;REGISTER MODIFICATION REFUSED (BIT #2)
;PARITY ERROR (BIT #3)
;FORMAT ERROR (BIT #4)
;WRITE CLOCK FAIL (BIT #5)
;ECC HARD ERROR (BIT #6)
;HEADER COMPARE ERROR (BIT #7)
;HEADER CRC ERROR (BIT #8)
;ADDRESS OVERFLOW ERROR (BIT #9)
;INVALID ADDRESS ERROR (BIT #10)
;WRITE LOCK ERROR (BIT #11)
;DRIVE TIMING ERROR (BIT #12)
;OPERATION INCOMPLETE (BIT #13)
;DRIVE UNSAFE (BIT #14)
;DATA CHECK ERROR (BIT 15)

;MAINTAINABILITY REGISTER (RPMR)(#03)

000001
000002
000004
000010
000020
000040
000200

DMD= 1
MCLK= 2
MINX= 4
MSTCK= 10
MRD= 20
MWR= 40
DTSY= 200

;DIAGINOSTIC MODE (BIT #0)
;MAINTAINABILITY CLOCK (BIT #1)
;MAINTAINABILITY INDEX (BIT #2)
;MAINTAINABILITY SECTOR CLOCK (BIT #3)
;MAINTAINABILITY READ (BIT #4)
;MAINTAINABILITY WRITE (BIT #5)
;MAINTAINABILITY SYNC DETECTED (BIT #7)

;ATTENTION SUMMARY PSEUDO-REGISTER (RPAS) (#04)

000001
000002
000004
000010
000020
000040
000100
000200

AT0= 1
AT1= 2
AT2= 4
AT3= 10
AT4= 20
AT5= 40
AT6= 100
AT7= 200

;DEVICE 0 (BIT #0)
;DEVICE 1 (BIT #1)
;DEVICE 2 (BIT #2)
;DEVICE 3 (BIT #3)
;DEVICE 4 (BIT #4)
;DEVICE 5 (BIT #5)
;DEVICE 6 (BIT #6)
;DEVICE 7 (BIT #7)

;DESIRED SECTOR/TRACK ADDRESS REGISTER (RPDA) (#05)
;(EACH BIT IS CALLED BY BIT NUMBER)

;DRIVE TYPE REGISTER (RPDT) (#06)

272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325

000001
000002
000004
000010
000020
000040
000100
000200
000400
004000
020000
040000
100000

DT00= 1
DT01= 2
DT02= 4
DT03= 10
DT04= 20
DT05= 40
DT06= 100
DT07= 200
DT08= 400
DRQ= 4000
MOH= 20000
TAP= 40000
NBA= 100000

; DRIVE TYPE NUMBER BIT 1
; DRIVE TYPE NUMBER BIT 2
; DRIVE TYPE NUMBER BIT 3
; DRIVE TYPE NUMBER BIT 4
; DRIVE TYPE NUMBER BIT 5
; DRIVE TYPE NUMBER BIT 6
; DRIVE TYPE NUMBER BIT 7
; DRIVE TYPE NUMBER BIT 8
; DRIVE TYPE NUMBER BIT 9
; DRIVE REQUEST REQUIRED (BIT #11)
; MOVING HEAD (BIT #13)
; TAPE DRIVE (BIT #14)
; NOT BLOCK ADDRESSED (BIT #15)

;LOOK-AHEAD REGISTER (RPLA) (#07)

000001
000002
000004
000010
000020
000040
000100
000200
001000
002000
004000
010000
020000
040000
100000

EXT1= 1
EXT2= 2
EXT4= 4
EXT10= 10
EXT20= 20
EXT40= 40
SC1= 100
SC2= 200
SC4= 400
SC10= 1000
SC20= 2000
TRK1= 4000
TRK2= 10000
TRK4= 20000
TRK10= 40000
TRK20= 100000

; EXTENSION 1 (BIT #0)
; EXTENSION 2 (BIT #1)
; EXTENSION 3 (BIT #2)
; EXTENSION 4 (BIT #3)
; EXTENSION 5 (BIT #4)
; EXTENSION 6 (BIT #5)
; SECTOR COUNT FIELD 0 (BIT #6)
; SECTOR COUNT FIELD 1 (BIT #7)
; SECTOR COUNT FIELD 2 (BIT #8)
; SECTOR COUNT FIELD 3 (BIT #9)
; SECTOR COUNT FIELD 4 (BIT #10)
; TRACK FIELD 1 (BIT #11)
; TRACK FIELD 2 (BIT #12)
; TRACK FIELD 3 (BIT #13)
; TRACK FIELD 4 (BIT #14)
; TRACK FIELD 5 (BIT #15)

;RP04 ERROR REGISTER #2 (RPER2) (#10)

000001
000002
000004
000010
000020
000040
000100
000200
000400
001000
002000
004000
010000
020000
100000

WCU= 1
CSF= 2
WSU= 4
CSU= 10
MSE= 20
TDF= 40
TUF= 100
FEN= 200
WRU= 400
MHS= 1000
NHS= 2000
IXE= 4000
VU30= 10000
PLU= 20000
ACU= 100000

; WRITE CURRENT UNSAFE (BIT #0)
; CURRENT SINK FAILURE (BIT #1)
; WRITE SELECT UNSAFE (BIT #2)
; CURRENT SWITCH UNSAFE (BIT #3)
; MOTOR SEQUENCE ERROR (BIT #4)
; TRANSITIONS DETECTOR FAILURE (BIT #5)
; TRANSITIONS UNSAFE (BIT #6)
; FAILSAFE ENABLED (BIT #7)
; WRITE READY UNSAFE (BIT #8)
; MULTIPLE HEAD SELECT (BIT #9)
; NO HEAD SELECTION (BIT #10)
; INDEX ERROR (BIT #11)
; 30VOLT UNSAFE (BIT #12)
; PLO UNSAFE (BIT #13)
; AC UNSAFE (BIT #15)

;RP05/6 ERROR REGISTER #02 (RPER2) (#10)

326	000001	WCU=	1	;WRITE CURRENT UNSAFE (BIT #0)
327	000002	CSF=	2	;CURRENT SINK FAILURE (BIT #1)
328	000004	WSU=	4	;WRITE SELECT UNSAFE (BIT #2)
329	000010	CSU=	10	;CURRENT SWITCH UNSAFE (BIT #3)
330	000020	RAW=	20	;READ AND WRITE (BIT #4)
331	000040	TDF=	40	;TRANSITIONS DETECTOR FAILURE (BIT #5)
332	000100	TUF=	100	;TRANSITIONS UNSAFE (BIT #6)
333	000200	ABS=	200	;ABNORMAL STOP (BIT #7)
334	000400	WRU=	400	;WRITE READY UNSAFE (BIT #8)
335	001000	MHS=	1000	;MULTIPLE HEAD SELECT (BIT #9)
336	002000	NHS=	2000	;NO HEAD SELECTION (BIT #10)
337	004000	IXE=	4000	;INDEX ERROR (BIT #11)
338	020000	PLU=	20000	;PLO UNSAFE (BIT #12)
339				
340				;OFFSET REGISTER (RPOF) (#11)
341				
342	000001	OF25=	1	;OFFSET 25 MICRO INCHES (BIT #0)
343	000002	OF50=	2	;OFFSET 50 MICRO INCHES (BIT #1)
344	000004	OF100=	4	;OFFSET 100 MICRO INCHES (BIT #2)
345	000010	OF200=	10	;OFFSET 200 MICRO INCHES (BIT #3)
346	000020	OF400=	20	;OFFSET 400 MICRO INCHES (BIT #4)
347	000040	OF800=	40	;OFFSET 800 MICRO INCHES (BIT #5)
348	000200	OFREV=	200	;OFFSET NEGATIVE (REVERSE) (BIT #5)
349	002000	HCI=	2000	;HEADER COMPARE INHIBIT (BIT #10)
350	004000	ECI=	4000	;ERROR CORRECTION CODE INHIBIT (BIT #11)
351	010000	FMT22=	10000	;FORMAT BIT (BIT #12)
352				
353				;DESIRED CYLINDER ADDRESS (RPCA) (#12)
354				; (EACH BIT IS CALLED BY BIT NUMBER)
355				
356				;CURRENT CYLINDER ADDRESS (RPCC) (#13)
357				; (EACH BIT IS CALLED BY BIT NUMBER)
358				
359				;SERIAL NUMBER REGISTER (RPSN) (#14)
360				; (EACH IS CALLED BY BIT NUMBER)
361				
362				;RPO4 ERROR REGISTER #03 (RPER3) (#15)
363				
364	000001	PSU=	1	;PACK SPEED UNSAFE (BIT #0)
365	000002	VUF=	2	;VELOCITY UNSAFE (BIT #1)
366	000010	UWR=	10	;ANY UNSAFE EXCEPT READ/WRITE (BIT #3)
367	000040	ACL=	40	;AC LOW (BIT #5)
368	000100	DCL=	100	;DC LOW (BIT #6)
369	040000	SKI=	40000	;SEEK INCOMPLETE (BIT #14)
370	100000	OCYL=	100000	;OFF CYLINDER (BIT #15)
371				
372				;RPO5/6 ERROR REGISTER #03 (RPER3) (#15)
373				
374	000001	DCU=	1	;DC UNSAFE (BIT #0)
375	000002	WAO=	2	;WRITE AND OFFSET (BIT #1)
376	000040	ACL=	40	;AC LOW (BIT #5)
377	000100	DCL=	100	;DC LOW (BIT #6)
378	020000	OPE=	20000	;OPERATOR PLUG ERROR (BIT #13)
379	040000	SKI=	40000	;SEEK INCOMPLETE (BIT #14)




```

380      100000      OCYL= 100000      ;OFF CYLINDER ERROR (BIT #15)
381
382      ;ECC POSITION REGISTER (RPEC1) (#16)
383      ;(EACH BIT IS CALLED BY BIT NUMBER)
384
385      ;ECC PATTERN REGISTER (RPEC2) (#17)
386      ;(EACH BIT IS CALLED BY BIT NUMBER)
387
388      ;*****
389
390      .SBTTL RPO4/5/6 DRIVER COMMANDS
391
392      ;*****
393
394      000101      RNOP      =      101      ;NO OPERATION
395      000103      UNLOAD   =      103      ;UNLOAD
396      000105      SEEK     =      105      ;SEEK
397      000107      RECAL    =      107      ;RECALIBRATE
398      000111      DRVCLR   =      111      ;DRIVE CLEAR
399      000113      RELSE    =      113      ;RELEASE
400      000115      OFFSET  =      115      ;OFFSET
401      000117      RTC     =      117      ;RETURN TO CENTER LINE
402      000121      READIN  =      121      ;READ IN PRESET
403      000123      ACK     =      123      ;PACK ACKNOWLEDGE
404      000131      SEARCH  =      131      ;SEARCH
405      000141      GETREG  =      141      ;GET REGISTERS
406      000143      SETFMT  =      143      ;SET FORMAT (& ECI OR HCI)
407      000145      SELDRV  =      145      ;SELECT DRIVE
408      000151      WCKD    =      151      ;WRITE CHECK DATA
409      000153      WCKHD   =      153      ;WRITE CHECK HEADER & DATA
410      000161      WRTDAT  =      161      ;WRITE DATA
411      000163      WRTHD   =      163      ;WRITE HEADER & DATA
412      000171      RDDAT   =      171      ;READ DATA
413      000173      RDHD    =      173      ;READ HEADER & DATA
414
415      .SBTTL TRAP CATCHER
416
417      000000      .=0
418      ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
419      ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
420      ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
421
422      000174      000000      .=174
423      000176      000000      DISPREG: .WORD 0      ;;SOFTWARE DISPLAY REGISTER
424      000176      000000      SWREG:  .WORD 0      ;;SOFTWARE SWITCH REGISTER
425      000200      000137      004076      .SBTTL STARTING ADDRESS(ES)
426      000204      000137      004066      JMP      @#START1      ;;JUMP TO STARTING ADDRESS OF PROGRAM
427      ;CHANGE THE RH11 UNIBUS ADDRESS
428      ;AFTER INITIAL START
429
430      .SBTTL ACT11 HOOKS
431
432      ;*****
433      000210      ;HOOKS REQUIRED BY ACT11
434      $SVPC=.      ;SAVE PC

```

434		000046
435	000046	005310
436		000052
437	000052	040000
438		000210

```

.=46
$ENDAD
.=52
.WORD 40000
.=SSVPC

```

```

;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
;;2)SET LOC.52 TO 40000
;; RESTORE PC

```


.SBTTL COMMON TAGS

*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
*USED IN THE PROGRAM.

439
440
441
442
443
444
445 001100
446 001100 000000
447 001100 000000
448 001102 000
449 001103 000
450 001104 000000
451 001106 000000
452 001110 000000
453 001112 000000
454 001114 000
455 001115 001
456 001116 000000
457 001120 000000
458 001122 000000
459 001124 000000
460 001126 000000
461 001130 000000
462 001132 000000
463 001134 000
464 001135 000
465 001136 000000
466 001140 177570
467 001142 177570
468 001144 177560
469 001146 177562
470 001150 177564
471 001152 177566
472 001154 000
473 001155 002
474 001156 012
475 001157 000
476 001160 177607 000377
477 001164 077
478 001165 015
479 001166 000012
480
481 000015
482 000012
483 001170 176700
484 001172 000254
485 001174 172540
486 001176 172542
487 001200 000104
488 001202 177546
489 001204 000100
490 001206 177777
491 001210 177777
492 001212 000074

.=1100
SCMTAG: .WORD 0
\$PASS: .WORD 0
\$STNM: .BYTE 00
\$ERFLG: .BYTE 00
\$ICNT: .WORD 00
\$LPADR: .WORD 00
\$LPERR: .WORD 00
\$ERTTL: .WORD 00
\$ITEMB: .BYTE 00
\$ERMAX: .BYTE 1
\$ERRPC: .WORD 00
\$GDADR: .WORD 00
\$BDADR: .WORD 00
\$GDDAT: .WORD 00
\$BDDAT: .WORD 00
\$AUTOB: .BYTE 00
\$INTAG: .BYTE 00
SWR: .WORD DSWR
DISPLAY: .WORD DDISP
\$TKS: 177560
\$TKB: 177562
\$TPS: 177564
\$TPB: 177566
\$NULL: .BYTE 0
\$FILLS: .BYTE 2
\$FILLC: .BYTE 12
\$STPFLG: .BYTE 0
\$BELL: .ASCIZ <207><377><377>
\$QUES: .ASCII /?/
\$CRLF: .ASCII <15>
\$LF: .ASCIZ <12>

CR = 15
LF = 12
\$RPADR: .WORD 176700
\$RPVEC: .WORD 254
\$LKCSR: .WORD 172540
\$LKCSB: .WORD 172542
\$LPVEC: .WORD 104
\$LKS: .WORD 177546
\$LLVEC: .WORD 100
PCLOCK: .WORD -1
CLKFLG: .WORD -1
HZ: .WORD 74

;; START OF COMMON TAGS
;; CONTAINS PASS COUNT
;; CONTAINS THE TEST NUMBER
;; CONTAINS ERROR FLAG
;; CONTAINS SUBTEST ITERATION COUNT
;; CONTAINS SCOPE LOOP ADDRESS
;; CONTAINS SCOPE RETURN FOR ERRORS
;; CONTAINS TOTAL ERRORS DETECTED
;; CONTAINS ITEM CONTROL BYTE
;; CONTAINS MAX. ERRORS PER TEST
;; CONTAINS PC OF LAST ERROR INSTRUCTION
;; CONTAINS ADDRESS OF 'GOOD' DATA
;; CONTAINS ADDRESS OF 'BAD' DATA
;; CONTAINS 'GOOD' DATA
;; CONTAINS 'BAD' DATA
;; RESERVED--NOT TO BE USED

;; AUTOMATIC MODE INDICATOR
;; INTERRUPT MODE INDICATOR

;; ADDRESS OF SWITCH REGISTER
;; ADDRESS OF DISPLAY REGISTER
;; TTY KBD STATUS
;; TTY KBD BUFFER
;; TTY PRINTER STATUS REG. ADDRESS
;; TTY PRINTER BUFFER REG. ADDRESS
;; CONTAINS NULL CHARACTER FOR FILLS
;; CONTAINS # OF FILLER CHARACTERS REQUIRED
;; INSERT FILL CHARS. AFTER A "LINE FEED"
;; "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
;; CODE FOR BELL
;; QUESTION MARK
;; CARRIAGE RETURN
;; LINE FEED

;; FIRST ADDRESS OF RH11/RP04/5/6 REGISTERS
;; RP04 VECTOR ADDRESS
;; ADDR OF KW11-P STATUS REGISTER
;; ADDR OF KW11-P COUNTER BUFFER
;; ADDR OF KW11-P VECTOR
;; ADDR OF KW11-L STATUS REGISTER
;; ADDR OF KW11-L VECTOR
;; '0' IF KW11-P IS ON SYSTEM
;; '0' IF A CLOCK IS AVAILABLE
;; 74 (8) IF 60 HZ SYSTEM; 62 (8) IF 50 HZ SYSTEM

```

493 001214 000000 STATIN: .WORD 0 ; 'TYPE STATISTICS' INDICATOR
494 001216 000000 PACK: .WORD 0 ; 'W' COMMAND INDICATOR
495 001220 000000 000000 000000 DATE: .WORD 0,0,0,0,0 ; OPERATOR ENTERED DATE
496 001226 000000 000000
497 001232 000000 000000 000000 OPERID: .WORD 0,0,0,0 ; OPERATOR ID
498 001240 000000
499 001242 000000 DRIVE: .WORD 0 ; DRIVE # STORAGE: ERRORS 1-5 & 10
500 001244 000000 ATTN: .WORD 0 ; ATTN REG STORAGE: ERRORS 1-5 & 10
501 001246 000000 UNIT: .WORD 0 ; DRIVE # STORAGE FOR PRINTOUT
502 001250 000000 MASK: .WORD 0 ; ERROR RETRY REGISTER MASK
503 001252 000 000 RETRY: .BYTE 0,0 ; ERROR RETRY LIMIT IN THE LOWER BYTE
504 ; RETRY COUNT IN THE UPPER BYTE
505 001254 000003 FAIRNS: .WORD 3 ; MAXIMUM TIME IN QUEUE VALUE
506 001256 000000 LSTAD: .WORD 0 ; STORE LAST MEMORY ADDRESS HERE
507 001260 000000 CHGADR: .WORD 0 ; CHANGE RH11 UNIBUS ADDRESS FLAG
508 001262 000000 CFLAG: .WORD 0 ; 'CONTROL C' FLAG
509 001264 000000 BADSEC: .WORD 0 ; BAD SECTOR/TRACK FLAG
510 001266 000000 HOUR: .WORD 0 ; HOUR COUNT STORED HERE (MAXIMUM - 999.)
511 001270 000000 MINUTE: .WORD 0 ; MINUTE'S COUNT STORED HERE
512 001272 000000 SECOND: .WORD 0 ; SECOND'S COUNT STORED HERE
513 001274 000000 SIXTEE: .WORD 0 ; TIMER ROUTINE COUNTER (FOR ONE SECOND)
514 001276 177777 ZROIND: .WORD -1 ; ZERO INDICATOR FOR THE DATA COMPARE ROUTINE
515 001300 000 FRSTER: .BYTE 0 ; DATA COMPARE ERROR FLAG
516 ; IF > 0, PROCESSING 'DCKER' OR CAN'T MATCH PATTERN
517 ; IF < 0, MISCOMPARSION FOUND
518 001301 000 .BYTE 0 ; MISCOMPARSION OR CAN'T MATCH PATTERN FLAG
519 ; IF < 0, ERROR IN BUFFER
520 001302 000000 SAVER1: .WORD 0 ; SAVE R1 HERE
521 001304 000000 SAVER5: .WORD 0 ; SAVE R5 HERE
522 001306 000000 ERCTR: .WORD 0 ; NUMBER OF ERRORS
523 001310 000000 LIMIT: .WORD 0 ; DISPLAY LIMIT
524 001312 000000 CMCNT: .WORD 0 ; WORD COUNT
525 001314 000000 CMCYL: .WORD 0 ; CYLINDER ADDRESS
526 001316 000 CMSEC: .BYTE 0 ; SECTOR ADDRESS
527 001317 000 CMTRK: .BYTE 0 ; TRACK ADDRESS
528 001320 000000 ECBIT: .WORD 0 ; ERROR BURST BIT OFFSET
529 001322 000000 ECSEC: .WORD 0 ; ERROR BURST WORD OFFSET (RELATIVE TO SECTOR)
530 001324 000000 ECMSK0: .WORD 0 ; CORRECTION MASK FOR FIRST ERROR WORD
531 001326 000000 ECMSK1: .WORD 0 ; CORRECTION MASK FOR SECOND ERROR WORD
532 001330 000000 ECWRD: .WORD 0 ; LOCATION OF FIRST ERROR WORD
533 001332 000000 ECGD: .WORD 0 ; GOOD DATA, FIRST WORD
534 001334 000000 ECBAD0: .WORD 0 ; BAD DATA, FIRST WORD
535 001336 000000 ECWRD1: .WORD 0 ; LOCATION OF SECOND ERROR WORD
536 001340 000000 ECGD1: .WORD 0 ; GOOD DATA, SECOND WORD
537 001342 000000 ECBAD1: .WORD 0 ; BAD DATA, SECOND WORD
538 001344 000025 SECLMT: .WORD 21. ; SECTOR ADDRESS LIMIT
539 001346 000022 TRKLMT: .WORD 18. ; TRACK ADDRESS LIMIT
540 001350 000632 CYLMT: .WORD 410. ; CYLINDER ADDRESS LIMIT FOR RPO4/5'S
; (CHANGED TO 814. FOR RPO6)

```

;;*****

.SBTTL COMMON PARAMETERS

541
542
543
544
545
546


```

547      ;:*****
548
549 001352 002740      ENDCON: .WORD 002740      ;1.875X10+8 WORDS (10) [3X10+9 BITS]
550 001354 005455      .WORD 005455      ;MSW
551 001356 143300      ENDSEK: .WORD 143300      ;3 X 10+6 SEEKS (LSW)
552 001360 000055      .WORD 55      ;MSW
553 001362 000001      PASCNT: .WORD 1      ;NUMBER OF PASSES TO END OF TEST
554 001364 000000      MAXDL: .WORD 0      ;MAXIMUM DATA TRANSFER SIZE IN WORDS
555      ;(FILLED BY PROGRAM AT STARTUP OR BY OPERATOR
556      ;DURING PARAMETER ENTRY DIALOG.)
557 001366 000144      MAXER: .WORD 100.      ;MAXIMUM ERRORS - 100(10)
558 001370 000005 000000      INTRVL: .WORD 5,0      ;FIRST WORD IS THE PERFORMANCE TYPEOUT INTERVAL
559      ;(IN MINUTES). SECOND WORD IS THE INTERVAL
560      ;COUNTER
561      ;COUNTER. UPPER BYTE IS VALUE.
562 001374 000004      CMPLMT: .WORD 4      ;NUMBER OF COMPARE ERRORS TYPED OUT
563 001376 000001      FORMAT: .WORD 1      ;IF NOT EQ 0, ALLOW WRITE HEADER & DATA ORDERS
564      ;IF EQ 0, DO NOT ALLOW WRITE HEADER & DATA ORDERS
565 001400 000000      WCSEL: .WORD 0      ;IF EQ TO 0, GENERATE A RANDOM WORD COUNT
566      ;FOR THE OPERATION.
567      ;IF NOT EQ TO 0, USE THE VALUE IN 'MAXDL' FOR
568      ;THE WORD COUNT
569 001402 000003      RATIO: .WORD 3      ;READ/WRITE RATIO [RANGE 0 - 7]
570      ;0 - 0/8 (READ/WRITE)
571      ;1 - 7/1
572      ;2 - 6/2
573      ;3 - 5/3
574      ;4 - 4/4
575      ;5 - 3/5
576      ;6 - 2/6
577      ;7 - 1/7
578 001404 000001      AUTOCK: .WORD 1      ;IF NOT EQ 0, DO AN APPROPRIATE WRITE
579      ;CHECK AFTER EACH WRITE ORDER.
580      ;IF EQ 0, SELECT WRITE CHECK ORDERS
581      ;RANDOMLY.
582 001406 000001      NOTPRT: .WORD 1      ;IF EQ 1, DO NOT PRINT DATA ERROR MESSAGES
583      ;ASSOCIATED WITH OPERATOR SPECIFIED
584      ;BAD PACK AREAS.
585      ;IF NOT EQ 0, PRINT ERROR MESSAGES RELATING TO
586      ;THESE AREAS.
587 001410 000001      ENDET: .WORD 1      ;IF NOT EQ 0, END OF PASS DETERMINED
588      ;BY THE 'WORDS READ' COUNT.
589      ;IF EQ 0, END OF PASS DETERMINED
590      ;BY THE SEEK COUNT.
591
592      ;:*****
593
594      .SBTTL VALUES FOR FIRST OPERATION
595
596      ;:*****
597
598 001412 000010      BEGPAT: .WORD 10      ;STARTING PATTERN CODE [RANGE 1 - 17 (OCTAL)]
599 001414 000005      BEGCOD: .WORD 5      ;STARTING COMMAND CODE [RANGE 0 - 5]
600      ;0 = WRITE CHECK DATA ('WCKD')

```

601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654

001416 000404

BEGSIZ: .WORD 404

:1 = WRITE CHECK HEADER & DATA ('WCHKHD')
:2 = WRITE DATA ('WRDAT')
:3 = WRITE HEADER & DATA ('WRTHD')
:4 = READ DATA ('RDDAT')
:5 = READ HEADER & DATA ('RDHD')
:STARTING RECORD SIZE [RANGE 4 - MAXMEM]
:NOTE: THE SIZE MUST BE AT LEAST 4 IF
:WRITE DATA OR READ DATA; THE SIZE MUST
:BE AT LEAST 8 IF WRITE HEADER AND
:DATA OR READ HEADER AND DATA.
:IF THE SIZE IS GREATER THAN 1 SECTOR, THE
:SIZE MUST ALLOW FOR OVERLAPPING 4 OR 8
:WORDS INTO THE LAST SECTOR USED.

;;*****

.SBTTL TABLES, CONSTANTS, AND VARIABLE LOCATIONS

;;*****

001420 000000 000000 000000 ORDERQ: .WORD 0,0,0,0,0,0,0,0,0 ;LIST OF DRIVES PERFORMING COMMANDS

001426 000000 000000 000000

001434 000000 000000 000000

001442 000000 ASNLST: .WORD 0 ;A BIT SET IS AN ASSIGNED DRIVE

001444 000000 000000 000000 DUNIT: .WORD 0,0,0,0,0,0,0,0,0 ;ADDRESSES OF DRIVES TO BE DEASSIGNED

001452 000000 000000 000000

001460 000000 000000 000000

001466 000000 000000 000000 NEWUNT: .WORD 0,0,0,0,0,0,0,0,0 ;ADDRESSES OF NEWLY ASSIGNED DRIVES

001474 000000 000000 000000

001502 000000 000000 000000

001510 000000 000000 000000 AVAIL: .WORD 0,0,0,0,0,0,0,0,0 ;LIST OF DRIVES WAITING FOR BUFFERS/PARAMETERS

001516 000000 000000 000000

001524 000000 000000 000000

001532 000000 000000 000000 WAIT: .WORD 0,0,0,0,0,0,0,0,0 ;LIST OF DRIVES WAITING FOR BUFFERS

001540 000000 000000 000000

001546 000000 000000 000000

001554 000000 000000 000000 PARQ: .WORD 0,0,0,0,0,0,0,0,0 ;LIST OF DRIVES WAITING FOR NEXT PARAMETERS

001562 000000 000000 000000

001570 000000 000000 000000

001576 000000 BUFTBL: .WORD 0 ;BUFFER ALLOCATION TABLE ENTRY COUNT

001600 000000 000000 .WORD 0,0

001604 000000 000000 .WORD 0,0

001610 000000 000000 .WORD 0,0

001614 000000 000000 .WORD 0,0

001620 000000 000000 .WORD 0,0

001624 000000 000000 .WORD 0,0

001630 000000 000000 .WORD 0,0

655	001634	000000	000000	.WORD	0,0	
656	001640	000000	000000	.WORD	0,0	
657	001644	000000	000000	.WORD	0,0	
658	001650	000000	000000	.WORD	0,0	
659	001654	000000	000000	.WORD	0,0	
660	001660	000000	000000	.WORD	0,0	
661	001664	000000	000000	.WORD	0,0	
662	001670	000000	000000	.WORD	0,0	
663	001674	000000	000000	.WORD	0,0	
664	001700	000000	000000	.WORD	0,0	
665	001704	000000	000000	.WORD	0,0	
666	001710	000000	000000	.WORD	0,0	
667	001714	000000	000000	.WORD	0,0	
668						
669	001720	041606		BLKADR: .WORD	DRIVE0	: ADDRESS OF THE BLOCK FOR DRIVE 0
670	001722	042112		.WORD	DRIVE1	: ADDRESS OF THE BLOCK FOR DRIVE 1
671	001724	042416		.WORD	DRIVE2	: ADDRESS OF THE BLOCK FOR DRIVE 2
672	001726	042722		.WORD	DRIVE3	: ADDRESS OF THE BLOCK FOR DRIVE 3
673	001730	043226		.WORD	DRIVE4	: ADDRESS OF THE BLOCK FOR DRIVE 4
674	001732	043532		.WORD	DRIVE5	: ADDRESS OF THE BLOCK FOR DRIVE 5
675	001734	044036		.WORD	DRIVE6	: ADDRESS OF THE BLOCK FOR DRIVE 6
676	001736	044342		.WORD	DRIVE7	: ADDRESS OF THE BLOCK FOR DRIVE 7
677						
678	001740	151		COMTBL: .BYTE	WCKD	: WRITE CHECK DATA
679	001741	153		.BYTE	WCKHD	: WRITE CHECK HEADER AND DATA
680	001742	161		.BYTE	WRTDAT	: WRITE DATA
681	001743	163		.BYTE	WRTHD	: WRITE HEADER AND DATA
682	001744	171		.BYTE	RDDAT	: READ DATA
683	001745	173		.BYTE	RDHD	: READ HEADER AND DATA
684						
685	001746	002		OPTBL: .BYTE	2	: UNLOAD
686	001747	004		.BYTE	4	: SEEK
687	001750	006		.BYTE	6	: RECAL
688	001751	010		.BYTE	10	: DRIVE CLEAR
689	001752	012		.BYTE	12	: RELEASE
690	001753	014		.BYTE	14	: OFFSET
691	001754	016		.BYTE	16	: RETURN TO CENTERLINE
692	001755	020		.BYTE	20	: READIN PRESET
693	001756	022		.BYTE	22	: PACK ACKNOWLEDGE
694	001757	030		.BYTE	30	: SEARCH
695	001760	050		.BYTE	50	: WRITE CHECK DATA
696	001761	052		.BYTE	52	: WRITE CHECK HEADER AND DATA
697	001762	060		.BYTE	60	: WRITE DATA
698	001763	062		.BYTE	62	: WRITE HEADER AND DATA
699	001764	070		.BYTE	70	: READ DATA
700	001765	072		.BYTE	72	: READ HEADER AND DATA
701	001766	377		.BYTE	-1	: TERMINATOR
702						
703		001770		.EVEN		
704						
705	001770	047125	047514	042101	MNTBL: .ASCIZ	/UNLOAD /
706	001776	000040				
707	002000	042523	045505	020040	.ASCIZ	/SEEK /
708	002006	000040				

709	002010	042522	040503	020114	.ASCIZ	/RECAL	/
710	002013	000040					
711	002020	051104	041526	051114	.ASCIZ	/DRVCLR	/
712	002026	000040					
713	002030	042522	051514	020105	.ASCIZ	/RELSE	/
714	002036	000040					
715	002040	043117	051506	052105	.ASCIZ	/OFFSET	/
716	002046	000040					
717	002050	052122	020103	020040	.ASCIZ	/RTC	/
718	002056	000040					
719	002060	042522	042101	047111	.ASCIZ	/READIN	/
720	002066	000040					
721	002070	040520	045503	020040	.ASCIZ	/PACK	/
722	002076	000040					
723	002100	042523	051101	044103	.ASCIZ	/SEARCH	/
724	002106	000040					
725	002110	041527	042113	020040	.ASCIZ	/WCKD	/
726	002116	000040					
727	002120	041527	044113	020104	.ASCIZ	/WCKHD	/
728	002126	000040					
729	002130	051127	042124	052101	.ASCIZ	/WRTDAT	/
730	002136	000040					
731	002140	051127	044124	020104	.ASCIZ	/WRTHD	/
732	002146	000040					
733	002150	042122	040504	020124	.ASCIZ	/RDDAT	/
734	002156	000040					
735	002160	042122	042110	020040	.ASCIZ	/RDHD	/
736	002166	000040					
737	002170	047516	042516	020040	.ASCIZ	/NONE	/
738	002176	000040					
739							

740	002200	000		
741	002201	010		
742	002202	210		
743	002203	020		
744	002204	220		
745	002205	030		
746	002206	230	000	
747	002210	020		
748	002211	220		
749	002212	040		
750	002213	240		
751	002214	060		
752	002215	260	000	
753		002220		

```

OFFCOD: .BYTE 0          :OFFSET CODE TABLE
         .BYTE 10       :+200 U INCHES
         .BYTE 210      :-200 U INCHES
         .BYTE 20       :+400 U INCHES
         .BYTE 220      :-400 U INCHES
         .BYTE 30       :+600 U INCHES
         .BYTE 230,0    :-600 U INCHES, TERMINATOR
         .BYTE 20       :+400 U INCHES
         .BYTE 220      :-400 U INCHES
         .BYTE 40       :+800 U INCHES
         .BYTE 240      :-800 U INCHES
         .BYTE 60       :+1200 U INCHES
         .BYTE 260,0    :-1200 U INCHES, TERMINATOR

```

754					.EVEN		
755	002220	002254			OFMTBL:	.WORD	OFMSG0
756	002222	002307				.WORD	OFMSG1
757	002224	002343				.WORD	OFMSG2
758	002226	002377				.WORD	OFMSG3
759	002230	002433				.WORD	OFMSG4
760	002232	002467				.WORD	OFMSG5
761	002234	002523				.WORD	OFMSG6
762	002236	002254				.WORD	OFMSG0

```

:1ST OFFSET MESSAGE
:2ND OFFSET MESSAGE
:3RD OFFSET MESSAGE
:4TH OFFSET MESSAGE
:5TH OFFSET MESSAGE
:6TH OFFSET MESSAGE
:7TH OFFSET MESSAGE
:1ST OFFSET MESSAGE

```


763	002240	002377			.WORD	OFMSG3	:4TH OFFSET MESSAGE
764	002242	002433			.WORD	OFMSG4	:5TH OFFSET MESSAGE
765	002244	002557			.WORD	OFMSG7	:8TH OFFSET MESSAGE
766	002246	002613			.WORD	OFMSG8	:9TH OFFSET MESSAGE
767	002250	002647			.WORD	OFMSG9	:10TH OFFSET MESSAGE
768	002252	002704			.WORD	OFMSGA	:11TH OFFSET MESSAGE
769							
770	002254	043101	042524	020122	OFMSG0:	.ASCIZ	/AFTER RETRY WITHOUT OFFSET/
771	002262	042522	051124	020131			
772	002270	044527	044124	052517			
773	002276	020124	043117	051506			
774	002304	052105	000				
775	002307	101	020124	043117	OFMSG1:	.ASCIZ	/AT OFFSET +200 MICRO-INCHES/
776	002314	051506	052105	025440			
777	002322	030062	020060	044515			
778	002330	051103	026517	047111			
779	002336	044103	051505	000			
780	002343	101	020124	043117	OFMSG2:	.ASCIZ	/AT OFFSET -200 MICRO-INCHES/
781	002350	051506	052105	026440			
782	002356	030062	020060	044515			
783	002364	051103	026517	047111			
784	002372	044103	051505	000			
785	002377	101	020124	043117	OFMSG3:	.ASCIZ	/AT OFFSET +400 MICRO-INCHES/
786	002404	051506	052105	025440			
787	002412	030064	020060	044515			
788	002420	051103	026517	047111			
789	002426	044103	051505	000			
790	002433	101	020124	043117	OFMSG4:	.ASCIZ	/AT OFFSET -400 MICRO-INCHES/
791	002440	051506	052105	026440			
792	002446	030064	020060	044515			
793	002454	051103	026517	047111			
794	002462	044103	051505	000			
795	002467	101	020124	043117	OFMSG5:	.ASCIZ	/AT OFFSET +600 MICRO-INCHES/
796	002474	051506	052105	025440			
797	002502	030066	020060	044515			
798	002510	051103	026517	047111			
799	002516	044103	051505	000			
800	002523	101	020124	043117	OFMSG6:	.ASCIZ	/AT OFFSET -600 MICRO-INCHES/
801	002530	051506	052105	026440			
802	002536	030066	020060	044515			
803	002544	051103	026517	047111			
804	002552	044103	051505	000			
805	002557	101	020124	043117	OFMSG7:	.ASCIZ	/AT OFFSET +800 MICRO-INCHES/
806	002564	051506	052105	025440			
807	002572	030070	020060	044515			
808	002600	051103	026517	047111			
809	002606	044103	051505	000			
810	002613	101	020124	043117	OFMSG8:	.ASCIZ	/AT OFFSET -800 MICRO-INCHES/
811	002620	051506	052105	026440			
812	002626	030070	020060	044515			
813	002634	051103	026517	047111			
814	002642	044103	051505	000			
815	002647	101	020124	043117	OFMSG9:	.ASCIZ	/AT OFFSET +1200 MICRO-INCHES/
816	002654	051506	052105	025440			

871	003050	000003	.WORD	000003
872	003052	000007	.WORD	000007
873	003054	000017	.WORD	000017
874	003056	000037	.WORD	000037
875	003060	000077	.WORD	000077
876	003062	000177	.WORD	000177
877	003064	000377	.WORD	000377
878	003066	000777	.WORD	000777
879	003070	001777	.WORD	001777
880	003072	003777	.WORD	003777
881	003074	007777	.WORD	007777
882	003076	017777	.WORD	017777
883	003100	037777	.WORD	037777
884	003102	077777	.WORD	077777
885	003104	177777	.WORD	177777
886				
887	003106	177776	.WORD	177776
888	003110	177774	.WORD	177774
889	003112	177770	.WORD	177770
890	003114	177760	.WORD	177760
891	003116	177740	.WORD	177740
892	003120	177700	.WORD	177700
893	003122	177600	.WORD	177600
894	003124	177400	.WORD	177400
895	003126	177000	.WORD	177000
896	003130	176000	.WORD	176000
897	003132	174000	.WORD	174000
898	003134	170000	.WORD	170000
899	003136	160000	.WORD	160000
900	003140	140000	.WORD	140000
901	003142	100000	.WORD	100000
902	003144	000000	.WORD	000000
903				
904	003146	000000	.WORD	000000
905	003150	000000	.WORD	000000
906	003152	000000	.WORD	000000
907	003154	177777	.WORD	177777
908	003156	177777	.WORD	177777
909	003160	177777	.WORD	177777
910	003162	000000	.WORD	000000
911	003164	000000	.WORD	000000
912	003166	177777	.WORD	177777
913	003170	177777	.WORD	177777
914	003172	000000	.WORD	000000
915	003174	177777	.WORD	177777
916	003176	000000	.WORD	000000
917	003200	177777	.WORD	177777
918	003202	000000	.WORD	000000
919	003204	177777	.WORD	177777
920				
921	003206	000000	.WORD	000000
922	003210	010421	.WORD	010421
923	003212	021042	.WORD	021042
924	003214	031463	.WORD	031463

;STANDARD PATTERN 2

;STANDARD PATTERN 3

;STANDARD PATTERN 4

925	003216	042104	.WORD	042104
926	003220	052525	.WORD	052525
927	003222	063146	.WORD	063146
928	003224	073567	.WORD	073567
929	003226	104210	.WORD	104210
930	003230	114631	.WORD	114631
931	003232	125252	.WORD	125252
932	003234	135673	.WORD	135673
933	003236	146314	.WORD	146314
934	003240	156735	.WORD	156735
935	003242	167356	.WORD	167356
936	003244	177777	.WORD	177777
937				
938	003246	052525	.WORD	052525
939	003250	052525	.WORD	052525
940	003252	052525	.WORD	052525
941	003254	125252	.WORD	125252
942	003256	125252	.WORD	125252
943	003260	125252	.WORD	125252
944	003262	052525	.WORD	052525
945	003264	052525	.WORD	052525
946	003266	125252	.WORD	125252
947	003270	125252	.WORD	125252
948	003272	052525	.WORD	052525
949	003274	125252	.WORD	125252
950	003276	052525	.WORD	052525
951	003300	125252	.WORD	125252
952	003302	052525	.WORD	052525
953	003304	125252	.WORD	125252
954				
955	003306	007417	.WORD	007417
956	003310	007417	.WORD	007417
957	003312	007417	.WORD	007417
958	003314	170360	.WORD	170360
959	003316	170360	.WORD	170360
960	003320	170360	.WORD	170360
961	003322	007417	.WORD	007417
962	003324	007417	.WORD	007417
963	003326	170360	.WORD	170360
964	003330	170360	.WORD	170360
965	003332	007417	.WORD	007417
966	003334	170360	.WORD	170360
967	003336	007417	.WORD	007417
968	003340	170360	.WORD	170360
969	003342	007417	.WORD	007417
970	003344	170360	.WORD	170360
971				
972	003346	026455	.WORD	026455
973	003350	026455	.WORD	026455
974	003352	026455	.WORD	026455
975	003354	151322	.WORD	151322
976	003356	151322	.WORD	151322
977	003360	151322	.WORD	151322
978	003362	026455	.WORD	026455

;STANDARD PATTERN 5

;STANDARD PATTERN 6

;STANDARD PATTERN 7

19

1

1

1

979	003364	026455	.WORD	026455	
980	003366	151322	.WORD	151322	
981	003370	151322	.WORD	151322	
982	003372	026455	.WORD	026455	
983	003374	151322	.WORD	151322	
984	003376	026455	.WORD	026455	
985	003400	151322	.WORD	151322	
986	003402	026455	.WORD	026455	
987	003404	151322	.WORD	151322	
988					
989	003406	165555	.WORD	165555	; STANDARD PATTERN 8
990	003410	133333	.WORD	133333	
991	003412	165555	.WORD	165555	
992	003414	133333	.WORD	133333	
993	003416	165555	.WORD	165555	
994	003420	133333	.WORD	133333	
995	003422	165555	.WORD	165555	
996	003424	133333	.WORD	133333	
997	003426	165555	.WORD	165555	
998	003430	133333	.WORD	133333	
999	003432	165555	.WORD	165555	
1000	003434	133333	.WORD	133333	
1001	003436	165555	.WORD	165555	
1002	003440	133333	.WORD	133333	
1003	003442	165555	.WORD	165555	
1004	003444	133333	.WORD	133333	
1005					
1006	003446	000001	.WORD	000001	; STANDARD PATTERN 9
1007	003450	000002	.WORD	000002	
1008	003452	000004	.WORD	000004	
1009	003454	000010	.WORD	000010	
1010	003456	000020	.WORD	000020	
1011	003460	000040	.WORD	000040	
1012	003462	000100	.WORD	000100	
1013	003464	000200	.WORD	000200	
1014	003466	000400	.WORD	000400	
1015	003470	001000	.WORD	001000	
1016	003472	002000	.WORD	002000	
1017	003474	004000	.WORD	004000	
1018	003476	010000	.WORD	010000	
1019	003500	020000	.WORD	020000	
1020	003502	040000	.WORD	040000	
1021	003504	100000	.WORD	100000	
1022					
1023	003506	177776	.WORD	177776	; STANDARD PATTERN 10
1024	003510	177775	.WORD	177775	
1025	003512	177773	.WORD	177773	
1026	003514	177767	.WORD	177767	
1027	003516	177757	.WORD	177757	
1028	003520	177737	.WORD	177737	
1029	003522	177677	.WORD	177677	
1030	003524	177577	.WORD	177577	
1031	003526	177377	.WORD	177377	
1032	003530	176777	.WORD	176777	

Handwritten mark

Handwritten mark

Handwritten mark

Handwritten mark

1033	003532	175777	.WORD	175777	
1034	003534	173777	.WORD	173777	
1035	003536	167777	.WORD	167777	
1036	003540	157777	.WORD	157777	
1037	003542	137777	.WORD	137777	
1038	003544	077777	.WORD	077777	
1039					
1040	003546	172666	.WORD	172666	; STANDARD PATTERN 11
1041	003550	155555	.WORD	155555	
1042	003552	172666	.WORD	172666	
1043	003554	155555	.WORD	155555	
1044	003556	172666	.WORD	172666	
1045	003560	155555	.WORD	155555	
1046	003562	172666	.WORD	172666	
1047	003564	155555	.WORD	155555	
1048	003566	172666	.WORD	172666	
1049	003570	155555	.WORD	155555	
1050	003572	172666	.WORD	172666	
1051	003574	155555	.WORD	155555	
1052	003576	172666	.WORD	172666	
1053	003600	155555	.WORD	155555	
1054	003602	172666	.WORD	172666	
1055	003604	155555	.WORD	155555	
1056					
1057	003606	077777	.WORD	077777	; STANDARD PATTERN 12
1058	003610	137777	.WORD	137777	
1059	003612	157777	.WORD	157777	
1060	003614	167777	.WORD	167777	
1061	003616	173777	.WORD	173777	
1062	003620	175777	.WORD	175777	
1063	003622	176777	.WORD	176777	
1064	003624	177377	.WORD	177377	
1065	003626	177577	.WORD	177577	
1066	003630	177677	.WORD	177677	
1067	003632	177737	.WORD	177737	
1068	003634	177757	.WORD	177757	
1069	003636	177767	.WORD	177767	
1070	003640	177773	.WORD	177773	
1071	003642	177775	.WORD	177775	
1072	003644	177776	.WORD	177776	
1073					
1074	003646	153333	.WORD	153333	; STANDARD PATTERN 13
1075	003650	066667	.WORD	066667	
1076	003652	153333	.WORD	153333	
1077	003654	066667	.WORD	066667	
1078	003656	153333	.WORD	153333	
1079	003660	066667	.WORD	066667	
1080	003662	153333	.WORD	153333	
1081	003664	066667	.WORD	066667	
1082	003666	153333	.WORD	153333	
1083	003670	066667	.WORD	066667	
1084	003672	153333	.WORD	153333	
1085	003674	066667	.WORD	066667	
1086	003676	153333	.WORD	153333	

Handwritten signature or initials

1087	003700	066667	.WORD	066667	
1088	003702	153333	.WORD	153333	
1089	003704	066667	.WORD	066667	
1090					
1091	003706	000000	.WORD	000000	; STANDARD PATTERN 14
1092	003710	177777	.WORD	177777	
1093	003712	177777	.WORD	177777	
1094	003714	177777	.WORD	177777	
1095	003716	177777	.WORD	177777	
1096	003720	177777	.WORD	177777	
1097	003722	177777	.WORD	177777	
1098	003724	177777	.WORD	177777	
1099	003726	177777	.WORD	177777	
1100	003730	177777	.WORD	177777	
1101	003732	177777	.WORD	177777	
1102	003734	177777	.WORD	177777	
1103	003736	177777	.WORD	177777	
1104	003740	177777	.WORD	177777	
1105	003742	177777	.WORD	177777	
1106	003744	177777	.WORD	177777	
1107					
1108	003746	177777	.WORD	177777	; STANDARD PATTERN 15
1109	003750	000000	.WORD	000000	
1110	003752	000000	.WORD	000000	
1111	003754	000000	.WORD	000000	
1112	003756	000000	.WORD	000000	
1113	003760	000000	.WORD	000000	
1114	003762	000000	.WORD	000000	
1115	003764	000000	.WORD	000000	
1116	003766	000000	.WORD	000000	
1117	003770	000000	.WORD	000000	
1118	003772	000000	.WORD	000000	
1119	003774	000000	.WORD	000000	
1120	003776	000000	.WORD	000000	
1121	004000	000000	.WORD	000000	
1122	004002	000000	.WORD	000000	
1123	004004	000000	.WORD	000000	
1124					

eg h. d. s. l.

1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178

.SBTTL ERROR POINTER TABLE

;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
;*LOCATION \$ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
;*NOTE1: IF \$ITEMB IS 0 THE ONLY PERTINENT DATA IS (\$ERRPC).
;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

;* EM ;;POINTS TO THE ERROR MESSAGE
;* DH ;;POINTS TO THE DATA HEADER
;* DT ;;POINTS TO THE DATA
;* DF ;;POINTS TO THE DATA FORMAT

\$ERRTB:
;ERROR 1

;ERROR 1

EM1
DH1
DT1
0

;RH11 INTERRUPT OCCURRED (RPAS = 0)

;ERROR 2

EM2
DH2
DT2
0

;UNEXPECTED ATTENTION OCCURRED

;ERROR 3

EM3
DH3
DT3
0

;MASSBUS PARITY ERROR (MCPE=1)

;ERROR 4

EM4
DH4
DT4
0

;MASSBUS PARITY ERROR (PAR=1)

;ERROR 5

EM5
DH2
DT2
0

;ADDRESS PLUG BIT CHANGED

;ERROR 6

Handwritten initials


```

1179 004056 045166          EM6          ;RH11 DIDN'T RESPOND TO ADDRESSING
1180 004060 047545          DH6
1181 004062 050074          DT6
1182 004064 000000          0
1183
1184 ;;*****
1185 .SBTTL  SETUP AND INITIALIZATION ROUTINE
1186
1187 ;       START ADDRESS = 200
1188 ;       ADDRESS TO CHANGE RH11 UNIBUS ADDRESS = 204
1189
1190 ;;*****
1191
1192
1193 004066 012737 177777 001260 START:  MOV    #-1,CHGADR      ;SET RH11 ADDRESS CHANGE FLAG
1194 004074 000402          BR      START2             ;START THE PROGRAM
1195 004076 005037 001260 START1: CLR    CHGADR       ;CLEAR THE RH11 ADDRESS CHANGE FLAG
1196 004102 000005          START2: RESET            ;CLEAR THE BUS
1197 .SBTTL  INITIALIZE THE COMMON TAGS
1198 ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
1199 004104 012706 001100          MOV    $CMTAG,R6         ;;FIRST LOCATION TO BE CLEARED
1200 004110 005026          CLR    (R6)+             ;;CLEAR MEMORY LOCATION
1201 004112 022706 001140          CMP    $SWR,R6 ;;DONE?
1202 004116 001374          BNE    -6                ;;LOOP BACK IF NO
1203 004120 012706 001100          MOV    $STACK,SP        ;;SETUP THE STACK POINTER
1204 ;;INITIALIZE A FEW VECTORS
1205 004124 012737 030460 000030          MOV    $ERROR,@#EMTVEC  ;;EMT VECTOR FOR ERROR ROUTINE
1206 004132 012737 000340 000032          MOV    #340,@#EMTVEC+2 ;;LEVEL 7
1207 004140 012737 033014 000034          MOV    $TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
1208 004146 012737 000340 000036          MOV    #340,@#TRAPVEC+2;LEVEL 7
1209 004154 012737 176543 032400          MOV    #176543,$HINUM   ;;PRIME THE RANDOM NUMBER GENERATOR
1210 004162 012737 123456 032402          MOV    #123456,$LONUM   ;;BOTH HIGH AND LOW WORDS
1211 ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
1212 ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
1213 004170 013746 000004          MOV    @#ERRVEC,-(SP)    ;;SAVE ERROR VECTOR
1214 004174 012737 004230 000004          MOV    #64$,@#ERRVEC    ;;SET UP ERROR VECTOR
1215 004202 012737 177570 001140          MOV    $DSWR,$SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
1216 004210 012737 177570 001142          MOV    $DDISP,$DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
1217 004216 022777 177777 174714          CMP    #-1,@$SWR        ;;TRY TO REFERENCE HARDWARE SWR
1218 004224 001012          BNE    66$              ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
1219 ;;AND THE HARDWARE SWR IS NOT = -1
1220 004226 000403          BR     65$              ;;BRANCH IF NO TIMEOUT
1221 004230 012716 004236          64$:  MOV    #65$,(SP)       ;;SET UP FOR TRAP RETURN
1222 004234 000002          RTI
1223 004236 012737 000176 001140          65$:  MOV    $SWREG,$SWR     ;;POINT TO SOFTWARE SWR
1224 004244 012737 000174 001142          MOV    $DISPR$,$DISPLAY
1225 004252 012637 000004          66$:  MOV    (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
1226
1227 004256 012737 000240 000032          MOV    #240,@#EMTVEC+2  ;;CHANGE EMT PRIORITY TO 5
1228 004264 012737 000240 000036          MOV    #240,@#TRAPVEC+2;CHANGE TRAP PRIORITY TO 5
1229 004272 005227 177777          INC    #-1              ;;FIRST START ?
1230 004276 001013          BNE    1$              ;;BR IF NOT
1231 004300 104401 055254          TYPE  ,TITLE           ;;NAME AND MANDEC NUMBER
1232 004304 005737 000042          TST   42               ;;AUTO ACCEPT OR CHAIN MODE ?

```



```

1233 004310 001006          BNE      1$          ;BR IF EITHER
1234 004312 122737 000011 000041  CMPB    #11,41      ;LOADED FROM AN RPO4/5/6 ?
1235 004320 001002          BNE      1$          ;BR IF NOT
1236 004322 104401 055354          TYPE    LOADRV      ;INSTRUCT THE OPERATOR ON HOW TO TEST DRIVE 0
1237 004326 004737 030000          JSR     PC,$TKINT   ;TURN ON THE KEYBOARD INTERRUPT
1238                                .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
1239 004332 005737 000042          TST     Q#42        ;ARE WE RUNNING UNDER XXDP/ACT?
1240 004336 001006          BNE      67$        ;BRANCH IF YES
1241 004340 023727 001140 000176  CMP     SWR,#SWREG  ;SOFTWARE SWITCH REG SELECTED?
1242 004346 001005          BNE      68$        ;BRANCH IF NO
1243 004350 104406          GTSWR                    ;GET SOFT-SWR SETTINGS
1244 004352 000403          BR      68$
1245 004354 112737 000001 001134 67$:  MOVB    #1,$AUTOB  ;;SET AUTO-MODE INDICATOR
1246 004362                                68$:
1247 004362 005227 177777          INC     #-1         ;FIRST START ?
1248 004366 001012          BNE      2$         ;BR IF NOT
1249 004370 004737 054662          JSR     PC,BUSADR   ;CHECK RH11 BUS ADDRESS
1250 004374 013737 001170 033250  MOV     $RPADR,RPADR ;RH11 ADDRESS
1251 004402 013737 001172 033252  MOV     $RPVEC,RPVEC ;RH11 VECTOR ADDRESS
1252 004410 004737 054404          JSR     PC,OPRDAT   ;GET THE DATE AND OPERATOR ID
1253 004414 005037 001214          CLR     STATIN      ;CLEAR PERFORMANCE SUMMARY TYPEOUT FLAG
1254 004420 012705 001420          MOV     #ORDERQ,R5 ;START OF AREA TO CLEAR
1255 004424 005025          CLR     (R5)+
1256 004426 022705 001720          CMP     #BLKADR,R5 ;LOOK FOR END OF CLEAR AREA
1257 004432 001374          BNE      3$         ;BR IF NOT FINISHED
1258 004434 012706 001100          MOV     #STACK,SP  ;SETUP THE STACK POINTER
1259 004440 005037 177776          CLR     PS          ;CLEAR THE PROCESSOR STATUS WORD
1260 004444 013737 001212 001274  MOV     HZ,SIXTEE   ;1/60 TH OR 1/50 TH SECOND COUNTER VALUE
1261 004452 005037 001266          CLR     HOUR        ;CLEAR THE HOUR'S COUNTER
1262 004456 005037 001270          CLR     MINUTE      ;CLEAR THE MINUTE'S COUNTER
1263 004462 005037 001272          CLR     SECOND      ;CLEAR THE SECOND'S COUNTER
1264 004466 005037 001372          CLR     INTRVL+2    ;CLEAR INTERVAL COUNTER
1265 004472 005037 001216          CLR     PACK        ;CLEAR THE 'R' OR 'W' COMMAND FLAG
1266 004476 005037 001262          CLR     CFLAG      ;CLEAR THE 'CONTROL C' FLAG
1267 004502 042737 170000 001366  BIC     #170000,MAXER ;MAKE SURE ERROR LIMITS ARE NOT TOO HIGH
1268
1269                                ;ROUTINE TO DETERMINE BUFFER AREA SIZE
1270
1271 004510 005227 177777          SIZMEM: INC     #-1         ;SEE IF TIME TO SIZE MEMORY
1272 004514 001005          BNE      1$         ;BR IF NOT
1273 004516 004737 054564          JSR     PC,$SIZE    ;SEE HOW MUCH MEMORY ON SYSTEM
1274 004522 013737 054660 001256  MOV     $LSTAD,LSTAD ;SAVE THE LAST ADDRESS
1275 004530 012737 000001 001576 1$:  MOV     #1,BUFTBL   ;LOAD NUMBER OF BUFFERS
1276 004536 012737 054404 001600  MOV     #ENDPGM,BUFTBL+2 ;STARTING ADDRESS OF BUFFER
1277 004544 013737 001256 001602  MOV     LSTAD,BUFTBL+4 ;LAST ADDR TO BUFFER ALLOCATION TABLE
1278 004552 162737 054404 001602  SUB     #ENDPGM,BUFTBL+4 ;SUBTRACT PROGRAM SPACE
1279 004560 000241          CLC
1280 004562 006037 001602          ROR     BUFTBL+4    ;CONVERT TO WORD COUNT
1281 004566 162737 000144 001602  SUB     #100.,BUFTBL+4 ;SAVE ROOM FOR THE 'ABS' LOADER
1282 004574 023727 001256 100000  CMP     LSTAD,#100000 ;16K ON THE SYSTEM ?
1283 004602 103406          BLO     3$         ;BR IF YES
1284 004604 105737 000041          TSTB   41          ;SEE WHO LOADED THE PROGRAM
1285 004610 001403          BEQ     3$         ;BR IF LOADED BY PAPER TAPE
1286 004612 162737 002570 001602  SUB     #1400.,BUFTBL+4 ;SUBTRACT 'XXDP' LOADER SIZE

```



```

1287 004620 005737 001364      3$:  TST    MAXDL      ;VALUE IN 'MAXDL' ?
1288 004624 001012                BNE    4$          ;BR IF VALUE IS
1289 004626 012737 013534 001364  MOV    #5980,MAXDL ;ASSUME FULL TRACK + 1 SEC MAXIMUM
1290 004634 023737 001364 001602  CMP    MAXDL,BUFTBL+4 ;IS THAT TOO LARGE ?
1291 004642 103403                BLO    4$          ;BR IF NOT
1292 004644 013737 001602 001364  MOV    BUFTBL+4,MAXDL ;USE MAX AVAIL MEMORY AS MAX BUFFER SIZE
1293 004652 013737 001602 053374  4$:  MOV    BUFTBL+4,PARLST+2 ;VALUE FOR THE PARAMETER TABLE
1294
1295 ;SEE IF THE OPERATOR WANTS TO CHANGE ANY PARAMETERS
1296
1297 004660 005737 000042      LKPAR: TST    @#42      ;'XXDP' CHAIN MODE OR 'ACT11' OPERATION ?
1298 004664 001022                BNE    SETVEC      ;BR IF YES
1299 004666 104401 053470                TYPE   ,ASKPAR     ;ASK FOR PARMETERS
1300 004672 104411                RDLIN                      ;READ THE ENTRY
1301 004674 012605                MOV    (SP)+,R5     ;ADDRESS OF ENTRY TO R5
1302 004676 122715 000131                CMPB  #'Y',(R5)    ;WAS ENTRY A 'Y' (YES)
1303 004702 001013                BNE    SETVEC      ;BR IF NOT 'Y'
1304
1305 004704 012703 053372      ENTPR: MOV    #PARLST,R3 ;PARAMETER TABLE ADDRESS
1306 004710 004737 026222                JSR    PC,PARENT   ;GET THE PARAMETER ENTRY
1307 004714 023727 001364 000004  CMP    MAXDL,#4    ;IS THE 'MAXDL' VALUE OK ?
1308 004722 103003                BHIS  SETVEC      ;BR IF IT IS
1309 004724 012737 000004 001364  MOV    #4,MAXDL   ;SET 'MAXDL' TO THE MINIMUM VALUE
1310
1311 ;DISPLAY DRIVE STATUS AND SET UP THE OTHER SYSTEM DEVICES THAT
1312 ; THE PROGRAM WILL USE
1313
1314 004732 004737 022342      SETVEC: JSR    PC,CKCLK ;START THE CLOCK
1315 004736 004737 033266                JSR    PC,RPINIT  ;INITIALIZE THE RPO4/5/6 DRIVER
1316 004742 012737 177777 033210  MOV    #-1,SAVEFG ;SET THE SAVE REGISTERS FLAG
1317 004750 062727 177777 000000  ADD    #-1,#0     ;CHECK FOR FIRST START
1318 004756 103004                BCC    11$        ;BR IF FIRST START
1319 004760 032777 000004 174152  BIT    #SW02,@SWR ;TYPEOUT THE DRIVE STATUS TABLE ?
1320 004766 001075                BNE    10$        ;BR IF NOT
1321 004770 012737 000340 177776  11$:  MOV    #PR7,PS    ;SET PRIORITY TO 7
1322 004776 005004                CLR    R4         ;DRIVE TABLE POINTER
1323 005000 104401 001165                TYPE  ,%CR-LF     ;CR-LF
1324 005004 104401 052262                TYPE  ,%SYSTAT   ;TYPE STATUS HEADING
1325
1326 005010 010446                1$:  MOV    R4,-(SP)   ;;SAVE R4 FOR TYPEOUT
1327
1328 005012 104403                TYPOS ;TYPE DRIVE NUMBER
1329 005014 002                .BYTE 2          ;GO TYPE--OCTAL ASCII
1330 005015 000                .BYTE 0          ;TYPE 2 DIGIT(S)
1331 005016 104401 052071                TYPE  ,LIN4SP    ;SUPPRESS LEADING ZEROS
1332 005022 105764 033122                TSTB  DRVSTA(R4) ;SPACES
1333 005026 100416                BMI   4$         ;CHECK DRIVE'S STATUS
1334 005030 001020                BNE  5$         ;BR IF UNSAFE
1335 005032 105764 033132                TSTB  DRVSTYP(R4);BR IF ONLINE
1336 005036 001404                BEQ   2$         ;SEE IF OFFLINE OR NONEXISTENT
1337 005040 100006                BPL  3$         ;BR IF NONEXISTENT
1338 005042 104401 052175                TYPE  ,NOTRP     ;BR IF OFFLINE
1339 005046 000440                BR    9$         ;DRIVE NOT AN RPO4/5/6
1340 005050 104401 052216                2$:  TYPE  ,NOTPRS   ;CHECK NEXT DRIVE
;DRIVE NOT PRESENT
    
```



```

1341 005054 000435          BR      9$      ;CHECK NEXT DRIVE
1342 005056 104401 052104  3$:     TYPE    ,UNTOFF ;DRIVE OFFLINE
1343 005062 000405          BR      6$      ;PRINT DRIVE TYPE
1344 005064 104401 052252  4$:     TYPE    ,NOTSAF ;DRIVE UNSAFE
1345 005070 000402          BR      6$      ;PRINT DRIVE TYPE
1346 005072 104401 052115  5$:     TYPE    ,UNTON  ;DRIVE ONLINE
1347 005076 104401 052073  6$:     TYPE    ,LINSF  ;SPACES
1348 005102 012737 052302 005146  MOV     #RPO4B,8$ ;ADDRESS OF RPO4 MESSAGE
1349 005110 132764 000001 033132  BITB   #BIT00,DRV TYP(R4) ;RPO4 ?
1350 005116 001012          BNE     7$      ;BR IF YES
1351 005120 012737 052307 005146  MOV     #RPO5,8$ ;ADDRESS OF RPO5 MESSAGE
1352 005126 132764 000002 033132  BITB   #BIT01,DRV TYP(R4) ;RPO5 ?
1353 005134 001003          BNE     7$      ;BR IF YES
1354 005136 012737 052314 005146  MOV     #RPO6,8$ ;ADDRESS OF RPO6 MESSAGE
1355 005144 104401          7$:     TYPE    ;TYPE THE DRIVE TYPE MESSAGE
1356 005146 000000          8$:     .WORD   0      ;MESSAGE ADDRESS HERE
1357 005150 104401 001165  9$:     TYPE    ,SCRLF  ;CR-LF
1358 005154 005204          INC     R4      ;INCREMENT DRIVE NUMBER/TABLE POINTER
1359 005156 020427 000010          CMP     R4,#8.  ;FINISHED ?
1360 005162 001312          BNE     1$      ;BR IF NOT
1361 005164 104401 001165  10$:    TYPE    ,SCRLF ;CR-LF
1362 005170 005037 177776          CLR     PS      ;SET PRIORITY BACK TO '0'
1363 005174 000137 005200          JMP     MONTR   ;CHECK FOR 'XXDP' OR 'ACT11' MONITOR
1364
1365 ;SETUP IF 'XXDP' OR 'ACT11' OPERATION
1366
1367 005200 005737 000042  MONTR:  TST     42      ;'XXDP' CHAIN MODE OR 'ACT11' AUTO ACCEPT
1368 005204 001402          BEQ     1$      ;BR IF NEITHER
1369 005206 004737 024224          JSR     PC,ASGN2 ;ASSIGN DRIVES
1370 005212 005227 177777  1$:     INC     #-1   ;FIRST START ?
1371 005216 001011          BNE     2$      ;BR IF NOT
1372 005220 105737 000041          TSTB   J#41    ;LOADED FROM PAPER TAPE ?
1373 005224 001406          BEQ     2$      ;BR IF YES
1374 005226 023727 001256 100000  CMP     LSTAD,#100000 ;MORE THAN 16K ON THE SYSTEM ?
1375 005234 103002          BHS     2$      ;BR IF YES
1376 005236 104401 055553          TYPE    ,NOLOAD ;TELL THE OPERATOR THAT THE 'XXDP' LOADER
1377 ;WILL BE OVERWRITTEN
1378 005242 004737 030000  2$:     JSR     PC,$TKINT ;INITIALIZE THE KEYBOARD INTERRUPT HANDLER
1379 005246 104401 053257          TYPE    ,INTDON ;TYPE 'INITIALIZE COMPLETE'
1380 005252 000137 005452          JMP     MAIN1   ;START THE PROGRAM
1381
1382 ;'XXDP' OR 'ACT11' END OF TEST ROUTINE
1383
1384 005256 013700 000042  $GET42: MOV     42,RO   ;MONITOR ADDRESS
1385 005262 001002          BNE     1$      ;BR IF MONITOR
1386 005264 000137 005452          JMP     MAIN1   ;NONE, CONTINUE
1387 005270 022700 005310  1$:     CMP     $SENDAD,RO ;IS MONITOR 'ACT11' ?
1388 005274 001005          BNE     $SENDAD ;BR IF NOT
1389 005276 022760 177777 000002  CMP     #-1,2(RO) ;LAST 'ACT11' PASS ?
1390 005304 001005          BNE     $DOAGN  ;NO, MAKE ANOTHER PASS
1391 005306 000005          RESET ;CLEAR EVERYTHING
1392 005310 004710          $SENDAD: JSR    PC,(RO) ;GO TO THE MONITOR
1393 005312 000240          NOP ;SAVE ROOM
1394 005314 000240          NOP ;FOR
    
```



```

1399 005316 000240          NUP          :ACT11
1399 005320 000137 004066 SDOAGN: JMP   START      :START AGAIN
1399
1399
1399
1400
1401
1402
1403
1404 005324 012703 000010 MAIN:  MOV    #8.,R3      :DRIVE COUNTER
1405 005330 012705 001444      MOV    #DUNIT,R5      :ADDRESS OF 'DROP DRIVE' TABLE
1406 005334 005715          1$:   TST    (R5)        :SEE IF ENTRY AT PRESENT POSITION
1407 005336 001011          BNE    3$             :BR IF THERE IS ONE
1408 005340 062705 000002      2$:   ADD    #2,R5      :INCREMENT TO NEXT TABLE POSITION
1409 005344 005303          DEC    R3            :DECREMENT DRIVE COUNTER
1410 005346 001372          BNE    1$            :BR IF MORE TO CHECK
1411 005350 005737 001442      TST    ASNLST        :ANY DRIVES ACTIVE ?
1412 005354 001036          BNE    MAIN1         :BR IF YES
1413 005356 000137 005256      JMP    $GET42        :CHECK FOR MONITOR RETURN
1414 005362 012701 001510      3$:   MOV    #AVAIL,R1  :ADDRESS OF 'AVAILABLE DRIVES' TABLE
1415 005366 005711          4$:   TST    (R1)        :SEE IF AT END OF TABLE
1416 005370 001405          BEQ    5$            :BR IF AT END: GO CHECK 'WAIT' TABLE
1417 005372 021115          CMP    (R1),(R5)     :IS DRIVE IN 'AVAIL' THE ONE TO BE DROPPED
1418 005374 001414          BEQ    7$            :BR IF YES
1419 005376 062701 000002      ADD    #2,R1        :INCREMENT 'AVAIL' TABLE ADDRESS
1420 005402 000771          BR     4$            :CONTINUE LOOKING
1421 005404 012701 001532      5$:   MOV    #WAIT,R1   :MOVE THE ADDRESS OF THE BUFFER WAIT TABLE
1422 005410 005711          6$:   TST    (R1)        :AT THE END OF THE 'WAIT' TABLE ?
1423 005412 001752          BEQ    2$            :BR IF YES: SEE IF ANY MORE 'DROP' REQUESTS
1424 005414 021115          CMP    (R1),(R5)     :DRIVE IN THE 'WAIT' TABLE ?
1425 005416 001403          BEQ    7$            :BR IF IT IS
1426 005420 062701 000002      ADD    #2,R1        :INCREMENT 'WAIT' TABLE ADDRESS
1427 005424 000771          BR     6$            :CONTINUE LOOK THROUGH THE 'WAIT' TABLE
1428 005426 011100          7$:   MOV    (R1),R0     :PUT THE DRIVE'S BLOCK ADDRESS IN R0
1429 005430 104401 052566      TYPE   DEASSG        :TYPE 'DRIVE DEASSIGNED'
1430 005434 004737 022626      JSR    PC,TYPEST     :TYPE THE DRIVE'S PERFORMANCE SUMMARY
1431 005440 005015          CLR    (R5)         :CLEAR THE 'DROP DRIVE' TABLE ENTRY
1432 005442 005011          CLR    (R1)         :REMOVE THE DRIVE FROM THE 'AVAIL' OR 'WAIT' TABLE
1433 005444 004737 017460      JSR    PC,COMPRES    :COMPRESS THE RESPECTIVE TABLE
1434 005450 000733          BR     2$            :SEE IF ANY MORE DRIVES
1435
1436
1437
1438 005452 012703 000010 MAIN1: MOV    #8.,R3      :DRIVE COUNT
1439 005456 005002          CLR    R2           :'AVAIL' INDEX
1440 005460 005004          CLR    R4           :ASSIGN LIST INDEX
1441 005462 005005          CLR    R5           :NEW DRIVE INDEX
1442 005464 005765 001466      1$:   TST    NEWUNT(R5) :NEW DRIVE IN THIS POSITION
1443 005470 001006          BNE    3$            :BR IF THERE IS
1444 005472 062705 000002      2$:   ADD    #2,R5      :INCREMENT R5
1445 005476 005204          INC    R4           :INCREMENT ASSIGN INDEX
1446 005500 005303          DEC    R3           :DECREMENT DRIVE COUNT
1447 005502 001370          BNE    1$            :BR IF MORE DRIVES
1448 005504 000432          BR     MAIN2        :START OPERATIONS FOR THE AVAILABLE DRIVES

```



```

1449 005506 104401 052076      3$:  TYPE      UNTMSG      ;'DRIVE'
1450 005512 010446                MOV      R4,-(SP)  ;SAVE R4 FOR TYPEOUT
1451                ;TYPE DRIVE NUMBER
1452                ;GO TYPE--OCTAL ASCII
1453 005514 104403                TYPOS
1454 005516      002                .BYTE 2          ;TYPE 2 DIGIT(S)
1455 005517      000                .BYTE 0          ;SUPPRESS LEADING ZEROS
1456 005520 104401 052636                TYPE      ASGND   ;ASSIGNED
1457 005524 005762 001510      4$:  TST      AVAIL(R2) ;AT END OF AVAILABLE TABLE
1458 005530 001403                BEQ      5$      ;BR IF YES
1459 005532 062702 000002                ADD      #2,R2   ;INCREMENT AVAILABLE TABLE INDEX
1460 005536 000772                BR      4$      ;CONTINUE LOOKING FOR END OF TABLE
1461 005540 016562 001466 001510 5$:  MOV      NEWUNT(R5),AVAIL(R2) ;MOVE ADDR OF DRIVE INTO AVAIL LST
1462 005546 005065 001466                CLR      NEWUNT(R5) ;TAKE DRIVE OUT OF NEW DRIVE TABLE
1463 005552 156437 033236 001442  BISS     ATABIT(R4),ASNLST ;SET DRIVE ASSIGNED INDICATOR
1464 005560 016200 001510                MOV      AVAIL(R2),R0 ;PUT STARTING ADDRESS OF BLOCK IN R0
1465 005564 062702 000002                ADD      #2,R2   ;INCREMENT AVAILABLE TABLE POINTER
1466 005570 000740                BR      2$      ;LOOK FOR MORE DRIVES

;GET PARAMETERS, BUFFER SPACE, AND START ORDERS FOR DRIVES IN
;THE 'AVAILABLE' QUEUE

1470 005572 005737 001532  MAIN2:  TST      WAIT      ;OUTSTANDING BUFFER REQUESTS
1471 005576 001113                BNE     MAIN3     ;BR IF THERE ARE
1472 005600 005002                CLR      R2       ;CLEAR DRIVE TABLE POINTER
1473 005602 005762 001510  1$:  TST      AVAIL(R2) ;ANY DRIVES WAITING FOR PARAMETERS
1474 005606 001551                BEQ     IDLE      ;BRANCH IF NONE
1475 005610 016200 001510  MOV      AVAIL(R2),R0 ;CONTROL BLOCK ADDR IN R0
1476 005614 005760 000104  TST      $NEXT(R0)  ;PARAMETERS BEEN SELECTED ?
1477 005620 001021                BNE     6$      ;BR IF THEY HAVE
1478 005622 105760 000026  TSTB     $PACK(R0) ;'R' OR 'W' COMMAND FOR THE DRIVE ?
1479 005626 001403                BEQ     2$      ;BR IF NOT
1480 005630 004737 017476  JSR      PC,WRTPK  ;GET DATA PACK PARAMETERS
1481 005634 000415                BR      7$      ;GET THE BUFFER
1482 005636 012701 001554  2$:  MOV      #PAR0,R1  ;ADDRESS OF THE PARAMETER QUEUE
1483 005642 020011  3$:  CMP      R0,(R1)  ;IS CURRENT DRIVE IN THE QUEUE ?
1484 005644 001403                BEQ     4$      ;BR IF IT IS
1485 005646 005721                TST     (R1)+    ;AT END OF THE QUEUE
1486 005650 001403                BEQ     5$      ;BR IF AT END
1487 005652 000773                BR      3$      ;CONTINUE LOOKING
1488 005654 004737 017460  4$:  JSR      PC,COMPRES ;COMPRESS THE TABLE
1489 005660 004737 016376  5$:  JSR      PC,SELPAR ;SELECT THE PARAMETERS
1490 005664 004737 017252  6$:  JSR      PC,GETPAR ;LOAD NEW PARAMETERS
1491 005670 005046  7$:  CLR      -(SP)   ;MAKE ROOM ON THE STACK FOR THE BUFFER ADDR
1492 005672 004737 015622  JSR      PC,GETBUF ;GET BUFFER
1493 005676 012660 000006  MOV      (SP)+,$BUF(R0) ;MOVE BUFFER ADDR TO DPB
1494 005702 001424                BEQ     8$      ;BR IF '0' ADDR (NO BUFFER)
1495 005704 004737 016172  JSR      PC,FILBUF ;FILL THE BUFFER
1496 005710 005060 000072  CLR      $FAIR(R0) ;CLEAR THE 'FAIRNESS' COUNT
1497 005714 004737 016320  JSR      PC,GODRIV ;PUT CURRENT DPB IN DRIVER
1498 005720 012705 001420  MOV      #ORDER0,R5 ;ADDRESS OF ORDER QUEUE IN R5
1499 005724 005725                TST     (R5)+    ;END OF QUEUE ?
1500 005726 001376                BNE     -2       ;BR IF NOT
1501 005730 010045                MOV     R0,-(R5) ;PUT BLOCK ADDRESS INTO QUEUE
1502 005732 105760 000026  TSTB     $PACK(R0) ;'R' OR 'W' COMMAND FOR DRIVE ?

```



```

1503 005736 001025 BNE 10$ ;BR IF EITHER
1504 005740 012705 001554 MOV #PARQ,R5 ;PUT BLOCK INTO THE PARAMETER QUEUE
1505 005744 005725 TST (R5)+ ;FIND THE END OF THE QUEUE
1506 005746 001376 BNE -2 ;BR IF NOT AT END OF QUEUE
1507 005750 010045 MOV RO,-(R5) ;PUT BLOCK ADDRESS INTO THE QUEUE
1508 005752 000417 BR 10$ ;CONTINUE LOOKING
1509 005754 026037 000072 001254 8$: CMP $FAIR(RO),FAIRNS ;ENTRY BEEN IN THE QUEUE LONG ENOUGH ?
1510 005762 001405 BEQ 9$ ;BR IF YES
1511 005764 005260 000072 INC $FAIR(RO) ;INCREMENT THE ENTRY COUNT
1512 005770 062702 000002 ADD #2,R2 ;INCREMENT THE POINTER
1513 005774 000702 BR 1$ ;LOOK FOR SOME MORE DRIVES
1514 005776 012705 001532 9$: MOV #WAIT,R5 ;'WAIT' QUEUE ADDRESS
1515 006002 005725 TST (R5)+ ;LOOK FOR AN OPENING
1516 006004 001376 BNE -2 ;BR IF NONE YET
1517 006006 016245 001510 MOV AVAIL(R2),-(R5) ;MOVE DRIVE'S BLOCK ADDRESS TO QUEUE
1518 006012 012701 001510 10$: MOV #AVAIL,R1 ;'AVAILABLE' TABLE ADDRESS
1519 006016 060201 ADD R2,R1 ;FORM ADDRESS OF LAST ENTRY
1520 006020 004737 017460 JSR PC,COMPRES ;COMPRESS THE TABLE
1521 006024 000666 BR 1$ ;CONTINUE LOOKING
1522
1523 ;GET BUFFER ASSIGNMENTS FOR DRIVES IN THE 'BUFFER WAIT' QUEUE
1524
1525 006026 013700 001532 MAIN3: MOV WAIT,RO ;MOVE THE 'WAIT' ENTRY TO RO
1526 006032 005046 CLR -(SP) ;MAKE ROOM ON THE STACK FOR THE BUFFER ADDR
1527 006034 004737 015622 JSR PC,GETBUF ;TRY TO GET A BUFFER
1528 006040 012660 000006 MOV (SP)+,$BUF(RO) ;MOVE THE BUFFER ADDR TO THE DPB
1529 006044 001002 BNE 1$ ;BR IF A BUFFER WAS ASSIGNED
1530 006046 000137 006132 JMP IDLE ;NO BUFFER AVAILABLE YET
1531 006052 004737 016172 1$: JSR PC,FILBUF ;FILL THE BUFFER
1532 006056 004737 016320 JSR PC,GODRIV ;PUT THE ENTRY IN THE DRIVER
1533 006062 005060 000072 CLR $FAIR(RO) ;CLEAR THE 'FAIRNESS' COUNT
1534 006066 012705 001420 MOV #ORDERQ,R5 ;ADDRESS OF ORDER QUEUE IN R5
1535 006072 005725 TST (R5)+ ;AT END OF THE QUEUE
1536 006074 001376 BNE -2 ;BR IF NOT
1537 006076 010045 MOV RO,-(R5) ;PUT BLOCK ADDRESS IN QUEUE
1538 006100 105760 000026 TSTB $PACK(RO) ;'R' OR 'W' COMMAND FOR DRIVE ?
1539 006104 001005 BNE 2$ ;BR IF YES, DON'T PUT BLOCK INTO 'PARQ'
1540 006106 012705 001554 MOV #PARQ,R5 ;FIND THE END OF THE PARAMETER QUEUE
1541 006112 005725 TST (R5)+ ;OPEN SLOT IN THE QUEUE ?
1542 006114 001376 BNE -2 ;BR IF NOT
1543 006116 010045 MOV RO,-(R5) ;PUT BLOCK ADDRESS INTO THE QUEUE
1544 006120 012701 001532 2$: MOV #WAIT,R1 ;ADDRESS OF TABLE TO COMPRESS
1545 006124 004737 017460 JSR PC,COMPRES ;COMPRESS THE WAIT TABLE
1546 006130 000620 BR MAIN2 ;LOOK FOR MORE ENTRIES
1547
1548 ;WAIT FOR AN ORDER TO FINISH
1549
1550 006132 012701 001420 IDLE: MOV #ORDERQ,R1 ;ADDRESS OF THE ORDER QUEUE IN R1
1551 006136 012100 1$: MOV (R1)+,RO ;PUT BLOCK ADDRESS INTO RO
1552 006140 001433 BEQ IDLE1 ;BR IF END OF QUEUE
1553 006142 005760 000016 TST $STATUS(RO) ;SEE IF DRIVE FINISHED
1554 006146 001773 BNE 1$ ;BR IF DRIVE NOT FINISHED
1555 006150 162701 000002 SUB #2,R1 ;CORRECT THE QUEUE POINTER
1556 006154 010146 MOV R1,-(SP) ;SAVE THE QUEUE ADDRESS

```



```

1557 006156 004737 015456      JSR      PC,STATIS      ;ACCUMULATE STATISTICS FOR DRIVE IN RO
1558 006162 000240              NOP                    ;DEBUGGING AID
1559 006164 004737 006470      JSR      PC,PROCES      ;PROCESS END OF ORDER
1560 006170 005037 001264      CLR      BADSEC        ;CLEAR THE BAD TRK/SEC ERROR INDICATOR
1561 006174 004737 026454      JSR      PC,ABNRML     ;SEE IF ANY DRIVES HAVE TOO MANY ERRORS
1562 006200 004737 026502      JSR      PC,EOP        ;SEE IF ANY DRIVE HAS XFERED 3X1019 BITS
1563 006204 012601              MOV      (SP)+,R1      ;RESTORE THE ORDER TABLE INDEX
1564 006206 012705 001510      MOV      #AVAIL,R5     ;FIND THE END OF THE 'AVAILABLE' TABLE
1565 006212 005725          2$:      TST      (R5)+        ;END OF THE TABLE ?
1566 006214 001376              BNE     2$            ;BR IF NOT AT END OF LIST
1567 006216 011145              MOV      (R1),-(R5)    ;MOVE THE BLOCK ADDRESS INTO THE TABLE
1568 006220 004737 017460      JSR      PC,CMPRES     ;COMPRESS THE ORDER QUEUE
1569 006224 004737 015756      JSR      PC,RELBUF     ;RESTORE BUFFER
1570 006230 005737 001262      IDLE1:  TST      CFLAG     ;'CONTROL C' FLAG ENTERED ?
1571 006234 001403              BEQ     1$            ;BR IF IT WAS
1572 006236 004737 023640      JSR      PC,KSR        ;SERVICE THE KEYBOARD
1573 006242 000733              BR      IDLE          ;SYSTEM WAS BUSY
1574 006244 032777 000004 172666 1$:      BIT      #SW02,JSWR    ;TYPE PERFORMANCE SUMMARY
1575 006252 001007              BNE     2$            ;BR IF NOT
1576 006254 005737 001214      TST      STATIN       ;TIME TO TYPE THE PERFORMANCE SUMMARY ?
1577 006260 001404              BEQ     2$            ;BR IF NOT
1578 006262 005037 001214      CLR      STATIN       ;CLEAR THE INDICATOR
1579 006266 004737 022544      JSR      PC,STATPR    ;TYPE THE SUMMARY
1580 006272 005737 001554          2$:      TST      PARQ        ;ENTRY IN THE PARAMETER QUEUE ?
1581 006276 001410              BEQ     3$            ;BR IF NOT
1582 006300 013700 001554      MOV      PARQ,RO      ;PUT THE BLOCK ADDRESS INTO RO
1583 006304 004737 016376      JSR      PC,SELPAR    ;GET THE PARAMETERS FOR NEXT OPERATION
1584 006310 012701 001554      MOV      #PARQ,R1     ;SETUP TO COMPRESS THE TABLE
1585 006314 004737 017460      JSR      PC,CMPRES    ;COMPRESS THE PARAMETER QUEUE
1586 006320 000137 005324          3$:      JMP      MAIN        ;CONTINUE THE LOOP
1587
1588          ;SETUP TO REFORMAT AN ERROR SECTOR
1589
1590 006324 032777 000001 172606  REFMT:  BIT      #SW0,JSWR    ;READ ONLY SWITCH SET ?
1591 006332 001055              BNE     REFMTX        ;BR IF IT IS
1592 006334 032777 000200 172576  BIT      #SW7,JSWR    ;SWITCH 7 SET ?
1593 006342 001051              BNE     REFMTX        ;BR IF IT IS
1594 006344 005737 001376      TST      FORMAT       ;WRITE HEADER & DATA ORDERS ALLOWED ?
1595 006350 001446              BEQ     REFMTX        ;BR IF NOT
1596 006352 016060 000272 000100  MOV      $RPCC(RO), $NICYL(RO) ;USE PRESENT CYLINDER
1597 006360 004737 022244      JSR      PC,READDR    ;GET CORRECTED SECTOR-TRACK ADDRESSES
1598 006364 112660 000077      MOVVB   (SP)+,$NTRK(RO) ;TRACK ADDR TO DPB
1599 006370 112660 000076      MOVVB   (SP)+,$NSEC(RO) ;SECTOR ADDR TO DPB
1600 006374 012760 000404 000102  MOV      #260,$NWRDL(RO) ;WORD COUNT FOR FORMAT
1601 006402 023727 001364 000404  CMP      MAXDL,#260.  ;CAN A FULL SECTOR BE WRITTEN ?
1602 006410 103003              BHIS   1$            ;BR IF IT CAN
1603 006412 013760 001364 000102  MOV      MAXDL,$NWRDL(RO) ;PUT TRANSFER SIZE INTO THE DPB
1604 006420 112760 000003 000074 1$:      MOVVB   #3,$NCODE(RO) ;COMMAND CODE
1605 006426 004737 017224      JSR      PC,GETPAT    ;GET A PATTERN
1606 006432 110560 000075      MOVVB   R5,$NPATC(RO) ;PATTERN CODE TO CONTROL BLOCK
1607 006436 012760 177777 000104  MOV      #-1,$NEXT(RO) ;SET PARAMETERS SELECTED INDICATOR
1608 006444 012701 001554      MOV      #PARQ,R1     ;SET UP TO SEE IF BLOCK IN THE PARAMETER QUEUE
1609 006450 005711          2$:      TST      (R1)        ;SEE IF AT END OF TABLE
1610 006452 001405              BEQ     REFMTX        ;BR IF AT END

```



```

1611 006454 020021          CMP      RO,(R1)+      ;SEE IF BLOCK AT PRESENT POSITION
1612 006456 001374          BNE      2$           ;BR IF NOT
1613 006460 005041          CLR      -(R1)       ;CLEAR THE ENTRY
1614 006462 004737 017460      JSR      PC,CMPRES   ;COMPRESS THE TABLE
1615 006466 000207          REFMTX: RTS      PC      ;RETURN
1616
1617                          ;PROCESS THE ORDER TERMINATION
1618
1619 006470 111037 001246      PROCES: MOVB      (RO),UNIT      ;DRIVE NUMBER FOR ANY ERROR MESSAGES
1620 006474 005760 000016      TST      $STATUS(RO)      ;SEE IF DRIVER SIGNALLED AN ERROR
1621 006500 100427          BMI      ERPROC        ;BR IF ERROR
1622 006502 032760 100000 000234  BIT      #BIT15,$RPCS1(RO) ;SEE IF 'SC' SET
1623 006510 001410          BEQ      1$           ;BR IF NOT SET
1624 006512 032760 040000 000234  BIT      #BIT14,$RPCS1(RO) ;SEE IF 'TRE' SET
1625 006520 001017          BNE      ERPROC        ;BR IF SET
1626 006522 032760 040000 000246  BIT      #BIT14,$RPDS1(RO) ;SEE IF 'ERR' SET
1627 006530 001013          BNE      ERPROC        ;BR IF SET
1628 006532 004737 012720      1$:      JSR      PC,CKERR      ;NO ERROR, CHECK ERROR BITS ANYWAY
1629 006536 004737 013020      JSR      PC,CKBUS      ;NO ERROR, CHECK BUS ADDR & WC
1630 006542 032777 000002 172370  BIT      #SW01,$SWR      ;DATA COMPARE ?
1631 006550 001002          BNE      2$           ;BR IF NOT
1632 006552 004737 013104      JSR      PC,CMPAR      ;NO ERROR, COMPARE DATA
1633 006556 000207          2$:      RTS      PC      ;RETURN
1634
1635                          ;ORDER TERMINATED WITH AN ERROR - PROCESS THE ERROR
1636
1637 006560 032760 000200 000016  ERPROC: BIT      #BIT07,$STATUS(RO) ;DONE BIT SET ?
1638 006566 001402          BEQ      ERPRC1      ;BR IF ORDER DIDN'T COMPLETE NORMALLY
1639 006570 000137 007214      JMP      DONE         ;PROCESS ERROR WITH 'DONE' BIT SET
1640
1641                          ;PROCESS ORDER COMPLETION WITH 'ERROR' & 'DONE NOT' BITS
1642
1643 006574 032760 010000 000016  ERPRC1: BIT      #BIT12,$STATUS(RO) ;SEE IF DRIVE WAS UNSAFE
1644 006602 001025          BNE      PUNSAF      ;BR IF YES
1645 006604 032760 004000 000016  BIT      #BIT11,$STATUS(RO) ;PARITY ERROR OCCURRED
1646 006612 001055          BNE      UCPAR      ;BR IF IT DID
1647 006614 032760 002000 000016  BIT      #BIT10,$STATUS(RO) ;FATAL PARITY ERROR?
1648 006622 001056          BNE      FALPAR      ;BR IF THERE IS ONE
1649 006624 032760 001000 000016  BIT      #BIT09,$STATUS(RO) ;TIMEOUT?
1650 006632 001076          BNE      SWTIM      ;BR IF YES
1651 006634 032760 040002 000016  BIT      #BIT14!BIT01,$STATUS(RO) ;DRIVE WENT OFFLINE ?
1652 006642 001111          BNE      OFLIN      ;BR IF IT DID
1653 006644 032760 000004 000016  BIT      #BIT2,$STATUS(RO) ;PORT REQUEST TIME OUT ?
1654 006652 001141          BNE      PRTIM      ;BR IF IT DID
1655 006654 000207          RTS      PC      ;ERROR. RETURN
1656
1657                          ;DRIVE IS PERSISTENTLY UNSAFE
1658
1659 006656 104401 001165      PUNSAF: TYPE      , $CRLF      ;CR-LF
1660 006662 104401 045326      TYPE      , EM12      ;'DRIVE UNSAFE' MESSAGE
1661 006666 104401 052611      TYPE      , DRNUM      ;DRIVE NUMBER
1662 006672 013746 001246      MOV      UNIT,-(SP)    ;SAVE UNIT FOR TYPEOUT
1663                          ;TYPE DRIVE NUMBER
1664 006676 104403          TYPOS      ;GO TYPE--OCTAL ASCII
    
```

```

1665 006700 002 .BYTE 2 ;;TYPE 2 DIGIT(S)
1666 006701 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
1667 006702 104401 001165 TYPE ,SCLF ;;CR-LF
1668 006706 004737 020130 JSR PC,LINE1 ;;PRINT LINE 1 OF ERROR MESSAGE
1669 006712 104414 045326 DISPLY EM12 ;;PERSISTENT DEVICE UNSAFE MESSAGE
1670 006716 004737 020174 JSR PC,LINE2 ;;PRINT LINE 2 OF ERROR MESSAGE
1671 006722 004737 020602 JSR PC,LINE3 ;;PRINT LINE 3 OF ERROR MESSAGE
1672 006726 004737 021256 JSR PC,LINE4 ;;PRINT LINE 4 OF THE ERROR MESSAGE
1673 006732 004737 023354 JSR PC,INCTOT ;;INCREMENT TOTAL ERROR COUNT
1674 006736 004737 021712 JSR PC,LINE7 ;;PRINT LINE 7 OF ERROR MESSAGE
1675 006742 000137 026376 JMP DROP ;;DROP THE DRIVE
1676
1677 ;UNCORRECTABLE MASSBUS PARITY ERROR OCCURRED
1678
1679 006746 104401 001165 UCPAR: TYPE ,SCLF ;;CR-LF
1680 006752 104401 045230 TYPE EM10 ;;'UNCORRECTABLE PARITY ERROR' MESSAGE
1681 006756 000404 BR FALPR1 ;;FINISH PROCESSING THE ERROR
1682
1683 ;'FATAL' MASSBUS PARITY ERROR OCCURRED
1684
1685 006760 104401 001165 FALPAR: TYPE ,SCLF ;;CR-LF
1686 006764 104401 045273 TYPE EM11 ;;'FATAL PARITY ERROR' MESSAGE
1687 006770 104401 052611 FALPR1: TYPE ,DRNUM ;;DRIVE NUMBER
1688 006774 013746 001246 MOV UNIT,-(SP) ;;SAVE UNIT FOR TYPEOUT
1689 ;;TYPE DRIVE NUMBER
1690 007000 104403 TYPOS ;;GO TYPE--OCTAL ASCII
1691 007002 002 .BYTE 2 ;;TYPE 2 DIGIT(S)
1692 007003 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
1693 007004 104401 001165 TYPE ,SCLF ;;CR-LF
1694 007010 004737 023354 JSR PC,INCTOT ;;INCREMENT TOTAL ERROR COUNT
1695 007014 032777 100000 172116 BIT #SW15,SWR ;;HALT ON ERROR ?
1696 007022 001401 BEQ IS ;;BR IF NOT
1697 007024 000000 HALT IS ;;ERROR HALT
1698 007026 000207 IS: RTS PC
1699
1700 ;SOFTWARE TIMEOUT OCCURRED
1701
1702 007030 004737 020130 SWTIM: JSR PC,LINE1 ;;PRINT LINE 1 OF ERROR MESSAGE
1703 007034 104414 045357 DISPLY EM13 ;;PRINT THE TIME OUT MESSAGE
1704 007040 004737 020174 JSR PC,LINE2 ;;PRINT LINE 2 OF ERROR MESSAGE
1705 007044 004737 020602 JSR PC,LINE3 ;;PRINT LINE 3 OF ERROR MESSAGE
1706 007050 004737 021256 JSR PC,LINE4 ;;PRINT LINE 4 OF ERROR MESSAGE
1707 007054 004737 023354 JSR PC,INCTOT ;;INCREMENT TOTAL ERROR COUNT
1708 007060 004737 021712 JSR PC,LINE7 ;;PRINT LINE 7 OF ERROR MESSAGE
1709 007064 000207 RTS PC ;;RETURN
1710
1711 ;DRIVE WENT OFFLINE
1712
1713 007066 104401 001165 OFLIN: TYPE ,SCLF ;;CR-LF
1714 007072 104401 045431 TYPE EM14 ;;'DRIVE WENT OFFLINE' MESSAGE
1715 007076 104401 052611 TYPE ,DRNUM ;;DRIVE NUMBER
1716 007102 013746 001246 MOV UNIT,-(SP) ;;SAVE UNIT FOR TYPEOUT
1717 ;;TYPE DRIVE NUMBER
1718 007106 104403 TYPOS ;;GO TYPE--OCTAL ASCII

```



```

1719 007110 002 .BYTE 2 ;:TYPE 2 DIGIT(S)
1720 007111 000 .BYTE 0 ;:SUPPRESS LEADING ZEROS
1721 007112 104401 001165 TYPE $CRLF ;:CR-LF
1722 007116 004737 020130 JSR PC,LINE1 ;:PRINT LINE 1 OF THE ERROR MESSAGE
1723 007122 104414 045431 DISPLY EM14 ;:PRINT OFFLINE MESSAGE
1724 007126 004737 020174 JSR PC,LINE2 ;:PRINT LINE 2 OF THE ERROR MESSAGE
1725 007132 004737 020602 JSR PC,LINE3 ;:PRINT LINE 3 OF THE ERROR MESSAGE
1726 007136 004737 021256 JSR PC,LINE4 ;:PRINT LINE 4 OF THE ERROR MESSAGE
1727 007142 004737 023354 JSR PC,INCTOT ;:INCREMENT TOTAL ERROR COUNT
1728 007146 004737 021712 JSR PC,LINE7 ;:PRINT LINE 7 OF THE ERROR MESSAGE
1729 007152 000137 026376 JMP DROP ;:DROP THE DRIVE
1730
1731 ;:PORT REQUEST TIMEOUT ERROR
1732
1733 007156 004737 020130 PRTIM: JSR PC,LINE1 ;:TYPE LINE 1 OF THE ERROR MESSAGE
1734 007162 104414 045454 DISPLY EM15 ;:PRINT PORT TIME OUT MESSAGE
1735 007166 004737 020174 JSR PC,LINE2 ;:TYPE LINE 2 OF THE ERROR MESSAGE
1736 007172 004737 020602 JSR PC,LINE3 ;:TYPE LINE 3 OF THE ERROR MESSAGE
1737 007176 004737 021256 JSR PC,LINE4 ;:TYPE LINE 4 OF THE ERROR MESSAGE
1738 007202 004737 023354 JSR PC,INCTOT ;:INCREMENT TOTAL ERROR COUNT
1739 007206 004737 021712 JSR PC,LINE7 ;:TYPE LINE 7 OF THE ERROR MESSAGE
1740 007212 000207 RTS ;:RETURN
1741
1742 ;:PROCESS ORDER COMPLETION WITH 'ERROR' & 'DONE' BITS SET
1743
1744 007214 032760 000030 000016 DONE: BIT #BIT04:BIT03,$STATUS(RO) ;:UNSAFE OCCURRED
1745 007222 001402 .+6 ;:BR IF NOT
1746 007224 000137 012416 JMP UNSAF ;:REPORT UNSAFE
1747 007230 032760 040000 000244 BIT #BIT14,$RPCS2(RO) ;:IS 'WCE' SET ?
1748 007236 001006 BNE 1$ ;:BR IF SET
1749 007240 032760 040000 000246 BIT #BIT14,$RPDS1(RO) ;:CHECK 'ERR'
1750 007246 001002 BNE 1$ ;:BR IF SET
1751 007250 000137 012162 JMP TRFER ;:PROCESS 'TRE'
1752 007254 032760 000400 000250 1$: BIT #BIT08,$RPER1(RO) ;:'HCRC' SET?
1753 007262 001402 .+6 ;:BR IF NOT
1754 007264 000137 010640 JMP HCRCER ;:PROCESS 'HCRC'
1755 007270 032760 000020 000250 BIT #BIT04,$RPER1(RO) ;:'FMT' SET?
1756 007276 001402 .+6 ;:BR IF NOT SET
1757 007300 000137 011022 JMP CKFMT ;:CHECK FORMAT ERROR
1758 007304 032760 000200 000250 BIT #BIT07,$RPER1(RO) ;:'HCE' SET?
1759 007312 001402 .+6 ;:BR IF NOT SET
1760 007314 000137 011216 JMP CKHCE ;:CHECK 'HCE' ERROR
1761 007320 032760 020000 000250 BIT #BIT13,$RPER1(RO) ;:'OPI' SET?
1762 007326 001402 .+6 ;:BR IF NOT SET
1763 007330 000137 011516 JMP OPIER ;:REPORT 'OPI'
1764 007334 032760 000010 000250 BIT #BIT3,$RPER1(RO) ;:'PAR' SET?
1765 007342 001402 .+6 ;:BR IF NOT SET
1766 007344 000137 011650 JMP PARER ;:REPORT 'PAR'
1767 007350 032760 000040 000250 BIT #BIT5,$RPER1(RO) ;:'WCF' SET?
1768 007356 001402 .+6 ;:BR IF NOT SET
1769 007360 000137 012320 JMP WCFER ;:REPORT 'WCF'
1770 007364 032760 002000 000250 BIT #BIT10,$RPER1(RO) ;:'IAE' SET?
1771 007372 001402 .+6 ;:BR IF NOT SET
1772 007374 000137 011742 JMP IAEER ;:REPORT 'IAE'

```

```

1773 007400 032760 004000 000250 BIT #BIT11,$RPER1(RO) ;'WLE' SET?
1774 007406 001402 BEQ .+6 ;BR IF NOT SET
1775 007410 000137 011774 JMP WLEER ;REPORT 'WLE'
1776 007414 032760 001000 000250 BIT #BIT9,$RPER1(RO) ;'AOE' SET?
1777 007422 001405 BEQ 2$ ;BR IF NOT SET
1778 007424 032760 002000 000246 BIT #BIT10,$RPDS1(RO) ;'LST' SET?
1779 007432 001401 BEQ 2$ ;BR IF NOT SET
1780 007434 000207 RTS PC ;'AOE' & 'LST' SET, EXIT
1781 007436 032760 010000 000250 2$: BIT #BIT12,$RPER1(RO) ;SEE IF 'DTE' SET
1782 007444 001402 BEQ .+6 ;BR IF NOT
1783 007446 000137 011626 JMP DTEER ;REPORT 'DTE' ERROR
1784 007452 032760 040000 000244 BIT #BIT14,$RPCS2(RO) ;SEE IF 'WCK' SET
1785 007460 001402 BEQ .+6 ;BR IF NOT SET
1786 007462 000137 010312 JMP WCKER ;REPORT 'WCK'
1787 007466 005760 000250 TST $RPER1(RO) ;SEE IF 'DCK' SET
1788 007472 100002 BPL .+6 ;BR IF NOT
1789 007474 000137 007520 JMP DCKER ;PROCESS 'DCK'
1790 007500 032760 140000 000276 BIT #BIT15!BIT14,$RPER3(RO) ;'SKI' OR 'OCYL' SET
1791 007506 001402 BEQ .+6 ;BR IF NOT SET
1792 007510 000137 012262 JMP SKIER ;REPORT ERROR
1793 007514 000137 010770 JMP DRIVER ;REPORT DRIVE ERROR
1794
1795 ;PROCESS DATA ('DCK') CHECK ERROR
1796
1797 007520 022760 010042 000300 DCKER: CMP #10042,$RPEC1(RO) ;VALID POSITION COUNT ?
1798 007526 101406 BLOS 1$ ;BR IF NOT VALID
1799 007530 005760 000300 TST $RPEC1(RO) ;POSITION COUNT 0 ?
1800 007534 001403 BEQ 1$ ;BR IF 0'S
1801 007536 005760 000302 TST $RPEC2(RO) ;VALUE IN PATTERN REGISTER ?
1802 007542 001026 BNE 4$ ;BR IF YES
1803 007544 004737 020130 1$: JSR PC,LINE1 ;TYPE FIRST LINE OF ERROR MESSAGE
1804 007550 104414 047061 DISPLY EM45 ;TYPE 'ECC LOGIC ERROR'
1805 007554 004737 020174 JSR PC,LINE2 ;TYPE LINE 2 OF ERROR MESSAGE
1806 007560 004737 023354 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
1807 007564 012737 000003 001252 MOV #3,RETRY ;RETRY COUNT
1808 007572 004737 015330 JSR PC,$RETRY ;RETRY THE ORDER
1809 007576 000403 BR 2$ ;RETRY WAS NOT SUCCESSFUL
1810 007600 004737 021630 JSR PC,LINE6C ;TYPE LINE 6C OF ERROR MESSAGE
1811 007604 000402 BR 3$ ;FINISH THE ERROR REPORT
1812 007606 004737 021636 2$: JSR PC,LINE6D ;TYPE LINE 6D OF ERROR MESSAGE
1813 007612 004737 021712 3$: JSR PC,LINE7 ;TYPE LINE 7 OF ERROR MESSAGE
1814 007616 000402 BR 5$ ;EXIT
1815 007620 004737 017772 4$: JSR PC,SPOTCK ;SEE IF ERROR AT A BAD SPOT ON THE PACK
1816 007624 000207 5$: RTS PC ;IT IS, DON'T REPORT IT
1817 007626 126027 000024 000001 CMPB $CODE(RO),#1 ;IS ORDER A WRITE CHECK ?
1818 007634 101002 BHI 6$ ;BR IF NOT
1819 007636 000137 010312 JMP WCKER ;REPORT ERROR UNDER WRITE CHECK PROCESSING
1820 007642 004737 020130 6$: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
1821 007646 104414 045531 DISPLY EM21 ;DATA CHECK ERROR
1822 007652 004737 020174 DCKER1: JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
1823 007656 004737 020602 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
1824 007662 004737 021256 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
1825 007666 004737 014772 JSR PC,PRTBAD ;SEE IF BAD SECTOR TO BE PRINTED
1826 007672 012737 110100 001250 MOV #BIT15!BIT12!BIT06,MASK ;LOAD ERROR MASK

```

4.9

1827	007700	032760	010100	000250		BIT	#BIT12!BIT06,\$RPER1(RO)	:CHECK 'DTE' & 'ECH'
1828	007706	001003				BNE	1\$:BR IF SET
1829	007710	004737	021602			JSR	PC,LINE6	:PRINT LINE 6 OF ERROR MESSAGE
1830	007714	000541				BR	8\$:FINISH THE ERROR REPORT
1831	007716	012737	000020	001252	1\$:	MOV	#16.,RETRY	:RETRY COUNT
1832	007724	005001				CLR	R1	:R1 IS OFFSET CODE POINTER
1833	007726	032760	000002	000262		BIT	#BIT01,\$RPDT(RO)	:IS DRIVE AN RPO6 ?
1834	007734	001002				BNE	16\$:BR IF IT IS
1835	007736	012701	000007			MOV	#7,R1	:INCREMENT PAST RPO6 OFFSET CODES
1836	007742	004737	015756		16\$:	JSR	PC,RELBUF	:RELEASE THE BUFFER
1837	007746	004737	022244			JSR	PC,READDR	:GET THE ADDRESS OF THE ERROR SECTOR
1838	007752	112660	000011			MOVB	(SP)+,\$TRK(RO)	:TRACK ADDRESS OF ERROR SECTOR
1839	007756	112660	000010			MOVB	(SP)+,\$SEC(RO)	:SECTOR ADDRESS OF ERROR SECTOR
1840	007762	016060	000272	000012		MOV	\$RPCC(RO),\$CYL(RO)	:PRESENT CYLINDER
1841	007770	026060	000022	000020		CMP	\$SSEC(RO),\$WRDL(RO)	:SEE IF TRANSFER LENGTH LESS THAN 1 SECTOR
1842	007776	103010				BHIS	15\$:BR IF IT IS; USE PRESENT TRANSFER LENGTH
1843	010000	016060	000022	000020		MOV	\$SSEC(RO),\$WRDL(RO)	:CHANGE TRANSFER SIZE TO 1 SECTOR
1844	010006	016060	000020	000004		MOV	\$WRDL(RO),\$WRDM(RO)	:SETUP WORD COUNT FOR OPERATION
1845	010014	005460	000004			NEG	\$WRDM(RO)	:CHANGE COUNT TO 2'S COMP
1846	010020	005046			15\$:	CLR	-(SP)	:SPACE FOR NEW BUFFER ADDRESS
1847	010022	004737	015622			JSR	PC,GETBUF	:GET A BUFFER
1848	010026	012660	000006			MOV	(SP)+,\$BUF(RO)	:NEW BUFFER ADDRESS TO DPB
1849	010032	004737	016320		2\$:	JSR	PC,GODRIV	:RETRY
1850	010036	005760	000016		3\$:	TST	\$STATUS(RO)	:TEST FOR DONE
1851	010042	001775				BEQ	3\$:BR IF NOT DONE
1852	010044	100075				BPL	10\$:BR IF NOT ERROR
1853	010046	032760	000200	000016		BIT	#BIT7,\$STATUS(RO)	:SEE IF ORDER TERMINIATED NORMALLY
1854	010054	001006				BNE	14\$:BR IF NOT
1855	010056	004737	023354			JSR	PC,INCTOT	:INCREMENT TOTAL ERROR COUNT
1856	010062	104414	051246			DISPLY	,LIN8M	:DIFFERENT ERROR DURING RETRY'
1857	010066	000137	006574			JMP	ERPRC1	:SEE WHICH ERROR
1858	010072	033760	001250	000250	14\$:	BIT	MASK,\$RPER1(RO)	:LOOK AT CURRENT ERROR
1859	010100	001430				BEQ	5\$:BR IF DIFFERENT ERROR
1860	010102	032760	010100	000250		BIT	#BIT12!BIT6,\$RPER1(RO)	: 'ECH' OR 'DTE' STILL SET ?
1861	010110	001437				BEQ	7\$:BR IF NEITHER SET
1862	010112	105237	001253			INCB	RETRY+1	:INCREMENT RETRY COUNT
1863	010116	123737	001252	001253		CMPB	RETRY,RETRY+1	:DONE ?
1864	010124	001342				BNE	2\$:BR IF NOT
1865	010126	005201				INC	R1	:INCREMENT TABLE INDEX
1866	010130	116137	002200	044647		MOVB	OFFCOD(R1),GENDPB+\$FMT	:OFFSET CODE
1867	010136	001435				BEQ	9\$:BR IF END OF OFFSET TABLE
1868	010140	062737	000002	001252		ADD	#2,RETRY	:NEW RETRY LIMIT
1869	010146	004737	015222			JSR	PC,OFFST	:OFFSET
1870	010152	005737	044664		4\$:	TST	GENDPB+\$STATUS	:SEE IF FINISHED WITH OFFSET
1871	010156	001775				BEQ	4\$:BR IF NOT
1872	010160	100324				BPL	2\$:BR IF NO ERROR PERFORMING OFFSET
1873	010162	004737	022160		5\$:	JSR	PC,LINE8	:PRINT LINE 8 OF ERROR MESSAGE
1874	010166	004737	023260		6\$:	JSR	PC,INCHRD	:INCREMENT 'HARD' ERROR COUNT
1875	010172	004737	023354			JSR	PC,INCTOT	:INCREMENT TOTAL ERROR COUNT
1876	010176	004737	021712			JSR	PC,LINE7	:PRINT LINE 7 OF ERROR MESSAGE
1877	010202	004737	014772			JSR	PC,PRTBAD	:PRINT THE BAD SECTOR
1878	010206	000436				BR	13\$:CLEAN UP AND RETURN
1879	010210	004737	021622		7\$:	JSR	PC,LINE6B	:PRINT LINE 6B OF ERROR MESSAGE
1880	010214	004737	021540			JSR	PC,LINE5B	:PRINT LINE 5B OF THE ERROR MESSAGE


```

1881 010220 004737 023234      8$: JSR PC,INCSOF ;INCREMENT 'SOFT' ERROR COUNT
1882 010224 004737 014232      JSR PC,ECC ;CORRECT THE ERROR USING ECC AND CHECK IT
1883 010230 000407              BR 11$ ;COMPARE THE BUFFER
1884 010232 004737 021636      9$: JSR PC,LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
1885 010236 000753              BR 6$ ;INCREMENT ERROR COUNT
1886 010240 004737 021614      10$: JSR PC,LINE6A ;PRINT LINE 6A OF ERROR MESSAGE
1887 010244 004737 023234      JSR PC,INCSOF ;INCREMENT 'SOFT' ERROR COUNT
1888 010250 012737 000001 001300 11$: MOV #1,FRSTER ;SET PROCESSING 'DCKER' INDICATOR
1889 010256 004737 013122      JSR PC,COMPARD ;COMPARE THE BUFFER
1890 010262 105737 001301      TSTB FRSTER+1 ;ERROR IN COMPARE ?
1891 010266 100406              BMI 13$ ;BRANCH IF ERROR
1892 010270 004737 023354      JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
1893 010274 104414 051460      DISPLY LIN9G ;'DATA COMPARE OK' MESSAGE
1894 010300 004737 021712      12$: JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
1895 010304 004737 006324      13$: JSR PC,REFMT ;REFORMAT THE ERROR SECTOR
1896 010310 000207              RTS PC ;RETURN
1897
1898 ;WRITE CHECK ERROR PROCESSING
1899
1900 010312 032760 100000 000250 WCKER: BIT #BIT15,$RPER1(RO) ;SEE IF 'DCK' SET ALSO
1901 010320 001034              BNE 2$ ;BR IF IT IS
1902 010322 004737 020130      JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
1903 010326 104414 045635      DISPLY ,EM23 ;PRINT WCE & DCK NOT
1904 010332 005037 001250      CLR MASK ;CLEAR ERROR MASK
1905 010336 004737 020174      JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
1906 010342 004737 020602      JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
1907 010346 004737 021256      JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
1908 010352 004737 021346      JSR PC,LINE5 ;PRINT LINE 5 OF ERROR MESSAGE
1909 010356 004737 023354      JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
1910 010362 012737 000003 001252 MOV #3,RETRY ;RETRY LIMIT
1911 010370 004737 015330      JSR PC,$RETRY ;RETRY THE OPERATION
1912 010374 000403              BR 1$ ;RETRY UNSUCCESSFUL
1913 010376 004737 021630      JSR PC,LINE6C ;PRINT LINE 6C OF ERROR MESSAGE
1914 010402 000502              BR 8$ ;FINISH PROCESSING THE ERROR
1915 010404 004737 021636      1$: JSR PC,LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
1916 010410 000506              BR 10$ ;FINISH PROCESSING THE ERROR
1917 010412 004737 017772      2$: JSR PC,SPOTCK ;SEE IF ERROR AT BAD SPOT ON THE PACK
1918 010416 000507              BR 11$ ;EXIT IF AT BAD SPOT ON PACK
1919 010420 004737 020130      JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
1920 010424 012737 045562 010452 MOV #EM22,13$ ;ASSUME THAT EM22 WILL BE PRINTED
1921 010432 032760 040000 000244 BIT #BIT14,$RPCS2(RO) ;DID 'WCK' ALSO SET ?
1922 010440 001003              BNE 12$ ;BR IF IT DID
1923 010442 012737 046463 010452 MOV #EM37,13$ ;MESSAGE FOR 'DCK' AND 'WCK' NOT DURING
1924 ;WRITE CHECK
1925 010450 104414      12$: DISPLY ;TYPE THE ERROR MESSAGE
1926 010452 000000      13$: .WORD 0 ;MESSAGE ADDRESS GOES HERE
1927 010454 004737 020174      JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
1928 010460 004737 020602      JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
1929 010464 004737 021256      JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
1930 010470 004737 021346      JSR PC,LINE5 ;PRINT LINE 5 OF ERROR MESSAGE
1931 010474 032760 000100 000250 BIT #BIT06,$RPER1(RO) ;ECH SET ALSO ?
1932 010502 001442              BEQ 8$ ;FINISH PROCESSING THE ERROR
1933 010504 012737 000020 001252 3$: MOV #16,RETRY ;RETRY LIMIT - 16 (10)
1934 010512 004737 016320      4$: JSR PC,GODRIV ;RETRY THE ORDER

```



```

1935 010516 005760 000016      5$:  TST      $STATUS(RO)      ;ORDER FINISHED ?
1936 010522 001775              BEQ      5$                ;BR IF NOT
1937 010524 100405              BMI      6$                ;BR IF ERROR ON ORDER
1938 010526 105237 001253      INCB     RETRY+1           ;INCREMENT RETRY COUNT
1939 010532 004737 021630      JSR     PC,LINE6C         ;PRINT LINE 6C OF ERROR MESSAGE
1940 010536 000431              BR       9$                ;FINISH ERROR PROCESSING
1941 010540 105237 001253      6$:  INCB     RETRY+1           ;INCREMENT RETRY COUNT
1942 010544 123737 001252 001253  CMPB     RETRY,RETRY+1     ;DONE ?
1943 010552 001714              BEQ      1$                ;BR IF AT RETRY LIMIT
1944 010554 032760 100000 000250  BIT     #BIT15,$RPER1(RO) ;'DCK' SET
1945 010562 001407              BEQ      7$                ;BR IF NOT - DIFFERENT ERROR
1946 010564 032760 000100 000250  BIT     #BIT06,$RPER1(RO) ;'ECH' ALSO SET ?
1947 010572 001347              SNE     4$                ;BR IF IT IS, RETRY ORDER
1948 010574 004737 021630      JSR     PC,LINE6C         ;PRINT LINE 6C OF ERROR MESSAGE
1949 010600 000403              BR       8$                ;FINISH PROCESSING ERROR
1950 010602 004737 022160      7$:  JSR     PC,LINE8         ;PRINT LINE 8 - 'DIFFERENT ERROR'
1951 010606 000405              BR       9$                ;FINISH PROCESSING ERROR
1952 010610 004737 023354      8$:  JSR     PC,INCTOT        ;INCREMENT TOTAL ERROR COUNT
1953 010614 004737 021712      JSR     PC,LINE7         ;FINISH THE ERROR MESSAGE
1954 010620 000406              BR       11$               ;EXIT
1955 010622 004737 023354      9$:  JSR     PC,INCTOT        ;INCREMENT TOTAL ERROR COUNT
1956 010626 004737 021712      10$: JSR     PC,LINE7         ;FINISH THE ERROR MESSAGE
1957 010632 004737 006324      JSR     PC,REFMT         ;REFORMAT THE SECTOR IN ERROR
1958 010636 000207      11$: RTS      PC           ;RETURN
1959
1960 ;REPORT 'HCRC' ERROR
1961
1962 010640 004737 017772      HRCER: JSR     PC,SPOTCK      ;SEE IF ERROR AT PACK BAD SPOT
1963 010644 000450              BR       3$                ;EXIT IF IT IS
1964 010646 004737 020130      JSR     PC,LINE1         ;PRINT LINE 1 OF ERROR MESSAGE
1965 010652 104414 045510      DISPLY  ,EM20            ;REPORT 'HCRC'
1966 010656 004737 020174      JSR     PC,LINE2         ;PRINT LINE 2 OF ERROR MESSAGE
1967 010662 004737 020602      JSR     PC,LINE3         ;PRINT LINE 3 OF ERROR MESSAGE
1968 010666 004737 021256      JSR     PC,LINE4         ;PRINT LINE 4 OF ERROR MESSAGE
1969 010672 032760 040000 000244  BIT     #BIT14,$RPCS2(RO) ;'WCE' ERROR ALSO ?
1970 010700 001402              BEQ      1$                ;BR IF NOT
1971 010702 004737 021346      JSR     PC,LINE5         ;DISPLAY WORDS WHICH CAUSED 'WCE'
1972 010706 004737 023234      1$:  JSR     PC,INCSOF        ;INCREMENT 'SOFT' ERROR COUNT
1973 010712 004737 023354      JSR     PC,INCTOT        ;INCREMENT TOTAL ERROR COUNT
1974 010716 012737 000400 001250  MOV     #BIT8,MASK        ;SET ERROR MASK
1975 010724 012737 000003 001252  MOV     #3,RETRY          ;RETRY LIMIT
1976 010732 004737 015330      JSR     PC,$RETRY        ;RETRY ORDER
1977 010736 000405              BR       2$                ;RETRY NOT SUCCESSFUL
1978 010740 004737 021630      JSR     PC,LINE6C         ;PRINT LINE 6C OF ERROR MESSAGE
1979 010744 004737 021712      JSR     PC,LINE7         ;PRINT LINE 7 OF ERROR MESSAGE
1980 010750 000406              BR       3$                ;EXIT
1981 010752 004737 021636      2$:  JSR     PC,LINE6D        ;PRINT LINE 6D OF ERROR MESSAGE
1982 010756 004737 021712      JSR     PC,LINE7         ;PRINT LINE 7 OF ERROR MESSAGE
1983 010762 004737 006324      JSR     PC,REFMT         ;REFORMAT THE ERROR SECTOR
1984 010766 000207      3$:  RTS      PC           ;RETURN
1985
1986 ;REPORT DRIVE ERROR
1987
1988 010770 004737 020130      DRIVER: JSR     PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE

```



```

2043 011270 000137 011442          JMP      POSER          ;REPORT POSITIONING ERROR
2044 011274 052737 010000 054374 2$:  BIS      #BIT12,CYLDER ;RESTORE THE FORMAT BIT
2045 011302 000137 012100          JMP      HCFER          ;REPORT 'HCE' ERROR
2046 011306 004737 017772          JSR      PC,SPOTCK     ;SEE IF ERROR AT BAD SPOT
2047 011312 000452                BR       6$           ;EXIT IF IT IS
2048 011314 004737 020130          JSR      PC,LINE1     ;PRINT LINE 1 OF ERROR MESSAGE
2049 011320 104414 045762          DISPLY  .EM25         ;HEADER READ ERROR - 'HCE' SET
2050 011324 004737 020174          JSR      PC,LINE2     ;PRINT LINE 2 OF ERROR MESSAGE
2051 011330 004737 020602          JSR      PC,LINE3     ;PRINT LINE 3 OF ERROR MESSAGE
2052 011334 004737 021256          JSR      PC,LINE4     ;PRINT LINE 4 OF ERROR MESSAGE
2053 011340 032760 040000 000244  BIT      #BIT14,$RPCS2(RO) ;'WCE' ERROR ALSO ?
2054 011346 001402                BEQ     4$           ;BR IF NOT
2055 011350 004737 021346          JSR      PC,LINES5   ;DISPLAY WORDS WHICH CAUSED 'WCE'
2056 011354 004737 021446          JSR      PC,LINE5A   ;PRINT LINE 5 OF ERROR MESSAGE
2057 011360 004737 023234          JSR      PC,INCSOF   ;INCREMENT SOFT ERROR COUNT
2058 011364 004737 023354          JSR      PC,INCTOT   ;INCREMENT TOTAL ERROR COUNT
2059 011370 012737 000200 001250  MOV      #BIT7,MASK   ;SET ERROR MASK
2060 011376 012737 000003 001252  MOV      #3,RETRY     ;RETRY LIMIT
2061 011404 004737 015330          JSR      PC,$RETRY   ;RETRY THE ORDER
2062 011410 000405                BR       5$           ;RETRY NOT SUCCESSFUL
2063 011412 004737 021630          JSR      PC,LINE6C   ;PRINT LINE 6C OF ERROR MESSAGE
2064 011416 004737 021712          JSR      PC,LINE7   ;PRINT LINE 7 OF ERROR MESSAGE
2065 011422 000406                BR       6$           ;EXIT
2066 011424 004737 021636          JSR      PC,LINE6D   ;PRINT LINE 6D OF ERROR MESSAGE
2067 011430 004737 021712          JSR      PC,LINE7   ;PRINT LINE 7 OF ERROR MESSAGE
2068 011434 004737 006324          JSR      PC,REFMT    ;REFORMAT THE ERROR SECTOR
2069 011440 000207                RTS       PC          ;RETURN
2070
2071          ;REPORT POSSIBLE POSITIONING ERROR
2072
2073 011442 004737 015170          POSER: JSR      PC,RECALT ;RECALIBRATE
2074 011446 004737 020130          JSR      PC,LINE1   ;PRINT LINE 1 OF ERROR MESSAGE
2075 011452 104414 047310          DISPLY  .EM51         ;PROGRAM DETECTED POSITIONING ERROR
2076 011456 004737 020174          JSR      PC,LINE2   ;PRINT LINE 2 OF ERROR MESSAGE
2077 011462 004737 020630          JSR      PC,LINE3C  ;PRINT LINE 3C OF ERROR MESSAGE
2078 011466 052737 010000 054374  BIS      #BIT12,CYLDER ;RESTORE THE FORMAT BIT
2079 011474 004737 021446          JSR      PC,LINE5A  ;PRINT LINE 5A OF THE ERROR MESSAGE
2080 011500 004737 023330          JSR      PC,INCMIS  ;INCREMENT MISPOSITIONING COUNT
2081 011504 004737 023354          JSR      PC,INCTOT  ;INCREMENT TOTAL ERROR COUNT
2082 011510 004737 022040          JSR      PC,LINE7A  ;PRINT LINE 7A OF ERROR MESSAGE
2083 011514 000207                RTS       PC          ;EXIT
2084
2085          ;REPORT 'OPI' ERROR
2086
2087 011516 004737 017772          OPIER: JSR      PC,SPOTCK ;SEE IF ERROR AT BAD SPOT
2088 011522 000207                RTS       PC          ;RETURN IF IT IS
2089 011524 004737 020130          JSR      PC,LINE1   ;PRINT LINE 1 OF ERROR MESSAGE
2090 011530 104414 046157          DISPLY  .EM31         ;'OPI' ERROR
2091 011534 004737 020174          JSR      PC,LINE2   ;PRINT LINE 2 OF ERROR MESSAGE
2092 011540 004737 020602          JSR      PC,LINE3   ;PRINT LINE 3 OF ERROR MESSAGE
2093 011544 004737 021256          JSR      PC,LINE4   ;PRINT LINE 4 OF ERROR MESSAGE
2094 011550 004737 023354          JSR      PC,INCTOT  ;INCREMENT TOTAL ERROR COUNT
2095 011554 012737 020000 001250  MOV      #BIT13,MASK ;ERROR MASK
2096 011562 012737 000003 001252  OPIER1: MOV     #3,RETRY ;RETRY LIMIT

```




```

2097 011570 004737 015330      JSR    PC,$RETRY      ;RETRY THE ORDER
2098 011574 000405              BR     1$             ;RETRY UNSUCCESSFUL
2099 011576 004737 021630      JSR    PC,LINE6C     ;PRINT LINE 6C OF ERROR MESSAGE
100  011602 004737 021712      JSR    PC,LINE7     ;PRINT LINE 7 OF ERROR MESSAGE
101  011606 000207              RTS     PC            ;EXIT
102  011610 004737 021636      1$:   JSR    PC,LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
103  011614 004737 021712      JSR    PC,LINE7     ;PRINT LINE 7 OF ERROR MESSAGE
104  011620 004737 006324      JSR    PC,REFMT     ;REFORMAT THE ERROR SECTOR
105  011624 000207              RTS     PC            ;RETURN

;REPORT 'DTE' ERROR

109  011626 004737 017772      DTEER: JSR    PC,SPOTCK ;SEE IF ERROR AT BAD SPOT
110  011632 000207              RTS     PC            ;RETURN IF IT IS
111  011634 004737 020130      JSR    PC,LINE1     ;PRINT LINE 1 OF ERROR MESSAGE
112  011640 104414 046222      DISPLY EM32         ;'DTE' ERROR
113  011644 000137 007652      JMP    DCKER1       ;FINISH PROCESSING THE 'DTE' ERROR

;REPORT 'PAR' ERROR

117  011650 004737 020130      PARER: JSR    PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
118  011654 104414 046255      DISPLY EM33         ;REPORT 'PAR'
119  011660 004737 020174      JSR    PC,LINE2     ;PRINT LINE 2 OF ERROR MESSAGE
120  011664 004737 020706      JSR    PC,LINE3E    ;PRINT LINE 3E OF ERROR MESSAGE
121  011670 004737 021256      JSR    PC,LINE4     ;PRINT LINE 4 OF ERROR MESSAGE
122  011674 004737 023354      JSR    PC,INCTOT    ;INCREMENT TOTAL ERROR COUNT
123  011700 012737 000010      MOV    #BIT03,MASK  ;ERROR MASK
124  011706 012737 000003      MOV    #3,RETRY     ;RETRY LIMIT
125  011714 004737 015330      JSR    PC,$RETRY    ;RETRY ORDER
126  011720 000405              BR     2$             ;RETRY UNSUCCESSFUL
127  011722 004737 021630      JSR    PC,LINE6C     ;RETRY SUCCESSFUL
128  011726 004737 021712      1$:   JSR    PC,LINE7     ;PRINT LINE 7 OF ERROR MESSAGE
129  011732 000207              RTS     PC            ;EXIT
130  011734 004737 021636      2$:   JSR    PC,LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
131  011740 000772              BR     1$             ;FINISH ERROR MESSAGE

;REPORT 'IAE' ERROR

135  011742 004737 020130      IAEER: JSR    PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
136  011746 10441  046374      DISPLY EM35         ;REPORT 'IAE'
137  011752 00473  020174      JSR    PC,LINE2     ;PRINT LINE 2 OF ERROR MESSAGE
138  011756 00473  020774      JSR    PC,LINE3F    ;PRINT LINE 3F OF ERROR MESSAGE
139  011762 004737 023354      JSR    PC,INCTOT    ;INCREMENT TOTAL ERROR COUNT
140  011766 004737 021712      JSR    PC,LINE7     ;PRINT LINE 7 OF ERROR MESSAGE
141  011772 000207              RTS     PC            ;RETURN

;REPORT 'WLE' ERROR

145  011774 004737 020130      WLEER: JSR    PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
146  012000 104414 046432      DISPLY EM36         ;REPORT 'WLE'
147  012004 004737 020174      JSR    PC,LINE2     ;PRINT LINE 2 OF ERROR MESSAGE
148  012010 004737 023354      JSR    PC,INCTOT    ;INCREMENT TOTAL ERROR COUNT
149  012014 004737 021712      JSR    PC,LINE7     ;PRINT LINE 7 OF ERROR MESSAGE
150  012020 000207              RTS     PC            ;RETURN

```

001250
001252

151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204

:REPORT FORMAT ERROR

```

FMTER: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
        DISPLY EM26 ;FORMAT ERROR
        JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
        JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
        JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
        BIT #BIT14,$RPCS2(RO) ;'WCE' ERROR ALSO ?
        BEQ 1$ ;BR IF NOT
        JSR PC,LINES ;DISPLAY WORDS WHICH CAUSED 'WCE'
1$: JSR PC,LINE5A ;PRINT LINE 5A OF ERROR MESSAGE
    JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
    JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
    RTS PC

```

:REPORT HEADER COMPARE ERROR

```

HCEER: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
        DISPLY EM27 ;HEADER COMPARE ERROR
        JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
        JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
        JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
        BIT #BIT14,$RPCS2(RO) ;'WCE' ERROR ALSO ?
        BEQ 1$ ;BR IF NOT
        JSR PC,LINES ;DISPLAY WORDS WHICH CAUSED 'WCE'
1$: JSR PC,LINE5A ;PRINT LINE 5A OF ERROR MESSAGE
    JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
    JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
    JSR PC,REFMT ;REFORMAT THE ERROR SECTOR
    RTS PC ;RETURN

```

:PROCESS CONTROL/INTERFACE TRANSFER ERROR

```

TRFER: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
        DISPLY EM40 ;RH11 OR UNIBUS TRANSFER ERROR
        JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
        JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
        JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
        JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
        BIT #BIT15:BIT13:BIT9:BIT8,$RPCS2(RO) ;'DLT','UPE','MXF','MOPE' SET ?
        BEQ 2$ ;BR IF NONE SET
        MOV #3,RETRY ;RETRY LIMIT
        CLR MASK ;CLEAR ERROR MASK
        JSR PC,$RETRY ;RETRY THE OPERATION
        BR 1$ ;RETURN HERE IF RETRY UNSUCCESSFUL
        JSR PC,LINE6C ;PRINT LINE 6C OF ERROR MESSAGE
        BR 2$ ;FINISH THE ERROR REPORT
1$: JSR PC,LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
2$: JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
    RTS PC

```

:PROCESS 'SKI' OR 'OCYL' ERRORS

DZRJD.019 MAIN PROGRAM

```

22205 012262 004737 020130 SKIER: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
22206 012266 104414 047221 DISPLY EM50 ;'SKI' OR 'OCYL' ERROR
22207 012272 004737 020174 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
22208 012276 004737 020616 JSR PC,LINE3B ;PRINT LINE 3B OF ERROR MESSAGE
22209 012302 004737 023354 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
22210 012306 004737 023304 JSR PC,INCSKI ;INCREMENT 'SKI' OR 'OCYL' ERROR COUNT
22211 012312 004737 022040 JSR PC,LINE7A ;PRINT LINE 7A OF ERROR MESSAGE
22212 012316 000207 RTS PC

```

;REPORT WRITE CLOCK FAILURE ('WCF')

```

22216 012320 004737 020130 WCFER: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
22217 012324 104414 046332 DISPLY EM34 ;REPORT WRITE CLOCK FAILURE
22218 012330 004737 020174 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
22219 012334 004737 020610 JSR PC,LINE3A ;PRINT LINE 3A OF ERROR MESSAGE
22220 012340 004737 021256 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
22221 012344 004737 023354 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
22222 012350 004737 014772 JSR PC,PRTBAD ;SEE IF BAD SECTOR TO BE PRINTED
22223 012354 012737 000003 MOV #3,RETRY ;RETRY COUNT
22224 012362 012737 000040 MOV #BIT05,MASK ;ERROR MASK
22225 012370 004737 015330 JSR PC,$RETRY ;RETRY THE ORDER
22226 012374 000405 BR 2$ ;RETURN HERE IF RETRY UNSUCCESSFUL
22227 012376 004737 021630 JSR PC,LINE6C ;PRINT LINE 6C OF ERROR MESSAGE
22228 012402 004737 021712 1$: JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
22229 012406 000207 RTS PC
22230 012410 004737 021636 2$: JSR PC,LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
22231 012414 000772 BR 1$

```

;PROCESS DRIVE UNSAFE ERROR

```

22235 012416 004737 020130 UNSAF: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
22236 012422 104414 047353 DISPLY EM60 ;REPORT DRIVE UNSAFE
22237 012426 004737 020174 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
22238 012432 004737 020602 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
22239 012436 004737 023354 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
22240 012442 012737 000003 MOV #3,RETRY ;RETRY COUNT
22241 012450 004737 015330 JSR PC,$RETRY ;RETRY THE ORDER
22242 012454 000403 BR 1$ ;RETRY WAS UNSUCCESSFUL
22243 012456 004737 021630 JSR PC,LINE6C ;PRINT LINE 6C OF ERROR MESSAGE
22244 012462 000402 BR 2$ ;CONTINUE WITH ERROR REPORT
22245 012464 004737 021636 1$: JSR PC,LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
22246 012470 004737 021712 2$: JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
22247 012474 000207 RTS PC ;RETURN

```

;REPORT AN 'UNKNOWN' DATA PATTERN

```

22251 012476 105737 001300 NOMTCH: TSTB FRSTER ;FIRST ERROR IN THE SECTOR ?
22252 012502 001013 BNE 1$ ;BR IF NOT OR IF PROCESSING 'DCKER'
22253 012504 004737 020130 JSR PC,LINE1 ;TYPE LINE 1 OF ERROR MESSAGE
22254 012510 104414 046730 DISPLY EM43 ;'CAN'T MATCH DATA WITH PATTERN'
22255 012514 004737 020174 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
22256 012520 004737 020610 JSR PC,LINE3A ;PRINT LINE 3A OF ERROR MESSAGE
22257 012524 004737 021256 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
22258 012530 000404 BR 2$ ;CONTINUE PROCESSING ERROR

```



```

2259 012532 104414 046730 1$: DISPLY ,EM43 ;'CAN'T MATCH DATA WITH PATTERN'
2260 012536 104414 001165 DISPLY ,$CRLF ;CR-LF
2261 012542 104414 051410 2$: DISPLY ,LIN9I ;HEADER FOR DATA PRINTOUT
2262 012546 010146 MOV R1,-(SP) ;ADDRESS OF WORD 1
2263 012550 004737 022172 JSR PC,LINOC ;TYPE WORD 1
2264 012554 104414 052073 DISPLY ,LINS ;SPACES
2265 012560 012146 MOV (R1)+,-(SP) ;ADDRESS OF WORD 1
2266 012562 004737 022172 JSR PC,LINOC ;TYPE WORD 1
2267 012566 104414 001165 DISPLY ,$CRLF ;CR-LF
2268 012572 010146 MOV R1,-(SP) ;ADDRESS OF WORD 2
2269 012574 004737 022172 JSR PC,LINOC ;TYPE WORD 2
2270 012600 104414 052073 DISPLY ,LINS ;SPACES
2271 012604 012146 MOV (R1)+,-(SP) ;ADDRESS OF WORD 2
2272 012606 004737 022172 JSR PC,LINOC ;TYPE WORD 2
2273 012612 104414 001165 DISPLY ,$CRLF ;CR-LF
2274 012616 010146 MOV R1,-(SP) ;ADDRESS OF WORD 3
2275 012620 004737 022172 JSR PC,LINOC ;TYPE WORD 3
2276 012624 104414 052073 DISPLY ,LINS ;SPACES
2277 012630 012146 MOV (R1)+,-(SP) ;ADDRESS OF WORD 3
2278 012632 004737 022172 JSR PC,LINOC ;TYPE WORD 3
2279 012636 104414 001165 DISPLY ,$CRLF ;CR-LF
2280 012642 010146 MOV R1,-(SP) ;ADDRESS OF WORD 4
2281 012644 004737 022172 JSR PC,LINOC ;TYPE WORD 4
2282 012650 104414 052073 DISPLY ,LINS ;SPACES
2283 012654 012146 MOV (R1)+,-(SP) ;ADDRESS OF WORD 4
2284 012656 004737 022172 JSR PC,LINOC ;TYPE WORD 4
2285 012662 104414 001165 DISPLY ,$CRLF ;CR-LF
2286 012666 062701 000770 ADD #(<252.*2.),R1 ;INCREMENT BUFFER POINTER
2287 012672 005002 CLR R2 ;CLEAR 'WORDS TO COMPARE' COUNT IN R2
2288 012674 112737 000001 001300 MOVB #1,FRSTER ;SET 'NOT FIRST ERROR' INDICATOR
2289 012702 112737 177777 001301 MOVB #-1,FRSTER+1 ;SET ERROR FOUND INDICATOR
2290 012710 013737 001374 001310 MOV CMLMT,LIMIT ;RESET THE COMPARE ERROR TIMEOUT LIMIT
2291 012716 000207 RTS PC ;RETURN
2292
2293 ;CHECK ERROR BITS IN THE RH11 & RPO4/5/6 REGISTERS
2294
2295 012720 032760 060000 000234 CKERR: BIT #60000,$RPCS1(R0) ;SEE IF 'TRE' OR 'MCPE' SET
2296 012726 001015 1$: BNE 1$ ;BR IF EITHER SET
2297 012730 032760 177400 000244 BIT #177400,$RPCS2(R0) ;SEE IF ERROR BITS IN CS2 SET
2298 012736 001011 1$: BNE 1$ ;BR IF ANY SET
2299 012740 005760 000250 TST $RPER1(R0) ;ANY BITS SET IN ER1
2300 012744 001006 1$: BNE 1$ ;BR IF ANY SET
2301 012746 005760 000274 TST $RPER2(R0) ;ANY BITS SET IN ER2 ?
2302 012752 001003 1$: BNE 1$ ;BR IF ANY SET
2303 012754 005760 000276 TST $RPER3(R0) ;ANY BITS SET IN ER3 ?
2304 012760 001416 2$: BEQ 2$ ;BR IF NONE SET
2305 012762 004737 020130 1$: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
2306 012766 104414 046775 DISPLY ,EM44 ;ERROR BITS SET, BUT 'SC' OR 'TRE' NOT SET
2307 012772 004737 020174 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
2308 012776 004737 020602 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
2309 013002 004737 021256 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
2310 013006 004737 023354 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
2311 013012 004737 021712 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
2312 013016 000207 2$: RTS PC ;RETURN

```

```

2313
2314 ;CHECK BUS ADDRESS REGISTER & WORD COUNT REGISTER
2315
2316 013020 005760 000236 CKBUS: TST $SRPWC(RO) ;CHECK WORD COUNT
2317 013024 001010 BNE 1$ ;BR IF NOT ZERO
2318 013026 016046 000020 MOV $WRDL(RO),-(SP) ;WORD LENGTH
2319 013032 006316 ASL (SP) ;CHANGE INTO BYTE COUNT
2320 013034 066016 000006 ADD $BUF(RO),(SP) ;ADD THE STARTING LOCATION
2321 013040 022660 000240 CMP (SP)+,$SRPBA(RO) ;BUFFER ADDRESS PROPER ?
2322 013044 001416 BEQ 2$ ;BR IF OK
2323 013046 004737 020130 1$: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
2324 013052 104414 046603 DISPLY EM41 ;BUS ADDRESS OR WORD COUNT INCORRECT
2325 013056 004737 020174 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
2326 013062 004737 020640 JSR PC,LINE3D ;PRINT LINE 3D OF ERROR MESSAGE
2327 013066 004737 021256 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
2328 013072 004737 023354 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
2329 013076 004737 021712 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
2330 013102 000207 2$: RTS PC
2331
2332 ;COMPARE THE BUFFER
2333
2334 013104 132760 000004 000024 CMPAR: BITB #BIT02,$CODE(RO) ;SEE IF READ ORDER
2335 013112 001001 BNE 1$ ;BR IF IT IS
2336 013114 000207 RTS PC ;RETURN
2337 013116 005037 001300 1$: CLR FRSTER ;CLEAR 'FIRST ERROR' INDICATOR
2338 013122 032777 000002 166010 CMPARD: BIT #SW01,$SWR ;IS SWITCH 1 SET?
2339 013130 001401 BEQ 1$ ;BR IF NOT
2340 013132 000207 RTS PC ;YES, DON'T COMPARE
2341 013134 005037 001306 1$: CLR ERCTR ;CLEAR THE ERROR COUNTER
2342 013140 016001 000006 MOV $BUF(RO),R1 ;BUFFER ADDRESS
2343 013144 016037 000020 001312 MOV $WRDL(RO),CMCNT ;WORD COUNT TO WORKING LOCATION
2344 013152 066037 000236 001312 ADD $SRPWC(RO),CMCNT ;CALCULATE ACTUAL WORDS TRANSFERED
2345 013160 016037 000012 001314 MOV $CYL(RO),CMCYL ;CYLINDER ADDRESS WORKING LOCATION
2346 013166 052737 010000 001314 BIS #BIT12,CMCYL ;SET FORMAT BIT
2347 013174 016037 000010 001316 MOV $SEC(RO),CMSEC ;SECTOR & TRACK ADDRESSES TO WORKING LOCNS
2348 013202 013737 001374 001310 MOV CMPLMT,LIMIT ;DISPLAY LIMIT
2349 013210 005237 001310 INC LIMIT ;CONVERT PARAMETER INTO LIMIT VALUE
2350 013214 012737 177777 001276 CMSTR: MOV #-1,ZROIND ;CLEAR THE 'ZERO'S' INDICATOR
2351 013222 005037 001302 CLR SAVER1 ;CLEAR THE R1 SAVE WORD
2352 013226 005037 001304 CLR SAVER5 ;CLEAR THE R5 SAVE WORD
2353 013232 023760 001312 000022 CMP CMCNT,$SSEC(RO) ;IS BUFFER SIZE GREATER THAN ONE SECTOR ?
2354 013240 101003 BHI 1$ ;BR IF IT IS
2355 013242 013702 001312 MOV CMCNT,R2 ;LESS THAN, USE REMAINING BUFFER
2356 013246 000402 BR 2$
2357 013250 016002 000022 1$: MOV $SSEC(RO),R2 ;COMPARE SECTOR
2358 013254 166037 000022 001312 2$: SUB $SSEC(RO),CMCNT ;DECREMENT WORD COUNT
2359 013262 126027 000024 000005 CMPB $CODE(RO),#5 ;READ HEADER & DATA?
2360 013270 001036 BNE CMHED ;BR IF NOT
2361 013272 023721 001314 CMHED: CMP CMCYL,(R1)+ ;CHECK CYLINDER
2362 013276 001402 BEQ 1$ ;BR IF COMPARE OK
2363 013300 004737 013354 JSR PC,5$ ;REPORT ERROR
2364 013304 023721 001316 1$: CMP CMSEC,(R1)+ ;COMPARE SECTOR & TRACK
2365 013310 001402 BEQ 2$ ;BR IF EQ
2366 013312 004737 013354 JSR PC,5$ ;REPORT ERROR
    
```




```

2421 013566 103612          BLO      CMSTR      ;BR IF NOT GREATER THAN MAX
2422 013570 105037 001316    CLRB     CMSEC     ;CLEAR SECTOR ADDRESS
2423 013574 105237 001317    INCB     CMTRK     ;INCREMENT TRACK
2424 013600 123727 001317 000023  CMPB     CMTRK,#19. ;TRACK GREATER THAN MAX ?
2425 013606 103602          BLO      CMSTR      ;BR IF NOT GREATER
2426 013610 105037 001317    CLRB     CMTRK     ;RESET TRACK ADDRESS
2427 013614 005237 001314    INC      CMCYL     ;INCREMENT CYLINDER ADDRESS
2428 013620 000137 013214    JMP      CMSTR     ;CONTINUE WITH COMPARE
2429 013624 004737 014106    JSR      PC,ENDCMP ;PRINT LAST LINE IF ERRORS
2430 013630 000207          RTS      PC
2431
2432          ;TYPE DATA COMPARE ERRORS
2433
2434 013632 005737 001302    CMPRT:   TST      SAVER1 ;PRINT SAVED VALUES ?
2435 013636 001010          BNE      2$        ;BR IF NOT
2436 013640 105737 001300    TSTB     FRSTER    ;FIRST ERROR?
2437 013644 100402          BMI      1$        ;BR IF NOT
2438 013646 004737 013726    JSR      PC,4$     ;PRINT INITIAL MESSAGE INFO
2439 013652 004737 014010    JSR      PC,8$     ;PRINT REMAINDER OF MESSAGE
2440 013656 000422          BR       3$        ;EXIT
2441 013660
2442 013660 010146          MOV      R1,-(SP)  ;;PUSH R1 ON STACK
2443 013662 010546          MOV      R5,-(SP)  ;;PUSH R5 ON STACK
2444 013664 013701 001302    MOV      SAVER1,R1 ;DISPLAY SAVED R1
2445 013670 013705 001304    MOV      SAVER5,R5 ;DISPLAY SAVED R5
2446 013674 004737 013726    JSR      PC,4$     ;PRINT INITIAL MESSAGE INFO
2447 013700 004737 014010    JSR      PC,8$     ;PRINT SAVED VALUES
2448 013704 005037 001302    CLR      SAVER1    ;CLEAR SAVED REGISTER INDICATORS
2449 013710 005037 001304    CLR      SAVER5    ;CLEAR THE OTHER ONE
2450 013714 012605          MOV      (SP)+,R5  ;;POP STACK INTO R5
2451 013716 012601          MOV      (SP)+,R1  ;;POP STACK INTO R1
2452 013720 004737 014010    JSR      PC,8$     ;PRINT REMAINDER OF MESSAGE
2453 013724 000207          RTS      PC        ;RETURN
2454 013726 105737 001300    3$:     TSTB     FRSTER    ;FIRST ERROR ?
2455 013732 100425          BMI      7$        ;BR IF NOT
2456 013734 001013          BNE      5$        ;BR IF FIRST ERROR AND PROCESSING 'DCK' ERROR
2457 013736 004737 020130    JSR      PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
2458 013742 104414 046647    DISPLY   ,EM42     ;DATA COMPARE ERROR
2459 013746 004737 020174    JSR      PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
2460 013752 004737 020610    JSR      PC,LINE3A ;PRINT LINE 3A OF ERROR MESSAGE
2461 013756 004737 021256    JSR      PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
2462 013762 000404          BR       6$        ;GO TO TYPE HEADER
2463 013764 104414 051303    5$:     DISPLY   ,LIN9B    ;HEADER MESSAGE OF PROCESSING 'DCK' ERROR
2464 013770 104414 001165    DISPLY   ,$CRLF    ;CR-LF
2465 013774 104414 051332    6$:     DISPLY   ,LIN9H    ;DISPLAY HEADER
2466 014000 012737 177777 001300    MOV      #-1,FRSTER ;SET ERROR FLAG
2467 014006 000207          RTS      PC        ;RETURN
2468 014010 005737 001310    7$:     TST      LIMIT     ;TYPEOUT LIMIT REACHED ?
2469 014014 001403          BEQ      9$        ;BR IF IT HAS
2470 014016 005337 001310    DEC      LIMIT     ;DECREMENT LIMIT COUNTER
2471 014022 001005          BNE      10$       ;BR IF NOT AT LIMIT
2472 014024 032777 000200 165106 9$:     BIT      #SW07,JSWR ;PRINT ALL DATA COMPARE ERRORS ?
2473 014032 001001          BNE      10$       ;BR IF YES
2474 014034 000207          RTS      PC        ;RETURN

```


2475 014036 010146
 2476 014040 162716 000002
 2477 014044 004737 022172
 2478 014050 104414 052073
 2479 014054 016546 177776
 2480 014060 004737 022172
 2481 014064 104414 052073
 2482 014070 016146 177776
 2483 014074 004737 022172
 2484 014100 104414 001165
 2485 014104 000207
 2486
 2487
 2488
 2489 014106 105737 001301
 2490 014112 001417
 2491 014114 005737 001306
 2492 014120 001410
 2493 014122 104414 051430
 2494 014126 013746 001306
 2495 014132 004737 022224
 2496 014136 104414 001165
 2497 014142 004737 023354
 2498 014146 004737 021712
 2499 014152 000207
 2500
 2501
 2502
 2503
 2504
 2505
 2506
 2507
 2508
 2509 014154 010146
 2510 014156 012704 000044
 2511 014162 011601
 2512 014164 162704 000002
 2513 014170 016405 002742
 2514 014174 001411
 2515
 2516 014176 012703 000004
 2517 014202 022125
 2518 014204 001366
 2519 014206 005303
 2520 014210 001374
 2521 014212 062704 002742
 2522 014216 000403
 2523 014220 062766 000002 000002
 2524 014226 012601
 2525 014230 000207
 2526
 2527
 2528

```

10$: MOV R1, -(SP) ;BUFFER ADDRESS
SUB #2, (SP) ;ADJUST ADDRESS
JSR PC, LINOCT ;TYPE IT
DISPLY ,LINSF ;2 SPACES
MOV -2(R5), -(SP) ;PUT GOOD DATA ON THE STACK
JSR PC, LINOCT ;TYPE IT
DISPLY ,LINSF ;2 SPACES
MOV -2(R1), -(SP) ;BAD DATA
JSR PC, LINOCT ;TYPE IT
DISPLY $CRLF ;CR-LF
RTS PC ;RETURN

;LAST LINE OF COMPARE ERROR REPORTING
ENDCMP: TSTB FRSTER+1 ;ANY COMPARE ERRORS FOUND ?
BEQ 2$ ;BR IF NOT
TST ERCTR ;SEE HOW MANY ERRORS
BEQ 1$ ;BR IF ONLY CAN'T MATCH PATTERN
DISPLY ,LINSF ;'NUMBER OF ERRORS='
MOV ERCTR, -(SP) ;NUMBER OF ERRORS
JSR PC, LINDEC ;TYPE IT
DISPLY $CRLF ;CR-LF
1$: JSR PC, INCTOT ;INCREMENT TOTAL ERROR COUNT
JSR PC, LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
2$: RTS PC ;RETURN

;ROUTINE TO MATCH THE DATA WITH A PATTERN
;CALL:
; MOV #BUFFER, R1 ;BUFFER ADDRESS
; JSR PC, MATCH
; RETURN1 ;PATTERN ADDRESS IN R4
; RETURN2 ;COULDN'T MATCH PATTERN

MATCH: MOV R1, -(SP) ;SAVE R1 ON THE STACK
MOV #4, R4 ;PATTERN TABLE INDEX
1$: MOV (SP), R1 ;RELOAD R1
SUB #2, R4 ;DECREMENT INDEX
MOV STNDAT(R4), R5 ;ADDRESS OF PATTERN ADDRESS
BEQ 3$ ;BR IF ALL PATTERNS CHECKED AND NO MATCH
;FOUND
2$: MOV #4, R3 ;NUMBER OF LOCATIONS TO CHECK
CMP (R1)+, (R5)+ ;COMPARE THE BUFFER AGAINST THE PATTERN
BNE 1$ ;BR IF NOT EQUAL, TRY NEXT PATTERN.
DEC R3 ;FINISHED CHECKING?
BNE 2$ ;BR IF NOT FINISHED
ADD #STNDAT, R4 ;MAKE PATTERN ADDRESS ABSOLUTE
BR 4$ ;EXIT
3$: ADD #2, 2(SP) ;INCREMENT RETURN ADDRESS
4$: MOV (SP)+, R1 ;RESTORE R1
RTS PC ;RETURN

;USE ECC TO CORRECT THE DATA ERROR
    
```

2529	014232	016037	000240	001322	ECC:	MOV	\$RPBA(RO),ECSEC	; ADDRESS OF LAST LOCN XFERED
2530	014240	016046	000236			MOV	\$RPWC(RO),-(SP)	; ACT WORDS XFERED (2'S COMP)
2531	014244	066016	000020			ADD	\$WRDL(RO),(SP)	; ADD WORDS REQUESTED
2532	014250	005046				CLR	-(SP)	; CLEAR NEXT STACK LOCN
2533	014252	016046	000022			MOV	\$SSEC(RO),-(SP)	; SECTOR SIZE
2534	014256	004737	027004			JSR	PC, LINKDV	; DIVIDE WORDS XFERED BY SECTOR SIZE
2535	014262	005716				TST	(SP)	; PARTIAL SECTOR XFERED ?
2536	014264	001413				BEQ	1\$; BR IF NOT
2537	014266	006316				ASL	(SP)	; CONVERT INTO NUMBER OF BYTES
2538	014270	161637	001322			SUB	(SP),ECSEC	; SUBTRACT SECTOR RESIDUE
2539	014274	126027	000024	000005		CMPB	\$CODE(RO),#5	; WAS OP READ HEAD & DATA
2540	014302	001007				BNE	2\$; BR IF NOT
2541	014304	062737	000010	001322		ADD	#8.,ECSEC	; ADD HEADER SIZE (IN BYTES) BACK IN
2542	014312	000403				BR	2\$; GO ADJUST THE STACK POINTER
2543	014314	162737	001000	001322	1\$:	SUB	#1000,ECSEC	; SUBTRACT SECTOR DATA FIELD SIZE (IN BYTES)
2544	014322	062706	000004		2\$:	ADD	#4,SP	; ADJUST THE STACK POINTER
2545	014326	016037	000300	001320		MOV	\$RPEC1(RO),ECBIT	; ECC POSITION COUNT
2546	014334	005337	001320			DEC	ECBIT	; ADJUST THE POSITION COUNT
2547	014340	013737	001320	001330		MOV	ECBIT,ECWRD	; LOAD THE WORD COUNT LOCATION
2548	014346	042737	177760	001320		BIC	#17,ECBIT	; SAVE THE BIT OFFSET COUNT
2549	014354	042737	000017	001330		BIC	#17,ECWRD	; CLEAR THE BIT OFFSET
2550	014362	006237	001330			ASR	ECWRD	; CHANGE TO BYTE COUNT
2551	014366	006237	001330			ASR	ECWRD	; CHANGE TO BYTE COUNT
2552	014372	006237	001330			ASR	ECWRD	; CHANGE TO BYTE COUNT
2553	014376	104414	051507			DISPLY	.LIN10A	; 'ERROR BURST BEGINS AT '
2554	014402	013746	001330			MOV	ECWRD, -(SP)	; PUT THE WORD COUNT ON THE STACK
2555	014406	006216				ASR	(SP)	; CONVERT TO WORD COUNT FOR MESSAGE
2556	014410	004737	027720			JSR	PC,\$SB2D	; CONVERT THE WORD COUNT
2557	014414	004737	027320			JSR	PC,\$SUPRS	; PRINT IT
2558	014420	104414	051543			DISPLY	.LIN10B	; ' IN DATA FIELD OF ERROR SECTOR'
2559	014424	063737	001322	001330		ADD	ECSEC,ECWRD	; FIND THE BEGINNING OF THE ERROR BURST
2560	014432	026037	000240	001330		CMP	\$RPBA(RO),ECWRD	; SEE IF BURST WAS IN DATA READ
2561	014440	101002				BHI	.+6	; BR IF IN DATA READ
2562	014442	000137	014760			JMP	ECC2	; NOT IN DATA READ - REPORT IT
2563	014446	016037	000302	001324		MOV	\$RPEC2(RO),ECMSK0	; GET THE ERROR MASK
2564	014454	005037	001326			CLR	ECMSK1	; CLEAR THE UPPER MASK WORD
2565	014460	005737	001320		3\$:	TST	ECBIT	; BIT OFFSET EQUAL ZERO
2566	014464	001407				BEQ	4\$; BR IF IT IS
2567	014466	005337	001320			DEC	ECBIT	; DECREMENT THE BIT OFFSET COUNT
2568	014472	006337	001324			ASL	ECMSK0	; SHIFT THE ERROR MASK
2569	014476	006137	001326			ROL	ECMSK1	; SHIFT THE LOWER INTO THE UPPER
2570	014502	000766				BR	3\$; CONTINUE THE SHIFT
2571	014504	017737	164620	001334	4\$:	MOV	2ECWRD,ECBADO	; SAVE THE INCORRECT WORD
2572	014512	005037	001336			CLR	ECWRD1	; CLEAR SECOND INCORRECT WORD ADDRESS
2573	014516	013746	001324			MOV	ECMSK0, -(SP)	; PUT LOWER MASK ON STACK
2574	014522	047716	164602			BIC	2ECWRD,(SP)	; CLEAR ERRONEOUS ONE BITS FROM MASK
2575	014526	043777	001324	164574		BIC	ECMSK0,2ECWRD	; CLEAR ERRONEOUS ONE BITS FROM BAD WORD
2576	014534	052677	164570			BIS	(SP)+,2ECWRD	; SET DROPPED BITS
2577	014540	005737	001326			TST	ECMSK1	; DOES BURST GO INTO NEXT WORD ?
2578	014544	001431				BEQ	ECC1	; BR IF BURST ONLY IN ONE WORD
2579	014546	013737	001330	001336		MOV	ECWRD,ECWRD1	; DUPLICATE ADDRESS
2580	014554	062737	000002	001336		ADD	#2,ECWRD1	; INCREMENT ERROR ADDRESS
2581	014562	026037	000240	001336		CMP	\$RPBA(RO),ECWRD1	; IS NEXT WORD IN THE BUFFER
2582	014570	101003				BHI	5\$; BR IF IT IS




```

2583 014572 005037 001336          CLR      ECWRD1      ;CLEAR 2ND WORD ADDRESS
2584 014576 000414          BR        ECC1       ;PRINT WORD CORRECTED
2585 014600 017737 164532 001342 5$:  MOV      @ECWRD1,ECBAD1 ;SAVE THE SECOND BAD WORD
2586 014606 013746 001326          MOV      ECMSK1,-(SP) ;PUT THE UPPER MASK ON THE STACK
2587 014612 047716 164520          BIC      @ECWRD1,(SP) ;CLEAR ERRONEOUS ONE BITS FROM UPPER MASK
2588 014616 043777 001325 164512  BIC      ECMSK1,@ECWRD1 ;CLEAR ERRONEOUS ONE BITS FROM DATA WORD
2589 014624 052677 164506          BIS      (SP)+,@ECWRD1 ;SET DROPPED BITS
2590 014630 104414 051711          ECC1:   DISPLY    ,LIN10H  ;HEADER
2591 014634 013746 001330          MOV      ECWRD,-(SP)  ;PUT ECWRD ON THE STACK
2592 014640 004737 022172          JSR      PC,LIN0CT   ;TYPE ECWRD
2593 014644 104414 052073          DISPLY   ,LINSF      ;SPACES
2594 014650 013746 001334          MOV      ECBADO,-(SP) ;PUT ECBADO ON THE STACK
2595 014654 004737 022172          JSR      PC,LIN0CT   ;TYPE ECBADO
2596 014660 104414 052073          DISPLY   ,LINSF      ;SPACES
2597 014664 017746 164440          MOV      @ECWRD,-(SP) ;PUT @ECWRD ON THE STACK
2598 014670 004737 022172          JSR      PC,LIN0CT   ;TYPE @ECWRD
2599 014674 104414 052073          DISPLY   ,LINSF      ;SPACES
2600 014700 005737 001336          TST      ECWRD1      ;PRINT THE NEXT WORD ?
2601 014704 001427          BEQ      ECCX        ;BR IF NOT
2602 014706 104414 001165          DISPLY   ,$CRLF      ;CR-LF
2603 014712 013746 001336          MOV      ECWRD1,-(SP) ;PUT ECWRD1 ON THE STACK
2604 014716 004737 022172          JSR      PC,LIN0CT   ;TYPE ECWRD1
2605 014722 104414 052073          DISPLY   ,LINSF      ;SPACES
2606 014726 013746 001342          MOV      ECBAD1,-(SP) ;PUT ECBAD1 ON THE STACK
2607 014732 004737 022172          JSR      PC,LIN0CT   ;TYPE ECBAD1
2608 014736 104414 052073          DISPLY   ,LINSF      ;SPACES
2609 014742 017746 164370          MOV      @ECWRD1,-(SP) ;PUT @ECWRD1 ON THE STACK
2610 014746 004737 022172          JSR      PC,LIN0CT   ;TYPE @ECWRD1
2611 014752 104414 052073          DISPLY   ,LINSF      ;SPACES
2612 014756 000402          BR        ECCX        ;EXIT
2613 014760 104414 051604          ECC2:   DISPLY    ,LIN10C ;ERROR BURST WAS NOT TRANSFERED TO MEMORY
2614 014764 104414 001165          ECCX:   DISPLY    ,$CRLF  ;CR-LF
2615 014770 000207          RTS      PC           ;RETURN
2616
2617
2618          ;ROUTINE TO DISPLAY THE SECTOR WHICH GAVE THE HARD ERROR
2619 014772 032777 000010 164140 PRTBAD: BIT      #SW3,@SWR      ;PRINT THE BAD SECTOR ?
2620 015000 001460          BEQ      6$          ;BR IF NOT
2621 015002 016001 000240          MOV      $RPBA(R0),R1 ;PUT THE END ADDRESS INTO R1
2622 015006 016046 000020          MOV      $WRDL(R0),-(SP) ;FIND THE BEGINNING OF THE SECTOR
2623 015012 066016 000236          ADD      $RPWC(R0),(SP) ;SUBTRACT THE WORDS NOT TRANSFERED
2624 015016 005046          CLR      -(SP)      ;MAKE THE UPPER DIVIDEND 0
2625 015020 016046 000022          MOV      $$SEC(R0),-(SP) ;DIVDE THE WORDS TRANSFERED BY THE SECTOR SIZE
2626 015024 004737 027004          JSR      PC,LINKDV   ;DIVIDE
2627 015030 005716          TST      (SP)      ;REMAINDER = 0 ?
2628 015032 001403          BEQ      1$          ;BR IF IT IS - COMPLETE SECTOR TRANSFERED
2629 015034 006316          ASL      (SP)      ;CONVERT THE RESIDUAL SECTOR SIZE INTO BYTE COUNT
2630 015036 161601          SUB      (SP),R1    ;SUBTRACT IT FROM THE END ADDRESS
2631 015040 000410          BR        2$          ;FINISH THE SIZING
2632 015042 162701 001000 1$:  SUB      #1000,R1    ;SUBTRACT FULL SECTOR SIZE FROM END ADDR
2633 015046 126027 000024 000005  CMPB     $CODE(R0),#5 ;WAS OPERATION READ HEADER & DATA ?
2634 015054 001002          BNE      2$          ;BR IF NOT
2635 015056 162701 000010          SUB      #10,R1     ;SUBTRACT HEADER SIZE FROM ADDR
2636 015062 062706 000004 2$:  ADD      #4,SP      ;RESTORE THE STACK POINTER

```

```

2637 015066 104414 051776          DISPLY ,LIN11H          ;PRINT THE HEADER
2638 015072 012702 000007          3$: MOV #7,R2          ;R2 CONTAINS THE WORDS/LINE COUNT
2639 015076 010146                   MOV R1,-(SP)          ;PUT THE ADDRESS ON THE STACK
2640 015100 004737 022172          JSR PC,LIN0CT         ;TYPE THE ADDRESS
2641 015104 020160 000240          4$: CMP R1,$RPBA(RO)  ;PRINTED ALL THE SECTOR ?
2642 015110 001412                   BEQ 5$                ;BR IF ALL PRINTED
2643 015112 104414 052073          DISPLY ,LINS          ;SPACES
2644 015116 012146                   MOV (R1)+,-(SP)      ;PUT THE DATA ON THE STACK
2645 015120 004737 022172          JSR PC,LIN0CT         ;TYPE THE DATA
2646 015124 005302                   DEC R2                ;DECREMENT THE HORIZONTAL COUNT
2647 015126 001366                   BNE 4$                ;BR IF NOT AT THE END OF THE LINE
2648 015130 104414 001165          DISPLY $CRLF          ;CR-LF
2649 015134 000756                   BR 3$                 ;RESTORE THE WORDS/LINE COUNT
2650 015136 104414 001165          5$: DISPLY $CRLF      ;PRINT WHAT REMAINS IN THE BUFFER
2651 015142 000207                   6$: RTS PC            ;RETURN
2652
2653          ;ROUTINE TO DO AN RTC - DRIVE SELECTED IN RO
2654          ;CALL:
2655          ;       MOV #DPB,RO          ;DPB ADDRESS
2656          ;       JSR PC,RTNCTR
2657          ;       RETURN
2658
2659 015144 111037 044646          RTNCTR: MOVB (RO),GENDPB ;MOVE THE DRIVE # TO THE GENERAL DPB
2660 015150 112737 000117          044650 MOVB #RTC,GENDPB+$COMND ;COMMAND CODE
2661 015156 004037 034036          1$: JSR RO,RPO4        ;DRIVER ENTRANCE
2662 015162 044646                   GENDPB                ;DPB ADDRESS FOR ORDER
2663 015164 000774                   BR 1$                 ;DRIVER DIDN'T ACCEPT ORDER
2664 015166 000207                   RTS PC                ;RETURN
2665
2666          ;ROUTINE TO DO A RECALIBRATE - DRIVE SELECTED IN RO
2667          ;CALL:
2668          ;       MOV #DPB,RO          ;DPB ADDRESS
2669          ;       JSR PC,RECALT
2670          ;       RETURN
2671
2672          ;OR
2673
2674          ;       MOV #DPB,RO          ;DPB ADDRESS
2675          ;       MOVB #DRIVE,GENDPB  ;DRIVE ADDRESS
2676          ;       JSR PC,RECALTO
2677          ;       RETURN
2678
2679 015170 111037 044646          RECALT: MOVB (RO),GENDPB ;MOVE THE DRIVE # TO THE GENERAL DPB
2680 015174 112737 000107          044650 RECALO: MOVB #RECAL,GENDPB+$COMND ;RECALIBRATE COMMAND
2681 015202 004037 034036          1$: JSR RO,RPO4        ;DRIVER ENTRANCE
2682 015206 044646                   GENDPB                ;DPB ADDRESS FOR ORDER
2683 015210 000774                   BR 1$                 ;DRIVER DIDN'T ACCEPT THE ORDER
2684 015212 005737 044664          2$: TST GENDPB+$STATUS ;SEE IF FINISHED
2685 015216 001775                   BEQ 2$                ;BR IF NOT FINISHED
2686 015220 000207                   RTS PC                ;RETURN
2687
2688          ;OFFSET THE DRIVE IN RO (OFFSET CODE PRELOADED INTO 'RPOF')
2689          ;CALL:
2690          ;       MOVB #OFFSET,GENDPB+$FMT ;OFFSET CODE

```



```

2691      :      MOV      #DPB,RO      ;DPB ADDRESS
2692      :      JSR      PC,OFFST
2693      :      RETURN
2694      :
2695 015222 111037 044646      OFFST: MOV      (RO),GENDPB      ;DRIVE # TO GENERAL DPB
2696 015226 112737 000115 044650 MOV      #OFFSET,GENDPB+$COMND ;COMMAND
2697 015234 004037 034036 1$: JSR      RO,RPO4      ;DRIVER ENTRANCE
2698 015240 044646      GENDPB      ;DPB ADDRESS FOR ORDER
2699 015242 000774      BR      1$      ;DRIVER DIDN'T ACCEPT ORDER
2700 015244 000207      RTS      PC
2701
2702      ;UTILITY READ HEADER ROUTINE
2703      ;CALL:
2704      :      MOV      #DPB,RO      ;DPB ADDRESS
2705      :      MOV      #SECTOR,-(SP)    ;SECTOR ADDRESS
2706      :      MOV      #TRACK,-(SP)    ;TRACK ADDRESS
2707      :      JSR      PC,READDR
2708      :      RETURN
2709
2710 015246 116637 000002 044657 READHD: MOV      2(SP),GENDPB+$TRK ;TRACK ADDRESS
2711 015254 116637 000004 044656      MOV      4(SP),GENDPB+$SEC ;SECTOR ADDRESS
2712 015262 111037 044646      MOV      (RO),GENDPB      ;DRIVE NUMBER
2713 015266 016037 000272 044660      MOV      $RPMC(RO),GENDPB+$CYL ;CYLINDER ADDRESS
2714 015274 112737 000173 044650      MOV      #RDHD,GENDPB+$COMND ;COMMAND
2715 015302 004037 034036 1$: JSR      RO,RPO4      ;DRIVER ENTRANCE
2716 015306 044646      GENDPB      ;DPB ADDRESS FOR ORDER
2717 015310 000774      BR      1$      ;DRIVER DIDN'T ACCEPT COMMAND
2718 015312 005737 044664 2$: TST      GENDPB+$STATUS ;FINISHED?
2719 015316 001775      BEQ      2$      ;BR IF NOT
2720 015320 012666 000002      MOV      (SP)+,2(SP)      ;ADJUST STACK FOR RETURN
2721 015324 005726      TST      (SP)+
2722 015326 000207      RTS      PC      ;RETURN
2723
2724      ;RETRY THE PRESENT OPERATION
2725      ;CALL:
2726      :      MOV      #COUNT,RETRY ;RETRY COUNT
2727      :      JSR      PC,$RETRY
2728      :      RETURN1
2729      :      RETURN2
2730      :
2731      :
2732      :
2733 015330 004737 016320      $RETRY: JSR      PC,GODRIV ;RE-START ORDER
2734 015334 005760 000016 1$: TST      $STATUS(RO) ;ORDER FINISHED?
2735 015340 001775      BEQ      1$      ;BR IF NOT
2736 015342 100405      BMI      2$      ;BR IF ERROR
2737 015344 105237 001253      INCB     RETRY+1 ;INCREMENT RETRY COUNT
2738 015350 062716 000002      ADD      #2,(SP) ;INCREMENT RETURN
2739 015354 000425      BR      5$      ;GO TO EXIT
2740 015356 032760 000200 000016 2$: BIT      #BIT7,$STATUS(RO) ;DID ORDER TERMINATE NORMALLY ?
2741 015364 001430      BEQ      7$      ;BR IF NOT
2742 015366 005737 001250      TST      MASK ;IS ERROR MASK 0 ?
2743 015372 001004      BNE      3$      ;BR IF NOT
2744 015374 005760 000250      TST      $RPER1(RO) ;MAKE SURE THAT THE DRIVE ERROR REG IS CLEAR
    
```

```

2745 015400 001014 BNE 6$ ;BR IF NOT
2746 015402 000404 BR 4$ ;CONTINUE RETRY
2747 015404 033760 001250 000250 3$: BIT MASK,$RPER1(RO) ;SAME ERROR?
2748 015412 001407 BEQ 6$ ;BR IF NOT
2749 015414 105237 001253 4$: INCB RETRY+1 ;INCREMENT RETRY COUNT
2750 015420 123737 001252 001253 CMPB RETRY,RETRY+1 ;DONE ?
2751 015426 001340 BNE $RETRY ;BR IF NOT DONE
2752 015430 000207 5$: RTS PC ;RETURN
2753 015432 004737 022160 6$: JSR PC,LINE8 ;REPORT DIFFERENT ERROR
2754 015436 004737 021712 JSR PC,LINE7 ;PRINT LINE 7
2755 015442 005726 TST (SP)+ ;ADJUST STACK POINTER FOR DIRECT RETURN
2756 015444 000207 RTS PC ;RETURN
2757 015446 104414 051246 7$: DISPLY LIN8M ;'DIFFERENT ERROR DURING RETRY'
2758 015452 000137 006574 JMP ERPRC1 ;REPORT THE ERROR
2759
2760 ;ROUTINE TO UPDATE THE PERFORMANCE SUMMARY STATISTICS
2761 ;CALL:
2762 ; MOV #DPB,RO ;DPB ADDRESS
2763 ; JSR PC,STATIS
2764 ; RETURN
2765
2766 015456 032760 000300 000016 STATIS: BIT #BIT07!BIT06,$STATUS(RO) ;CHECK FOR DATA TERMINATION
2767 015464 001454 BEQ 3$ ;BR IF NOT DATA TERMINATION
2768 015466 016037 000240 015620 MOV $RPBA(RO),FACTOR ;STORE THE FINAL BUFFER ADDRESS
2769 015474 166037 000006 015620 SUB $BUF(RO),FACTOR ;SUBTRACT THE INITIAL ADDRESS
2770 015502 001434 BEQ 2$ ;BR IF NO DATA TRANSFER
2771 015504 006237 015620 ASR FACTOR ;CONVERT TO A WORD COUNT
2772 015510 063760 015620 000046 ADD FACTOR,$TRANS(RO) ;UPDATE WORD COUNT
2773 015516 005560 000050 ADC $TRANS+2(RO) ;ADD ANY CARRY
2774 015522 132760 000002 000024 BITB #BIT01,$CODE(RO) ;SEE IF ORDER READ OR WRITE
2775 015530 001021 BNE 2$ ;BRANCH IF ORDER WRITE
2776 015532 005737 001404 TST AUTOCK ;AUTO WRITE CHECKS BEING PERFORMED
2777 015536 001411 BEQ 1$ ;BR IF NOT
2778 015540 126027 000024 000001 CMPB $CODE(RO),#1 ;PRESENT OPERATION AN AUTOMATIC WRITE CHECK ?
2779 015546 101005 BHI 1$ ;BR IF NOT
2780 015550 066060 000020 000046 ADD $WRDL(RO),$TRANS(RO) ;ADD WORDS WRITTEN
2781 015556 005560 000050 ADC $TRANS+2(RO) ;ADD A CARRY
2782 015562 063760 015620 000052 1$: ADD FACTOR,$READ(RO) ;UPDATE THE READ WORD COUNT
2783 015570 005560 000054 ADC $READ+2(RO) ;ADD ANY CARRY
2784 015574 026060 000012 000270 2$: CMP $CYL(RO),$RPCA(RO) ;DID MID-TRANSFER SEEK OCCUR
2785 015602 001405 BEQ 3$ ;BR IF NOT
2786 015604 062760 000001 000042 ADD #1,$POSIT(RO) ;INCREMENT SEEK COUNT
2787 015612 005560 000044 ADC $POSIT+2(RO) ;ADD CARRY TO UPPER WORD
2788 015616 000207 3$: RTS PC
2789
2790 015620 000000 FACTOR: .WORD 0 ;USED FOR WORDS TRANSFERED
2791
2792 ;ROUTINE TO GET A BUFFER
2793 ;CALL:
2794 ; MOV #DPB,RO ;DPB ADDRESS
2795 ; CLR -(SP) ;CLEAR THE STACK
2796 ; JSR PC,GETBUF
2797 ; RETURN ;BUFFER ADDRESS WILL BE ON THE STACK
2798 ; ;STACK WILL BE ZERO IF NO BUFFER AVAILABLE
    
```



```

015622 010146 GETBUF: MOV R1,-(SP) ;SAVE R1
015624 010246 MOV R2,-(SP) ;SAVE R2
015626 010346 MOV R3,-(SP) ;SAVE R3
015630 013702 001576 MOV BUFTBL,R2 ;NUMBER OF SEPARATE BUFFERS
015634 001444 BEQ 6$ ;BR IF NONE AVAILABLE
015636 012701 001600 MOV #BUFTBL+2,R1 ;FIRST ADDRESS OF ALLOCATION TABLE
015642 026061 000020 000002 1$: CMP $WRDL(R0),2(R1) ;SEE IF THERE IS A BLOCK LARGE ENOUGH
015650 101405 BLOS 3$ ;BRANCH IF IT IS
015652 005302 DEC R2 ;DECREMENT TABLE COUNT
015654 001434 BEQ 6$ ;BR IF THROUGH TABLE
015656 062701 000004 ADD #4,R1 ;INCREMENT TABLE POINTER
015662 000767 BR 1$ ;CONTINUE LOOKING
015664 011166 000010 3$: MOV (R1),10(SP) ;BUFFER ADDRESS TO STACK
015670 166061 000020 000002 SUB $WRDL(R0),2(R1) ;ADJUST BUFFER SIZE
015676 001407 BEQ 4$ ;BR IF DIFFERENCE IS ZERO
015700 006360 000020 ASL $WRDL(R0) ;CONVERT # WORDS TO BYTES
015704 066011 000020 ADD $WRDL(R0),(R1) ;MAKE NEW STARTING ADDRESS
015710 006260 000020 ASR $WRDL(R0) ;RETURN # BYTES TO WORDS
015714 000414 BR 6$ ;RETURN
015716 005337 001576 4$: DEC BUFTBL ;DECREMENT ENTRIES COUNT
015722 001411 BEQ 6$ ;BR IF ALLOCATION TABLE EMPTY
015724 005302 DEC R2 ;DECREMENT TABLE COUNT
015726 001407 BEQ 6$ ;BR IF ITEM WERE LAST ENTRY
015730 010103 MOV R1,R3 ;MOVE TABLE POINTER
015732 062703 000004 ADD #4,R3 ;POINT TO NEXT ENTRY
015736 012321 5$: MOV (R3)+,(R1)+ ;MOVE ITEMS
015740 012321 MOV (R3)+,(R1)+
015742 005302 DEC R2 ;DECREMENT TABLE COUNT
015744 001374 BNE 5$ ;CONTINUE IF NOT AT END OF TABLE
015746 012603 6$: MOV (SP)+,R3 ;RESTORE R3
015750 012602 MOV (SP)+,R2 ;RESTORE R2
015752 012601 MOV (SP)+,R1 ;RESTORE R1
015754 000207 RTS PC ;RETURN

```

:ROUTINE TO PUT BUFFER BACK IN TABLE

```

:CALL:
: MOV #DPB,R0 ;DPB ADDRESS
: JSR PC,RELBUF
: RETURN

```

```

015756 010146 RELBUF: MOV R1,-(SP) ;SAVE R1
015760 012701 001600 MOV #BUFTBL+2,R1 ;BEGINNING OF TABLE
015764 013702 001576 MOV BUFTBL,R2 ;ENTRY COUNT
015770 001424 BEQ 2$ ;BR IF EMPTY TABLE
015772 016003 000020 MOV $WRDL(R0),R3 ;TRIAL ADDRESS
015776 006303 ASL R3 ;CHANGE TO BYTE COUNT
016000 066003 000006 ADD $BUF(R0),R3 ;ADDRESS OF HIGHER ADJACENT BLOCK
016004 021103 1$: CMP (R1),R3 ;UPPER ADJACENT BLOCK
016006 001424 BEQ 4$ ;BR IF YES
016010 062701 000004 ADD #4,R1 ;INCREMENT POINTER
016014 005302 DEC R2 ;DECREMENT ENTRY COUNT
016016 001372 BNE 1$ ;CONTINUE SEARCHING

```



```

28953 016020 016011 000006      MOV      $BUF(R0), (R1) ;PUT THE BUFFER BLOCK INTO THE TABLE
28954 016024 016061 000020 000002      MOV      $WRDL(R0), 2(R1) ;BLOCK SIZE
28955 016032 005237 001576      INC      BUFTBL ;INCREMENT ENTRY COUNT
28956 016036 005202      INC      R2 ;INCREMENT R2 FOR USE LATER
28957 016040 000414      BR      5$ ;SEE IF A LOWER ADJACENT BLOCK IS IN THE TABLE
28958 016042 016021 000006 2$:      MOV      $BUF(R0), (R1)+ ;BLOCK ADDRESS TO TABLE
28959 016046 016021 000020      MOV      $WRDL(R0), (R1)+ ;SIZE TO TABLE
28960 016050 005237 001576      INC      BUFTBL ;INCREMENT ENTRY COUNT
28961 016056 000443      BR      10$ ;EXIT
28962 016060 016011 000006 4$:      MOV      $BUF(R0), (R1) ;RELEASED BUFFER IS LOWER ADJACENT
28963 016064 066061 000020 000002      ADD      $WRDL(R0), 2(R1) ;INCREMENTED SIZE
28964 016072 010246 5$:      MOV      R2, -(SP) ;SAVE R2
28965 016074 013702 001576      MOV      BUFTBL, R2 ;ENTRY COUNT
28966 016100 012705 001600      MOV      #BUFTBL+2, R5 ;BEGINNING OF TABLE
28967 016104 016504 000002 6$:      MOV      2(R5), R4 ;BLOCK SIZE (IN WORDS)
28968 016110 006304      ASL      R4 ;CHANGE TO BYTE COUNT
28969 016112 061504      ADD      (R5), R4 ;ADD BLOCK BEGINNING ADDRESS
28970 016114 020411      CMP      R4, (R1) ;R1 STILL POINTS TO INSERTED ENTRY
28971 016116 001406      BEQ      8$ ;LOWER ADJACENT IN TABLE
28972 016120 062705 000004      ADD      #4, R5 ;INCREMENT POINTER
28973 016124 005302      DEC      R2 ;DECREMENT ENTRY COUNT
28974 016126 001366      BNE      6$ ;CONTINUE LOOKING
28975 016130 005726      TST      (SP)+ ;RESTORE STACK POINTER
28976 016132 000415      BR      10$ ;END
28977 016134 012602 8$:      MOV      (SP)+, R2 ;RESTORE R2
28978 016136 066165 000002 000002      ADD      2(R1), 2(R5) ;INCREMENT LOWER BLOCK LENGTH
28979 016144 005337 001576      DEC      BUFTBL ;DECREMENT ENTRY COUNT
28980 016150 010105      MOV      R1, R5 ;GET READY TO COMPRESS
28981 016152 062705 000004      ADD      #4, R5 ;INCREMENT TO NEXT ENTRY
28982 016156 012521 9$:      MOV      (R5)+, (R1)+ ;COMPRESS TABLE
28983 016160 012521      MOV      (R5)+, (R1)+ ;MOVE SIZE FIELD DOWN
28984 016162 005302      DEC      R2 ;DECREMENT ENTRY COUNT
28985 016164 001374      BNE      9$ ;BR IF NOT FINISHED
28986 016166 012601 10$:     MOV      (SP)+, R1 ;RESTORE R1
28987 016170 000207      RTS      PC ;RETURN

```

;FILL THE ASSIGNED BUFFER (IF WRITE OR WRITE CHECK ORDER)

;CALL:

```

:      MOV      #DPB, R0 ;DPB ADDRESS
:      MOV      #BUFADR, $BUF(R0) ;LOAD BUFFER ADDRESS INTO THE DPB
:      MOVB     #PATTERN, $PATTC(R0) ;PATTERN CODE
:      JSR      PC, FILBUF
:      RETURN

```

FILBUF: SAVREG

```

28998 016172 104412      SAVREG ;SAVE THE REGISTERS
28999 016174 132760 000004 000024      BITB     #BIT02, $CODE(R0) ;SEE IF READ ORDER
29000 016202 001044      BNE      4$ ;BR IF READ
29001 016204 016001 000006 1$:      MOV      $BUF(R0), R1 ;BUFFER ADDRESS
29002 016210 016002 000020      MOV      $WRDL(R0), R2 ;POSITIVE WORD COUNT
29003 016214 132760 000001 000024      BITB     #BIT00, $CODE(R0) ;SEE IF WRITE HEADER TYPE ORDER
29004 016222 001413      BEQ      2$ ;BR IF NOT
29005 016224 016011 000012      MOV      $CYL(R0), (R1) ;CYLINDER ADDRESS
29006 016230 052721 010000      BIS      #BIT12, (R1)+ ;SET FMT22 BIT

```



```

2961 016434 000406          BR      3$          ;CONTINUE WITH THE SELECTION
2962 016436 013705 032402 2$: MOV    $LONUM,R5 ;SELECT READ OPERATION CODE
2963 016442 042705 177776   BIC    #1C1,R5     ;MASK OUT ALL BUT BIT 0
2964 016446 062705 000004   ADD    #4,R5       ;TABLE OFFSET FOR READ CODE
2965 016452 110560 000074 3$: MOVB  R5,$NCODE(RO) ;ORDER SELECTION CODE TO CONTROL BLOCK

;GENERATE A RANDOM SECTOR ADDRESS BETWEEN VALUES 'MINSEC' & 'MAXSEC'
2966
2967
2968
2969
2970 016456 016005 000116 RANSEC: MOV    MAXSEC(RO),R5 ;GET MAXIMUM SECTOR ADDRESS
2971 016462 026005 000120   CMP    MINSEC(RO),R5 ;'MINSEC' AND 'MAXSEC' THE SAME ?
2972 016466 001417         BEQ    2$          ;BR IF THEY ARE
2973 016470 166005 000120   SUB    MINSEC(RO),R5 ;SUBTRACT MINIMUM SECTOR ADDRESS
2974 016474 100002         BPL    1$          ;BR IF MAX LARGER THAN MIN
2975 016502 005205         ADD    #22.,R5     ;CORRECT THE NUMBER
2976 016504 004737 026756 1$: INC    R5        ;INCREMENT DIFFERENCE TO USE AS DIVISOR
2977 016510 066005 000120   JSR    PC,GETREM   ;GET THE RANDOM AUGMENT
2978 016514 020527 000025   ADD    MINSEC(RO),R5 ;NEW ADDRESS
2979 016520 101402         CMP    R5,#21.    ;IS VALUE TOO LARGE ?
2980 016522 162705 000026   BLOS  2$          ;BR IF NOT
2981 016526 110560 000076   SUB    #22.,R5     ;CORRECT VALUE
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001 016606 012737 000633 001350   MOV    #411,CYLIMT ;ASSUME AN RPO4/5
3002 016614 032760 000002 000262   BIT    #BIT01,$RPDT(RO) ;SEE IF RPO6
3003 016622 001403         BEQ    RANCYL     ;BR IF NOT
3004 016624 012737 001457 001350   MOV    #815.,CYLIMT ;CHANGE CYLINDER LIMIT
3005 016632 016005 000106 RANCYL: MOV    MAXCYL(RO),R5 ;GET MAXIMUM CYLINDER ADDRESS
3006 016636 026005 000110   CMP    MINCYL(RO),R5 ;'MINCYL' AND 'MAXCYL' THE SAME ?
3007 016642 001417         BEQ    2$          ;BR IF THEY ARE
3008 016644 166005 000110   SUB    MINCYL(RO),R5 ;SUBTRACT MINIMUM CYLINDER ADDRESS
3009 016650 100002         BPL    1$          ;BR IF MAX LARGER THAN MIN
3010 016652 063705 001350   ADD    CYLIMT,R5   ;CORRECT THE NUMBER
3011 016656 005205         INC    R5        ;INCREMENT DIFFERENCE TO USE AS DIVISOR
3012 016660 004737 026756 1$: JSR    PC,GETREM   ;GET THE RANDOM AUGMENT
3013 016664 066005 000110   ADD    MINCYL(RO),R5 ;NEW CYLINDER ADDRESS
3014 016670 023705 001350   CMP    CYLIMT,R5  ;IS VALUE TOO LARGE ?
    
```



```

3015 016674 101002          BHI      2$          ;BR IF NOT
3016 016676 163705 001350  SUB      CYLIMT,R5   ;CORRECT VALUE
3017 016702 010560 000100 2$: MOV     R5,$N$CYL(RO) ;STORE CYLINDER ADDRESS IN DPB
3018 016706 122760 000003 000074 CMPB    #3,$N$CODE(RO) ;WRITE HEADER & DATA ?
3019 016714 001013          BNE      RANSIZ     ;BR IF NOT
3020 016716 012760 000404 000102 MOV     #260,$N$WRDL(RO) ;CHANGE WORD LENGTH TO 260 FOR WRTHD ORDER
3021 016724 023727 001364 000404 CMP     MAXDL,#260.   ;CAN A FULL SECTOR BE WRITTEN ?
3022 016732 103062          BHIS    RANPAT      ;BR IF IT CAN
3023 016734 013760 001364 000102 MOV     MAXDL,$N$WRDL(RO) ;CHANGE TRANSFER SIZE
3024 016742 000456          BR       RANPAT     ;CONTINUE WITH THE SELECTION
3025
3026 ;GENERATE A RANDOM BUFFER LENGTH BETWEEN 4 & THE VALUE IN 'MAXDL'
3027
3028 016744 013705 001364  RANSIZ: MOV    MAXDL,R5 ;GET BUFFER SIZE
3029 016750 005737 001400  TST     WCSEL      ;SELECT A RANDOM WORD COUNT ?
3030 016754 001010          BNE     1$         ;BR IF NOT
3031 016756 005205          INC     R5         ;INCREMENT THE MAXIMUM SIZE
3032 016760 004737 026756  JSR     PC,GETREM  ;DIVIDE BY MAX VALUE
3033 016764 005705          TST     R5         ;IS THE REMAINDER 0 ?
3034 016766 001003          BNE     1$         ;NOT 0, CONTINUE
3035 016770 004737 032302  JSR     PC,$RAND   ;CYCLE THE RANDOM NUMBER GENERATOR
3036 016774 000763          BR      RANSIZ    ;TRY AGAIN
3037 016776 010560 000102 1$: MOV     R5,$N$WRDL(RO) ;WORD LENGTH TO CONTROL BLOCK
3038 017002 010546          MOV     R5,-(SP)  ;NEW WORD LENGTH ON STACK FOR CHECK
3039 017004 005046          CLR    -(SP)     ;MAKE UPPER DIVIDEND ZERO
3040 017006 012746 000400  MOV     #256,-(SP) ;SECTOR SIZE IS THE DIVISOR
3041 017012 132760 000001 000074 BITB    #1,$N$CODE(RO) ;SEE IF NEXT ORDER IS A HEADER ORDER
3042 017020 001402          BEQ    2$         ;BR IF NOT
3043 017022 062716 000004  ADD     #4,(SP)   ;ADD HEADER SIZE TO SECTOR SIZE
3044 017026 004737 027004 2$: JSR     PC,LINKDV ;DIVIDE BUFFER SIZE BY SECTOR SIZE
3045 017032 012616          MOV     (SP)+,(SP) ;MOV REMAINDER UP THE STACK
3046 017034 021627 000004  CMP     (SP),#4 ;SEE IF REMAINDER LESS THAN 4
3047 017040 103012          BHIS   4$         ;BR IF NOT
3048 017042 005737 001400  TST     WCSEL     ;SELECTING RANDOM TRANSFER SIZES ?
3049 017046 001403          BEQ    3$         ;BR IF YES
3050 017050 161660 000102  SUB     (SP),$N$WRDL(RO) ;ADJUST WORD LENGTH DOWNWARD
3051 017054 000404          BR     4$         ;CONTINUE
3052 017056 005726          TST    (SP)+     ;CORRECT THE STACK POINTER
3053 017060 004737 032302  JSR     PC,$RAND   ;CYCLE THE RANDOM NUMBER GENERATOR
3054 017064 000727          BR     RANSIZ    ;TRY AGAIN
3055 017066 005726          TST    (SP)+     ;CORRECT THE STACK POINTER
3056 017070 122760 000002 000074 CMPB    #2,$N$CODE(RO) ;SEE IF WRITE DATA
3057 017076 001004          BNE    RANXIT    ;BR IF NOT WRITE DATA
3058
3059 ;GET A RANDOM PATTERN NUMBER
3060
3061 017100 004737 017224  RANPAT: JSR    PC,GETPAT ;GET PATTERN CODE
3062 017104 110560 000075  MOVB   R5,$N$PATC(RO) ;MOVE PATTERN CODE TO CONTROL BLOCK
3063 017110 012760 177777 000104 RANXIT: MOV    #-1,$N$NEXT(RO) ;SET PARAMETERS SELECTED INDICATOR
3064 017116 000207          RTS     PC        ;RETURN
3065
3066 ;ROUTINE TO SELECT A WRITE (OR WRITE CHECK) OPERATION.
3067
3068 017120 012705 000004  RANWRT: MOV    #4,R5 ;WRITE OPERATION SELECTION DIVISOR

```

```

3069 017124 004737 026756 JSR PC,GETREM ;GET SELECTION CODE
3070 017130 005737 001404 TST AUTOCK ;ARE WRITE CHECK ORDERS TO BE SELECTED
3071 ;RANDOMLY ?
3072 017134 001403 BEQ 1$ ;BR IF THEY ARE
3073 017136 152705 000002 BISS #2,R5 ;SET CODE TO EXCLUDE WRITE CHECK ORDERS
3074 017142 000420 BR 3$ ;COMPLETE SELECTION
3075 017144 020527 000001 1$: CMP R5,#1 ;WRITE CHECK SELECTED ?
3076 017150 101015 BHI 3$ ;BR IF NOT
3077 017152 132760 000002 000024 BITB #2,$CODE(RO) ;PREVIOUS WRITE OPERATION ?
3078 017160 001407 BEQ 2$ ;BR IF PREVIOUS WAS READ OR WRITE CHECK
3079 017162 116060 000024 000074 MOVB $CODE(RO),$NCODE(RO) ;MOVE CODE TO 'NEXT CODE'
3080 017170 142760 000002 000074 BICB #2,$NCODE(RO) ;CHANGE WRITE TO WRITE CHECK
3081 017176 000411 BR 5$ ;EXIT
3082 017200 052705 000002 2$: BIS #2,R5 ;CHANGE WRITE CHECK TO WRITE
3083 017204 005737 001376 3$: TST FORMAT ;WRITE HEADER ORDERS ALLOWED ?
3084 017210 001002 BNE 4$ ;BR IF THEY ARE
3085 017212 042705 000001 BIC #1,R5 ;ALTER POSSIBLE WRITE HEADER
3086 017216 110560 000074 4$: MOVB R5,$NCODE(RO) ;SETUP 'NEXT' CODE
3087 017222 000207 5$: RTS PC ;RETURN
3088
3089 ;ROUTINE TO SELECT A PATTERN
3090
3091 017224 012705 000020 GETPAT: MOV #20,R5 ;SELECT PATTERN
3092 017230 004737 026756 JSR PC,GETREM ;GET CODE
3093 017234 005705 TST R5 ;WAS PATTERN ZERO SELECTED ?
3094 017236 001003 BNE 1$ ;BR IF NOT ZERO
3095 017240 004737 032302 JSR PC,$RAND ;CYCLE THE RANDOM NUMBER GENERATOR
3096 017244 000767 BR GETPAT ;TRY AGAIN
3097 017246 006305 1$: ASL R5 ;MAKE CODE INTO TABLE INDEX
3098 017250 000207 RTS PC
3099
3100 ;ROUTINE TO GET THE PREVIOUSLY SELECTED PARAMETER VALUES
3101 ;CALL:
3102 ;
3103 ; MOV #DPB,RO ;DPB ADDRESS
3104 ; JSR PC,SELPAR ;SELECT THE PARAMETERS
3105 ; JSR PC,GETPAR
3106 ; RETURN
3107
3107 017252 010546 GETPAR: MOV R5,-(SP) ;SAVE R5
3108 017254 116060 000234 000027 MOVB $RPCS1(RO),$PREV0(RO) ;SAVE CURRENT PARAMETERS
3109 017262 032760 000006 000074 BIT #6,$NCODE(RO) ;SEE IF NEXT OPERATION IS READ OR WRITE
3110 017270 001007 BNE 1$ ;BR IF EITHER
3111 017272 016060 000012 000034 MOV $CYL(RO),$PREVA+2(RO) ;SAVE STARTING CYLINDER
3112 017300 016060 000010 000032 MOV $SEC(RO),$PREVA(RO) ;SAVE STARTING SECTOR AND TRACK
3113 017306 000411 BR 2$
3114 017310 004737 022244 1$: JSR PC,READR ;GET THE DECREMENTED SECTOR AND TRACK ADDRESSES
3115 017314 112660 000033 MOVB (SP)+,$PREVA+1(RO) ;TRACK ADDRESS
3116 017320 112660 000032 MOVB (SP)+,$PREVA(RO) ;SECTOR ADDRESS
3117 017324 016060 000272 000034 MOV $RPCC(RO),$PREVA+2(RO) ;CURRENT CYLINDER
3118 017332 032777 000100 161600 2$: BIT #SW06,$SWR ;SWITCH 6 SET ?
3119 017340 001043 BNE 3$ ;BR IF SET
3120 017342 116060 000074 000024 MOVB $NCODE(RO),$CODE(RO) ;LOGICAL CODE FOR OPERATION
3121 017350 116005 000074 MOVB $NCODE(RO),R5 ;LOAD R5 FOR USE AS TABLE INDEX
3122 017354 116560 001740 000002 MOVB COMTBL(R5),$COMND(RO) ;RPO4 COMMAND CODE

```

Handwritten notes and signatures on the right margin, including a large '3' and some illegible scribbles.

H08

```

3123 017362 116060 000075 000030      MOVB   $NPATC(RO), $PATT(RO) ; PATTERN CODE
3124 017370 016060 000076 000010      MOV    $NSEC(RO), $SEC(RO)  ; TRACK AND SECTOR ADDRESSES
3125 017376 016060 000100 000012      MOV    $NCYL(RO), $CYL(RO)  ; CYLINDER ADDRESS
3126 017404 016060 000102 000020      MOV    $NWRDL(RO), $WRDL(RO) ; BUFFER SIZE
3127 017412 016060 000102 000004      MOV    $NWRDL(RO), $WRDM(RO) ; WORD COUNT FOR THE RH11
3128 017420 005460 000004      NEG    $WRDM(RO)           ; COMPLEMENT IT
3129 017424 012760 000400 000022      MOV    #256, $SSEC(RO)     ; INITIAL VALUE OF SECTOR SIZE
3130 017432 032760 000001 000024      BIT    #1, $CODE(RO)       ; HEADER OPERATION ?
3131 017440 001403      BEQ    3$,                  ; BR IF NOT
3132 017442 062760 000004 000022      ADD    #4, $SSEC(RO)       ; ADD HEADER SIZE
3133 017450 005060 000104      CLR    $NEXT(RO)          ; RESET 'PARAMETERS LOADED' INDICATOR
3134 017454 012605      MOV    (SP)+, R5          ; RESTORE R5
3135 017456 000207      RTS    PC                 ; RETURN

; ROUTINE TO COMPRESS A LIST
3137      :CALL:
3138      :
3139      :       MOV    #ADDRS, R1      ; COMPRESS LIST STARTING AT THIS ADDRESS
3140      :       JSR    PC, CMPRES
3141      :       RETURN
3142      :
3143 017460 016111 000002      CMPRES: MOV    2(R1), (R1)      ; COMPRESS THE TABLE IN R1
3144 017464 001403      BEQ    1$,                  ; BR WHEN ZERO FOUND
3145 017466 062701 000002      ADD    #2, R1              ; INCREMENT R1
3146 017472 000772      BR    CMPRES              ; CONTINUE COMPRESSING TABLE
3147 017474 000207      1$:    RTS    PC           ; RETURN

; ROUTINE TO SETUP PARAMETERS FOR A SEQUENTIAL READ OR WRITE OF THE DISK
3149      :CALL:
3150      :
3151      :       MOV    #DPB, RO      ; DPB ADDRESS
3152      :       MOV    #-1, $PACK(RO) ; 'WRITE PACK' FLAG
3153      :       OR
3154      :       MOV    #1, $PACK(RO)  ; 'READ PACK' FLAG
3155      :       JSR    PC, WRTPK
3156      :       RETURN
3157      :
3158 017476 004737 032302      WRTPK: JSR    PC, $RAND        ; CYCLE THE RANDOM NUMBER GENERATOR
3159 017502 005760 000040      TST    $OPERC+2(RO)        ; SEE IF FIRST OPERATION
3160 017506 001007      BNE    WRTPK1              ; BR IF UPPER WORD OF COUNTER NOT ZERO
3161 017510 005760 000036      TST    $OPERC(RO)         ; LOWER WORD ZERO ?
3162 017514 001004      BNE    WRTPK1              ; BR IF NOT 1ST OPERATION
3163 017516 105760 000026      TSTB   $PACK(RO)          ; SEE WHICH - 'R' OR 'W'
3164 017522 100503      BMI    WRTPK3              ; BR IF 'W'
3165 017524 000470      BR    WRTPK2              ; 'R' OPERATION
3166 017526 116060 000234 000027      WRTPK1: MOVB   $RPCS1(RO), $PREV0(RO) ; SAVE CURRENT PARAMETERS
3167 017534 004737 022244      JSR    PC, READR          ; GET THE DECREMENTED SECTOR AND TRACK ADDRESSES
3168 017540 112660 000033      MOVB   (SP)+, $PREVA+1(RO) ; TRACK ADDRESS
3169 017544 112660 000032      MOVB   (SP)+, $PREVA(RO)   ; SECTOR ADDRESS
3170 017550 016060 000272 000034      MOV    $RPCC(RO), $PREVA+2(RO) ; CURRENT CYLINDER
3171 017556 016060 000242 000010      MOV    $RPDA(RO), $SEC(RO) ; NEW SECTOR & TRACK ADDRESS
3172 017564 016060 000270 000012      MOV    $RPCA(RO), $CYL(RO) ; NEW CYLINDER ADDRESS
3173 017572 026060 000012 000106      CMP    $CYL(RO), MAXCYL(RO) ; SEE IF AT END
3174 017600 103427      BLO    2$,                  ; BR IF LESS THAN 'MAXCYL'
3175 017602 101004      BHI    1$,                  ; BR IF GREATER THAN 'MAXCYL'
3176 017604 126060 000011 000112      CMPB   $TRK(RO), MAXTRK(RO) ; SEE IF AT MAX TRACK
  
```

```

3177 017612 101422          BLOS      2$          ;BR IF NOT GREATER
3178 017614 116060 000114 000011 1$:      MOVB     MINTRK(RO),$STRK(RO) ;RESET TRACK ADDRESS
3179 017622 116060 000120 000010          MOVB     MINSEC(RO),$SECC(RO) ;RESET SECTOR ADDRESS
3180 017630 016060 000110 000012          MOV      MINCYL(RO),$CYL(RO) ;RESET CYLINDER ADDRESS
3181 017636 004737 026556          JSR      PC,EOP2          ;DROP THE DRIVE (NORMAL TERMINATION)
3182 017642 032777 000020 161270          BIT      #SW04,$SWR      ;IS SWITCH 4 SET ?
3183 017650 001003          BNE      2$          ;BR IF SET
3184 017652 005726          TST     (SP)+          ;INCREMENT THE STACK POINTER
3185 017654 000137 005324          JMP      MAIN          ;RETURN DIRECTLY TO 'MAIN'
3186 017660 013760 001364 000020 2$:      MOV      MAXDL,$WRDL(RO) ;BUFFER SIZE IS MAXIMUM
3187 017666 013760 001364 000004          MOV      MAXDL,$WRDM(RO) ;WORD COUNT
3188 017674 005460 000004          NEG      $WRDM(RO)      ;CHANGE WORD COUNT TO 2'S COMPLEMENT
3189 017700 105760 000026          TSTB    $PACK(RO)      ;READ OR WRITE ?
3190 017704 100412          BMI     WRTPK3         ;BR IF WRITE
3191 017706 012760 000404 000022 WRTPK2: MOV      #260,$SSEC(RO) ;SECTOR SIZE FOR READ
3192 017714 112760 000005 000024          MOVB    #5,$CODE(RO)  ;CODE FOR READ HEADER & DATA
3193 017722 112760 000173 000002          MOVB    #RDHD,$COMND(RO) ;DRIVE CODE FOR OPERATION
3194 017730 000415          BR      WRTPK4         ;SET UP FOR EXIT
3195 017732 012760 000400 000022 WRTPK3: MOV      #256,$SSEC(RO) ;SECTOR SIZE
3196 017740 112760 000002 000024          MOVB    #2,$CODE(RO)  ;CODE FOR WRTDAT
3197 017746 112760 000161 000002          MOVB    #WRTDAT,$COMND(RO) ;OP CODE
3198 017754 004737 017224          JSR      PC,GETPAT     ;GET PATTERN CODE
3199 017760 110560 000030          MOVB    RS,$PATTC(RO) ;PATTERN CODE
3200 017764 005060 000104          WRTPK4: CLR     $NEXT(RO) ;CLEAR 'PARAMETER SELECTED' INDICATOR
3201 017770 000207          RTS      PC           ;RETURN
3202
3203 ;ROUTINE TO DETERMINE OF ERROR IS AT A LOCATION ON THE PACK DEFINED
3204 ; IN THE BAD TRACK/SECTOR TABLE FOR THE DRIVE.
3205 ;CALL:
3206 ;
3207 ; JSR      PC,SPOTCK
3208 ; RETURN1 ;ERROR AT AN ADDRESS IN TABLE
3209 ; RETURN2 ;NO TABLE ENTRY FOR ERROR ADDRESS OR
3210 ; ;PARAMETER 'NOTPRT' IS 0
3211 SPOTCK:
3212 017772 010146          MOV      R1,-(SP)      ;:PUSH R1 ON STACK
3213 017774 010246          MOV      R2,-(SP)      ;:PUSH R2 ON STACK
3214 017776 012701 000124          MOV      #SBDSEC,R1   ;:INCREMENT FOR BAD SECTOR TABLE
3215 020002 060001          ADD      RO,R1        ;:ADD THE BLOCK'S STARTING ADDRESS
3216 020004 012702 000010          MOV      #8,R2        ;:BAD SECTOR TABLE SIZE COUNT
3217 020010 021160 000272 1$:      CMP      (R1),$RPCC(RO) ;:IS CYLINDER IN THE TABLE ?
3218 020014 00102?          BNE      4$          ;:BR IF NOT
3219 020016 105761 000003          TSTB    3(R1)        ;:TRACK ENTRY ?
3220 020022 100426          BMI     5$          ;:BR IF NOT
3221 020024 004737 022244          JSR      PC,READDR    ;:DECREMENT THE SECTOR/TRACK ADDRESS
3222 020030 122661 000003          CMPB    (SP)+,3(R1)   ;:COMPARE THE TRACK ADDRESS
3223 020034 001011          BNE      3$          ;:BR IF IT IS NOT EQUAL
3224 020036 105761 000002          TSTB    2(R1)        ;:IS A SECTOR ADDRESS IN THE TABLE ?
3225 020042 100002          BPL     2$          ;:BR IF ONE IS
3226 020044 005726          TST     (SP)+          ;:INCREMENT THE STACK POINTER
3227 020046 000414          BR      5$          ;:DISPLAY THE MESSAGE
3228 020050 122661 000002 2$:      CMPB    (SP)+,2(R1)   ;:COMPARE THE SECTOR ADDRESS
3229 020054 001002          BNE      4$          ;:BR IF NOT EQUAL
3230 020056 000410          BR      5$          ;:CHECK 'NOTPRT'

```



```

3231 020060 005726          3$:   TST      (SP)+      ; INCREMENT THE STACK POINTER
3232 020062 062701 000004  4$:   ADD      #4,R1      ; GO TO THE NEXT LOCATION IN THE TABLE
3233 020066 005711          TST      (R1)         ; PAST THE TABLE ENTRIES ?
3234 020070 100411          BMI      6$           ; BR IF PAST
3235 020072 005302          DEC      R2           ; DECREMENT THE MAXIMUM ENTRY COUNT
3236 020074 001345          BNE      1$           ; BR IF MORE TO CHECK
3237 020076 000406          BR       6$           ; END, EXIT
3238 020100 005737 001406  5$:   TST      NOTPRT      ; PRINT THE ERROR ANYWAY ?
3239 020104 001006          BNE      7$           ; BR IF NOT
3240 020106 012737 177777 001264  MOV     #-1,BADSEC    ; SET THE INDICATOR FOR THE IDENTIFICATION LINE
3241 020114 062766 000002 000004  6$:   ADD      #2,4(SP)   ; INCREMENT THE RETURN
3242 020122
3243 020122 012602          MOV     (SP)+,R2     ; POP STACK INTO R2
3244 020124 012601          MOV     (SP)+,R1     ; POP STACK INTO R1
3245 020126 000207          RTS      PC          ; RETURN

```

;;*****

.SBTTL ERROR MESSAGE GENERATION ROUTINES

;;*****

;PRINT LINE 1 OF ERROR MESSAGE:

; 'HH:MM:SS'

```

3256 020130 032777 002000 161002 LINE1: BIT      #SW10,DSWR    ; SWITCH 10 SET ?
3257 020136 001402          BEQ      1$           ; BR IF NOT
3258 020140 104401 001160          TYPE     ,SBELL      ; RING THE BELL
3259 020144 032777 020000 160766 1$:   BIT      #SW13,DSWR    ; INHIBIT TYPEOUT ?
3260 020152 001403          BEQ      2$           ; BR IF NOT
3261 020154 104414 001165          DISPLY   ,$CRLF      ; CR-LF
3262 020160 000404          BR       3$           ; EXIT
3263 020162 004737 023400  2$:   JSR      PC,$TIME    ; TYPE THE TIME
3264 020166 104414 052074          DISPLY   ,LINSPO     ; SPACES
3265 020172 000207          RTS      PC          ; RETURN & TYPE DESCRIPTION

```

;PRINT LINE 2 OF ERROR MESSAGE

; 'PRESENT ORDER = XXXX PREVIOUS ORDER = XXXX'

; '* ERROR AT BAD TRACK/SECTOR'

; 'DRV RPCS1 RPCS2 RPDS1 RPER1 RPER2 RPER3 RPEC1 RPEC2'

; 'RPWC RPBA RPDA RPAS RPLA RPDB RPMR RPDT'

; 'RPSN RPOF RPCA RPCC STATUS'

; 'BUS ADDRESS OR WORD COUNT NOT CONSISTENT'

; 'RPBA = XXXXXX RPWC = XXXXXX'

; 'BUFFER ADR = XXXXXX SIZE = XXXX ACTUAL NMBR WRDS XFRD = XXX'

LINE2:

```

3277 020174
3278 020174 010346          MOV     R3,-(SP)     ; PUSH R3 ON STACK
3279 020176 010446          MOV     R4,-(SP)     ; PUSH R4 ON STACK
3280 020200 010546          MOV     R5,-(SP)     ; PUSH R5 ON STACK
3281 020202 104414 001165          DISPLY   , $CRLF     ; CR-LF
3282 020206 005037 020334          CLR     4$           ; CLEAR MESSAGE ADDRESS STORAGE
3283 020212 005004          CLR     R4           ; WORKING REGISTER
3284 020214 012737 050160 020334          MOV     #LIN2C,4$    ; ADDRESS OF 'PRESENT ORDER = ' MSG

```

3285	020222	116004	000234		MOVB	\$RPCS1(RO),R4	;GET THE OPCODE
3286	020226	042704	177701		BIC	#↑C76,R4	;SAVE ONLY SIGNIFICANT BITS
3287	020232	004737	020270		JSR	PC,1\$;TYPE THE FIRST MNEMONIC
3288	020236	005737	020340		TST	5\$;SEE IF MNEMONIC ENTRY FOUND
3289	020242	001440			BEQ	LINE2A	;BR IF NOT
3290	020244	012737	050201	020334	MOV	#LIN2P,4\$;ADDRESS OF 'PREVIOUS ORDER = ' MSG
3291	020252	116004	000027		MOVB	\$PREV0(RO),R4	;PREVIOUS OPERATION CODE
3292	020256	042704	177701		BIC	#↑C76,R4	;SAVE ONLY SIGNIFICANT BITS
3293	020262	004737	020270		JSR	PC,1\$;TYPE THE PREVIOUS MNEMONIC
3294	020266	000426			BR	LINE2A	;CONTINUE
3295	020270	005005		1\$:	CLR	R5	;CLEAR THE TABLE INDEX
3296	020272	126504	001746	2\$:	CMPB	OPTBL(R5),R4	;LOOK FOR THE OPCODE
3297	020276	001405			BEQ	3\$;BR WHEN OPCODE COUNT EQUALS OPCODE
3298	020300	105765	001746		TSTB	OPTBL(R5)	;LOOK FOR END OF TABLE
3299	020304	100402			BMI	3\$;BR IF END
3300	020306	005205			INC	R5	;INCREMENT THE POINTER
3301	020310	000770			BR	2\$;CONTINUE - NOT END OF TABLE
3302	020312	006305		3\$:	ASL	R5	;SHIFT INDEX
3303	020314	006305			ASL	R5	;SHIFT THE INDEX
3304	020316	006305			ASL	R5	;SHIFT THE INDEX
3305	020320	012737	001770	020340	MOV	#MNTBL,5\$;ADDRESS OF ASCII TEXT TABLE
3306	020326	060537	020340		ADD	R5,5\$;ADD THE INDEX
3307	020332	104414			DISPLY		;TYPE IT
3308	020334	000000		4\$:	.WORD	0	;ADDRESS OF 'PRESENT' OR 'PREVIOUS' MESSAGE
3309	020336	104414			DISPLY		;TYPE THE OPERATION MNEMONIC
3310	020340	000000		5\$:	.WORD	0	;ADDRESS OF MESSAGE
3311	020342	000207			RTS	PC	;RETURN TO MAIN ROUTINE
3312	020344	005737	001264	LINE2A:	TST	BADSEC	;PRINT THE BAD SECTOR LINE ?
3313	020350	001404			BEQ	LINE2B	;BR IF NOT
3314	020352	104414	001165		DISPLY	,\$CRLF	;CR-LF
3315	020356	104414	050225		DISPLY	,LIN2S	;ERROR ADDRESS DEFINED AS BAD AREA
3316	020362	104414	001165	LINE2B:	DISPLY	,\$CRLF	;CR-LF
3317	020366	104414	047554		DISPLY	,DH14	;STANDARD RPO4/5/6 REGISTER HEADER
3318	020372	104414	052074		DISPLY	,LINSPO	;TYPE A SPACE
3319	020376	013746	001246		MOV	UNIT,-(SP)	;PUT THE DRIVE NUMBER ON THE STACK
3320	020402	004737	022224		JSR	PC,LINDEC	;TYPE DRIVE NUMBER
3321	020406	104414	052073		DISPLY	,LINSR	;SPACES
3322	020412	012705	050100		MOV	#DT14,R5	;REGISTER INDEXES
3323	020416	004737	020546		JSR	PC,3\$;PRINT THE REGISTERS
3324	020422	032777	000040	160510	BIT	#S005,0SWR	;PRINT THE OPTIONAL REGISTERS ?
3325	020430	001014			BNE	1\$;BR IF NOT
3326	020432	104414	047660		DISPLY	,DH15	
3327	020436	012705	050122		MOV	#DT15,R5	;SECOND DATA LINE
3328	020442	004737	020546		JSR	PC,3\$;PRINT THEM
3329	020446	104414	047757		DISPLY	,DH15	
3330	020452	012705	050144		MOV	#DT16,R5	;THIRD DATA LINE
3331	020456	004737	020546		JSR	PC,3\$;PRINT THE REGISTERS
3332	020462	032760	000100	000016	1\$:	BIT	#BIT6,\$STATUS(RO)
3333	020470	001422			BEQ	2\$;DATA ERROR ?
3334	020472	016046	000020		MOV	\$WRDL(RO),-(SP)	;BR IF NOT
3335	020476	066016	000236		ADD	\$RPWC(RO),(SP)	;TRANSFER SIZE
3336	020502	006316			ASL	(SP)	;ADD REMAINING WORD COUNT
3337	020504	066016	000006		ADD	\$BUF(RO),(SP)	;CONVERT TO AN BYTE INCREMENT
3338	020510	022660	000240		CMP	(SP)+,\$RPBA(RO)	;BUFFER STARTING ADDRESS
							;CORRECT BUFFER ADDRESS ?


```

3339 020514 001410          BEQ      2$          ;BR IF YES
3340 020516 104414 047147  DISPLY  ,EM46      ;'BUS ADDRESS AND WORD COUNT ARE NOT CONSISTENT'
3341 020522 104414 001165  DISPLY  , $CRLF    ;CR-LF
3342 020526 004737 020640  JSR     PC,LIN3D   ;PRINT LINE 3D OF ERROR MESSAGE
3343 020532 004737 021256  JSR     PC,LIN4   ;PRINT LINE 4 OF ERROR MESSAGE
3344 020536
3345 020536 012605          2$:      MOV     (SP)+,R5  ;;POP STACK INTO R5
3346 020540 012604          MOV     (SP)+,R4  ;;POP STACK INTO R4
3347 020542 012603          MOV     (SP)+,R3  ;;POP STACK INTO R3
3348 020544 000207          RTS     PC        ;RETURN TO ERROR PROCESSING ROUTINE
3349 020546 012546          3$:      MOV     (R5)+,-(SP) ;PUT THE REGISTER INDEX ON THE STACK
3350 020550 060016          ADD     RD,(SP)   ;ADD DRIVE'S TABLE ADDRESS
3351 020552 017646 000000  MOV     @$(SP),-(SP) ;VALUE
3352 020556 004737 022172  JSR     PC,LIN0CT ;TYPE IT
3353 020562 005726          TST     (SP)+    ;CORRECT THE STACK POINTER
3354 020564 104414 052073  DISPLY  ,LINS    ;PRINT 2 SPACES
3355 020570 005715          TST     (R5)    ;AT END OF LINE ?
3356 020572 001365          BNE     3$      ;BR IF NOT
3357 020574 104414 001165  4$:      DISPLY  , $CRLF ;CR-LF
3358 020600 000207          RTS     PC        ;RETURN
3359
3360          ;PRINT LINE 3 OF ERROR MESSAGE
3361          ;'ERROR AT CCC TT SS PREVIOUS ADR = CCC TT SS'
3362
3363 020602 104414 050261  LINE3:  DISPLY  ,LINM3 ;LINE 3 ENTRANCE
3364 020606 000517          BR      LIN3.1    ;FINISH PRINTOUT
3365
3366          ;PRINT LINE 3A OF ERROR MESSAGE
3367          ;'START CYL = CCC END CYL = CCC'
3368
3369 020610 104414 050277  LINE3A: DISPLY  ,LINN3 ;LINE 3A ENTRANCE
3370 020614 000514          BR      LIN3.1    ;FINISH ERROR LINE
3371
3372          ;PRINT LINE 3B OF ERROR MESSAGE
3373          ;'START CYL = CCC END CYL = CCC ACTUAL CYL = CCC'
3374
3375 020616 004737 021160  LINE3B: JSR     PC,LIN3.3 ;LINE 3B ENTRANCE
3376 020622 104414 001165  DISPLY  , $CRLF
3377 020626 000207          RTS     PC
3378
3379          ;PRINT LINE 3C OF ERROR MESSAGE
3380          ;'START CYL = CCC END CYL = CCC ACTUAL CYL = CCC TRK = TT'
3381
3382 020630 004737 021160  LINE3C: JSR     PC,LIN3.3 ;LINE 3C ENTRANCE
3383 020634 000137 021212  JMP     LIN3.4    ;FINISH MESSAGE
3384
3385          ;PRINT LINE 3D OF ERROR MESSAGE
3386          ;'RPBA = XXXXXX RPWC = XXXXXX'
3387
3388 020640 032777 000040 160272 LINE3D: BIT     #SW05,@SWR ;SWITCH 5 SET ?
3389 020646 001416          BEQ     1$        ;BR IF IT IS
3390 020650 104414 050450  DISPLY  ,LINB3    ;'RPBA = '
3391 020654 016046 000240  MOV     $RPBA(R0),-(SP) ;BUFFER ADDR REG CONTENTS
3392 020660 004737 022172  JSR     PC,LIN0CT ;CONVERT TO OCTAL AND TYPE IT
    
```

3393 020664 104414 050460
 3394 020670 016046 000236
 3395 020674 004737 022172
 3396 020700 104414 001165
 3397 020704 000207
 3398
 3399
 3400
 3401
 3402 020706 104414 050344
 3403 020712 016046 000012
 3404 020716 004737 022224
 3405 020722 104414 052073
 3406 020726 104414 050472
 3407 020732 005046
 3408 020734 116016 000011
 3409 020740 004737 022224
 3410 020744 104414 052073
 3411 020750 104414 050507
 3412 020754 005046
 3413 020756 116016 000010
 3414 020762 004737 022224
 3415 020766 104414 001165
 3416 020772 000207
 3417
 3418
 3419
 3420
 3421 020774 032777 000040 160136
 3422 021002 001420
 3423 021004 104414 050440
 3424 021010 016046 000242
 3425 021014 004737 022172
 3426 021020 104414 052073
 3427 021024 104414 050427
 3428 021030 016046 000270
 3429 021034 004737 022172
 3430 021040 104414 001165
 3431 021044 000207
 3432
 3433
 3434
 3435 021046 016046 000272
 3436 021052 004737 022224
 3437 021056 104414 050274
 3438 021062 004737 022244
 3439 021066 004737 022224
 3440 021072 104414 050320
 3441 021076 004737 022224
 3442 021102 104414 050323
 3443 021106 016046 000034
 3444 021112 004737 022224
 3445 021116 104414 050274
 3446 021122 005046

```

DISPLY LINW3 ;' RPWC = '
MOV $RPWC(RO),-(SP) ;WORD COUNT REGISTER CONTENTS
JSR PC,LINOC ;CONVERT TO OCTAL AND TYPE IT
DISPLY $CRLF
RTS PC

;PRINT LINE 3E OF ERROR MESSAGE
;'START CYL = CCC START TRK = TT START SEC = SS'

LINE3E: DISPLY LINS3 ;'START CYL = '
MOV $CYL(RO),-(SP) ;MOVE CYL TO STACK
JSR PC,LINDEC ;TYPE IT IN DECIMAL
DISPLY ,LINS3 ;SPACES
DISPLY ,LINST3 ;'START TRK = '
CLR -(SP) ;CLEAR STACK
MOVB $TRK(RO),(SP) ;TRACK TO STACK
JSR PC,LINDEC ;TYPE IT IN DECIMAL
DISPLY ,LINS3 ;SPACES
DISPLY ,LINS3 ;'START SEC = '
CLR -(SP) ;CLEAR STACK
MOVB $SEC(RO),(SP) ;SECTOR ADDR TO STACK
JSR PC,LINDEC ;TYPE IT IN DECIMAL
DISPLY $CRLF
RTS PC

;PRINT LINE 3F OF ERROR MESSAGE
;'RPDA = XXXXXX RPCA = XXXXXX'

LINE3F: BIT #SWS,SWR ;SWITCH 5 SET ?
BEQ 1$ ;BR IF NOT
DISPLY LINDA3 ;'RPDA = '
MOV $RPDA(RO),-(SP) ;PUT SECTOR/TRACK ADDRESS ON THE STACK
JSR PC,LINOC ;TYPE IT
DISPLY ,LINS3 ;SPACES
DISPLY LINDA3 ;'RPCA = '
MOV $RPCA(RO),-(SP) ;PUT DESIRED CYLINDER ADDRESS ON THE STACK
JSR PC,LINOC ;TYPE IT
DISPLY $CRLF
RTS PC

;'CCC TT SS PREV ADR = CCC TT SS'

LIN3.1: MOV $RPCC(RO),-(SP) ;PUT CYLINDER ADDR ON STACK
JSR PC,LINDEC ;TYPE IT IN DECIMAL
DISPLY T ;PRINT 'T'
JSR PC,READDR ;DECREMENT TRACK AND SECTOR ADDRESSES
JSR PC,LINDEC ;TYPE TRACK IN DECIMAL
DISPLY S ;PRINT 'S'
JSR PC,LINDEC ;TYPE SECTOR ADDRESS
DISPLY LINS3 ;PRINT 'PREV ADDR'
MOV $PREVA+2(RO),-(SP) ;PREVIOUS CYLINDER
JSR PC,LINDEC ;TYPE IT IN DECIMAL
DISPLY T ;PRINT 'T'
CLR -(SP) ;MAKE ROOM ON THE STACK
    
```



```

3447 021124 116016 000033      MOVB   $PREVA+1(RO), (SP)  ;PREVIOUS TRACK ADDRESS
3448 021130 004737 022224      JSR    PC,LINDEC          ;TYPE IT IN DECIMAL
3449 021134 104414 050320      DISPLY ,S                 ;PRINT 'S'
3450 021140 005046                CLR    -(SP)              ;MAKE ROOM ON THE STACK
3451 021142 116016 000032      MOVB   $PREVA(RO), (SP)  ;PREVIOUS SECTOR DDRESS
3452 021146 004737 022224      JSR    PC,LINDEC          ;TYPE IT IN DECIMAL
3453 021152 104414 001165      DISPLY $CRLF
3454 021156 000207      RTS    PC
3455
3456                                ;'START CYL = CCC  END CYL = CCC'
3457
3458 021160 104414 050344      LIN3.3: DISPLY ,LINS3     ;LINE '3B & 3C' ENTRANCE
3459 021164 016046 000034      MOV    $PREVA+2(RO), -(SP) ;PREVIOUS CYLINDER
3460 021170 004737 022224      JSR    PC,LINDEC          ;TYPE IT IN DECIMAL
3461 021174 104414 050361      DISPLY ,LINS3            ;PRINT 'END CYL'
3462 021200 016046 000272      MOV    $RPCC(RO), -(SP)  ;PRESENT CYLINDER
3463 021204 004737 022224      JSR    PC,LINDEC          ;TYPE IT IN DECIMAL
3464 021210 000207      RTS    PC
3465
3466                                ;'ACTUAL CYL = CCC  TRK = TT'
3467
3468 021212 104414 050376      LIN3.4: DISPLY ,LINA3     ;PRINT 'ACTUAL'
3469 021216 013746 054374      MOV    CYLDER, -(SP)     ;ACTUAL CYLINDER
3470 021222 042716 010000      BIC    #BIT12, (SP)      ;CLEAR THE FORMAT BIT
3471 021226 004737 022224      JSR    PC,LINDEC          ;TYPE IT IN DECIMAL
3472 021232 104414 050416      DISPLY ,LINT3            ;PRINT TRACK
3473 021236 005046                CLR    -(SP)              ;CLEAR STACK WORD
3474 021240 116016 000243      MOVB   $RPDA+1(RO), (SP) ;PUT TRACK ON STACK
3475 021244 004737 022224      JSR    PC,LINDEC          ;TYPE IT IN DECIMAL
3476 021250 104414 001165      DISPLY $CRLF
3477 021254 000207      RTS    PC
3478
3479                                ;PRINT LINE 4 OF ERROR MESSAGE
3480                                ;'BUFFER ADR = XXXXXX  SIZE = XXXX  ACTUAL NMBR WRDS XFRD = XXX'
3481
3482 021256 032760 000100 000016 LINE4: BIT    #BIT06, $STATUS(RO) ;DATA ERROR ?
3483 021264 001427                BEQ    1$                 ;BR IF NOT
3484 021266 104414 050524      DISPLY ,LINM4            ;'PRINT BUFFER'
3485 021272 016046 000006      MOV    $BUF(RO), -(SP)   ;BUFFER ADDR ON STACK
3486 021276 004737 022172      JSR    PC,LINOC†        ;CONVERT TO OCTAL & PRINT
3487 021302 104414 050543      DISPLY ,LINS4            ;PRINT 'SIZE'
3488 021306 016046 000020      MOV    $WRDL(RO), -(SP)  ;BUFFER SIZE
3489 021312 004737 022224      JSR    PC,LINDEC          ;TYPE IT IN DECIMAL
3490 021316 104414 050555      DISPLY ,LINX4            ;'ACTUAL NMBR WRDS XFRD = '
3491 021322 016046 000240      MOV    $RPBA(RO), -(SP)  ;VALUE IN BUFFER ADDR REGISTER
3492 021326 166016 000006      SUB    $BUF(RO), (SP)    ;SUBTRACT STARTING ADDRESS
3493 021332 006216                ASR    (SP)               ;CONVERT INTO A WORD COUNT
3494 021334 004737 022224      JSR    PC,LINDEC          ;TYPE IT IN DECIMAL
3495 021340 104414 001165      DISPLY $CRLF
3496 021344 000207      1$: RTS    PC             ;CR-LF
3497                                ;RETURN
3498                                ;PRINT LINE 5 OF ERROR MESSAGE
3499                                ;'GOOD DATA = XXXXXX  BAD DATA = XXXXXX  SECT POS = XXX'
3500

```


3501 021346 104414 050610
 3502 021352 162760 000002
 3503 021360 017046 000240
 3504 021364 004737 022172
 3505 021370 104414 050625
 3506 021374 016046 000256
 3507 021400 004737 022172
 3508 021404 016046 000236
 3509 021410 066016 000020
 3510 021414 005046
 3511 021416 016046 000022
 3512 021422 004737 027004
 3513 021426 012616
 3514 021430 104414 050643
 3515 021434 004737 022224
 3516 021440 104414 001165
 3517 021444 000207
 3518
 3519
 3520
 3521
 3522
 3523 021446 104414 050661
 3524 021452 013746 054374
 3525 021456 004737 022172
 3526 021462 104414 052073
 3527 021466 013746 054376
 3528 021472 004737 022172
 3529 021476 104414 052073
 3530 021502 013746 054400
 3531 021506 004737 022172
 3532 021512 104414 052073
 3533 021516 013746 054402
 3534 021522 004737 022172
 3535 021526 104414 052073
 3536 021532 104414 001165
 3537 021536 000207
 3538
 3539
 3540
 3541 021540 104414 050715
 3542 021544 016046 000300
 3543 021550 004737 022172
 3544 021554 104414 052073
 3545 021560 104414 050726
 3546 021564 016046 000302
 3547 021570 004737 022172
 3548 021574 104414 001165
 3549 021600 000207
 3550
 3551
 3552
 3553
 3554 021602 104414 050740

```

LINES:  DISPLY  .LINDS      ;PRINT 'GOOD DATA'
SUB      #2,$RPBA(RO) ;BACK THE ADDRESS UP
MOV      JSRPBA(RO),-(SP) ;'GOOD' DATA - AT THE BUFFER LOCATION
JSR      PC,LINOC1      ;TYPE IT
DISPLY   .LINB5        ;PRINT 'BAD DATA'
MOV      $RPDB(RO),-(SP) ;BAD DATA FROM BUFFER
JSR      PC,LINOC1      ;TYPE IT
MOV      $RPWC(RO),-(SP) ;WORD LENGTH ON STACK
ADD      $NRDL(RO),(SP)  ;MAKE INTO A POSITIVE NUMBER
CLR      -(SP)          ;UPPER DIVIDEND TO ZERO
MOV      $SSEC(RO),-(SP) ;SECTOR SIZE ON THE STACK
JSR      PC,LINKDV      ;DIVIDE WORDS XFERED BY SECTOR SIZE
MOV      (SP)+(SP)      ;MOVE REMAINDER UP THE STACK
DISPLY   .LINP5        ;PRINT 'SECT POS'
JSR      PC,LINDEC      ;TYPE THE POSITION
DISPLY   $CRLF
RTS      PC
  
```

```

;PRINT LINE 5A OF THE ERROR MESSAGE
;'HEADER FROM ERROR SECTOR  XXXXXX  XXXXXX  XXXXXX  XXXXXX'
  
```

```

LINESA: DISPLY  .LINS5      ;'HEADER CONTENTS OF ERROR SECTOR'
MOV      CYLDER, -(SP)     ;HEADER POSITION
JSR      PC,LINOC1      ;TYPE IT
DISPLY   .LINS5          ;SPACES
MOV      CYLDER+2, -(SP)   ;HEADER POSITION +2
JSR      PC,LINOC1      ;TYPE IT
DISPLY   .LINS5          ;SPACES
MOV      CYLDER+4, -(SP)   ;HEADER POSITION +4
JSR      PC,LINOC1      ;TYPE IT
DISPLY   .LINS5          ;SPACES
MOV      CYLDER+6, -(SP)   ;HEADER POSITION +6
JSR      PC,LINOC1      ;TYPE IT
DISPLY   .LINS5          ;SPACES
DISPLY   $CRLF
RTS      PC
  
```

```

;PRINT LINE 5B OF ERROR MESSAGE
;'RPEC1 = XXXXXX  RPEC2 = -XXXXXX'
  
```

```

LINESB: DISPLY  .LINEP5     ;'RPEC1 = '
MOV      $RPEC1(RO),-(SP) ;PUT REGISTER CONTENTS ON THE STACK
JSR      PC,LINOC1      ;TYPE IT
DISPLY   .LINEP5        ;SPACES
DISPLY   .LINEP5        ;' RPEC2 = '
MOV      $RPEC2(RO),-(SP) ;PUT REGISTER CONTENTS ON THE STACK
JSR      PC,LINOC1      ;TYPE IT
DISPLY   $CRLF
RTS      PC              ;RETURN
  
```

```

;PRINT LINE 6 OF ERROR MESSAGE
;'SECTOR IS ECC CORRECTABLE'
  
```

```

LINE6:  DISPLY  .LINB6      ;ECC CORRECTABLE
  
```



```

3555 021606 104414 001165          DISPLY  ,SCLRF
3556 021612 000207          RTS      PC
3557
3558          :PRINT LINE 6A OF THE ERROR MESSAGE
3559          : 'SECTOR READ CORRECTLY AT OFFSET N'
3560
3561 021614 104414 050773          LINE6A: DISPLY  ,LINC6          :PRINT 'READ CORRECTLY AT OFFSET N'
3562 021620 000411          BR      LIN6.1          :TYPE THE REST OF THE LINE
3563
3564          :PRINT LINE 6B OF THE ERROR MESSAGE
3565          : 'SECTOR IS ECC CORRECTABLE AT OFFSET N'
3566
3567 021622 104414 050740          LINE6B: DISPLY  ,LINB6          :PRINT 'SECTOR IS ECC CORRECTABLE '
3568 021626 000406          BR      LIN6.1
3569
3570          :PRINT LINE 6C OF THE ERROR MESSAGE
3571          : 'CORRECTED ON NTH RETRY'
3572
3573 021630 104414 051022          LINE6C: DISPLY  ,LING6          : 'CORRECTED ON NTH RETRY'
3574 021634 000414          BR      LIN6.2          :TYPE THE REST OF THE LINE
3575
3576          :PRINT LINE 6D OF THE ERROR MESSAGE
3577          : 'UNCORRECTABLE AFTER N RETRIES'
3578
3579 021636 104414 051051          LINE6D: DISPLY  ,LINU06          : 'UNCORRECTABLE AFTER N RETRIES'
3580 021642 000411          BR      LIN6.2          :FINISH
3581
3582          :TYPE THE OFFSET VALUE IN MICRO-INCHES
3583
3584 021644 006301          LINE6.1: ASL      R1          :DOUBLE THE OFFSET TABLE INDEX
3585 021646 016137 002220 021656          MOV      OFMTBL(R1),1$      :ADDRESS OF OFFSET POSITION MESSAGE
3586 021654 104414          DISPLY
3587 021656 000000          1$:      WORD      0          :OFFSET VALUE
3588 021660 104414 001165          DISPLY  ,SCLRF
3589 021664 000207          RTS      PC
3590
3591          :RETRY COUNT TYPEOUT
3592
3593 021666 005046          LINE6.2: CLR      -(SP)          :CLEAR STACK
3594 021670 113716 001253          MOVB     RETRY+1,(SP)       :RETRY COUNT
3595 021674 004737 022224          JSR      PC,LINDEC         :TYPE IT IN DECIMAL
3596 021700 104414 051040          DISPLY  ,LINR6          : 'RETRY'
3597 021704 104414 001165          DISPLY  ,SCLRF
3598 021710 000207          RTS      PC
3599
3600          :PRINT LINE 7 OF THE ERROR MESSAGE
3601          : 'ORDERS:XXXXX TOTAL ERRORS:XXX WRDS XFRD:XXXXXX WRDS READ:XXXXXXX'
3602
3603 021712 104414 051124          LINE7:  DISPLY  ,LIN70          :PRINT ORDER COUNT
3604 021716 012746 000036          MOV      %$OPERC,-(SP)     :TO STACK
3605 021722 060016          ADD      RD,(SP)          :ADD THE BASE ADDRESS
3606 021724 004737 032500          JSR      PC,$DB2D         :CONVERT IT
3607 021730 004737 027320          JSR      PC,$SUPRS        :PRINT IT
3608 021734 104414 051203          DISPLY  ,LIN7T          :TOTAL ERRORS

```



```

3609 021740 016046 000056      MOV      $TOTAL(RO),-(SP)      ;TO STACK
3610 021744 004737 022224      JSR      PC,LINDEC            ;TYPE IT IN DECIMAL
3611 021750 104414 051215      DISPLY   ,LIN7X              ;PRINT 'WRDS XFR'
3612 021754 012746 000046      MOV      $STRANS,-(SP)       ;ADDRESS OF LOW WORD ON STACK
3613 021760 060016                ADD      RO,(SP)
3614 021762 004737 032500      JSR      PC,$DB2D            ;CONVERT
3615 021766 004737 027320      JSR      PC,$SUPRS           ;PRINT
3616 021772 104414 051231      DISPLY   ,LIN7R              ;'BITS READ'
3617 021776 012746 000052      MOV      $$READ,-(SP)       ;LOW WORD ADDR $$$
3618 022002 060016                ADD      RO,(SP)
3619 022004 004737 032500      JSR      PC,$DB2D            ;CONVERT
3620 022010 004737 027320      JSR      PC,$SUPRS           ;PRINT
3621 022014 104414 001165      DISPLY   ,$CRLF
3622 022020 104414 001165      DISPLY   ,$CRLF
3623 022024 032777 100000      BIT      $SW15,$SWR          ;SEE IF 'HALT ON ERROR' - SWITCH 15
3624 022032 001401                BEQ      IS
3625 022034 000000                HALT
3626 022036 000207                IS:   RTS      PC

```

```

;PRINT LINE 7A OF ERROR MESSAGE
;'ORDERS:XXXXX TOTAL SEEKS=XXXXX TOTAL MISPOS ERR = XXX TOTAL SKI,OCYL ERR = XXX'

```

```

3631 022040 104414 051124      LINE7A: DISPLY ,LIN7O        ;'ORDERS = '
3632 022044 012746 000036      MOV      $$OPERC,-(SP)      ;ORDER COUNT INCREMENT
3633 022050 060016                ADD      RO,(SP)            ;ADD BASE ADDRESS
3634 022052 004737 032500      JSR      PC,$DB2D            ;CONVERT THE COUNT
3635 022056 004737 027320      JSR      PC,$SUPRS           ;PRINT IT
3636 022062 104414 051134      DISPLY   ,LIN7P            ;'TOTAL SEEKS = '
3637 022066 012746 000042      MOV      $$POSIT,-(SP)      ;TOTAL SEEKS
3638 022072 060016                ADD      RO,(SP)            ;DEVICE TABLE ADDRESS
3639 022074 004737 032500      JSR      PC,$DB2D            ;CONVERT THE SEEK COUNT
3640 022100 004737 027320      JSR      PC,$SUPRS           ;PRINT IT
3641 022104 104414 051076      DISPLY   ,LIN7M            ;' TOTAL MISPOS ERR = '
3642 022110 016046 000066      MOV      $MISPO(RO),-(SP)   ;TOTAL ERRORS
3643 022114 004737 022224      JSR      PC,LINDEC            ;TYPE IT IN DECIMAL
3644 022120 104414 051154      DISPLY   ,LIN7S            ;' TOTAL SKI,OCYL ERR = '
3645 022124 016046 000064      MOV      $SKI(RO),-(SP)     ;CONVERT & PRINT IT
3646 022130 004737 022224      JSR      PC,LINDEC            ;TYPE IT IN DECIMAL
3647 022134 104414 001165      DISPLY   , $CRLF
3648 022140 104414 001165      DISPLY   , $CRLF
3649 022144 032777 100000      BIT      $SW15,$SWR          ;SEE IF HALT ON ERROR - SWITCH 15 SET
3650 022152 001401                BEQ      IS
3651 022154 000000                HALT
3652 022156 000207                IS:   RTS      PC

```

```

;PRINT LINE 8 OF THE ERROR MESSAGE
;'DIFFERENT ERROR DURING RETRY'

```

```

3657 022160 104414 051246      LINE8:  DISPLY   ,LIN8M
3658 022164 004737 020174      JSR      PC,LINE2           ;PRINT LINE 2 OF ERROR MESSAGE
3659 022170 000207                RTS      PC

```

```

;OCTAL TYPEOUT ROUTINE
;CALL:

```



```

3663      :      MOV      NUM,-(SP)      ;PUT THE NUMBER ON THE STACK
3664      :      JSR      PC,LINOCCT
3665      :      RETURN
3666
3667 022172 016646 000002      LINOCCT: MOV      2(SP),-(SP)      ;PUT NUMBER IN PROPER LOCATION ON STACK
3668 022176 004737 027750      JSR      PC,$$B20      ;CONVERT THE NUMBER TO OCTAL
3669 022202 012637 022216      MOV      (SP)+,1$      ;GET THE ADDRESS OF THE ASCII STRING
3670 022206 062737 000005 022216      ADD      #5.,1$      ;ADDRESS THE LAST 6 ASCII DIGITS
3671 022214 104414      DISPLY      ;TYPE IT
3672 022216 000000      1$: .WORD      0      ;ADDRESS
3673 022220 012616      MOV      (SP)+,(SP)      ;CORRECT THE STACK
3674 022222 000207      RTS      PC      ;RETURN
3675
3676      ;ROUTINE TO CONVERT THE INPUT NUMBER TO DECIMAL AND TYPE IT WITH
3677      ;LEADING ZERO SUPPRESSION
3678      ;CALL:
3679      :      MOV      NUM,-(SP)      ;PUT THE NUMBER ON THE STACK
3680      :      JSR      PC,LINDEC
3681      :      RETURN
3682
3683 022224 016646 000002      LINDEC: MOV      2(SP),-(SP)      ;SET UP STACK FOR CONVERT
3684 022230 004737 027720      JSR      PC,$$B20      ;CONVERT IT TO DECIMAL
3685 022234 004737 027320      JSR      PC,$$SUPRS      ;TYPE IT (WITH LEADING ZEROS SUPRESSED)
3686 022240 012616      MOV      (SP)+,(SP)      ;RESTORE STACK POINTER
3687 022242 000207      RTS      PC
3688
3689      ;:*****
3690
3691      .SBTTL  GENERAL SUPPORT SUBROUTINES
3692
3693      ;:*****
3694
3695      ;DECREMENT THE SECTOR-TRACK ADDRESS
3696      ;CALL:
3697      :      MOV      #DPB,RO      ;DPB ADDRESS
3698      :      JSR      PC,READDR
3699      :      RETURN
3700      :      (SP) CONTAINS THE TRACK ADDRESS
3701      :      2(SP) CONTAINS THE SECTOR ADDRESS
3702
3703 022244 162706 000004      READDR: SUB      #4,SP      ;DECREMENT THE STACK POINTER
3704 022250 016616 000004      MOV      4(SP),(SP)      ;MOVE THE RETURN ADDR DOWN THE STACK
3705 022254 005066 000004      CLR      4(SP)      ;CLEAR STACK FOR SECTOR
3706 022260 005066 000002      CLR      2(SP)      ;CLEAR STACK FOR TRACK
3707 022264 116066 000242 000004      MOVB     $RPDA(RO),4(SP)      ;INCREMENTED SECTOR ON STACK
3708 022272 005366 000004      DEC      4(SP)      ;DECREMENT THE SECTOR ADDRESS
3709 022276 100015      BPL      1$      ;BR IF SECTOR GREATER THAN 0
3710 022300 012766 000025 000004      MOV      #21,4(SP)      ;JAM SECTOR ADDRESS TO 21(10)
3711 022306 116066 000243 000002      MOVB     $RPDA+1(RO),2(SP)      ;TRACK ADDRESS
3712 022314 005366 000002      DEC      2(SP)      ;DECREMENT TRACK ADDRESS
3713 022320 100007      BPL      2$      ;BR IF IT DIDN'T GO NEG
3714 022322 012766 000022 000002      MOV      #18.,2(SP)      ;RESET TRACK TO 18(10)
3715 022330 000403      BR      2$
3716 022332 116066 000243 000002      1$:  MOVB     $RPDA+1(RO),2(SP)      ;TRACK ADDRESS

```

201

```

3717 022340 000207          2$:  RTS      PC          ;RETURN
3718
3719          ;ROUTINE TO CHECK FOR KW11-L OR KW11-P CLOCKS
3720
3721 022342 012737 177777 001210 CKCLK:  MOV      #-1,CLKFLG      ;CLEAR CLOCK AVAILABILITY FLAG
3722 022350 012737 177777 001206      MOV      #-1,PCLOCK      ;CLEAR KW11-P CLOCK AVAILABILITY FLAG
3723 022356 012737 022436 000004      MOV      #CKCLK1,ERRVEC  ;SET UP VECTOR FOR CLOCK CHECK
3724 022364 005037 000006      CLR      @#ERRVEC+2      ;NEW PSW
3725 022370 005777 156600      TST      @SLKCSR         ;CHECK FOR KW11-P
3726 022374 005037 001210      CLR      CLKFLG         ;SET CLOCK AVAILABILITY FLAG
3727 022400 005037 001206      CLR      PCLOCK         ;SET KW11-P CLOCK FLAG
3728 022404 013701 001200      MOV      $LPVEC,R1      ;KW11-P VECTOR ADDRESS
3729 022410 012721 023476      MOV      #CLOCK,(R1)+   ;SET UP KW11-P VECTOR
3730 022414 012711 000300      MOV      #300,(R1)      ;PSW - PRI 6
3731 022420 012777 174575 156550      MOV      #-1667,@SLKCSB ;LOAD COUNTER BUFFER WITH 16.67
3732 022426 012777 000131 156540      MOV      #131,@SLKCSR   ;SET CLOCK - CNT UP, IOUS, CONT INT
3733 022434 000437
3734 022436 062706 000004          CKCLK1:  BR       CKCLK3
3735 022442 012737 022504 000004      ADD      #4,SP          ;RESTORE THE STACK POINTER
3736 022450 005777 156526      MOV      #CKCLK2,@ERRVEC ;CHANGE ERROR VECTOR TO CHECK FOR KW11-L
3737 022454 005037 001210      TST      @SLKS          ;LOOK FOR KW11-L
3738 022460 013701 001204      CLR      CLKFLG         ;SET CLOCK FLAG
3739 022464 012721 023476      MOV      $LLVEC,R1      ;KW11-L VECTOR ADDRESS
3740 022470 012711 000300      MOV      #CLOCK,(R1)+   ;SET UP KW11-L VECTOR
3741 022474 012777 000100 156500      MOV      #300,(R1)      ;PSW - PRI 6
3742 022502 000414          BR       CKCLK3         ;SET KW11-L INTERRUPT
3743 022504 062706 000004          CKCLK2:  ADD      #4,SP          ;RESTORE THE STACK POINTER
3744 022510 104401 052651          TYPE      ,NEDCLK       ;'P OR L CLOCK MUST BE ON SYSTEM'
3745 022514 005737 000042          TST      42             ;UNDER MONITOR CONTROL ?
3746 022520 001402          BEQ      1$             ;BR IF NOT
3747 022522 000137 005256          JMP      $GET42         ;ABORT PROGRAM
3748 022526 000000          1$:      HALT           ;HALT
3749 022530 000137 004066          JMP      START         ;TRY AGAIN
3750 022534 012737 000006 000004 CKCLK3:  MOV      #6,@ERRVEC     ;RESTORE THE ERROR VECTOR
3751 022542 000207          RTS      PC
3752
3753          ;ROUTINE TO DISPLAY STATISTICS FOR ALL DRIVES ASSIGNED
3754          ;CALL:
3755          ;      JSR      PC,STATPR
3756          ;      RETURN
3757
3758 022544 010046          STATPR:  MOV      R0,-(SP)     ;SAVE R0
3759 022546 010446          MOV      R4,-(SP)     ;SAVE R4
3760 022550 005737 001442          TST      ASNLST        ;ANY DRIVES ASSIGNED ?
3761 022554 001421          BEQ      3$            ;BR IF NOT
3762 022556 004737 022654          JSR      PC,SHDTYP     ;TYPE THE HEADING
3763 022562 005004          CLR      R4           ;CLEAR THE DRIVE INDEX
3764 022564 006304          1$:      ASL      R4           ;CHANGE TO INDEX WORDS
3765 022566 016400 001720          MOV      BLKADR(R4),R0 ;GET THE DRIVE'S BLOCK ADDRESS
3766 022572 006204          ASR      R4           ;RESTORE R4
3767 022574 136437 033236 001442          BITB     ATABIT(R4),ASNLST ;IS THIS DRIVE ASSIGNED ?
3768 022602 001402          BEQ      2$            ;BR IF NOT
3769 022604 004737 022676          JSR      PC,SDETAL     ;TYPE THE PERFORMANCE SUMMARY
3770 022610 005204          2$:      INC      R4           ;INCREMENT THE INDEX
    
```



```

3771 022612 020427 000010          CMP      R4,#8.          ;FINISHED ?
3772 022616 001362          BNE      1$            ;BR IF NOT
3773 022620 012604          3$:     MOV      (SP)+,R4 ;RESTORE R4
3774 022622 012600          MOV      (SP)+,R0     ;RESTORE R0
3775 022624 000207          RTS       PC           ;RETURN
3776
3777          ;ROUTINE TO TYPE STATISTICS FOR AN INDIVIDUAL DRIVE
3778          ;CALL:
3779          ;       MOV      #DPB,R0          ;DPB ADDRESS
3780          ;       JSR      PC,TYPEST
3781          ;       RETURN
3782
3783 022626 010046          TYPEST: MOV      R0,-(SP) ;SAVE R0
3784 022630 010446          MOV      R4,-(SP)     ;SAVE R4
3785 022632 004737 022654          JSR      PC,SHDTYP    ;TYPE THE HEADING
3786 022636 005004          CLR      R4           ;CLEAR R4 FOR DRIVE NUMBER
3787 022640 111004          MOVB     (R0),R4      ;DRIVE NUMBER
3788 022642 004737 022676          JSR      PC,SDETAL    ;TYPE THE STATISTICS
3789 022646 012604          MOV      (SP)+,R4     ;RESTORE R4
3790 022650 012600          MOV      (SP)+,R0     ;RESTORE R0
3791 022652 000207          RTS       PC           ;RETURN
3792
3793          ;TYPE THE HEADER FOR THE DRIVE PERFORMANCE SUMMARY TYPEOUT:
3794          ;CALL:
3795          ;       JSR      PC,SHDTYP
3796          ;       RETURN
3797
3798 022654 004737 023400          SHDTYP: JSR      PC,$TIME ;TYPE THE TIME OF DAY
3799 022660 004537 027360          JSR      R5,TYPRI4   ;TYPE AT PRIORITY 4
3800 022664 001165          $CRLF
3801 022666 004537 027360          JSR      R5,TYPRI4   ;TYPE THE HEADER
3802 022672 052321          STATHD
3803 022674 000207          RTS       PC           ;RETURN
3804
3805          ;TYPE THE PERFORMANCE SUMMARY DATE LINE
3806          ;CALL:
3807          ;       MOV      #DRIVE,R4      ;DRIVE NUMBER
3808          ;       MOV      #DPB,R0        ;DPB ADDRESS
3809          ;       RETURN
3810
3811 022676 010246          SDETAL: MOV      R2,-(SP) ;SAVE R2
3812 022700 010002          MOV      R0,R2        ;DPB ADDRESS
3813 022702 062702 000036          ADD      #SOPERC,R2  ;FIRST STATISTICAL FIELD
3814 022706 010446          MOV      R4,-(SP)     ;SAVE R4 FOR TYPEOUT
3815
3816 022710 104403          TYPOS
3817 022712          .BYTE 2             ;GO TYPE--OCTAL ASCII
3818 022713          .BYTE 0             ;TYPE 2 DIGIT(S)
3819 022714 104401 052073          TYPE     LINSF        ;SUPPRESS LEADING ZEROS
3820 022720 016046 000070          MOV      $PASSC(R0),-(SP) ;SPACES
3821 022724 004737 027720          JSR      PC,$SB2D     ;PUT THE PASS COUNT ON THE STACK
3822 022730 004537 027230          JSR      R5,REPLZ    ;CONVERT IT
3823 022734 000003          .WORD 3              ;TYPE IT
3824 022736 104401 052073          TYPE     ,LINSF       ;TYPE 3 DIGITS
                          ;SPACES

```

3825	022742	010246		MOV	R2,-(SP)	:PUT \$OPERC ON THE STACK
3826	022744	004737	032500	JSR	PC,\$DB2D	:CONVERT IT
3827	022750	004537	027230	JSR	R5,REPLZ	:TYPE \$OPERC
3828	022754	000006		.WORD	6	:TYPE 6 DIGITS
3829	022756	104401	052073	TYPE	,LINSF	:SPACES
3830	022762	062702	000004	ADD	#4,R2	:INCREMENT R2
3831	022766	010246		MOV	R2,-(SP)	:PUT \$POSIT ON THE STACK
3832	022770	004737	032500	JSR	PC,\$DB2D	:CONVERT IT
3833	022774	004537	027230	JSR	R5,REPLZ	:TYPE \$POSIT
3834	023000	000006		.WORD	6	:TYPE 6 DIGITS
3835	023002	104401	052073	TYPE	,LINSF	:SPACES
3836	023006	062702	000004	ADD	#4,R2	:INCREMENT R2
3837	023012	010246		MOV	R2,-(SP)	:PUT \$TRANS ON THE STACK
3838	023014	004737	032500	JSR	PC,\$DB2D	:CONVERT \$TRANS
3839	023020	004537	027230	JSR	R5,REPLZ	:TYPE IT
3840	023024	000012		.WORD	10	:TYPE 10 DIGITS
3841	023026	104401	052073	TYPE	,LINSF	:SPACES
3842	023032	062702	000004	ADD	#4,R2	:INCREMENT R2
3843	023036	010246		MOV	R2,-(SP)	:PUT \$READ ON THE STACK
3844	023040	004737	032500	JSR	PC,\$DB2D	:CONVERT \$READ
3845	023044	004537	027230	JSR	R5,REPLZ	:TYPE IT
3846	023050	000012		.WORD	10	:TYPE 10 DIGITS
3847	023052	104401	052074	TYPE	,LINSPO	:1 SPACE
3848	023056	062702	000004	ADD	#4,R2	:INCREMENT R2
3849	023062	062702	000002	ADD	#2,R2	:INCREMENT R2 AGAIN
3850	023066	012246		MOV	(R2)+,-(SP)	:PUT \$SOFT ON THE STACK
3851	023070	004737	027720	JSR	PC,\$SB2D	:CONVERT \$SOFT
3852	023074	004537	027230	JSR	R5,REPLZ	:TYPEOUT \$SOFT
3853	023100	000004		.WORD	4	:TYPE 4 DIGITS
3854	023102	104401	052074	TYPE	,LINSPO	:SPACES
3855	023106	012246		MOV	(R2)+,-(SP)	:PUT \$HARD ON THE STACK
3856	023110	004737	027720	JSR	PC,\$SB2D	:CONVERT \$HARD
3857	023114	004537	027230	JSR	R5,REPLZ	:TYPEOUT \$HARD
3858	023120	000004		.WORD	4	:TYPE 4 DIGITS
3859	023122	104401	052074	TYPE	,LINSPO	:SPACES
3860	023126	012246		MOV	(R2)+,-(SP)	:PUT \$SKI ON THE STACK
3861	023130	004737	027720	JSR	PC,\$SB2D	:CONVERT \$SKI
3862	023134	004537	027230	JSR	R5,REPLZ	:TYPEOUT \$SKI
3863	023140	000004		.WORD	4	:TYPE 4 DIGITS
3864	023142	104401	052074	TYPE	,LINSPO	:SPACES
3865	023146	012246		MOV	(R2)+,-(SP)	:PUT \$MISPO ON THE STACK
3866	023150	004737	027720	JSR	PC,\$SB2D	:CONVERT \$MISPO
3867	023154	004537	027230	JSR	R5,REPLZ	:TYPEOUT \$MISPO
3868	023160	000004		.WORD	4	:TYPE 4 DIGITS
3869	023162	104401	052074	TYPE	,LINSPO	:SPACES
3870	023166	016046	000056	MOV	\$TOTAL(R0),-(SP)	:CALCULATE NUMBER OF OTHER ERRORS
3871	023172	166016	000060	SUB	\$SOFT(R0),(SP)	:SUBTRACT \$SOFT FROM \$TOTAL
3872	023176	166016	000062	SUB	\$HARD(R0),(SP)	:SUBTRACT \$HARD FROM \$TOTAL
3873	023202	166016	000064	SUB	\$SKI(R0),(SP)	:SUBTRACT \$SKI FROM \$TOTAL
3874	023206	166016	000066	SUB	\$MISPO(R0),(SP)	:SUBTRACT \$MISPO FROM \$TOTAL
3875	023212	004737	027720	JSR	PC,\$SB2D	:CONVERT 'OTHER' COUNT
3876	023216	004537	027230	JSR	R5,REPLZ	:TYPE IT
3877	023222	000004		.WORD	4	:TYPE 4 DIGITS
3878	023224	104401	001165	TYPE	,\$SRLF	


```

3879 023230 012602      MOV      (SP)+,R2      ;RESTORE R2
3880 023232 000207      RTS        PC
3881
3882
3883
3884      ;ROUTINE TO INCREMENT $SOFT
3885
3886      ;NOTE: $SOFT WILL NOT BE INCREMENTED BEYOND 9999 (10)
3887
3888 023234 005737 001264      INCSOF: TST      BADSEC      ;SEE IF BAD TRK/SEC INDICATOR SET
3889 023240 001006      BNE      1$              ;BR IF IT'S SET, DON'T INCREMENT COUNT
3890 023242 026027 000060 023417      CMP      $SOFT(RO),#9999.  ;IS $SOFT ALREADY AT MAXIMUM ?
3891 023250 103002      BHIS     1$              ;BR IF IT IS
3892 023252 005260 000060      INC      $SOFT(RO)      ;INCREMENT $SOFT
3893 023256 000207      1$:      RTS        PC      ;RETURN
3894
3895
3896      ;ROUTINE TO INCREMENT $SHARD
3897
3898      ;NOTE: $SHARD WILL NOT BE INCREMENTED BEYOND 9999 (10)
3899
3900 023260 005737 001264      INCHRD: TST      BADSEC      ;SEE IF BAD TRK/SEC INDICATOR SET
3901 023264 001006      BNE      1$              ;BR IF IT'S SET, DON'T INCREMENT COUNT
3902 023266 026027 000062 023417      CMP      $SHARD(RO),#9999. ;IS $SHARD ALREADY AT MAXIMUM ?
3903 023274 103002      BHIS     1$              ;BR IF IT IS
3904 023276 005260 000062      INC      $SHARD(RO)     ;INCREMENT $SHARD
3905 023302 000207      1$:      RTS        PC      ;RETURN
3906
3907
3908      ;ROUTINE TO INCREMENT $SKI
3909
3910      ;NOTE: $SKI WILL NOT BE INCREMENTED BEYOND 9999 (10)
3911
3912 023304 005737 001264      INCSKI: TST      BADSEC      ;SEE IF BAD TRK/SEC INDICATOR SET
3913 023310 001006      BNE      1$              ;BR IF IT'S SET, DON'T INCREMENT COUNT
3914 023312 026027 000064 023417      CMP      $SKI(RO),#9999.  ;IS $SKI ALREADY AT MAXIMUM ?
3915 023320 103002      BHIS     1$              ;BR IF IT IS
3916 023322 005260 000064      INC      $SKI(RO)       ;INCREMENT $SKI
3917 023326 000207      1$:      RTS        PC      ;RETURN
3918
3919
3920      ;ROUTINE TO INCREMENT $MISPO
3921
3922      ;NOTE: $MISPO WILL NOT BE INCREMENTED BEYOND 9999 (10)
3923
3924 023330 005737 001264      INCMIS: TST      BADSEC      ;SEE IF BAD TRK/SEC INDICATOR SET
3925 023334 001006      BNE      1$              ;BR IF IT'S SET, DON'T INCREMENT COUNT
3926 023336 026027 000066 023417      CMP      $MISPO(RO),#9999. ;IS $MISPO ALREADY AT MAXIMUM ?
3927 023344 103002      BHIS     1$              ;BR IF IT IS
3928 023346 005260 000066      INC      $MISPO(RO)     ;INCREMENT $MISPO
3929 023352 000207      1$:      RTS        PC      ;RETURN
3930
3931
3932      ;ROUTINE TO INCREMENT $TOTAL
    
```

```

3933
3934
3935
3936 023354 005737 001264
3937 023360 001006
3938 023362 026027 000056 023417
3939 023370 103002
3940 023372 005260 000056
3941 023376 000207
3942
3943
3944
3945
3946
3947
3948
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3980
3981
3982
3983
3984
3985
3986
    
```

```

;NOTE: $TOTAL WILL NOT BE INCREMENTED BEYOND 9999 (10)
INCTOT: TST     BADSEC           ;SEE IF BAD TRK/SEC INDICATOR SET
        BNE     1$              ;BR IF IT'S SET, DON'T INCREMENT COUNT
        CMP     $TOTAL(RO),#9999 ;IS $TOTAL ALREADY AT MAXIMUM ?
        BHIS   1$              ;BR IF IT IS
        INC     $TOTAL(RO)      ;INCREMENT $TOTAL
1$:     RTS     PC              ;RETURN
    
```

;ROUTINE TO TYPE THE TIME

```

$TIME: TST     CLKFLG           ;CLOCK ON THE SYSTEM ?
        BNE     1$              ;BR IF NOT
        TYPE   , $CRLF          ;CR-LF
        MOV     HOUR, -(SP)     ;PUT 'HOURS' ON THE STACK
        JSR    PC, $SB2D        ;CONVERT TO DECIMAL
        JSR    R5, REPLZ        ;TYPE IT
        .WORD  2                ;TYPE 2 DIGITS
        TYPE   , COLON          ;
        MOV     MINUTE, -(SP)   ;PUT 'MINUTES' ON THE STACK
        JSR    PC, $SB2D        ;CONVERT TO DECIMAL
        JSR    R5, REPLZ        ;TYPE IT
        .WORD  2                ;TYPE 2 DIGITS
        TYPE   , COLON          ;
        MOV     SECOND, -(SP)   ;PUT SECONDS ON THE STACK
        JSR    PC, $SB2D        ;CONVERT TO DECIMAL
        JSR    R5, REPLZ        ;TYPE IT
        .WORD  2                ;TYPE 2 DIGITS
1$:     RTS     PC
    
```

;CLOCK HANDLER ROUTINE

```

CLOCK: DEC     SIXTEE           ;INCREMENT THE 1/60 SECOND COUNTER
        BNE     1$              ;BR IF A SECOND NOT COUNTED
        MOV     HZ, SIXTEE      ;RESTORE THE VALUE
        INC     SECOND          ;COUNT THE SECOND
        CMP     #60., SECOND    ;AT MAXIMUM ?
        BNE     1$              ;BR IF NOT
        CLR     SECOND          ;CLEAR THE SECOND'S COUNTER
        INC     INTRVL+2        ;COUNT THE PERFORMANCE SUMMARY INTERVAL
        INC     MINUTE          ;COUNT THE MINUTE
        CMP     #60., MINUTE    ;AT MAXIMUM ?
        BNE     1$              ;BR IF NOT
        CLR     MINUTE          ;CLEAR THE MINUTE'S COUNTER
        INC     HOUR            ;COUNT THE HOURS
        CMP     #999., HOUR     ;AT MAXIMUM
        BNE     1$              ;BR IF NOT
        CLR     HOUR            ;CLEAR THE HOURS
        MOV     #17., -(SP)     ;17 MS ON THE STACK
        JSR    PC, RPTMR        ;DRIVER TIMER ROUTINE
        TST    INTRVL           ;DISPLAY THE PERFORMANCE SUMMARY ?
        BEQ    2$              ;BR IF NOT
        
```



```

3987 023614 023737 001370 001372      CMP      INTRVL,INTRVL+2 ;DISPLAY INTERVAL FINISHED ?
3988 023622 001005                      BNE      2$ ;BR IF NOT
3989 023624 012737 177777 001214      MOV      #-1,STATIN ;SET PERFORMANCE SUMMARY DISPLAY FLAG
3990 023632 005037 001372                      CLR      INTRVL+2 ;CLEAR THE PERFORMANCE INTERVAL COUNTER
3991 023636 000002                      2$:     RTI ;RETURN
3992
3993 ;COMMAND DECODE ROUTINE
3994 ;CALL:
3995 ;     MOV      #-1,CFLAG ;'CFLAG' IS NORMALLY SET BY THE TTY SERVICE
3996 ;     ;ROUTINE IN INTERRUPT MODE
3997 ;     JSR      PC,KSR
3998 ;     ;SYSTEM BUSY RETURN
3999 ;     RETURN1 ;RETURN AFTER KEYBOARD SERVICED
4000 ;     RETURN2
4001 023640 005737 001420      KSR:    TST      ORDERQ ;ANY OPERATIONS ACTIVE ?
4002 023644 001003                      BNE      1$ ;BR IF SOME ARE
4003 023646 005037 023672                      CLR      3$ ;CLEAR THE LOOP COUNTER
4004 023652 000410                      BR       KSR1 ;PROCESS THE KEYBOARD REQUEST
4005 023654 062737 000001 023672 1$:     ADD      #1,3$ ;COUNT THE TIMES THROUGH THE LOOP
4006 023662 001002                      BNE      2$ ;BR IF NOT ENOUGH
4007 023664 104401 053236                      TYPE     'BUSY' ;'SYSTEM BUSY...'
4008 023670 000207                      2$:     RTS      PC ;PROCESS ANY COMPLETED DRIVES
4009 023672 000000                      3$:     .WORD   0 ;LOOP COUNTER
4010 023674 104412                      KSR1:   SAVREG ;SAVE THE REGISTERS
4011 023676 012737 000200 177776      MOV      #PR4,PS ;SET PRIORITY TO 4
4012 023704 005037 001262                      CLR      CFLAG ;CLEAR THE 'CONTROL C' FLAG
4013 023710 004737 023400                      JSR      PC,$TIME ;TYPE THE TIME
4014 023714 005777 155226                      TST      @STKB ;CLEAR ANY GARBAGE IN THE TTY BUFFER
4015 023720 104401 052751                      TYPE     'ENTCOM' ;'ENTER COMMANDS'
4016 023724 104411                      RDLIN   ;READ THE KEYBOARD
4017 023726 012605                      MOV      (SP)+,R5 ;GET ADDRESS OF INPUT STRING
4018 023730 005737 001262                      TST      CFLAG ;CHECK THE CONTROL C FLAG
4019 023734 001065                      BNE      7$ ;EXIT IF 'CONTROL C' ENTERED
4020 023736 005205                      INC      R5 ;POINT TO SECOND CHARACTER
4021 023740 122715 000101                      CMPB     #'A',(R5) ;EQ TO AN 'A'
4022 023744 001410                      BEQ      1$ ;BR IF IT IS
4023 023746 121527 000067                      CMPB     (R5),#'7' ;DRIVE NUMBER GREATER THAN AN ASCII 7 ?
4024 023752 101054                      BHI      6$ ;BR IF IT IS
4025 023754 121527 000060                      CMPB     (R5),#'0' ;DRIVE NUMBER LESS THAN AN ASCII 0 ?
4026 023760 103451                      BLO      6$ ;BR IF IT IS
4027 023762 142715 177770                      BICB     #7,(R5) ;LEAVE ONLY LOWER 3 BITS IF CHAR NOT 'A'
4028 023766 122765 000124 177777 1$:     CMPB     #'T',-1(R5) ;EQ TO 'T'
4029 023774 001003                      BNE      2$ ;BR IF NOT EQ
4030 023776 004737 024502                      JSR      PC,NEWASN ;ASSIGN DRIVE FOR TEST
4031 024002 000442                      BR       7$ ;EXIT
4032 024004 122765 000104 177777 2$:     CMPB     #'D',-1(R5) ;EQ TO 'D' ?
4033 024012 001003                      BNE      3$ ;BR IF NOT EQ
4034 024014 004737 024512                      JSR      PC,DEASGN ;DEASSIGN DRIVE
4035 024020 000433                      BR       7$ ;EXIT
4036 024022 122765 000123 177777 3$:     CMPB     #'S',-1(R5) ;EQ TO 'S'
4037 024030 001003                      BNE      4$ ;BR IF NOT EQ
4038 024032 004737 024620                      JSR      PC,SCMND ;TYPE STATISTICS
4039 024036 000424                      BR       7$ ;EXIT
4040 024040 122765 000127 177777 4$:     CMPB     #'W',-1(R5) ;EQ TO 'W'

```

```

4041 024046 001007          BNE      5$      ;BR IF NOT EQ
4042 024050 032777 000001 155062  BIT      #SWD,ASWR ;IS SWITCH 0 SET ?
4043 024056 001012          BNE      6$      ;BR IF SET, CAN'T DO 'W' COMMAND
4044 024060 004737 025070  JSR      PC,DATAPK ;WRITE A DATA PACK
4045 024064 000411          BR       7$      ;EXIT
4046 024066 122765 000122 177777 5$:  CMPB    #'R,-1(R5) ;EQ TO 'R' ?
4047 024074 001003          BNE      6$      ;BR IF NOT EQ
4048 024076 004737 025102  JSR      PC,REDAPK ;READ A DATA PACK
4049 024102 000402          BR       7$      ;EXIT
4050 024104 104401 052727          6$:  TYPE   ,INVLD    ;TYPE 'INVALID COMMAND' MESSAGE
4051 024110 104413          7$:  RESREG ;RESTORE R0 - R5
4052 024112 062716 000002  ADD      #2,(SP)  ;INCREMENT THE RETURN ADDRESS
4053 024116 005777 155024  TST      AS$TKB  ;CLEAR THE TTY BUFFER
4054 024122 052777 000100 155014  BIS      #BIT06,AS$TKS ;SET TTY INTERRUPT ENABLE
4055 024130 005037 177776  CLR      PS      ;SET PRIORITY BACK TO ZERO
4056 024134 000207          RTS      PC      ;RETURN
4057
4058                                     ;ROUTINE TO PROCESS THE ASSIGN REQUEST ('T', 'R', OR 'W' COMMANDS)
4059
4060 024136 122715 000101  ASSIGN:  CMPB    #'A,(R5) ;ASSIGN ALL DRIVES?
4061 024142 001430          BEQ      ASGN2   ;BR IF ALL DRIVES
4062 024144 111504          ASGN1:  MOVB    (R5),R4 ;PUT DRIVE # IN R4
4063 024146 012737 052147 026366  MOV      #UNTASN,ASNMSG ;'DRIVE ASSIGNED' MESSAGE ADDRESS
4064 024154 012703 000001  MOV      #1,R3    ;RELOAD R3 FOR 1 UNIT
4065 024160 136437 033236 001442  BITB    ATABIT(R4),ASNLST ;DRIVE ALREADY ASSIGNED ?
4066 024166 001013          BNE      2$      ;BR IF IT IS
4067 024170 005704          TST      R4      ;TRYING TO ASSIGN DRIVE 0 ?
4068 024172 001007          BNE      1$      ;BR IF NOT
4069 024174 012737 052233 026366  MOV      #NOTAVL,ASNMSG ;'NOT AVAILABLE' MESSAGE ADDRESS
4070 024202 122737 000011 000041  CMPB    #11,41   ;SEE IF LOADED FROM AN RPO4/5/6
4071 024210 001402          BEQ      2$      ;BR IF RPO4/5/6 IS THE LOAD DEVICE
4072 024212 004737 024276          1$:  JSR      PC,ASGN3 ;SEE IF DRIVE ON THE SYSTEM
4073 024216 000137 026346          2$:  JMP      ASNERR  ;RETURN HERE IF DRIVE NOT AVAIL
4074 024222 000207          RTS      PC      ;EXIT
4075 024224 122737 000011 000041  ASGN2:  CMPB    #11,41 ;LOADED FROM AN RPO4/5/6 ?
4076 024232 001005          BNE      1$      ;BR IF NOT
4077 024234 012704 000001  MOV      #1,R4    ;START WITH DRIVE 1
4078 024240 012703 000007  MOV      #7.,R3   ;SETUP FOR ONLY 7 DRIVES
4079 024244 000403          BR       2$      ;CONTINUE
4080 024246 005004          1$:  CLR      R4      ;START WITH DRIVE 0
4081 024250 012703 000010  MOV      #8.,R3   ;DRIVE COUNT FOR 8 DRIVES
4082 024254 004737 024276          2$:  JSR      PC,ASGN3 ;ASSIGN ALL UNASSIGNED, AVAIL DRIVES
4083 024260 000137 024266          3$:  JMP      4$      ;DRIVE NOT ON SYSTEM
4084 024264 000207          RTS      PC      ;RETURN
4085 024266 012746 024260          4$:  MOV      #3$,-(SP) ;PUT RETURN ADDRESS ON THE STACK
4086 024272 000137 024414          JMP      ASGN4   ;LOOK FOR MORE DRIVES
4087 024276 136437 033236 001442  ASGN3:  BITB    ATABIT(R4),ASNLST ;DRIVE ALREADY ASSIGNED ?
4088 024304 001043          BNE      ASGN4   ;BR IF IT IS
4089 024306 005737 033234          1$:  TST      DTUW   ;DATA TRANSFER UNDER WAY ?
4090 024312 100375          BPL     1$      ;BR IF IT IS
4091 024314 110437 044646          MOVB    R4,GENDPB ;DRIVE NUMBER
4092 024320 004737 015174          JSR      PC,RECALO ;RECALIBRATE DRIVE
4093 024324 105764 033122          TSTB    DRVSTA(R4) ;DRIVE AVAILABLE?
4094 024330 001444          BEQ      ASGN7   ;BR IF DRIVE OFFLINE OR NONEXISTENT

```



```

4095 024332 100437      BMI      ASGN6      ;BR IF DRIVE UNSAFE
4096 024334 006304      ASL      R4          ;MAKE R4 INTO WORD INDEX
4097 024336 016464 001720 001466      MOV      BLKADR(R4),NEWUNT(R4) ;DPB ADDRESS
4098 024344 016400 001720      MOV      BLKADR(R4),R0      ;PUT BLOCK'S ADDR INTO R0
4099 024350 004737 025114      JSR      PC,CLRDPB      ;CLEAR BLOCK FOR DRIVE JUST ASSIGNED
4100 024354 004737 025310      JSR      PC,DRVPRM      ;GET THE DRIVE'S ADDRESS LIMITS
4101 024360 004737 025572      JSR      PC,GETID      ;GET DRIVE I.D.
4102 024364 004737 025702      JSR      PC,GETADR      ;GET BAD SECTOR ADDRESSES
4103 024370 012760 000001 000070      MOV      #1,$PASSC(R0)      ;PRESET PASS COUNT TO 1
4104 024376 005737 001216      TST      PACK          ;WRITE DATA PACK ?
4105 024402 001403      BEQ      2$           ;BR IF NOT
4106 024404 113760 001216 000026      MOVB     PACK,$PACK(R0)      ;SET READ/WRITE DATA PACK INDICATOR
4107 024412 006204      2$:      ASR      R4          ;RESTORE DRIVE ADDRESS
4108 024414 005303      ASGN4:   DEC      R3          ;DECREMENT DRIVE COUNT
4109 024416 001402      BEQ      ASGN5        ;BR IF FINISHED
4110 024420 005204      INC      R4          ;INCREMENT DRIVE NUMBER
4111 024422 000725      BR       ASGN3        ;CONTINUE
4112 024424 062716 000004      ASGN5:   ADD      #4,(SP)      ;INCREMENT RETURN
4113 024430 000207      RTS      PC          ;RETURN
4114 024432 012737 052252 026366      ASGN6:   MOV      #NOTSAF,ASNMSG ;'UNSAFE' MESSAGE ADDRESS
4115 024440 000207      RTS      PC          ;RETURN
4116 024442 105764 033132      ASGN7:   TSTB     DRVTP(R4)      ;DRIVE PRESENT?
4117 024446 001405      BEQ      1$           ;BR IF NOT
4118 024450 100010      BPL      2$           ;BR IF DRIVE OFFLINE
4119 024452 012737 052175 026366      MOV      #NOTRP,ASNMSG ;ADDRESS OF 'NOT RPO4/5/6' MSG
4120 024460 000407      BR       3$           ;EXIT
4121 024462 012737 052216 026366      1$:     MOV      #NOTPRS,ASNMSG ;ADDRESS OF 'NOT PRESENT' MSG
4122 024470 000403      BR       3$           ;EXIT
4123 024472 012737 052104 026366      2$:     MOV      #UNTOFF,ASNMSG ;ADDRESS OF 'DRIVE OFFLINE' MESSAGE
4124 024500 000207      3$:     RTS      PC          ;ERROR RETURN
4125
4126      ;'T' COMMAND (ROUTINE TO ASSIGN A DRIVE)
4127
4128 024502 005037 001216      NEWASN:  CLR      PACK          ;CLEAR 'W' COMMAND INDICATOR
4129 024506 000137 024136      JMP      ASSIGN       ;GO TO THE ASSIGN ROUTINE
4130
4131      ;'D' COMMAND (ROUTINE TO DEASSIGN A DRIVE)
4132
4133 024512 005004      DEASGN:  CLR      R4          ;DEASSIGN ALL DRIVES ?
4134 024514 122715 000101      CMPB     #'A,(R5)      ;DEASSIGN ALL DRIVES ?
4135 024520 001434      BEQ      5$           ;BR IF YES
4136 024522 012703 000001      MOV      #BIT00,R3      ;SET R3 FOR ONE UNIT
4137 024526 111504      MOVB     (R5),R4        ;GET DRIVE NUMBER
4138 024530 136437 033236 001442      1$:     BITB     ATABIT(R4),ASNLST ;DRIVE ASSIGNED ?
4139 024536 001414      BEQ      3$           ;BR IF NOT
4140 024540 146437 033236 001442      BICB     ATABIT(R4),ASNLST ;DELETE THE DRIVE FROM THE ASSIGNED LIST
4141 024546 006304      ASL      R4          ;MAKE ADDR INTO A WORD INDEX
4142 024550 016464 001720 001444      MOV      BLKADR(R4),DUNIT(R4) ;PUT ADDRESS IN DEASSIGN LIST
4143 024556 006204      ASR      R4          ;ANY MORE DRIVES ?
4144 024560 005303      2$:     DEC      R3          ;BR IF NOT
4145 024562 001412      BEQ      4$           ;BR IF NOT
4146 024564 005204      INC      R4          ;ANY MORE DRIVES ?
4147 024566 000760      BR       1$           ;BR IF NOT
4148 024570 122715 000101      3$:     CMPB     #'A,(R5)      ;DEASSIGN ALL DRIVES ?

```



```

4149 024574 001771          BEQ      2$          ;BR IF YES
4150 024576 012737 052125 026366  MOV      #UNTNOT,ASNMSG ;ADDR OF 'NOT ASSIGNED' MESSAGE
4151 024604 004737 026346          JSR      PC,ASNEER   ;REPORT IT
4152 024610 000207          4$:     RTS      PC
4153 024612 012703 000010          5$:     MOV      #8.,R3          ;SET UNIT COUNT TO 8
4154 024616 000744          BR       1$
4155
4156          ;'S' COMMAND (ROUTINE TO TYPE DRIVE PERFORMANCE SUMMARY)
4157
4158 024620 005004          SCMND:  CLR      R4
4159 024622 122715 000101          CMPB    #'A,(R5)          ;ALL STATISTICS ?
4160 024626 001421          BEQ      2$          ;BR IF YES
4161 024630 111504          MOVB    (R5),R4          ;GET DRIVE NUMBER
4162 024632 136437 033236 001442  BITB    ATABIT(R4),ASNLST ;SEE IF DRIVE ASSIGNED
4163 024640 001406          BEQ      1$          ;BR IF NOT
4164 024642 006304          ASL     R4          ;MAKE DRIVE ADDR INTO WORD INDEX
4165 024644 015400 001720          MOV     BLKADR(R4),R0    ;ADDR OF BLOCK
4166 024650 004737 022626          JSR     PC,TYPEST       ;TYPE DRIVE STATISTICS
4167 024654 000504          BR      9$          ;EXIT
4168 024656 012737 052125 026366 1$:     MOV     #UNTNOT,ASNMSG  ;ADDR OF 'NOT ASSIGNED' MSG
4169 024664 004737 026346          JSR     PC,ASNEER       ;TYPE ERROR MESSAGE
4170 024670 000476          BR      9$          ;EXIT
4171 024672 012703 000010          2$:     MOV     #8.,R3          ;DRIVE COUNT
4172 024676 136437 033236 001442 3$:     BITB    ATABIT(R4),ASNLST ;SEE IF DRIVE ASSIGNED
4173 024704 001004          BNE     4$          ;BR IF YES
4174 024706 005204          INC     R4          ;INCREMENT DRIVE ADDRESS
4175 024710 005303          DEC     R3          ;DECREMENT COUNTER
4176 024712 001371          BNE     3$          ;MORE TO CHECK
4177 024714 000464          BR      9$          ;NONE ASSIGNED, RETURN
4178 024716 004737 022544          4$:     JSR     PC,STATPR    ;TYPE ALL STATISTICS
4179 024722 105737 001220          TSTB   DATE          ;SEE IF 'DATE' ENTERED
4180 024726 001404          BEQ     11$         ;BR IF NOT
4181 024730 104401 053127          TYPE   ,DATEIS       ;'DATE:'
4182 024734 104401 001220          TYPE   ,DATE          ;THE OPERATOR ENTERED DATE
4183 024740 105737 001232          11$:   TSTB   OPERID        ;SEE IF OPERATOR I.D. ENTERED
4184 024744 001404          BEQ     12$         ;BR IF NOT
4185 024746 104401 053140          TYPE   ,IDIS         ;'OPERATOR I.D.:'
4186 024752 104401 001232          TYPE   ,OPERID       ;THE OPERATOR I.D.
4187 024756 104401 053162          12$:   TYPE   ,HEDLIN      ;HEADER LINE
4188 024762 012737 042032 025036  MOV     #DRIVED+$DRVID,6$ ;DRIVE I.D. FIELD ADDRESS
4189 024770 005004          CLR     R4          ;DRIVE ADDRESS
4190 024772 012703 000010          MOV     #8.,R3          ;COUNTER
4191 024776 136437 033236 001442 5$:     BITB    ATABIT(R4),ASNLST ;SEE IF DRIVE ASSIGNED
4192 025004 001417          BEQ     7$          ;BR IF NOT ASSIGNED
4193 025006 010446          MOV     R4,-(SP)      ;SAVE R4 FOR TYPEOUT
4194
4195 025010 104403          TYPOS  ;TYPE DRIVE NUMBER
4196 025012 002          .BYTE  2          ;GO TYPE--OCTAL ASCII
4197 025013 000          .BYTE  0          ;TYPE 2 DIGIT(S)
4198 025014 104401 052071          TYPE   ,LIN4SP       ;SUPPRESS LEADING ZEROS
4199 025020 105777 000012          TSTB   26$          ;4 SPACES
4200 025024 001003          BNE     10$         ;SEE IF DRIVE I.D. ENTERED
4201 025026 104401 053205          TYPE   ,NONE         ;BR IF DRIVE I.D. PRESENT
4202 025032 000404          BR      7$          ;TYPE 'NONE'
                          ;CONTINUE
    
```



```

42203 025034 104401          10$: TYPE                ;TYPE THE DRIVE I.D.
42204 025036 000000          6$:  WORD 0              ;ADDRESS OF DRIVE I.D. FIELD HERE
42205 025040 104401 001165    TYPE $CRLF              ;CR-LF
42206 025044 005303          7$: DEC R3              ;DECREMENT THE COUNTER
42207 025046 001405          BEQ R5                  ;BR IF AT END
42208 025050 062737 000304 025036 ADD $RPEC2+2,6$        ;INCREMENT THE MESSAGE FIELD ADDRESS
42209 025056 005204          INC R4                  ;INCREMENT DRIVE ADDRESS
42210 025060 000746          BR R5                   ;CONTINUE
42211 025062 104401 001165    8$: TYPE $CRLF          ;CR-LF
42212 025066 000207          9$: RTS PC
;
; 'W' COMMAND (ROUTINE TO WRITE A DATA PACK)
42215 025070 012737 177777 001216 DATAPK: MOV $-1,PACK    ;SET THE 'W' COMMAND INDICATOR
42216 025076 000137 024136    JMP ASSIGN              ;ASSIGN REQUESTED DRIVE
;
; 'R' COMMAND (ROUTINE TO READ A DATA PACK)
42219 025102 012737 000001 001216 REDAPK: MOV $1,PACK    ;SET THE 'READ' INDICATOR
42220 025110 000137 024136    JMP ASSIGN              ;ASSIGN THE REQUESTED DRIVE
;
; ROUTINE TO CLEAR THE DPB FOR THE ASSIGNED DRIVE
; CALL:
;       MOV $DPB RD      ;DPB ADDRESS
;       JSR PC,CLRDPB
;       RETURN
;
CLRDPB:
42223 025114          MOV R3,-(SP)           ;;PUSH R3 ON STACK
42224 025114 010346          MOV R4,-(SP)           ;;PUSH R4 ON STACK
42225 025116 010446          MOV R5,-(SP)           ;;PUSH R5 ON STACK
42226 025120 010546          MOV R0,R4              ;GET THE DPB ADDRESS
42227 025122 010004          ADD $2,R4              ;ADDRESS OF FIRST LOCN TO BE CLEARED
42228 025124 062704 000002    MOV $5,R3              ;NUMBER OF LOCNS TO BE CLEARED
42229 025130 012703 000005    1$: CLR (R4)+           ;CLEAR THE LOCATION
42230 025134 005024          DEC R3                 ;DECREMENT THE COUNTER
42231 025136 005303          BNE 1$                 ;BR IF NOT FINISHED
42232 025140 001375          ADD $2,R4              ;MOVE THE ADDRESS PAST THE 'REG' ADDR
42233 025142 062704 000002    MOV $NEXT-$REG,R3     ;NUMBER OF LOCNS TO BE CLEARED
42234 025146 012703 000070    2$: CLR (R4)+           ;CLEAR
42235 025152 005024          SUB $2,R3              ;DECREMENT THE LOCN COUNTER
42236 025154 162703 000002    BNE 2$                 ;BR IF NOT FINISHED
42237 025160 001374          ADD $12,R4             ;MOVE PAST ADDRESS LIMITS
42238 025162 062704 000014    MOV $RPEC2-MINSEC,R3 ;NUMBER OF LOCNS TO BE CLEARED
42239 025166 012703 000162    3$: CLR (R4)+           ;CLEAR A LOCATION
42240 025172 005024          SUB $2,R3              ;DECREMENT THE COUNTER
42241 025174 162703 000002    BNE 3$                 ;BR IF NOT DONE
42242 025200 001374          MOVB BEGCOD,$CODE(R0) ;INITIAL COMMAND CODE
42243 025202 113760 001414 000024 MOV BEGCOD,R1          ;GET THE ACTUAL OP CODE
42244 025210 013701 001414    MOV COMB(R1),$COMND(R0); OPERATION CODE
42245 025214 116160 001740 000002 MOVB BEGPAT,$PATIC(R0); PATTERN CODE
42246 025222 113760 001412 000030 MOVB $PATIC(R0)        ;CONVERT CODE TO A TABLE INDEX
42247 025230 106360 000030    ASLB BEGSIZ,$WRDL(R0) ;BEGINNING RECORD SIZE
42248 025234 013760 001416 000020

```



```

4257 025242 013760 001416 000004      MOV     BEGSIZ,$WRDM(R0)      ;VALUE FOR DATA TRANSFER
4258 025250 005460 000004      NEG     $WRDM(R0)           ;MAKE IT INTO 2'S COMPLEMENT
4259 025254 012760 000400 000022      MOV     #256,$SSEC(R0)     ;INITIAL VALUE OF SECTOR SIZE
4260 025262 132760 000001 000024      BITB   #1,$CODE(R0)       ;HEADER ORDER ?
4261 025270 001403      BEQ     4$                 ;BR IF NOT
4262 025272 062760 000004 000022      ADD     #4,$SSEC(R0)      ;ADD HEADER SIZE TO SECTOR SIZE
4263 025300      4$:
4264 025300 012605      MOV     (SP)+,R5           ;:POP STACK INTO R5
4265 025302 012604      MOV     (SP)+,R4           ;:POP STACK INTO R4
4266 025304 012603      MOV     (SP)+,R3           ;:POP STACK INTO R3
4267 025306 000207      RTS     PC                 ;:RETURN
4268
4269
4270      ;ROUTINE TO GET ADDRESS LIMITS FROM THE OPERATOR
4271 025310 010346      DRVPRM: MOV     R3,-(SP)        ;SAVE R3
4272 025312 010446      MOV     R4,-(SP)        ;SAVE R4
4273 025314 005737 000042      TST     42                ;RUNNING UNDER MONITOR CONTROL
4274 025320 001035      BNE     3$                ;BR IF YES
4275 025322 104401 053023      TYPE   ,ENTLMT           ;'ENTER ADDRESSES'
4276 025326 006204      ASR     R4                ;CONVERT INDEX TO DRIVE NUMBER
4277 025330 010446      MOV     R4,-(SP)        ;SAVE R4 FOR TYPEOUT
4278
4279      ;TYPE DRIVE NUMBER
4280 025332 104403      TYPOS  ;GO TYPE--OCTAL ASCII
4281 025334 002      .BYTE 2                 ;TYPE 2 DIGIT(S)
4282 025335 000      .BYTE 0                 ;SUPPRESS LEADING ZEROS
4283 025336 104401 053514      TYPE   ,SLASH           ;/
4284 025342 012737 052302 025406      MOV     #RP04B,2$        ;ADDRESS OF 'RP04' MESSAGE
4285 025350 132764 000001 033132      BITB   #BIT00,DRVTP(R4)  ;RP04 ?
4286 025356 001012      BNE     1$                ;BR IF YES
4287 025360 012737 052307 025406      MOV     #RP05,2$        ;ADDRESS OF 'RP05' MESSAGE
4288 025366 132764 000002 033132      BITB   #BIT01,DRVTP(R4)  ;RP05 ?
4289 025374 001003      BNE     1$                ;BR IF YES
4290 025376 012737 052314 025406      MOV     #RP06,2$        ;ADDRESS OF 'RP06' MESSAGE
4291 025404 104401      1$: TYPE   ,MESSAGE WHICH FOLLOWS
4292 025406 000000      2$: .WORD 0             ;MESSAGE ADDRESS
4293 025410 104401 001165      TYPE   ,$CRLF           ;CR-LF
4294 025414 012737 000632 001350      3$: MOV     #410,CYLIMT     ;ASSUME AN RP04/5
4295 025422 132764 000004 033132      BITB   #BIT02,DRVTP(R4)  ;SEE IF RP06
4296 025430 001403      BEQ     4$                ;BR IF NOT
4297 025432 012737 001456 001350      MOV     #814,CYLIMT     ;CHANGE LIMIT TO 814
4298 025440 062760 177777 000122      4$: ADD     #-1,$FIRST(R0) ;SEE IF FIRST TIME STARTED
4299 025446 103417      BCS     5$                ;BR IF NOT
4300 025450 013760 001350 000106      MOV     CYLIMT,MAXCYL(R0) ;LOAD MAXIMUM CYLINDER
4301 025456 005060 000110      CLR     MINCYL(R0)       ;CLEAR MINIMUM CYLINDER
4302 025462 013760 001346 000112      MOV     TRKLMT,MAXTRK(R0) ;LOAD MAXIMUM TRACK
4303 025470 005060 000114      CLR     MINTRK(R0)       ;CLEAR MINIMUM TRACK
4304 025474 013760 001344 000116      MOV     SECLMT,MAXSEC(R0) ;LOAD MAXIMUM SECTOR
4305 025502 005060 000120      CLR     MINSEC(R0)       ;CLEAR MINIMUM SECTOR
4306 025506 006304      5$: ASL     R4                ;SETUP TO ADDRESS WORDS
4307 025510 016403 053674      MOV     TABLE(R4),R3    ;PARAMETER TABLE ADDRESS
4308 025514 013763 001350 000002      MOV     CYLIMT,2(R3)     ;LOAD CYLINDER LIMIT FOR LAST CYLINDER
4309 025522 013763 001350 000010      MOV     CYLIMT,10(R3)    ;LOAD CYLINDER LIMIT FOR STARTING CYLINDER
4310 025530 005737 000042      TST     42                ;UNDER MONITOR CONTROL ?
4311 025534 001002      BNE     6$                ;BR IF YES

```



```

DZRJD.019          GENERAL SUPPORT SUBROUTINES
4311 025536 004737 026222          JSR      PC PARENT          ;GET THE DRIVE'S PARAMETERS
4312 025542 116060 000120 000010 6$:  MOVB   MINSEC(RO), $SEC(RO) ;INITIAL SECTOR VALUE
4313 025550 116060 000114 000011      MOVB   MINTRK(RO), $TRK(RO) ;INITIAL TRACK VALUE
4314 025556 016060 000110 000012      MOV   MINCYL(RO), $CYL(RO) ;INITIAL CYLINDER VALUE
4315 025564 012604          MOV   (SP)+, R4          ;RESTORE R4
4316 025566 012603          MOV   (SP)+, R3          ;RESTORE R3
4317 025570 000207          RTS      PC              ;RETURN
4318
4319
4320          ;ROUTINE TO GET THE DRIVE I.D. FROM THE OPERATOR
4321 025572 010546          GETID:  MOV   R5, -(SP)      ;SAVE R5
4322 025574 005737 000042          TST   42                ;UNDER MONITOR CONTROL ?
4323 025600 001036          BNE   2$                ;BR IF NOT
4324 025602 005037 001262          1$:  CLR   CFLAG           ;CLEAR THE 'CONTROL C' FLAG
4325 025606 104401 052776          TYPE ,ENTDRV           ;'ENTER DRV I.D.:'
4326 025612 005046          CLR   -(SP)            ;CLEAR THE STACK
4327 025614 111016          MOVB  (RO), (SP)       ;PUT THE DRIVE NUMBER ON THE STACK
4328 025616 104403          TYPOS ;TYPE THE DRIVE NUMBER
4329 025620          .BYTE 2              ;TYPE 2 DIGITS
4330 025621          .BYTE 0              ;SUPPRESS LEADING ZEROS
4331 025622 104401 001165          TYPE , $CRLF          ;CR-LF
4332 025626 104411          RDLIN ;READ THE ENTRY
4333 025630 012605          MOV   (SP)+, R5       ;GET THE ENTRY ADDRESS
4334 025632 005737 001262          TST   CFLAG           ;'CONTROL C' ENTERED ?
4335 025636 001361          BNE   1$              ;BR IF IT WAS
4336 025640 121527 000056          CMPB  (R5), #'.       ;PERIOD ENTERED ?
4337 025644 001414          BEQ   2$              ;BR IF YES
4338 025646 112560 000224          MOVB  (R5)+, $DRVID(RO) ;STORE THE I.D.
4339 025652 112560 000225          MOVB  (R5)+, $DRVID+1(RO) ;STORE THE I.D.
4340 025656 112560 000226          MOVB  (R5)+, $DRVID+2(RO) ;STORE THE I.D.
4341 025662 112560 000227          MOVB  (R5)+, $DRVID+3(RO) ;STORE THE I.D.
4342 025666 112560 000230          MOVB  (R5)+, $DRVID+4(RO) ;STORE THE I.D.
4343 025672 112560 000231          MOVB  (R5)+, $DRVID+5(RO) ;STORE THE I.D.
4344 025676 012605 2$:  MOV   (SP)+, R5       ;RESTORE R5
4345 025700 000207          RTS      PC              ;RETURN
4346
4347          ;ROUTINE TO GET THE ADDRESSES OF ANY BAD SECTORS (UP TO A MAX OF 16)
4348
4349          GETADR:
4350 025702 010146          MOV   R1, -(SP)       ;; PUSH R1 ON STACK
4351 025704 010246          MOV   R2, -(SP)       ;; PUSH R2 ON STACK
4352 025706 010346          MOV   R3, -(SP)       ;; PUSH R3 ON STACK
4353 025710 010446          MOV   R4, -(SP)       ;; PUSH R4 ON STACK
4354 025712 005737 000042          TST   42                ;UNDER MONITOR CONTROL ?
4355 025716 001012          BNE   14$             ;BR IF YES
4356 025720 005037 001262          14$:  CLR   CFLAG           ;CLEAR 'CONTROL C' FLAG
4357 025724 104401 053064          TYPE ,ENTADR          ;ENTER SECTOR ADDRESSES
4358 025730 005046          CLR   -(SP)            ;CLEAR THE STACK
4359 025732 111016          MOVB  (RO), (SP)       ;PUT THE DRIVE NUMBER ON THE STACK
4360 025734 104403          TYPOS ;TYPE THE DRIVE NUMBER
4361 025736          .BYTE 2              ;TYPE 2 DIGITS
4362 025737          .BYTE 0              ;SUPPRESS LEADING ZEROS
4363 025740 104401 001165          TYPE , $CRLF          ;CR-LF
4364 025744 012703 000040          15$:  MOV   #32., R3       ;NUMBER OF LOCATIONS IN THE TABLE TO PRESET

```

4365	025750	012704	000124		MOV	#\$BDSEC,R4	:TABLE INCREMENT
4366	025754	060004			ADD	R0,R4	:BLOCK STARTING ADDRESS
4367	025756	012724	177777	1\$:	MOV	#-1,(R4)+	:SET LOCATION TO 1'S
4368	025762	005303			DEC	R3	:DECREMENT TABLE SIZE COUNT
4369	025764	001374			BNE	1\$:BR IF NOT FINISHED WITH TABLE
4370	025766	005737	000042		TST	42	:UNDER MONITOR CONTROL ?
4371	025772	001106			BNE	13\$:BR IF YES
4372	025774	162704	000100		SUB	#64.,R4	:SET POINTER TO BEGINNING OF TABLE
4373	026000	012703	000020		MOV	#16.,R3	:NUMBER OF ADDRESSES IN TABLE
4374	026004	104411		2\$:	RDLIN		:GET ADDRESS FROM OPERATOR
4375	026006	012601			MOV	(SP)+,R1	:TEXT POINTER
4376	026010	005737	001262		TST	CFLAG	: 'CONTROL C' ENTERED ?
4377	026014	001341			BNE	14\$:BR IF IT WAS
4378	026016	013702	001350		MOV	CYLIMT,R2	:UPPER LIMIT OF INPUT
4379	026022	004537	027562		JSR	R5,CK.DIG	:CHECK THE DIGIT(S)
4380	026026	026170			12\$:CARRIAGE RETURN ONLY ENTERED
4381	026030	026210			13\$:PERIOD ONLY ENTERED
4382	026032	026170			12\$:ILLEGAL INPUT
4383	026034	026045			4\$:TERMINATED WITH A CARRIAGE RETURN
4384	026036	026052			5\$:TERMINATED WITH A "."
4385	026040	026042			3\$:TERMINATED WITH A ":
4386	026042	010214		3\$:	MOV	R2,(R4)	:CYLINDER ADDRESS
4387	026044	000461			BR	13\$:EXIT, PERIOD ENTERED
4388	026046	010214		4\$:	MOV	R2,(R4)	:CYLINDER ADDRESS
4389	026050	000442			BR	11\$:FINISHED WITH THIS ADDRESS, 'CR' ENTERED
4390	026052	010214		5\$:	MOV	R2,(R4)	:CYLINDER ADDRESS
4391	026054	013702	001346		MOV	TRKLMT,R2	:UPPER LIMIT OF INPUT
4392	026060	004537	027562		JSR	R5,CK.DIG	:CHECK THE DIGIT(S)
4393	026064	026170			12\$:CARRIAGE RETURN ONLY ENTERED
4394	026066	026210			13\$:PERIOD ONLY ENTERED
4395	026070	026170			12\$:ILLEGAL INPUT
4396	026072	026106			7\$:TERMINATED WITH A CARRIAGE RETURN
4397	026074	026114			8\$:TERMINATED WITH A "."
4398	026076	026100			6\$:TERMINATED WITH A ":
4399	026100	110264	000003	6\$:	MOVB	R2,3(R4)	:TRACK ADDRESS
4400	026104	000441			BR	13\$:EXIT, ENTRY TERMINATED BY PERIOD
4401	026106	110264	000003	7\$:	MOVB	R2,3(R4)	:TRACK ADDRESS
4402	026112	000421			BR	11\$:FINISHED WITH THIS ADDRESS, 'CR' ENTERED
4403	026114	110264	000003	8\$:	MOVB	R2,3(R4)	:TRACK ADDRESS
4404	026120	013702	001344		MOV	SECLMT,R2	:UPPER LIMIT OF INPUT
4405	026124	004537	027562		JSR	R5,CK.DIG	:CHECK THE DIGIT(S)
4406	026130	026170			12\$:CARRIAGE RETURN ONLY ENTERED
4407	026132	026210			13\$:PERIOD ONLY ENTERED
4408	026134	026170			12\$:ILLEGAL INPUT
4409	026136	026152			10\$:TERMINATED WITH A CARRIAGE RETURN
4410	026140	026170			12\$:TERMINATED WITH A "."
4411	026142	026144			9\$:TERMINATED WITH A ":
4412	026144	110264	000002	9\$:	MOVB	R2,2(R4)	:SECTOR ADDRESS
4413	026150	000417			BR	13\$:EXIT, ENTRY TERMINATED BY PERIOD
4414	026152	110264	000002	10\$:	MOVB	R2,2(R4)	:SECTOR ADDRESS
4415	026156	005303		11\$:	DEC	R3	:MORE ENTRIES ?
4416	026160	001413			BEQ	13\$:BR IF NOT
4417	026162	062704	000004		ADD	#4,R4	:INCREMENT THE TABLE POINTER
4418	026166	000706			BR	2\$:CONTINUE


```

4419 026170 012714 177777 12$: MOV #-1,(R4) ;CLEAR PRESENT TABLE ENTRY
4420 026174 012764 177777 000002 MOV #-1,2(R4) ;CLEAR PRESENT TABLE ENTRY
4421 026202 104401 053214 TYPE ,BADENT ;'INVALID ENTRY'
4422 026206 000676 BR 2$ ;TRY AGAIN
4423 026210 13$:
4424 026210 012604 MOV (SP)+,R4 ;:POP STACK INTO R4
4425 026212 012603 MOV (SP)+,R3 ;:POP STACK INTO R3
4426 026214 012602 MOV (SP)+,R2 ;:POP STACK INTO R2
4427 026216 012601 MOV (SP)+,R1 ;:POP STACK INTO R1
4428 026220 000207 RTS PC ;RETURN
4429
4430 ;PARAMETER ENTRY ROUTINE
4431 ;CALL
4432 ;
4433 ; MOV #ADR,R3 ;PARAMETER TABLE ADDRESS
4434 ; JSR PC,PARENT ;GET THE PARAMETERS
4435 026222 010346 PARENT: MOV R3,-(SP) ;SAVE THE PARAMETER TABLE ADDRESS
4436 026224 005037 001262 CLR CFLAG ;CLEAR THE 'CONTROL C' FLAG
4437 026230 012337 026240 1$: MOV (R3)+,3$ ;ADDRESS OF PARAMETER NAME
4438 026234 001442 BEQ 9$ ;BR IF AT END OF TABLE
4439 026236 104401 TYPE ;TYPE THE PARAMETER NAME
4440 026240 000000 3$: .WORD 0 ;ADDRESS OF PARAMETER NAME TEXT
4441 026242 012302 MOV (R3)+,R2 ;MAXIMUM PARAMETER VALUE
4442 026244 012305 MOV (R3)+,R5 ;ADDRESS OF PARAMETER
4443 026246 011546 MOV (R5),-(SP) ;CURRENT VALUE OF PARAMETER
4444 026250 104405 TYPDS ;TYPE THE CURRENT VALUE OF THE PARAMETER
4445 026252 104401 053514 TYPE ,SLASH ; /
4446 026256 104411 RDLIN ;READ THE KEYBOARD
4447 026260 012601 MOV (SP)+,R1 ;INPUT ASCII STRING ADDRESS
4448 026262 005737 001262 TST CFLAG ;'CONTROL C' ENTERED ?
4449 026266 001021 BNE 8$ ;BR IF IT WAS
4450 026270 004537 027562 JSR R5,CK.DIG ;CHECK THE DIGIT(S)
4451 026274 026230 1$ ;CARRIAGE RETURN ONLY ENTERED
4452 026276 026342 9$ ;PERIOD ONLY ENTERED
4453 026300 026314 6$ ;ILLEGAL INPUT
4454 026302 026310 5$ ;TERMINATED WITH A CARRIAGE RETURN
4455 026304 026314 6$ ;TERMINATED WITH A "."
4456 026306 026326 7$ ;TERMINATED WITH A " "
4457 026310 010215 5$: MOV R2,(R5) ;MOVE NEW VALUE TO PARAMETER LOCATION
4458 026312 000746 1$ BR 1$ ;GET MORE PARAMETERS
4459 026314 104401 053214 6$: TYPE ,BADENT ;'BAD ENTRY'
4460 026320 162703 000006 SUB #6,R3 ;DECREMENT THE TABLE POINTER
4461 026324 000741 1$ BR 1$ ;TRY AGAIN
4462 026326 010215 7$: MOV R2,(R5) ;NEW VALUE
4463 026330 000404 9$ BR 9$ ;EXIT
4464 026332 005037 001262 8$: CLR CFLAG ;CLEAR THE 'CONTROL C' FLAG
4465 026336 011603 MOV (SP),R3 ;RELOAD THE PARAMETER TABLE ADDRESS
4466 026340 000733 1$ BR 1$ ;TRY AGAIN
4467 026342 005726 9$: TST (SP)+ ;CORRECT THE STACK POINTER
4468 026344 000207 RTS PC ;RETURN
4469
4470 ;TYPEOUT ASSIGN/DEASSIGN ERROR MESSAGE
4471 ;CALL
4472 ; MOV #MESADR,ASNMSG ;ERROR MESSAGE ADDRESS
    
```

```

4473      :      JSR      PC,ASNERR
4474      :      RETURN
4475
4476      026346 104401 052725      ASNERR: TYPE      ,QUES      ;QUESTION MARK
4477      026352 104401 052076      TYPE      'UNTMSG      ;TYPE 'DRIVE'
4478      026356 010446      MOV      R4,-(SP)      ;SAVE R4 FOR TYPEOUT
4479      :      :      :      :      :TYPE DRIVE NUMBER
4480      026360 104403      TYPOS      :      :      :      :      :GO TYPE--OCTAL ASCII
4481      026362      002      .BYTE      2      :      :      :      :      :TYPE 2 DIGIT(S)
4482      026363      000      .BYTE      0      :      :      :      :      :SUPPRESS LEADING ZEROS
4483      026364 104401      TYPE      :      :      :      :      :TYPE SPECIFIC MESSAGE
4484      026366 000000      ASNMSG: .WORD      0      :      :      :      :MESSAGE ADDRESS
4485      026370 104401 001165      TYPE      ,SCLRF
4486      026374 000207      RTS      PC
4487
4488      :      :      :      :      :DEASSIGN DRIVE IF A FATAL ERROR OCCURS
4489      :      :      :      :      :CALL
4490      :      :      :      :      :
4491      :      :      :      :      :JSR      PC,DROP
4492      :      :      :      :      :RETURN
4493
4493      026376 005004      DROP:  CLR      R4      ;CLEAR R4 FOR DRIVE NUMBER
4494      026400 111004      MOV      (R0),R4      ;MOVE DRIVE NUMBER TO R4
4495      026402 146437 033236 001442      BIC      ATABIT(R4),ASNLS      ;REMOVE DRIVE FROM ASSIGNED LIST
4496      026410 006304      ASL      R4      ;MAKE DRIVE NUMBER INTO A TABLE INDEX
4497      026412 010064 001444      MOV      R0,DUNIT(R4)      ;PUT DRIVE IN DROP LIST
4498      026416 104401 001165      TYPE      ,SCLRF
4499      026422 104401 001165      TYPE      ,SCLRF
4500      026426 104401 052500      TYPE      ,DROPN      ;TYPE 'DROPPING DRIVE'
4501      026432 104401 052611      TYPE      ,DRNUM      ;'DRIVE #'
4502      026436 006204      ASR      R4      ;DRIVE NUMBER
4503      026440 010446      MOV      R4,-(SP)      ;SAVE R4 FOR TYPEOUT
4504
4505      026442 104403      TYPOS      :      :      :      :      :TYPE DRIVE NUMBER
4506      026444      002      .BYTE      2      :      :      :      :      :GO TYPE--OCTAL ASCII
4507      026445      000      .BYTE      0      :      :      :      :      :TYPE 2 DIGIT(S)
4508      026446 104401 001165      TYPE      ,SCLRF      ;SUPPRESS LEADING ZEROS
4509      026452 000207      RTS      PC
4510
4511      :      :      :      :      :ROUTINE TO DEASSIGN DRIVE IF ERRORS BECOMES EXCESSIVE
4512
4513      026454 032777 000020 152456      ABNRL: BIT      #SW04,ASWR      ;SEE IF SWITCH 4 SET
4514      026462 001006      BNE      1$      ;BR IF IT'S SET
4515      026464 023760 001366 000056      CMP      MAXER,$TOTAL(R0)      ;CHECK TOTAL ERROR VALUE
4516      026472 103002      BHS      1$      ;BR IF ERRORS DONOT EXCEED MAX
4517      026474 000137 026376      JMP      DROP      ;DEASSING THE DRIVE
4518      026500 000207      1$:      RTS      PC      ;RETURN
4519
4520      :      :      :      :      :ROUTINE TO CHECK FOR END OF PASS AND END OF TEST
4521
4522      026502 005737 001410      EOP:  TST      ENDET      ;END OF PASS DETERMINED BY SEEKS OR WORDS ?
4523      026506 001412      BEQ      EOP1      ;BR IF SEEKS
4524      026510 026037 000054 001354      CMP      $READ+2(R0),ENDCON+2      ;CHECK MSW OF WORDS READ COUNT
4525      026516 101017      BHI      EOP2      ;BR IF MSW GREATER THAN LIMIT
4526      026520 103405      BLO      EOP1      ;BR IF MSW LESS THAN LIMIT

```


H10

4527	026522	026037	000052	001352		CMP	\$READ(RO),ENDCON	:CHECK LSW AGAINST LIMIT
4528	026530	103012				BHIS	EOP2	:BR IF EQUAL OR GREATER
4529	026532	000510				BR	EOPX	:EXIT
4530	026534	026037	000044	001360	EOP1:	CMP	\$POSIT+2(RO),ENDSEK+2	:CHECK MSW OF SEEK COUNT
4531	026542	101005				BHI	EOP2	:BR IF MSW GREATER THAN LIMIT
4532	026544	103503				BLO	EOPX	:EXIT IF MSW LESS THAN LIMIT
4533	026546	026037	000042	001356		CMP	\$POSIT(RO),ENDSEK	:CHECK LSW OF SEEK COUNT
4534	026554	103477				BLO	EOPX	:EXIT IF LSW LESS THAN LIMIT
4535	026556	104401	001165		EOP2:	TYPE	,\$SRLF	:CR-LF
4536	026562	104401	052534			TYPE	,ENDPAS	:END OF PASS FOR THE DRIVE
4537	026566	016046	000070			MOV	\$PASSC(RO),-(SP)	:PUT PASS COUNT ON THE STACK
4538	026572	104405				TYPDS		:CONVERT PASS COUNT TO DECIMAL AND TYPE IT
4539	026574	111037	001246			MOVB	(RO),UNIT	:STORE THE DRIVE NUMBER
4540	026600	032777	000020	152332		BIT	#SW04,@SWR	:SWITCH 4 SET ?
4541	026606	001017				BNE	1\$:BR IF SET
4542	026610	026037	000070	001362		CMP	\$PASSC(RO),PASCNT	:SEE IF AT END OF TEST
4543	026616	103413				BLO	1\$:BR IF NOT
4544	026620	104401	052550			TYPE	,ENDTST	:TYPE 'END OF TEST'
4545	026624	104401	052611			TYPE	,DRNUM	: 'DRIVE #'
4546	026630	013746	001246			MOV	UNIT,-(SP)	:SAVE UNIT FOR TYPEOUT
4547								:TYPE DRIVE NUMBER
4548	026634	104403				TYPOS		:GO TYPE--OCTAL ASCII
4549	026636	002				.BYTE	2	:TYPE 2 DIGIT(S)
4550	026637	000				.BYTE	0	:SUPPRESS LEADING ZEROS
4551	026640	104401	001165			TYPE	,\$SRLF	:CR-LF
4552	026644	000431				BR	3\$:DEASSIGN THE DRIVE
4553	026646	104401	052611		1\$:	TYPE	,DRNUM	: 'DRIVE #'
4554	026652	013746	001246			MOV	UNIT,-(SP)	:SAVE UNIT FOR TYPEOUT
4555								:TYPE DRIVE NUMBER
4556	026656	104403				TYPOS		:GO TYPE--OCTAL ASCII
4557	026660	002				.BYTE	2	:TYPE 2 DIGIT(S)
4558	026661	000				.BYTE	0	:SUPPRESS LEADING ZEROS
4559	026662	104401	001165			TYPE	,\$SRLF	:CR-LF
4560	026666	004737	022626			JSR	PC,TYPEST	:TYPE THE DRIVE'S STATISTICS
4561	026672	010346				MOV	R3,-(SP)	:SAVE R3
4562	026674	010446				MOV	R4,-(SP)	:SAVE R4
4563	026676	010004				MOV	RO,R4	:DRIVE'S BLOCK ADDRESS
4564	026700	062704	000036			ADD	,\$OPERC,R4	:ADD THE STARTING ADDR OF SECTIONS TO CLEAR
4565	026704	012703	000010			MOV	#8.,R3	:NUMBER OF LOCNS TO BE CLEARED
4566								: (ERROR COUNTERS NOT CLEARED)
4567	026710	005024			2\$:	CLR	(R4)+	:CLEAR THE LOCN
4568	026712	005303				DEC	R3	:DECREMENT THE LOCATION COUNTER
4569	026714	001375				BNE	2\$:BR IF MORE TO GO
4570	026716	012604				MOV	(SP)+,R4	:RESTORE R4
4571	026720	012603				MOV	(SP)+,R3	:RESTORE R3
4572	026722	005260	000070			INC	\$PASSC(RO)	:INCREMENT THE PASS COUNT
4573	026726	000412				BR	EOPX	:EXIT
4574	026730	104401	001165		3\$:	TYPE	,\$SRLF	
4575	026734	005004				CLR	R4	:CLEAR R4 FOR DRIVE NUMBER
4576	026736	111004				MOVB	(RO),R4	:MOVE DRIVE NUMBER
4577	026740	146437	033236	001442		BICB	ATABIT(R4),ASNLS	:DELETE DRIVE FROM ASSIGNED LIST
4578	026746	006304				ASL	R4	:MAKE DRIVE NUMBER INTO TABLE INDEX
4579	026750	010064	001444			MOV	RO,DUNIT(R4)	:PUT BLOCK ADDRESS INTO DROP LIST
4580	026754	000207			EOPX:	RTS	PC	:RETURN

```

4581
4582
4583
4584
4585
4586
4587
4588 026756 013746 032402
4589 026762 013746 032400
4590 026766 010546
4591 026770 004737 027004
4592 026774 012605
4593 026776 005726
4594 027000 000240
4595 027002 000207
4596
4597
4598
4599
4600
4601 027004 104412
4602 027006 016605 000026
4603 027012 005004
4604 027014 016602 000030
4605 027020 016603 000032
4606 027024 005000
4607 027026 005001
4608 027030 004737 027052
4609 027034 010166 000030
4610 027040 010366 000032
4611 027044 104413
4612 027046 012616
4613 027050 000207
4614
4615
4616
4617
4618
4619
4620
4621
4622 027052 012746 000040
4623 027056 010446
4624 027060 010546
4625 027062 005466 000002
4626 027066 005416
4627 027070 005666 000002
4628 027074 061601
4629 027076 005500
4630 027100 066600 000002
4631 027104 103445
4632 027106 005046
4633 027110 006103
4634 027112 006102

;ROUTINE TO GET THE REMAINDER OF THE RANDOM NUMBER
;CALL
;      MOV      NUMBER,R5      ;DIVISOR INTO R5
;      JSR      PC,GETREM
;      RETURN
;      REMAINDER IS IN R5
GETREM: MOV      $LONUM,-(SP)    ;STORE RANDOM NUMBER ON THE STACK FOR DIVIDE
        MOV      $HINUM,-(SP)  ;UPPER PART
        MOV      R5,-(SP)      ;PUT THE DIVISOR ONTO THE STACK
        JSR      PC,LINKDV     ;DIVIDE THE RANDOM NUMBERS
        MOV      (SP)+,R5      ;PUT THE REMAINDER INTO R5
        TST      (SP)+         ;ADJUST THE STACK POINTER
        NOP
        RTS      PC           ;FOR DEBUGGING HALT

;LINK ROUTINE TO THE DIVISION UTILITY SUBROUTINE
;THIS ROUTINE ALLOWS THE 'SYSMAC' DIVIDE ROUTINE
;CALLING SEQUENCE TO BE USED
LINKDV: SAVREG
        MOV      26(SP),R5     ;STORE R0 - R5
        CLR      R4           ;DIVISOR
        MOV      30(SP),R2     ;OTHER DIVISOR WORD
        MOV      32(SP),R3     ;UPPER DIVIDEND WORD
        CLR      R0           ;LOWER DIVIDEND WORD
        CLR      R1           ;CLEAR OTHER DIVIDEND REGISTERS
        JSR      PC,M.DPID     ;GO TO THE DIVIDE ROUTINE
        MOV      R1,30(SP)     ;REMAINDER ON THE STACK
        MOV      R3,32(SP)     ;QUOTIENT ON THE STACK
        RESREG
        MOV      (SP)+,(SP)    ;RESTORE R0 - R5
        RTS      PC           ;MOVE RETURN UP THE STACK

;DIVISION UTILITY SUBROUTINE
;R0-R1-R2-R3=DIVIDEND
;R4-R5=DIVISOR
;R0-R1=REMAINDER AFTER DIVISION
;R2-R3=QUOTIENT AFTER DIVISION
;ENTER WITH JSR PC,M.DPID
M.DPID: MOV      #40,-(SP)     ;COUNTER FOR DIVISION CYCLES
        MOV      R4,-(SP)     ;HIGH ORDER
        MOV      R5,-(SP)     ;LOW ORDER DIVISOR TO THE STACK
        NEG      2(SP)        ;FORM NEGATIVE
        NEG      @SP          ;VERSION OF THE DIVISOR
        SBC      2(SP)
        ADD      @SP,R1
        ADC      R0           ;PERFORM THE INITIAL SUBTRACTION
        ADD      2(SP),R0
        BCS      M.DP50       ;IF CARRY THEN OVERFLOW HAS OCCURRED
        CLR      -(SP)        ;THIS IS A LONGER LASTING CARRY BIT
M.DP40: ROL      R3
        ROL      R2

```



```

4635 027114 006101      ROL    R1
4636 027116 006100      ROL    R0
4637 027120 005716      TST    QSP                ;TEST "CARRY" INDICATOR
4638 027122 001410      BEQ    M.DP41            ;IF NO "CARRY" THEN ADD ELSE SUBTRACT
4639 027124 005016      CLR    QSP                ;CLEAR UP FOR NEXT TIME
4640 027126 066601 000002  ADD    2(SP),R1
4641 027132 005500      ADC    R0                ;ADD -(DIVISOR)
4642 027134 005516      ADC    QSP                ;SET "CARRY"
4643 027136 066600 000004  ADD    4(SP),R0;<- I
4644 027142 000404      BR     M.DP42
4645 027144 060501      M.DP41: ADD   R5,R1
4646 027146 005500      ADC    R0                ;ADD +(DIVISOR)
4647 027150 005516      ADC    QSP                ;SET "CARRY"
4648 027152 060400      M.DP42: ADD   R4,R0;<- I
4649 027154 005516      ADC    QSP                ;SET "CARRY"
4650 027156 005716      TST    QSP                ;TEST THE UPDATE INDICATOR
4651 027160 001401      BEQ    .+4                ;IF ZERO FORGET IT
4652 027162 005203      INC    R3                ;NO CARRY POSSIBLE HERE
4653 027164 005366 000006  DEC    6(SP)                ;DECREMENT COUNTER
4654 027170 003347      BGT    M.DP40            ;BRANCH IF MORE TO DO
4655 027172 006003      ROR    R3
4656 027174 103404      BCS    M.DP44
4657 027176 060501      ADD    R5,R1
4658 027200 005500      ADC    R0
4659 027202 060400      ADD    R4,R0
4660 027204 000241      CLC
4661 027206 006103      M.DP44: ROL   R3
4662 027210 062706 000010  ADD    #10,SP            ;ADJUST STACK BY 4 WORDS
4663 027214 000242      CLV
4664 027216 000207      RTS    PC
4665 027220 062706 000006  M.DP50: ADD   #6,SP
4666 027224 000262      SEV
4667 027226 000207      RTS    PC
4668
4669
4670      ;ROUTINE TO REPLACE LEADING ZEROS IN A NUMERIC STRING WITH SPACES
4671      ;CALL
4672      ;      MOV    #ADR, -(SP)      ;ADDRESS OF NUMBER (IN ASCII)
4673      ;      JSR    R5, REPLZ
4674      ;      .WORD  N                ;'N' IS NUMBER OF DIGITS TO BE TYPED
4675
4676 027230 010046      REPLZ: MOV    R0, -(SP)      ;SAVE R0
4677 027232 012746 000012  MOV    #10, -(SP)        ;MAXIMUM NUMBER OF DIGITS TO BE TYPED
4678 027236 162516      SUB    (R5)+, (SP)        ;SUBTRACT DIGITS TO FORM INDEX
4679 027240 016600 000006  MOV    6(SP), R0          ;ADDRESS OF NUMBER TO R0
4680 027244 122710 000060  1$:   CMPB  #'0', (R0)      ;BYTE EQUAL TO ASCII '0' ?
4681 027250 001004      BNE    2$                ;BR IF NOT
4682 027252 112710 000040  MOVB  #40, (R0)          ;REPLACE THE ZERO WITH A SPACE
4683 027256 005200      INC    R0                ;INCREMENT THE BYTE ADDRESS
4684 027260 000771      BR     1$                ;GO BACK AND LOOK FOR MORE LEADING ZEROS
4685 027262 105710      2$:   TSTB  (R0)            ;SEE IF ZERO BYTE TERMINATOR
4686 027264 001003      BNE    3$                ;BR IF NOT
4687 027266 005300      DEC    R0                ;BACKUP STRING POINTER
4688 027270 112710 000060  MOVB  #'0', (R0)        ;PUT A ZERO BACK IN
    
```

```

4689 027274 016637 000006 027310 3$: MOV 6(SP),4$ ;PUT ADDRESS IN LOCATION FOR TYPEOUT
4690 027302 062637 027310 ADD (SP)+,4$ ;BEGINNING OF SIGNIFICANT DIGITS
4691 027306 104401 TYPE ;TYPE THE NUMBER
4692 027310 000000 4$: .WORD 0 ;ADDRESS OF NUMBER
4693 027312 012600 MOV (SP)+,R0 ;RESTORE R0
4694 027314 012616 MOV (SP)+,(SP) ;MOVE RETURN ADDRESS
4695 027316 000205 RTS R5 ;RETURN
4696
4697 ;TYPE NUMERICAL ASCIZ STRING SUPPRESS LEADING ZEROS
4698
4699 ;CALL
4700 ; MOV #NUMADR, -(SP) ;FIRST ADDRESS OF ASCIZ STRING
4701 ; JSR PC,$SUPRS
4702
4703 027320 010046 $SUPRS: MOV R0, -(SP) ;SAVE R0
4704 027322 016600 000004 MOV 4(SP),R0 ;PICKUP THE POINTER
4705 027326 105710 1$: TSTB (R0) ;TERMINATOR ?
4706 027330 001403 BEQ 2$ ;BR IF YES
4707 027332 122720 000060 CMPB #'0,(R0)+ ;IS THIS AN ASCII '0' ?
4708 027336 001773 BEQ 1$ ;BR IF YES
4709 027340 005300 2$: DEC R0 ;BACKUP BY '1'
4710 027342 010037 027350 MOV R0,3$ ;SAVE FOR TYPING
4711 027346 104414 DISPLY ;GO PRINT
4712 027350 000000 3$: .WORD 0 ;ASCIZ POINTER GOES HERE
4713 027352 012600 MOV (SP)+,R0 ;RESTORE R0
4714 027354 012616 MOV (SP)+,(SP) ;RESTORE THE STACK
4715 027356 000207 RTS PC ;RETURN
4716
4717 ;ROUTINE TO TYPE AT PRIORITY 4
4718
4719 027360 013746 177776 177776 TYPRI4: MOV @#PS, -(SP) ;SAVE THE PRESENT STATUS
4720 027364 012737 000200 MOV #200,@#PS ;CHANGE THE PRIORITY TO 4
4721 027372 012537 027402 MOV (R5)+,1$ ;MESSAGE ADDRESS
4722 027376 004737 030752 JSR PC,$TYPE ;TYPE THE MESSAGE
4723 027402 000000 1$: .WORD 0 ;MESSAGE ADDRESS GOES HERE
4724 027404 000205 RTS R5 ;RETURN
4725
4726 ;ROUTINE TO TYPE ERRORS
4727 ;CALL
4728 ; DISPLY ;MUST DEFINED IN 'TRAP' TABLE
4729 ; MESADR ;ADDRESS OF MESSAGE
4730 ; RETURN
4731
4732 027406 032777 020000 151524 $DSPLY: BIT #BIT13,@SWR ;INHIBIT ERROR TYPEOUT ?
4733 027414 001004 BNE 1$ ;BR IF YES
4734 027416 005037 177776 CLR @#PS ;SET PRIORITY TO ZERO
4735 027422 000137 030752 JMP $TYPE ;TYPE THE MESSAGE
4736 027426 062716 000002 1$: ADD #2,(SP) ;INCREMENT THE RETURN
4737 027432 000002 RTI ;RETURN
4738
4739 ;THIS ROUTINE IS USED TO CHECK IF AN
4740 ;ASCII CHARACTER IS A DIGIT BETWEEN 0 AND 7.
4741 ;CALL
4742 ; MOV #ADR,R1 ;ADDRESS OF ASCII CHARACTER

```



```

4743      :      JSR      R5,CK.OCT      ;CHECK THE CHARACTER
4744      :      RETURN1      ;CHARACTER IS NOT BETWEEN 0-7
4745      :      RETURN2      ;CHARACTER IS IN R2 AS A
4746      :      :      ;OCTAL DIGIT
4747      :
4748 027434 121127 000060      CK.OCT: CMPB      (R1),#'0      ;LESS THAN ZERO?
4749 027440 103407              BLO          1$          ;YES -- BRANCH
4750 027442 121127 000067      CMPB      (R1),#'7      ;GREATER THAN SEVEN?
4751 027446 101004              BHI          1$          ;YES -- BRANCH
4752 027450 111102              MOVB      (R1),R2      ;GET THE CHARACTER
4753 027452 042702 177770      BIC          #1C7,R2    ;STRIP AWAY THE ASCII
4754 027456 005725              TST          (R5)+     ;ADJUST FOR RETURN
4755 027460 000205              1$:      RTS          R5      ;RETURN
4756
4757      :THIS ROUTINE IS USED TO CHECK AN ASCII CHARACTER
4758      :AND DETERMINE IF IT IS A DIGIT BETWEEN 0 AND 9.
4759      :CALL
4760      :      MOV          #ADR,R1      ;ADDRESS OF ASCII CHARACTER
4761      :      JSR          R5,CK.DEC    ;CHECK THE CHARACTER
4762      :      RETURN1      ;NOT BETWEEN 0 AND 9
4763      :      RETURN2      ;BETWEEN 0 AND 9
4764      :      :      ;R2 = DIGIT
4765      :
4766 027462 121127 000060      CK.DEC: CMPB      (R1),#'0      ;LESS THAN ZERO?
4767 027466 103407              BLO          1$          ;YES -- BRANCH
4768 027470 121127 000071      CMPB      (R1),#'9      ;GREATER THAN NINE?
4769 027474 101004              BHI          1$          ;YES -- BRANCH
4770 027476 111102              MOVB      (R1),R2      ;GET THE CHARACTER
4771 027500 042702 000060      BIC          #'0,R2    ;STRIP AWAY THE ASCII
4772 027504 005725              TST          (R5)+     ;ADJUST FOR RETURN
4773 027506 000205              1$:      RTS          R5      ;RETURN
4774
4775      :THIS ROUTINE WILL CHECK AN ASCII CHARACTER TO
4776      :DETERMINE WHAT IT IS.
4777      :CALL
4778      :      MOV          #ADR,R1      ;ADDRESS OF ASCII CHARACTER
4779      :      JSR          R5,CK.CHR    ;CHECK CHARACTER
4780      :      RETURN      ADR1      ;UNKNOWN CHARACTER
4781      :      RETURN      ADR2      ;CARRIAGE RETURN * (R1)=ADR+1
4782      :      RETURN      ADR3      ;COMMA * (R1)=ADR+1
4783      :      RETURN      ADR4      ;PERIOD * (R1)=ADR+1
4784      :      RETURN      ADR5      ;DIGIT BETWEEN 0 AND 7.
4785      :      RETURN      ADR6      ;DIGIT BETWEEN 8 AND 9.
4786      :      :      ;R2 = DIGIT * (R1)=ADR+1
4787      :
4788 027510 105711              CK.CHR: TSTB      (R1)      ;"CARRIAGE RETURN"?
4789 027512 001417              BEQ          3$          ;YES -- BRANCH
4790 027514 121127 000054      CMPB      (R1),#',      ;"COMMA"?
4791 027520 001413              BEQ          2$          ;YES -- BRANCH
4792 027522 121127 000056      CMPB      (R1),#'.      ;"PERIOD"?
4793 027526 001407              BEQ          1$          ;YES -- BRANCH
4794 027530 004537 027462      JSR          R5,CK.DEC    ;"DIGIT"?
4795 027534 000410              BR          4$          ;NO -- BRANCH
4796 027536 004537 027434      JSR          R5,CK.OCT    ;OCTAL ?

```

M10

```

4797 027542 005725      TST      (R5)+      ;DIGIT BETWEEN 8-9
4798 027544 005725      TST      (R5)+      ;DIGIT BETWEEN 0-7
4799 027546 005725      1$: TST      (R5)+      ;PERIOD
4800 027550 005725      2$: TST      (R5)+      ;COMMA
4801 027552 005725      3$: TST      (R5)+      ;CARRIAGE RETURN
4802 027554 005201      INC      R1          ;MOVE POINTER TO NEXT CHARACTER
4803 027556 011505      4$: MOV      (R5),R5    ;UNKNOWN CHARACTER
4804 027560 000205      RTS       R5        ;RETURN
4805
4806
4807
4808
4809
4810
4811
4812
4813
4814
4815
4816
4817
4818
4819 027562 010446      CK.DIG: MOV     R4,-(SP)    ;SAVE R4
4820 027564 010346      MOV     R3,-(SP)    ;SAVE R3
4821 027566 010246      MOV     R2,-(SP)    ;SAVE THE MAX. SIZE ON THE STACK
4822 027570 005002      CLR     R2          ;START WITH 0
4823 027572 005003      CLR     R3
4824 027574 005004      CLR     R4
4825 027576 004537 027510 JSR     R5,CK.CHR    ;CHECK ONE CHARACTER
4826 027602 027676      6$:      ;ILLEGAL CHARACTER
4827 027604 027704      9$:      ;CARRIAGE RETURN
4828 027606 027676      6$:      ;"
4829 027610 027700      7$:      ;"
4830 027612 027616      1$:      ;DIGIT 0-7
4831 027614 027616      1$:      ;DIGIT 8-9
4832 027616 062705 000004 1$: ADD     #4,R5      ;STEP RETURN POINTER PAST "CR" & "PERIOD" RETURNS
4833 027622 006303      2$: ASL     R3        ;INPUT NUMBER *2
4834 027624 010346      MOV     R3,-(SP)    ;SAVE *2
4835 027626 006303      ASL     R3          ;*4
4836 027630 006303      ASL     R3          ;*8
4837 027632 062603      ADD     (SP)+,R3    ;(*2)+(*8) = *10
4838 027634 060203      ADD     R2,R3      ;UPDATE THE INPUT NUMBER
4839 027636 004537 027510 JSR     R5,CK.CHR    ;CHECK ONE CHARACTER
4840 027642 027702      8$:      ;ILLEGAL CHARACTER
4841 027644 027666      5$:      ;CARRIAGE RETURN
4842 027646 027664      4$:      ;"
4843 027650 027656      3$:      ;"
4844 027652 027622      2$:      ;DIGIT 0-7
4845 027654 027622      2$:      ;DIGIT 8-9
4846 027656 105711      3$: TSTB    (R1)      ;DOES A "CR" FOLLOW THE "PERIOD"
4847 027660 001010      BNE     8$         ;BR IF NOT
4848 027662 005724      TST     (R4)+      ;INCREMENT THE RETURN
4849 027664 005724      4$: TST     (R4)+      ;INCREMENT THE RETURN
4850 027666 005724      5$: TST     (R4)+      ;INCREMENT THE RETURN

```

THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL CHARACTERS AND FORMS A DECIMAL VALUE BINARY NUMBER IN R2.

```

CALL
MOV     #ADR,R1      ;ADDRESS OF ASCIZ STRING
MOV     #NUM,R2      ;MAX. MAGNITUDE OF INPUT NUMBER
JSR     R5,CK.DIG    ;CHECK DIGITS
RETURN  ADR1         ;"CR" ONLY ENTERED -- R2=0
RETURN  ADR2         ;"PERIOD" ONLY ENTERED -- R2=0
RETURN  ADR3         ;ILLEGAL CHARACTER OR INPUT TOO LARGE -- R2=?
RETURN  ADR4         ;"CR" -- R2 = NUMBER
RETURN  ADR5         ;"COMMA" -- R2 = NUMBER
RETURN  ADR6         ;"PERIOD" -- R2 = NUMBER

```



```

4851 027670 020316      CMP      R3,(SP)      ;CHECK THE MAGNITUDE OF THE NUMBER
4852 027672 101004      BHI      9$           ;BR IF ENTERED NUMBER TOO LARGE
4853 027674 000402      BR       9$           ;BYPASS INCREMENT
4854 027676 005725      6$:     TST      (R5)+  ;INCREMENT RETURN PAST INVALID RETURN
4855 027700 005725      7$:     TST      (R5)+  ;INCREMENT RETURN
4856 027702 060405      8$:     ADD      R4,R5  ;SETUP RETURN POINTER
4857 027704 010302      9$:     MOV      R3,R2  ;ENTERED VALUE
4858 027706 005726      TST      (SP)+      ;CLEAN MAX. SIZE OFF OF STACK
4859 027710 012603      MOV      (SP)+,R3   ;RESTORE R3
4860 027712 012604      MOV      (SP)+,R4   ;RESTORE R4
4861 027714 011505      MOV      (R5),R5    ;GET RETURN ADDRESS
4862 027716 000205      RTS      R5         ;RETURN
4863
4864 ;THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
4865 ;UNSIGNED DECIMAL ASCIZ NUMBER.
4866 ;CALL
4867 ;
4868 ;
4869 ;
4870 ;
4871 ;NOTE: THE PROGRAM REQUIRES THIS FORM OF '$SB2D', NOT THE VERSION ON
4872 ;THE SYSMAC LIBRARY, REV C AND LATER
4873
4874 027720 016637 000002 027744 $SB2D: MOV      2(SP),1$      ;SAVE THE BINARY NUMBER
4875 027726 012746 027744      MOV      #1$,-(SP)   ;SET THE POINTER
4876 027732 004737 032500      JSR      PC,$DB2D    ;CALL THE DOUBLE LENGTH CONVERT
4877 027736 012666 000002      MOV      (SP)+,2(SP) ;PICKUP THE POINTER
4878 027742 000207      RTS      PC         ;RETURN
4879 027744 000000 000000      1$:     .WORD    0,0
4880
4881 ;THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
4882 ;UNSIGNED OCTAL ASCIZ NUMBER.
4883 ;CALL
4884 ;
4885 ;
4886 ;
4887 ;
4888 ;NOTE: THE PROGRAM REQUIRES THIS FORM OF '$SB20', NOT THE VERSION ON
4889 ;THE SYSMAC LIBRARY, REV C AND LATER
4890
4891 027750 016637 000002 027774 $SB20: MOV      2(SP),1$      ;SAVE THE BINARY NUMBER
4892 027756 012746 027774      MOV      #1$,-(SP)   ;SET THE POINTER
4893 027762 004737 032674      JSR      PC,$DB2D    ;CALL THE DOUBLE LENGTH CONVERT
4894 027766 012666 000002      MOV      (SP)+,2(SP) ;PICKUP THE POINTER
4895 027772 000207      RTS      PC         ;RETURN
4896 027774 000000 000000      1$:     .WORD    0,0
4897
4898 ;KEYBOARD INTERRUPT INITIALIZATION ROUTINE
4899 ;CALL
4900 ;
4901 ;
4902 ;
4903 030000 012737 030030 000060 $TKINT: MOV      #$TKSRV,TKVEC ;SETUP VECTOR
4904 030006 012737 000240 000062      MOV      #PR5,TKVEC+2 ;PRIORITY TO 5

```



```

4905 030014 005777 151126          TST      3$TKB          ;CLEAR THE BUFFER
4906 030020 012777 000100 151116      MOV      #BIT06,3$TKS ;SET INTERRUPT ENABLE
4907 030026 000207          RTS      PC           ;RETURN

;KEYBOARD INTERRUPT SERVICE ROUTINE
;CALL
;
;      ENTER VIA INTERRUPT

$TKSRV: RDCHR          ;READ THE KEYBOARD
         MOV      (SP)+,5$ ;GET THE CHARACTER
         CMP      5$,#3    ;'CONTROL C' ?
         BNE     1$       ;BR IF NOT
         TYPE    ,SCRLF   ;CR-LF
         TYPE    ,SCNTLC  ;'C'
         MOV      #-1,CFLAG ;SET THE 'CONTROL C' FLAG
         CLR     3$TKS    ;CLEAR THE TTY INTERRUPT
         BR      4$       ;EXIT
         CMP      SWR,#SWREG ;SOFTWARE SWITCH REGISTER IN USE ?
         BNE     3$       ;BR IF NOT
         CMP      5$,#7    ;'CONTROL G' ?
         BNE     3$       ;BR IF NOT
         TYPE    ,SCRLF   ;CR-LF
         TYPE    ,SCNTLG  ;'G'
         MOV      PS,-(SP) ;PUT THE STATUS WORD ON THE STACK
         MOV      #2$,-(SP) ;RETURN ADDRESS
         CLR     3$TKS    ;CLEAR THE TTY INTERRUPT ENABLE
         JMP     $GTSWR   ;GET THE SWITCH REGISTER ENTRY
         MOV      #100,3$TKS ;ENABLE TTY KEYBOARD INTERRUPT
         BR      4$       ;EXIT
         TYPE    ,5$      ;ECHO THE CHARACTER
         RTI          ;RETURN

5$:      .WORD      0          ;ENTERED CHARACTER

;THIS ROUTINE WILL INPUT A STRING FROM THE TTY
;CALL:
;
;      RDLIN
;      RETURN HERE
;
;      ;INPUT A STRING FROM THE TTY
;      ;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
;      ;TERMINATOR WILL BE A BYTE OF ALL 0'S

$RDLIN: MOV      R3,-(SP) ;SAVE R3
         CLR     -(SP)    ;CLEAR THE RUBOUT KEY
1$:      MOV      #$TTYIN,R3 ;GET ADDRESS
2$:      CMP      #$TTYIN+10.,R3 ;BUFFER FULL?
         BLOS    4$       ;BR IF YES
         RDCHR   ;GO READ ONE CHARACTER FROM THE TTY
         MOV      (SP)+,(R3) ;GET CHARACTER
         CMPB   #177,(R3) ;IS IT A RUBOUT
         BNE     5$       ;BR IF NO
         TST    (SP)      ;IS THIS THE FIRST RUBOUT?
         BNE     6$       ;BR IF NO
         MOVB  #' \,9$    ;TYPE A BACK SLASH
         TYPE   ,9$      ;
         MOV    #-1,(SP) ;SET THE RUBOUT KEY
    
```



```

4959 030234 005303          5$: DEC R3 ;BACKUP BY ONE
4960 030236 020327 030440  CMP R3,#STTYIN ;STACK EMPTY?
4961 030242 03445  BLO 4$ ;BR IF YES
4962 030244 111337 030436  MOVB (R3),9$ ;SETUP TO TYPEOUT THE DELETED CHAR.
4963 030250 104401 030436  TYPE 9$ ;GO TYPE
4964 030254 000746  BR 2$ ;GO READ ANOTHER CHAR.
4965 030256 005716          5$: TST (SP) ;RUBOUT KEY SET?
4966 030260 001406  BEQ 7$ ;BR IF NO
4967 030262 112737 000134 030436  MOVB #' \,9$ ;TYPE A BACK SLASH
4968 030270 104401 030436  TYPE 9$
4969 030274 005016  CLR (SP) ;CLEAR THE RUBOUT KEY
4970 030276 122713 000025          7$: CMPB #25,(R3) ;IS CHARACTER A CTRL U?
4971 030302 001003  BNE 10$ ;BR IF NO
4972 030304 104401 032246  TYPE $CNTLU ;TYPE A CONTROL "U"
4973 030310 000726  BR 1$ ;GO START OVER
4974 030312 122713 000003          10$: CMPB #3,(R3) ;IS CHARACTER A CTRL C ?
4975 030316 001006  BNE 8$ ;BR IF NOT
4976 030320 012737 177777 001262  MOV #-1,CFLAG ;SET CNTRL C FLAG
4977 030326 104401 030452  TYPE $CNTLC ;ECHO IT
4978 030332 000427  BR 11$ ;EXIT
4979 030334 122713 000012          8$: CMPB #12,(R3) ;IS CHARACTER A "LF"?
4980 030340 001011  BNE 3$ ;BRANCH IF NO
4981 030342 105013  CLRB (R3) ;CLEAR THE CHARACTER
4982 030344 104401 001165  TYPE $CRLF ;TYPE A "CR" & "LF"
4983 030350 104401 030440  TYPE $TTYIN ;TYPE THE INPUT STRING
4984 030354 000706  BR 2$ ;GO PICKUP ANOTHER CHARACTER
4985 030356 104401 001164          4$: TYPE $QUES ;TYPE A '?'
4986 030362 000701  BR 1$ ;CLEAR THE BUFFER AND LOOP
4987 030364 111337 030436          3$: MOVB (R3),9$ ;ECHO THE CHARACTER
4988 030370 104401 030436  TYPE 9$
4989 030374 122723 000015  CMPB #15,(R3)+ ;CHECK FOR RETURN
4990 030400 001274  BNE 2$ ;LOOP IF NOT RETURN
4991 030402 105063 177777  CLRB -1(R3) ;CLEAR RETURN (THE 15)
4992 030406 104401 001166  TYPE $LF ;TYPE A LINE FEED
4993 030412 005726          11$: TST (SP)+ ;CLEAN RUBOUT KEY FROM THE STACK
4994 030414 012603  MOV (SP)+,R3 ;RESTORE R3
4995 030416 011646  MOV (SP)-,(SP) ;ADJUST THE STACK AND PUT ADDRESS OF THE
4996 030420 016666 000004 000002  MOV 4(SP),2(SP) ; FIRST ASCII CHARACTER ON IT
4997 030426 012766 030440 000004  MOV #STTYIN,4(SP)
4998 030434 000002  RTI ;RETURN
4999 030436 000          9$: .BYTE 0 ;STORAGE FOR ASCII CHAR. TO TYPE
5000 030437 000          .BYTE 0 ;TERMINATOR
5001 030440 000012  STTYIN: .BLKB 10. ;RESERVE 10 BYTES FOR TTY INPUT
5002 030452 041536 005015 000 SCNTLC: .ASCIZ /<CR><LF> ;CONTROL "C"
5003
5004 030460 .EVEN
5005
5006 ;:*****
5007
5008 .SBTTL MACRO ROUTINES
5009
5010 .SBTTL ERROR HANDLER ROUTINE
5011
5012 ;:*****

```



```

5013
5014
5015
5016
5017
5018
5019
5020
5021
5022
5023
5024 030460
5025 030460 104407
5026 030462 010337 001244
5027 030466 010137 001242
5028 030472 032777 020000 150440
5029 030500 001002
5030 030502 004737 023400
5031 030506 105237 001103
5032 030512 001775
5033 030514 013777 001102 150420
5034 030522 032777 002000 150410
5035 030530 001402
5036 030532 104401 001160
5037 030536 005237 001112
5038 030542 011637 001116
5039 030546 162737 000002 001116
5040 030554 117737 150336 001114
5041 030562 032777 020000 150350
5042 030570 001004
5043 030572 004737 030616
5044 030576 104401 001165
5045 030602
5046 030602 005777 150332
5047 030606 100002
5048 030610 000000
5049 030612 104407
5050 030614 000002
5051
5052
5053
5054
5055
5056
5057
5058
5059 030616
5060 030616 104401 001165
5061 030622 010046
5062 030624 005000
5063 030626 153700 001114
5064 030632 001004
5065
5066 030634 013746 001116

```

```

;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT.
;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
;*AND GO TO $ERRTYP ON ERROR
;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
;*SW15=1 HALT ON ERROR
;*SW13=1 INHIBIT ERROR TYPEOUTS
;*SW10=1 BELL ON ERROR
;*CALL
;* ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER

```

```

$ERROR:
CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
MOV R3,ATTN ;;SAVE THE ATTENTION REGISTER CONTENTS
MOV R1,DRIVE ;;DRIVE NUMBER
BIT #SW13,$SWR ;;INHIBIT PRINTOUTS ?
BNE .+6 ;;BR IF YES
JSR PC,$TIME ;;TYPE THE TIME
7$: INCB $ERFLG ;;SET THE ERROR FLAG
BEQ 7$ ;;DON'T LET THE FLAG GO TO ZERO
MOV $TSTNM,$DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
BIT #BIT10,$SWR ;;BELL ON ERROR?
BEQ 1$ ;;NO - SKIP
TYPE $BELL ;;RING BELL
1$: INC $ERTTL ;;COUNT THE NUMBER OF ERRORS
MOV (SP),$ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
SUB #2,$ERRPC
MOVB @ $ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
BIT #BIT13,$SWR ;;SKIP TYPEOUT IF SET
BNE 20$ ;;SKIP TYPEOUTS
JSR PC,$ERRTYP ;;GO TO USER ERROR ROUTINE
TYPE , $CRLF
20$:
2$: TST $SWR ;;HALT ON ERROR
BPL 3$ ;;SKIP IF CONTINUE
HALT ;;HALT ON ERROR!
CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
3$: RTI ;;RETURN

```

.SBTTL ERROR MESSAGE TYPEOUT ROUTINE

```

;*****
;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.

```

```

$ERRTYP:
TYPE , $CRLF ;; "CARRIAGE RETURN" & "LINE FEED"
MOV RO,-(SP) ;;SAVE RO
CLR RO ;;PICKUP THE ITEM INDEX
BISB @ $ITEMB,RO
BNE 1$ ;;IF ITEM NUMBER IS ZERO, JUST
MOV $ERRPC,-(SP) ;;TYPE THE PC OF THE ERROR
;;SAVE $ERRPC FOR TYPEOUT

```



```

5067                                     ;; ERROR ADDRESS
5068 030640 104402      TYP0C           ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
5069 030642 000426      BR              6$           ;; GET OUT
5070 030644 005300      1$: DEC         RO           ;; ADJUST THE INDEX SO THAT IT WILL
5071 030646 006300      ASL         RO           ;; WORK FOR THE ERROR TABLE
5072 030650 006300      ASL         RO
5073 030652 006300      ASL         RO
5074 030654 062700 004006  ADD      #ERRTB,RO      ;; FORM TABLE POINTER
5075 030660 012037 030670  MOV      (RO)+,2$      ;; PICKUP "ERROR MESSAGE" POINTER
5076 030664 001404      BEQ         3$           ;; SKIP TYPEOUT IF NO POINTER
5077 030666 104401      TYPE           ;; TYPE THE "ERROR MESSAGE"
5078 030670 000000      2$: .WORD      0           ;; "ERROR MESSAGE" POINTER GOES HERE
5079 030672 104401 001165  TYPE      $CRLF      ;; "CARRIAGE RETURN" & "LINE FEED"
5080 030676 012037 030706  3$: MOV      (RO)+,4$      ;; PICKUP "DATA HEADER" POINTER
5081 030702 001404      BEQ         5$           ;; SKIP TYPEOUT IF 0
5082 030704 104401      TYPE           ;; TYPE THE "DATA HEADER"
5083 030706 000000      4$: .WORD      0           ;; "DATA HEADER" POINTER GOES HERE
5084 030710 104401 001165  TYPE      $CRLF      ;; "CARRIAGE RETURN" & "LINE FEED"
5085 030714 011000      5$: MOV      (RO),RO      ;; PICKUP "DATA TABLE" POINTER
5086 030716 001004      BNE         7$           ;; GO TYPE THE DATA
5087 030720 012600      6$: MOV      (SP)+,RO      ;; RESTORE RO
5088 030722 104401 001165  TYPE      $CRLF      ;; "CARRIAGE RETURN" & "LINE FEED"
5089 030726 000207      RTS         PC           ;; RETURN
5090 030730
5091 030730 013046      MOV      2(RO)+,-(SP)      ;; SAVE 2(RO)+ FOR TYPEOUT
5092 030732 104402      TYP0C           ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
5093 030734 005710      TST         (RO)         ;; IS THERE ANOTHER NUMBER?
5094 030736 001770      BEQ         6$           ;; BR IF NO
5095 030740 104401 030746  TYPE      8$           ;; TYPE TWO(2) SPACES
5096 030744 000771      BR         7$           ;; LOOP
5097 030746 020040 000      8$: .ASCIZ  / /           ;; TWO(2) SPACES
5098 030752
5099
100 .SBTTL TYPE ROUTINE
101
102 ;;*****
103 ;;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
104 ;;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
105 ;;*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
106 ;;*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
107 ;;*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
108 ;;*
109 ;;*CALL:
110 ;;*1) USING A TRAP INSTRUCTION
111 ;;* TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
112 ;;*OR
113 ;;* TYPE
114 ;;* MESADR
115 ;;*
116
117 030752 105737 001157 $TYPE: TSTB  STPFLG      ;; IS THERE A TERMINAL?
118 030756 100002      BPL         1$           ;; BR IF YES
119 030760 000000      HALT        ;; HALT HERE IF NO TERMINAL
120 030762 000407      BR         3$           ;; LEAVE

```

```

S121 030764 010046      1$:  MOV    RO,-(SP)      ;;SAVE RO
S122 030766 017600 000002  MOV    22(SP),RO      ;;GET ADDRESS OF ASCIZ STRING
S123 030772 112046      2$:  MOVB   (RO)+,-(SP)  ;;PUSH CHARACTER TO BE TYPED ONTO STACK
S124 030774 001005      BNE    4$              ;;BR IF IT ISN'T THE TERMINATOR
S125 030776 005726      TST    (SP)+          ;;IF TERMINATOR POP IT OFF THE STACK
S126 031000 012600      60$:  MOV    (SP)+,RO      ;;RESTORE RO
S127 031002 062716 000002  3$:  ADD    #2,(SP)       ;;ADJUST RETURN PC
S128 031006 000002      RTI                    ;;RETURN
S129 031010 122716 000011  4$:  CMPB   #HT,(SP)      ;;BRANCH IF <HT>
S130 031014 001430      BEQ    8$              ;;BRANCH IF NOT <CRLF>
S131 031016 122716 000200  CMPB   #CRLF,(SP)    ;;BRANCH IF NOT <CRLF>
S132 031022 001006      BNE    5$              ;;PCP <CR><LF> EQUIV
S133 031024 005726      TST    (SP)+          ;;TYPE A CR AND LF
S134 031026 104401      TYPE                    ;;TYPE A CR AND LF
S135 031030 001165      $CRLF
S136 031032 105037 031166  CLRB   $CHARCNT      ;;CLEAR CHARACTER COUNT
S137 031036 000755      BR     2$              ;;GET NEXT CHARACTER
S138 031040 004737 031122  5$:  JSR    PC,$TYPEPC    ;;GO TYPE THIS CHARACTER
S139 031044 123726 001156  6$:  CMPB   $FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?
S140 031050 001350      BNE    2$              ;;IF NO GO GET NEXT CHAR.
S141 031052 013746 001154  MOV    $NULL,-(SP)    ;;GET # OF FILLER CHARS. NEEDED
S142                                ;;AND THE NULL CHAR.
S143 031056 105366 000001  7$:  DECB   1(SP)         ;;DOES A NULL NEED TO BE TYPED?
S144 031062 002770      BLT    6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
S145 031064 004737 031122  JSR    PC,$TYPEPC    ;;GO TYPE A NULL
S146 031070 105337 031166  DECB   $CHARCNT      ;;DO NOT COUNT AS A COUNT
S147 031074 000770      BR     7$              ;;LOOP

```

;HORIZONTAL TAB PROCESSOR

```

S151 031076 112716 000040  8$:  MOVB   #' ,(SP)     ;;REPLACE TAB WITH SPACE
S152 031102 004737 031122  9$:  JSR    PC,$TYPEPC    ;;TYPE A SPACE
S153 031106 132737 000007 031166 BITB   #7,$CHARCNT    ;;BRANCH IF NOT AT
S154 031114 001372      BNE    9$              ;;TAB STOP
S155 031116 005726      TST    (SP)+          ;;POP SPACE OFF STACK
S156 031120 000724      BR     2$              ;;GET NEXT CHARACTER
S157 031122 105777 150022  $TYPEPC: TSTB   2$TPS    ;;WAIT UNTIL PRINTER IS READY
S158 031126 100375      BPL    $TYPEPC
S159 031130 116677 000002 150014 MOVB   2(SP),2$TPB    ;;LOAD CHAR TO BE TYPED INTO DATA REG.
S160 031136 122766 000015 000002 CMPB   #CR,2(SP)      ;;IS CHARACTER A CARRIAGE RETURN?
S161 031144 001003      BNE    1$              ;;BRANCH IF NO
S162 031146 105037 031166  CLRB   $CHARCNT      ;;YES--CLEAR CHARACTER COUNT
S163 031152 000406      BR     $TYPEPC        ;;EXIT
S164 031154 122766 000012 000002 1$:  CMPB   #LF,2(SP)     ;;IS CHARACTER A LINE FEED?
S165 031162 001402      BEQ    $TYPEPC        ;;BRANCH IF YES
S166 031164 105227      INCB   (PC)+          ;;COUNT THE CHARACTER
S167 031166 000000  $CHARCNT: .WORD 0    ;;CHARACTER COUNT STORAGE
S168 031170 000207  $TYPEPC: RTS    PC

```

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

;*****
;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT

```

S169
S170
S171
S172
S173
S174

G11

```

5175      *OCTAL (ASCII) NUMBER AND TYPE IT.
5176      *$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
5177      *CALL:
5178      *      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
5179      *      TYPOS      ;;CALL FOR TYPEOUT
5180      *      .BYTE   N      ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
5181      *      .BYTE   M      ;;M=1 OR 0
5182      *                               ;;1=TYPE LEADING ZEROS
5183      *                               ;;0=SUPPRESS LEADING ZEROS
5184
5185      *$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
5186      *$TYPOS OR $TYPOC
5187      *CALL:
5188      *      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
5189      *      TYPON      ;;CALL FOR TYPEOUT
5190
5191      *$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
5192      *CALL:
5193      *      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
5194      *      TYPOC      ;;CALL FOR TYPEOUT
5195
5196      031172  017646  000000
5197      031176  116637  000001  031415  $TYPOS: MOV      2(SP),-(SP)      ;;PICKUP THE MODE
5198      031204  112637  031417      MOVVB   1(SP),$OFILL      ;;LOAD ZERO FILL SWITCH
5199      031210  062716  000002      MOVVB   (SP)+,$OMODE+1    ;;NUMBER OF DIGITS TO TYPE
5200      031214  000406      ADD     #2,(SP)          ;;ADJUST RETURN ADDRESS
5201      031216  112737  000001  031415  $TYPOC: BR      $TYPON
5202      031224  112737  000006  031417      MOVVB   #1,$OFILL      ;;SET THE ZERO FILL SWITCH
5203      031232  112737  000005  031414  $TYPON: MOVVB   #6,$OMODE+1  ;;SET FOR SIX(6) DIGITS
5204      031240  010346      MOV     R3,-(SP)        ;;SET THE ITERATION COUNT
5205      031242  010446      MOV     R4,-(SP)        ;;SAVE R3
5206      031244  010546      MOV     R5,-(SP)        ;;SAVE R4
5207      031246  113704  031417      MOVVB   $OMODE+1,R4    ;;SAVE R5
5208      031252  005404      NEG     R4              ;;GET THE NUMBER OF DIGITS TO TYPE
5209      031254  062704  000006      ADD     #6,R4          ;;SUBTRACT IT FOR MAX. ALLOWED
5210      031260  110437  031416      MOVVB   R4,$OMODE      ;;SAVE IT FOR USE
5211      031264  113704  031415      MOVVB   $OFILL,R4     ;;GET THE ZERO FILL SWITCH
5212      031270  016605  000012      MOV     12(SP),R5     ;;PICKUP THE INPUT NUMBER
5213      031274  005003      CLR     R3              ;;CLEAR THE OUTPUT WORD
5214      031276  006105      1$:    ROL     R5          ;;ROTATE MSB INTO "C"
5215      031300  000404      BR      3$             ;;GO DO MSB
5216      031302  006105      2$:    ROL     R5          ;;FORM THIS DIGIT
5217      031304  006105      ROL     R5
5218      031306  006105      ROL     R5
5219      031310  010503      MOV     R5,R3
5220      031312  006103      3$:    ROL     R3          ;;GET LSB OF THIS DIGIT
5221      031314  105337  031416      DECB   $OMODE          ;;TYPE THIS DIGIT?
5222      031320  100016      BPL     7$             ;;BR IF NO
5223      031322  042703  177770      BIC     #177770,R3    ;;GET RID OF JUNK
5224      031326  001002      BNE     4$            ;;TEST FOR 0
5225      031330  005704      TST     R4             ;;SUPPRESS THIS 0?
5226      031332  001403      BEQ     5$            ;;BR IF YES
5227      031334  005204      4$:    INC     R4          ;;DON'T SUPPRESS ANYMORE 0'S
5228      031336  052703  000060      BIS     #'0,R3        ;;MAKE THIS DIGIT ASCII
  
```

DZRJD.019

BINARY TO OCTAL (ASCII) AND TYPE

```

5229 031342 052703 000040      5$:  BIS      #' R3      ;; MAKE ASCII IF NOT ALREADY
5230 031346 110337 031412      MOVB     R3,8$      ;; SAVE FOR TYPING
5231 031352 104401 031412      TYPE     8$        ;; GO TYPE THIS DIGIT
5232 031356 105337 031414      7$:  DECB     $OCNT   ;; COUNT BY 1
5233 031362 003347          BGT      2$        ;; BR IF MORE TO DO
5234 031364 002402          BLT      6$        ;; BR IF DONE
5235 031366 005204          INC      R4        ;; INSURE LAST DIGIT ISN'T A BLANK
5236 031370 000744          BR       2$        ;; GO DO THE LAST DIGIT
5237 031372 012605      6$:  MOV      (SP)+,R5  ;; RESTORE R5
5238 031374 012604      MOV      (SP)+,R4  ;; RESTORE R4
5239 031376 012603      MOV      (SP)+,R3  ;; RESTORE R3
5240 031400 016666 000002 000004  MOV      2(SP),4(SP) ;; SET THE STACK FOR RETURNING
5241 031406 012616      MOV      (SP)+,(SP)
5242 031410 000002          RTI              ;; RETURN
5243 031412          000          8$:  .BYTE     0        ;; STORAGE FOR ASCII DIGIT
5244 031413          000          .BYTE     0        ;; TERMINATOR FOR TYPE ROUTINE
5245 031414          000      $OCNT:  .BYTE     0        ;; OCTAL DIGIT COUNTER
5246 031415          000      $OFILL: .BYTE     0        ;; ZERO FILL SWITCH
5247 031416 000000      $OMODE: .WORD     0        ;; NUMBER OF DIGITS TO TYPE

```

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```

*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
*REPLACED WITH SPACES.

```

```

*CALL:
*      MOV      NUM,-(SP)      ;; PUT THE BINARY NUMBER ON THE STACK
*      TYPDS                    ;; GO TO THE ROUTINE

```

```

$TYPDS:
MOV      R0,-(SP)      ;; PUSH R0 ON STACK
MOV      R1,-(SP)      ;; PUSH R1 ON STACK
MOV      R2,-(SP)      ;; PUSH R2 ON STACK
MOV      R3,-(SP)      ;; PUSH R3 ON STACK
MOV      R5,-(SP)      ;; PUSH R5 ON STACK
MOV      #20200,-(SP)   ;; SET BLANK SWITCH AND SIGN
MOV      20(SP),R5     ;; GET THE INPUT NUMBER
BPL      1$           ;; BR IF INPUT IS POS.
NEG      R5           ;; MAKE THE BINARY NUMBER POS.
MOVB     #'-,1(SP)    ;; MAKE THE ASCII NUMBER NEG.
1$:  CLR      R0           ;; ZERO THE CONSTANTS INDEX
MOV      #0BLK,R3     ;; SETUP THE OUTPUT POINTER
MOVB     #' ,(R3)+    ;; SET THE FIRST CHARACTER TO A BLANK
2$:  CLR      R2           ;; CLEAR THE BCD NUMBER
MOV      $DTBL(R0),R1 ;; GET THE CONSTANT
3$:  SUB      R1,R5       ;; FORM THIS BCD DIGIT
BLT      4$           ;; BR IF DONE
INC      R2           ;; INCREASE THE BCD DIGIT BY 1
4$:  ADD      R1,R5       ;; ADD BACK THE CONSTANT
TST      R2           ;; CHECK IF BCD DIGIT=0

```

```

5261 031420          000000
5262 031420 010046
5263 031422 010146
5264 031424 010246
5265 031426 010346
5266 031430 010546
5267 031432 012746 020200
5268 031436 016605 000020
5269 031442 100004
5270 031444 005405
5271 031446 112766 000055 000001
5272 031454 005000
5273 031456 012703 031634
5274 031462 112723 000040
5275 031466 005002
5276 031470 016001 031624
5277 031474 160105
5278 031476 002402
5279 031500 005202
5280 031502 000774
5281 031504 060105
5282 031506 005702

```



```

5283 031510 001002      BNE      5$          ;; FALL THROUGH IF 0
5284 031512 105716      TSTB     (SP)        ;; STILL DOING LEADING 0'S?
5285 031514 100407      BMI      7$          ;; BR IF YES
5286 031516 106316      5$: ASLB     (SP)        ;; MSD?
5287 031520 103003      BCC      6$          ;; BR IF NO
5288 031522 116663 000001 177777  MOVB     1(SP),-1(R3) ;; YES--SET THE SIGN
5289 031530 052702 000060 6$: BIS     #'0,R2     ;; MAKE THE BCD DIGIT ASCII
5290 031534 052702 000040 7$: BIS     #' ,R2     ;; MAKE IT A SPACE IF NOT ALREADY A DIGIT
5291 031540 110223      MOVB     R2,(R3)+   ;; PUT THIS CHARACTER IN THE OUTPUT BUFFER
5292 031542 005720      TST      (R0)+      ;; JUST INCREMENTING
5293 031544 020027 000010  CMP      R0,#10     ;; CHECK THE TABLE INDEX
5294 031550 002746      BLT      2$          ;; GO DO THE NEXT DIGIT
5295 031552 003002      BGT      8$          ;; GO TO EXIT
5296 031554 010502      MOV      R5,R2      ;; GET THE LSD
5297 031556 000764      BR       6$          ;; GO CHANGE TO ASCII
5298 031560 105726      8$: TSTB     (SP)+    ;; WAS THE LSD THE FIRST NON-ZERO?
5299 031562 100003      BPL      9$          ;; BR IF NO
5300 031564 116663 177777 177776  MOVB     -1(SP),-2(R3) ;; YES--SET THE SIGN FOR TYPING
5301 031572 105013      9$: CLRB     (R3)      ;; SET THE TERMINATOR
5302 031574 012605      MOV      (SP)+,R5   ;; POP STACK INTO R5
5303 031576 012603      MOV      (SP)+,R3   ;; POP STACK INTO R3
5304 031600 012602      MOV      (SP)+,R2   ;; POP STACK INTO R2
5305 031602 012601      MOV      (SP)+,R1   ;; POP STACK INTO R1
5306 031604 012600      MOV      (SP)+,R0   ;; POP STACK INTO R0
5307 031606 104401 031634  TYPE     $DBLK       ;; NOW TYPE THE NUMBER
5308 031612 016666 000002 000004  MOV      2(SP),4(SP) ;; ADJUST THE STACK
5309 031620 012616      MOV      (SP)+,(SP)
5310 031622 000002      RTI                          ;; RETURN TO USER
5311 031624 023420      $DTBL: 10000.
5312 031626 001750      1000.
5313 031630 000144      100.
5314 031632 000012      10.
5315 031634 000004      $DBLK: .BLKW 4
5316
5317      .SBTTL  TTY INPUT ROUTINE
5318
5319      ;;*****
5320      .ENABL  LSB
5321
5322      ;;*****
5323      *SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
5324      *ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
5325      *SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
5326      *WHEN OPERATING IN TTY FLAG MODE.
5327 031644 022737 000176 001140 $CKSWR: CMP      #SWREG,SWR  ;; IS THE SOFT-SWR SELECTED?
5328 031652 001074      BNE      15$        ;; BRANCH IF NO
5329 031654 105777 147264      TSTB     3$TKS     ;; CHAR THERE?
5330 031660 100071      BPL      15$        ;; IF NO, DON'T WAIT AROUND
5331 031662 117746 147260      MOVB     3$TKB,-(SP) ;; SAVE THE CHAR
5332 031666 042716 177600      BIC      #177,(SP)  ;; STRIP-OFF THE ASCII
5333 031672 022726 000007      CMP      #7,(SP)+   ;; IS IT A CONTROL G?
5334 031676 001062      BNE      15$        ;; NO, RETURN TO USER
5335 031700 123727 001134 000001  CMPB     $AUTOB,#1  ;; ARE WE RUNNING IN AUTO-MODE?
5336 031706 001456      BEQ      15$        ;; BRANCH IF YES

```

```

5337
5338 031710 104401 032253
5339 031714 104401 032260 $GTSWR: TYPE , $CNTLG ;; ECHO THE CONTROL-G (↑G)
5340 031720 013746 000176 MOV $MSWR ;; TYPE CURRENT CONTENTS
5341 031724 104402 TYPOC $WREG, -(SP) ;; SAVE SWREG FOR TYPEOUT
5342 031726 104401 032271 TYPE , $MNEW ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
5343 031732 005046 19$: CLR -(SP) ;; PROMPT FOR NEW SWR
5344 031734 005046 CLR -(SP) ;; CLEAR COUNTER
5345 031736 105777 147202 7$: TSTB $TKS ;; THE NEW SWR
5346 031742 100375 BPL 7$ ;; CHAR THERE?
5347 ;; IF NOT TRY AGAIN
5348 031744 117746 147176 MOVB $TKB, -(SP) ;; PICK UP CHAR
5349 031750 042716 177600 BIC #1C177, (SP) ;; MAKE IT 7-BIT ASCII
5350
5351
5352
5353 031754 021627 000025 9$: CMP (SP), #25 ;; IS IT A CONTROL-U?
5354 031760 001005 BNE 10$ ;; BRANCH IF NOT
5355 031762 104401 032246 TYPE , $CNTLU ;; YES, ECHO CONTROL-U (↑U)
5356 031766 062706 000006 20$: ADD #6, SP ;; IGNORE PREVIOUS INPUT
5357 031772 000757 BR 19$ ;; LET'S TRY IT AGAIN
5358
5359
5360 031774 021627 000015 10$: CMP (SP), #15 ;; IS IT A <CR>?
5361 032000 001022 BNE 16$ ;; BRANCH IF NO
5362 032002 005766 000004 TST 4(SP) ;; YES, IS IT THE FIRST CHAR?
5363 032006 001403 BEQ 11$ ;; BRANCH IF YES
5364 032010 016677 000002 147122 MOV 2(SP), $SWR ;; SAVE NEW SWR
5365 032016 062706 000006 11$: ADD #6, SP ;; CLEAR UP STACK
5366 032022 104401 001165 14$: TYPE $CRLF ;; ECHO <CR> AND <LF>
5367 032026 123727 001135 000001 CMPB $INTAG, #1 ;; RE-ENABLE TTY KBD INTERRUPTS?
5368 032034 001003 BNE 15$ ;; BRANCH IF NOT
5369 032036 012777 000100 147100 MOV #100, $TKS ;; RE-ENABLE TTY KBD INTERRUPTS
5370 032044 000002 15$: RTI ;; RETURN
5371 032046 004737 031122 16$: JSR PC, $TYPEC ;; ECHO CHAR
5372 032052 021627 000060 CMP (SP), #60 ;; CHAR < 0?
5373 032056 002420 BLT 18$ ;; BRANCH IF YES
5374 032060 021627 000067 CMP (SP), #67 ;; CHAR > 7?
5375 032064 003015 BGT 18$ ;; BRANCH IF YES
5376 032066 042726 000060 BIC #60, (SP)+ ;; STRIP-OFF ASCII
5377 032072 005766 000002 TST 2(SP) ;; IS THIS THE FIRST CHAR
5378 032076 001403 BEQ 17$ ;; BRANCH IF YES
5379 032100 006316 ASL (SP) ;; NO, SHIFT PRESENT
5380 032102 006316 ASL (SP) ;; CHAR OVER TO MAKE
5381 032104 006316 ASL (SP) ;; ROOM FOR NEW ONE.
5382 032106 005266 000002 17$: INC 2(SP) ;; KEEP COUNT OF CHAR
5383 032112 056616 177776 BIS -2(SP), (SP) ;; SET IN NEW CHAR
5384 032116 000707 BR 7$ ;; GET THE NEXT ONE
5385 032120 104401 001164 18$: TYPE $QUES ;; TYPE ?<CR><LF>
5386 032124 000720 BR 20$ ;; SIMULATE CONTROL-U
5387 .DSABL LSB
5388
5389
5390 ;*****

```



```

5391 ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
5392 ;*CALL:
5393 ;* RDCHR ;: INPUT A SINGLE CHARACTER FROM THE TTY
5394 ;* RETURN HERE ;: CHARACTER IS ON THE STACK
5395 ;* ;: WITH PARITY BIT STRIPPED OFF
5396 ;*
5397 ;*
5398 032126 011646 $RDCHR: MOV (SP),-(SP) ;: PUSH DOWN THE PC
5399 032130 016666 000004 000002 MOV 4(SP),2(SP) ;: SAVE THE PS
5400 032136 105777 147002 1$: TSTB @STKS ;: WAIT FOR
5401 032142 100375 BPL 1$ ;: A CHARACTER
5402 032144 117766 146776 000004 MOVB @STKB,4(SP) ;: READ THE TTY
5403 032152 042766 177600 000004 BIC #1C<177>,4(SP) ;: GET RID OF JUNK IF ANY
5404 032160 026627 000004 000023 CMP 4(SP),#23 ;: IS IT A CONTROL-S?
5405 032166 001013 BNE 3$ ;: BRANCH IF NO
5406 032170 105777 146750 2$: TSTB @STKS ;: WAIT FOR A CHARACTER
5407 032174 100375 BPL 2$ ;: LOOP UNTIL ITS THERE
5408 032176 117746 146744 MOVB @STKB,-(SP) ;: GET CHARACTER
5409 032202 042716 177600 BIC #1C177,(SP) ;: MAKE IT 7-BIT ASCII
5410 032206 022627 000021 CMP (SP)+,#21 ;: IS IT A CONTROL-Q?
5411 032212 001366 BNE 2$ ;: IF NOT DISCARD IT
5412 032214 000750 BR 1$ ;: YES, RESUME
5413 032216 026627 000004 000140 3$: CMP 4(SP),#140 ;: IS IT UPPER CASE?
5414 032224 002407 BLT 4$ ;: BRANCH IF YES
5415 032226 026627 000004 000175 CMP 4(SP),#175 ;: IS IT A SPECIAL CHAR?
5416 032234 003003 BGT 4$ ;: BRANCH IF YES
5417 032236 042766 000040 000004 BIC #40,4(SP) ;: MAKE IT UPPER CASE
5418 032244 000002 RTI ;: GO BACK TO USER
5419 032246 052536 005015 000 $CNTLU: .ASCIZ /1U<15><12> ;: CONTROL "U"
5420 032253 136 006507 000012 $CNTLG: .ASCIZ /1G<15><12> ;: CONTROL "G"
5421 032260 005015 053523 020122 $MSWR: .ASCIZ <15><12>/SWR = /
5422 032266 020075 000
5423 032271 040 047040 053505 $MNEW: .ASCIZ / NEW = /
5424 032276 036440 000040

```

.SBTTL RANDOM NUMBER GENERATOR ROUTINE

```

5425 ;:*****
5426 ;:*****
5427 ;:*****
5428 ;:*****
5429 ;:*****
5430 ;:*****
5431 ;:*****
5432 ;:*****
5433 ;:*****
5434 ;:*****
5435 ;:*****
5436 ;:*****
5437 ;:*****
5438 ;:*****
5439 ;:*****
5440 ;:*****
5441 ;:*****
5442 ;:*****
5443 ;:*****
5444 ;:*****

```

```

5437 032302 $RAND: MOV R0,-(SP) ;: PUSH R0 ON STACK
5438 032302 010046 MOV R1,-(SP) ;: PUSH R1 ON STACK
5439 032304 010146 MOV R2,-(SP) ;: PUSH R2 ON STACK
5440 032306 010246 MOV $LONUM,R0 ;: SET R0 WITH LOW
5441 032310 013700 032402 MOV $SHINUM,R1 ;: SET R1 WITH HIGH
5442 032314 013701 032400 MOV #-7,R2 ;: SET SHIFT COUNT
5443 032320 012702 177771 1$: ASL R0 ;: SHIFT R0 LEFT AND
5444 032324 006300

```

```

5445 032326 006101          ROL    R1          ;; ROTATE CARRY INTO R1 AND
5446 032330 005202          INC    R2          ;; CHECK FOR DONE
5447 032332 001374          BNE   1$          ;; CONTINUE SHIFT LOOP
5448 032334 063700 032402    ADD   $LONUM,R0   ;; ADD NUMBER TO MAKE X 129
5449 032340 005501          ADC   R1          ;; PROPOGATE CARRY
5450 032342 063701 032400    ADD   $HINUM,R1   ;; ADD NUMBER TO MAKE X 129
5451 032346 062700 001057    ADD   #1057,R0    ;; ADD LOW CONSTANT
5452 032352 005501          ADC   R1          ;; PROPOGATE CARRY
5453 032354 062701 047401    ADD   #47401,R1   ;; ADD HIGH CONSTANT
5454 032360 010037 032402    MOV   R0,$LONUM   ;; SAVE R0
5455 032364 010137 032400    MOV   R1,$HINUM   ;; SAVE R1
5456 032370 012602          MOV   (SP)+,R2    ;; POP STACK INTO R2
5457 032372 012601          MOV   (SP)+,R1    ;; POP STACK INTO R1
5458 032374 012600          MOV   (SP)+,R0    ;; POP STACK INTO R0
5459 032376 000207          RTS   PC          ;; RETURN
5460 032400 176543          $HINUM: .WORD    176543
5461 032402 123456          $LONUM: .WORD    123456

```

.SBTTL SAVE AND RESTORE R0-R5 ROUTINES

```

*****
; *SAVE R0-R5
; *CALL:
; * SAVREG
; *UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
; *
; *TOP---(+16)
; * +2---(+18)
; * +4---R5
; * +6---R4
; * +8---R3
; * +10---R2
; * +12---R1
; * +14---R0

```

```

5480 032404          $SAVREG:
5481 032404 010046          MOV   R0,-(SP)    ;; PUSH R0 ON STACK
5482 032406 010146          MOV   R1,-(SP)    ;; PUSH R1 ON STACK
5483 032410 010246          MOV   R2,-(SP)    ;; PUSH R2 ON STACK
5484 032412 010346          MOV   R3,-(SP)    ;; PUSH R3 ON STACK
5485 032414 010446          MOV   R4,-(SP)    ;; PUSH R4 ON STACK
5486 032416 010546          MOV   R5,-(SP)    ;; PUSH R5 ON STACK
5487 032420 016646 000022    MOV   22(SP),-(SP) ;; SAVE PS OF MAIN FLOW
5488 032424 016646 000022    MOV   22(SP),-(SP) ;; SAVE PC OF MAIN FLOW
5489 032430 016646 000022    MOV   22(SP),-(SP) ;; SAVE PS OF CALL
5490 032434 016646 000022    MOV   22(SP),-(SP) ;; SAVE PC OF CALL
5491 032440 000002          RTI

```

```

; *RESTORE R0-R5
; *CALL:
; * RESREG
$RESREG:
5496 032442          MOV   (SP)+,22(SP) ;; RESTORE PC OF CALL
5497 032442 012666 000022    MOV   (SP)+,22(SP) ;; RESTORE PS OF CALL
5498 032446 012666 000022

```



```

5499 032452 012666 000022      MOV      (SP)+,22(SP)      ;;RESTORE PC OF MAIN FLOW
5500 032456 012666 000022      MOV      (SP)+,22(SP)      ;;RESTORE PS OF MAIN FLOW
5501 032462 012605                MOV      (SP)+,R5          ;;POP STACK INTO R5
5502 032464 012604                MOV      (SP)+,R4          ;;POP STACK INTO R4
5503 032466 012603                MOV      (SP)+,R3          ;;POP STACK INTO R3
5504 032470 012602                MOV      (SP)+,R2          ;;POP STACK INTO R2
5505 032472 012601                MOV      (SP)+,R1          ;;POP STACK INTO R1
5506 032474 012600                MOV      (SP)+,R0          ;;POP STACK INTO R0
5507 032476 000002                RTI
    
```

.SBTTL DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

```

5508
5509
5510
5511
5512
5513
5514
5515
5516
5517
5518
5519
5520
5521
5522 032500 104412                $DB2D: SAVREG                ;;SAVE REGISTERS
5523 032502 016602 000002      MOV      2(SP),R2          ;;PICKUP THE DATA POINTER
5524 032506 012700 032660      MOV      #$DECVL,R0        ;;GET ADDRESS OF "$DECVL" STRING
5525 032512 010066 000002      MOV      R0,2(SP)          ;;PUT ADDRESS OF ASCII STRING ON STACK
5526 032516 012201                MOV      (R2)+,R1          ;;PICKUP THE BINARY NUMBER
5527 032520 012202                MOV      (R2)+,R2
5528 032522 012737 000012 032576      MOV      #10.,4$          ;;SET UP TO DO 10 CONVERSIONS
5529 032530 012704 032610      MOV      $TNPWR,R4         ;;ADDRESS OF TEN POWER
5530 032534 012705 032612      MOV      $TNPWR+2,R5
5531 032540 005003                1$: CLR      R3              ;;CLEAR PARTIAL
5532 032542 161401                2$: SUB      (R4),R1        ;;SUBTRACT TEN POWER
5533 032544 005602                SBC      R2
5534 032546 161502                SUB      (R5),R2
5535 032550 002402                BLT      3$                ;;BR IF TEN POWER TOO LARGE
5536 032552 005203                INC      R3                ;;ADD 1 TO PARTIAL
5537 032554 000772                BR       2$                ;;LOOP
5538 032556 062401                3$: ADD      (R4)+,R1        ;;RESTORE SUBTRACTED VALUE
5539 032560 005502                ADC      R2
5540 032562 062402                ADD      (R4)+,R2
5541 032564 022525                CMP      (R5)+,(R5)+      ;;MOVE TO NEXT TEN POWER
5542 032566 052703 000060      BIS      #'0,R3            ;;CHANGE PARTIAL TO ASCII
5543 032572 110320                MOV      R3,(R0)+         ;;SAVE IT
5544 032574 005327                DEC      (PC)+            ;;DONE?
5545 032576 000000                4$: .WORD 0
5546 032600 001357                BNE     1$                ;;BR IF NO
5547 032602 105020                CLRB    (R0)+            ;;TERMINATOR
5548 032604 104413                RESREG                ;;RESTORE REGISTERS
5549 032606 000207                RTS      PC              ;;RETURN
5550 032610 145000                $TNPWR: 145000            ;;1.0E09
5551 032612 035632                35632
5552 032614 160400                160400                    ;;1.0E08
    
```

5553	032616	002765	2765	
5554	032620	113200	113200	::1.0E07
5555	032622	000230	230	
5556	032624	041100	041100	::1.0E06
5557	032626	000017	17	
5558	032630	103240	103240	::1.0E05
5559	032632	000001	1	
5560	032634	023420	23420	::1.0E04
5561	032636	000000	0	
5562	032640	001750	1750	::1.0E03
5563	032642	000000	0	
5564	032644	000144	144	::1.0E02
5565	032646	000000	0	
5566	032650	000012	12	::1.0E01
5567	032652	000000	0	
5568	032654	000001	1	::1.0E00
5569	032656	000000	0	
5570	032660	000014		\$DECVL: .BLKB 12. ::RESERVE STORAGE FOR ASCIZ STRING

.SBTTL DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE

::*****
::*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
::*UNSIGNED OCTAL ASCIZ NUMBER.

```

::*CALL
::* MOV #PNTR, -(SP) :: POINTER TO LOW WORD OF BINARY NUMBER
::* JSR PC, @#$DB20 :: CALL THE ROUTINE
::* RETURN :: THE ADDRESS OF THE FIRST ASCIZ CHAR. IS ON THE STACK

```

5583	032674	104412		\$DB20: SAVREG :: SAVE ALL REGISTERS
5584	032676	016601	000002	MOV 2(SP), R1 :: PICKUP THE POINTER TO LOW WORD
5585	032702	012705	033013	MOV #SOCTVL+13., R5 :: POINTER TO DATA TABLE
5586	032706	012704	000014	MOV #12., R4 :: DO ELEVEN CHARACTERS
5587	032712	012703	177770	MOV #C7, R3 :: MASK
5588	032716	012100		MOV (R1)+, R0 :: LOWER WORD
5589	032720	012101		MOV (R1)+, R1 :: HIGH WORD
5590	032722	005002		CLR R2 :: TERMINATOR
5591	032724	110245		1\$: MOVB R2, -(R5) :: PUT CHARACTER IN DATA TABLE
5592	032726	010002		MOV R0, R2 :: GET THIS DIGIT
5593	032730	005304		DEC R4 :: COUNT THIS CHARACTER
5594	032732	003007		BGT 3\$:: BR IF NOT THE LAST DIGIT
5595	032734	001405		BEQ 2\$:: BR IF IT IS THE LAST DIGIT
5596	032736	005205		INC R5 :: ALL DIGITS DONE-ADJUST POINTER FOR FIRST
5597	032740	010566	000002	MOV R5, 2(SP) :: ASCIZ CHAR. & PUT IT ON THE STACK
5598	032744	104413		RESREG :: RESTORE ALL REGISTERS
5599	032746	000207		RTS PC :: RETURN TO USER
5600	032750	006203		2\$: ASR R3 :: POSITION THE MASK FOR THE LAST DIGIT
5601	032752	006001		3\$: ROR R1 :: POSITION THE BINARY NUMBER FOR
5602	032754	006000		ROR R0 :: THE NEXT OCTAL DIGIT
5603	032756	006001		ROR R1
5604	032760	006000		ROR R0
5605	032762	006001		ROR R1
5606	032764	006000		ROR R0

000007 032766 040302
000008 032770 062702 000060
000009 032774 000753
000010 032776 000016

BIC R3,R2 ;:MASK OUT ALL JUNK
ADD #0,R2 ;:MAKE THIS CHAR. ASCII
BR 1\$;:GO PUT IT IN THE DATA TABLE
\$OCTVL: .BLKB 14. ;:RESERVE DATA TABLE

.SBTTL TRAP DECODER

*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
*GO TO THAT ROUTINE.

000011 033014 010046
000012 033016 016600 000002
000013 033022 005740
000014 033024 111000
000015 033026 006300
000016 033030 016000 033050
000017 033034 000200

\$TRAP: MOV RO, -(SP) ;:SAVE RO
MOV 2(SP),RO ;:GET TRAP ADDRESS
TST -(RO) ;:BACKUP BY 2
MOVB (RO),RO ;:GET RIGHT BYTE OF TRAP
ASL RO ;:POSITION FOR INDEXING
MOV \$TRPAD(RO),RO ;:INDEX TO TABLE
RTS RO ;:GO TO ROUTINE

;;THIS IS USE TO HANDLE THE "GETPRI" MACRO

000018 033036 011646
000019 033040 016666 000004 000002
000020 033046 000002

\$TRAP2: MOV (SP), -(SP) ;:MOVE THE PC DOWN
MOV 4(SP),2(SP) ;:MOVE THE PSW DOWN
RTI ;:RESTORE THE PSW

.SBTTL TRAP TABLE

*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
*BY THE "TRAP" INSTRUCTION.

000021 033050 033036
000022 033052 030752
000023 033054 031216
000024 033056 031172
000025 033060 031232
000026 033062 031420
000027 033064 031714
000028 033066 031644
000029 033070 032126
000030 033072 030162
000031 033074 032404
000032 033076 032442
000033 033100 027406
000034 000032

ROUTINE

\$TRPAD: .WORD \$TRAP2
\$TYPE ;:CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
\$TYPOC ;:CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
\$TYPOS ;:CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
\$TYPON ;:CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
\$TYPDS ;:CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)

\$GTSWR ;:CALL=GTSWR TRAP+6(104406) GET SOFT-SWR SETTING

\$CKSWR ;:CALL=CKSWR TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
\$RDCHR ;:CALL=RDCHR TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
\$ROLIN ;:CALL=ROLIN TRAP+11(104411) TTY TYPEIN STRING ROUTINE
\$SAVREG ;:CALL=SAVREG TRAP+12(104412) SAVE RO-R5 ROUTINE
\$RESREG ;:CALL=RESREG TRAP+13(104413) RESTORE RO-R5 ROUTINE
\$DSPLY ;:CALL=DISPLY TRAP+14(104414) ROUTINE TO TYPE ERROR MESSAGES

\$TERM=.-\$TRPAD

5661
5662
5663
5664
5665
5666
5667
5668
5669
5670
5671
5672
5673
5674
5675
5676
5677
5678
5679
5680
5681
5682
5683
5684
5685
5686
5687
5688
5689
5690
5691
5692
5693
5694
5695
5696
5697
5698
5699
5700
5701
5702
5703
5704
5705
5706
5707
5708
5709
5710
5711
5712
5713
5714

.SBTTL SINGLE/DUAL PORT RH11/RP04/5/6 DRIVER (REV 1.0)

:COPYRIGHT (C) 1976
:DIGITAL EQUIPMENT CORP.
:MAYNARD, MA 01754
:AUTHOR(S): JIM LACEY/CHUCK HESS

::*****

:STORAGE FOR RPDS1, RPER1, RPER2, AND RPER3 ON AN ERROR "2"

:RPERRS = RPDS1
:RPERRS+2 = RPER1
:RPERRS+4 = RPER2
:RPERRS+6 = RPER3

033102 000000 000000 000000 RPERRS: .WORD 0,0,0,0
033110 000000

:TABLE OF DRIVE ACTIVE INDICATORS (DRVACT=8 BYTES)

:DRVACT=0 IF DRIVE IS IDLE
:DRVACT>0 IF DRIVE IS ACTIVE WITH A COMMAND
:DRVACT<0 IF DRIVE IS ACTIVE WITH AN ERROR RECOVERY OPERATION

033112 000 :DRVACT: .BYTE 0 :DRIVE 0
033113 000 :.BYTE 0 :DRIVE 1
033114 000 :.BYTE 0 :DRIVE 2
033115 000 :.BYTE 0 :DRIVE 3
033116 000 :.BYTE 0 :DRIVE 4
033117 000 :.BYTE 0 :DRIVE 5
033120 000 :.BYTE 0 :DRIVE 6
033121 000 :.BYTE 0 :DRIVE 7

:TABLE OF DRIVE STATUS INDICATORS (DRVSTA=8 BYTES)

:DRVSTA=0 IF DRIVE IS OFFLINE OR NONEXISTENT
:DRVSTA>0 IF DRIVE IS ONLINE
:DRVSTA<0 IF DRIVE IS UNSAFE

033122 000 DRVSTA: .BYTE 0 :DRIVE 0
033123 000 :.BYTE 0 :DRIVE 1
033124 000 :.BYTE 0 :DRIVE 2
033125 000 :.BYTE 0 :DRIVE 3
033126 000 :.BYTE 0 :DRIVE 4
033127 000 :.BYTE 0 :DRIVE 5
033130 000 :.BYTE 0 :DRIVE 6
033131 000 :.BYTE 0 :DRIVE 7

:TABLE OF DRIVE TYPES (DRV TYP=8 BYTES)

:DRV TYP=0 IF DRIVE IS NONEXISTENT (DRVSTA=0, ALSO)
:DRV TYP=1 IF DRIVE IS RP04
:DRV TYP=2 IF DRIVE IS RP05
:DRV TYP=4 IF DRIVE IS RP06
:DRV TYP=-1 IF NOT RP04/5/6


```

5715
5716 033132 000
5717 033133 000
5718 033134 000
5719 033135 000
5720 033136 000
5721 033137 000
5722 033140 000
5723 033141 000
5724
5725
5726
5727
5728
5729 033142 000
5730 033143 000
5731 033144 000
5732 033145 000
5733 033146 000
5734 033147 000
5735 033150 000
5736 033151 000
5737
5738
5739
5740
5741
5742 033152 000
5743 033153 000
5744 033154 000
5745 033155 000
5746 033156 000
5747 033157 000
5748 033160 000
5749 033161 000
5750
5751
5752
5753
5754
5755 033162 000000
5756
5757
5758
5759
5760
5761
5762
5763 033164 000000
5764
5765
5766
5767
5768

```

```

DRVTP: .BYTE 0          :DRIVE 0
        .BYTE 0          :DRIVE 1
        .BYTE 0          :DRIVE 2
        .BYTE 0          :DRIVE 3
        .BYTE 0          :DRIVE 4
        .BYTE 0          :DRIVE 5
        .BYTE 0          :DRIVE 6
        .BYTE 0          :DRIVE 7

```

```

;TABLE OF DUAL PORT INITIALIZATION INDICATORS
;DPINT=0 IF INITIALIZATION IS NOT ACTIVE ON THE DRIVE
;DPINT<0 IF INITIALIZATION IS IN PROGRESS

```

```

DPINT: .BYTE 0          :DRIVE 0
        .BYTE 0          :DRIVE 1
        .BYTE 0          :DRIVE 2
        .BYTE 0          :DRIVE 3
        .BYTE 0          :DRIVE 4
        .BYTE 0          :DRIVE 5
        .BYTE 0          :DRIVE 6
        .BYTE 0          :DRIVE 7

```

```

;TABLE OF PENDING DUAL PORT REQUESTS
;DPRQS=0 IF THAT A DUAL PORT REQUEST IS NOT PENDING FOR THAT DRIVE
;DPRQS<0 IF THAT A DUAL PORT REQUEST IS PENDING FOR THAT DRIVE

```

```

DPRQS: .BYTE 0          :DRIVE 0
        .BYTE 0          :DRIVE 1
        .BYTE 0          :DRIVE 2
        .BYTE 0          :DRIVE 3
        .BYTE 0          :DRIVE 4
        .BYTE 0          :DRIVE 5
        .BYTE 0          :DRIVE 6
        .BYTE 0          :DRIVE 7

```

```

;TRANSFER WAIT FLAG (TRNSWT=1 WORD)
;THIS IS A ONE WORD QUEUE. IT WILL CONTAIN THE ADDRESS OF
;"DPB" OF THE I/O OPERATION.

```

```
TRNSWT: .WORD 0
```

```

;SEARCH WAIT KEYS (SRCHWT=1 WORD)
;THIS IS A ONE WORD QUEUE THAT WILL CONTAIN A KEY FOR EACH OF
;THE DRIVES THAT ARE PERFORMING A SEARCH COMMAND FOR THE I/O
;REQUEST THAT IS AT THE TOP OF THEIR REQUEST QUEUE.
;EACH DRIVE IS ASSIGNED ONE BIT, STARTING AT BIT00 FOR DRIVE 0.

```

```
SRCHWT: .WORD 0
```

```

;RPO4/5/6 DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)
;ACTDRV=0 IF DRIVER IS INACTIVE
;ACTDRV>0 IF DRIVER IS ACTIVE

```

```

5769 033166 000 ACTDRV: .BYTE 0
5770
5771 ;SOFTWARE TIMER ROUTINE ACTIVE FLAG (ACTSTR=1 BYTE)
5772 ;ACTSTR=0 IF SOFTWARE TIMER ROUTINE IS INACTIVE
5773 ;ACTSTR>0 IF SOFTWARE TIMER ROUTINE IS ACTIVE
5774
5775 033167 000 ACTSTR: .BYTE 0
5776
5777 ;UNLOAD FLAG (ULDFLG=8 BYTES)
5778 ;ULDFLG=0 IF NO UNLOAD COMMAND
5779 ;ULDFLG>0 IF UNLOAD COMMAND IN PROGRESS
5780 ;ULDFLG<0 IF UNLOAD COMMAND IN WAIT QUEUE
5781
5782 033170 000 ULDFLG: .BYTE 0 ;DRIVE 0
5783 033171 000 .BYTE 0 ;DRIVE 1
5784 033172 000 .BYTE 0 ;DRIVE 2
5785 033173 000 .BYTE 0 ;DRIVE 3
5786 033174 000 .BYTE 0 ;DRIVE 4
5787 033175 000 .BYTE 0 ;DRIVE 5
5788 033176 000 .BYTE 0 ;DRIVE 6
5789 033177 000 .BYTE 0 ;DRIVE 7
5790
5791 ;LOOK AHEAD COUNT (LACNT=8 BYTES)
5792 ;LACNT WILL INDICATE THE NUMBER OF LOOK AHEADS PERFORMED
5793
5794 033200 000 LACNT: .BYTE 0 ;DRIVE 0
5795 033201 000 .BYTE 0 ;DRIVE 1
5796 033202 000 .BYTE 0 ;DRIVE 2
5797 033203 000 .BYTE 0 ;DRIVE 3
5798 033204 000 .BYTE 0 ;DRIVE 4
5799 033205 000 .BYTE 0 ;DRIVE 5
5800 033206 000 .BYTE 0 ;DRIVE 6
5801 033207 000 .BYTE 0 ;DRIVE 7
5802
5803 ;SAVE REGISTERS FLAG (SAVEFG =1 WORD)
5804 ;SAVEFG <0 IF SAVE THE RH11/RPO4/5/6 REGISTERS WHEN THE
5805 ;OPERATION IS COMPLETED AS PER (DPB+14).
5806 ;SAVEFG=0 IF SAVE THE RH11/RPO4/5/6 REGISTERS, AS PER
5807 ;(DPB+14), AFTER AN ERROR.
5808
5809 033210 000000 SAVEFG: .WORD 0
5810
5811 ;SEEK FLAG (SEEKFG=1 WORD)
5812 ;SEEKFG=0 IF WHEN THE DISK ADDRESS ISN'T IN THE WINDOW
5813 ;FOR A DATA TRANSFER START A SEARCH COMMAND
5814 ;SEEKFG<0 IF DATA TRANSFER WILL DO IMPLIED SEEKS,
5815 ;DISREGARD THE WINDOW
5816
5817 033212 000000 SEEKFG: .WORD 0
5818
5819 ;TIMEOUT TABLE (TIMER=8 WORDS)
5820 ;THIS TABLE CONTAINS THE TIME ALLOWED FOR AN OPERATION
5821
5822 033214 177777 TIMER: .WORD -1 ;DRIVE 0

```


5823	033216	177777	.WORD	-1	:DRIVE	1
5824	033220	177777	.WORD	-1	:DRIVE	2
5825	033222	177777	.WORD	-1	:DRIVE	3
5826	033224	177777	.WORD	-1	:DRIVE	4
5827	033226	177777	.WORD	-1	:DRIVE	5
5828	033230	177777	.WORD	-1	:DRIVE	6
5829	033232	177777	.WORD	-1	:DRIVE	7

;DATA TRANSFER UNDERWAY INDICATOR (DTUW=1 WORD)
 ;DTUW<0 IF NO DATA TRANSFER UNDERWAY
 ;DTUW=+N (WHERE N=0 TO 7) IMPLIES DATA TRANSFER UNDERWAY ON DRIVE N

5835	033234	177777	DTUW:	.WORD	-1
------	--------	--------	-------	-------	----

;ATTENTION BITS TABLE (ATABIT=8 BYTES)
 ;THIS TABLE CONTAINS THE CORRESPONDING BIT TO EACH DRIVES
 ;ATTENTION BIT

5841	033236	001	ATABIT:	.BYTE	1	:DRIVE	0
5842	033237	002		.BYTE	2	:DRIVE	1
5843	033240	004		.BYTE	4	:DRIVE	2
5844	033241	010		.BYTE	10	:DRIVE	3
5845	033242	020		.BYTE	20	:DRIVE	4
5846	033243	040		.BYTE	40	:DRIVE	5
5847	033244	100		.BYTE	100	:DRIVE	6
5848	033245	200		.BYTE	200	:DRIVE	7

;RPO4/5/6 TO RH11 "MASSBUS CONTROL BUS PARITY ERRORS" (MCPE) ALLOWED BEFORE
 ;CALLING IT FATAL (MCPEMX=1 WORD)

5853	033246	000003	MCPEMX:	.WORD	3
------	--------	--------	---------	-------	---

;STORAGE FOR RPADR (THE FIRST ADDRESS (776700) OF THE RH11/RPO4/5/6),
 ;RPVEC (THE VECTOR ADDRESS (254)), AND RPVEC+2 (THE BR LEVEL (5)).

5858	033250	176700	RPADR:	.WORD	176700
5859	033252	000254	RPVEC:	.WORD	254,5*32.

;MAXIMUM NUMBER OF LOOK AHEADS ALLOWED IS 4 (MXLACT=1 WORD)

5863	033256	000004	MXLACT:	.WORD	4
------	--------	--------	---------	-------	---

;MAXIMUM DELTA DELAY IS 8 SECTORS (MXDLTA=1 WORD)

5866	033260	001000	MXDLTA:	.WORD	8.*64.
------	--------	--------	---------	-------	--------

;MINIMUM DELTA DELAY IS 2 SECTORS (MNDLTA=1 WORD)

5869	033262	000200	MNDLTA:	.WORD	2*64.
------	--------	--------	---------	-------	-------

;MAXIMUM SEARCH FOR I/O WINDOW IS 5 SECTORS (MXWNDW=1 WORD)

5872	033264	000005	MXWNDW:	.WORD	5
------	--------	--------	---------	-------	---

;DEFINITIONS OF THE RH11/RPO4/5/6 ADDRESS INDEXES

5876		000000	RPCS1=0			;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 00)
------	--	--------	---------	--	--	---

5877	000002	RPWC=2	:WORD COUNT REGISTER (NOT A DRIVE REG)
5878	000004	RPBA=4	:UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)
5879	000006	RPDA=6	:DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 05)
5880	000010	RPCS2=10	:CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
5881	000012	RPDS1=12	:DRIVE STATUS REGISTER (DRIVE REG 01)
5882	000014	RPER1=14	:ERROR REGISTER #1 (DRIVE REG. 02)
5883	000016	RPAS=16	:ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 04)
5884	000020	RPLA=20	:LOOK AHEAD REGISTER (DRIVE REG. 07)
5885	000022	RPDB=22	:DATA BUFFER REGISTER (NOT A DRIVE REG.)
5886	000024	RPMR=24	:MAINTAINABILITY REGISTER (DRIVE REG. 03)
5887	000026	RPDT=26	:DRIVE TYPE REGISTER (DRIVE REG. 06)
5888	000030	RPSN=30	:SERIAL NUMBER REGISTER (DRIVE REG. 10)
5889	000032	RPOF=32	:OFFSET REGISTER (DRIVE REG. 11)
5890	000034	RPCA=34	:DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
5891	000036	RPCC=36	:CURRENT CYLINDER ADDRESS REGISTER (DRIVE REG. 13)
5892	000040	RPER2=40	:ERROR REGISTER #2 (DRIVE REG. 14)
5893	000042	RPER3=42	:ERROR REGISTER #3 (DRIVE REG. 15)
5894	000044	RPEC1=44	:ECC POSITION REGISTER (DRIVE REG. 16)
5895	000046	RPEC2=46	:ECC PATTERN REGISTER (DRIVE REG. 17)

```

:RH11/RPO4/5/6 DRIVER INITIALIZATION CODE
:THIS ROUTINE WILL DETERMINE WHICH RPO4/5/6 DRIVES ARE
:AVAILABLE FOR TESTING AND SET THE DRVSTA INDICATOR
:TO THE PROPER STATE FOR EACH DRIVE.
:NOTE: THIS ROUTINE CALLS DRVINT

```

:CALL

```

:      JSR    PC,RPINIT
:      RETURN

```

:NOTE: THE 'P' OR 'L' CLOCK MUST BE STARTED

5910	033266	104412	RPINIT: SAVREG	:SAVE R0 - R5
5911	033270	013746	MOV	:SAVE THE PRESENT PROCESSOR STATUS
5912	033274	012737	MOV	:CHANGE THE PRIORITY TO 5
5913	033302	004737	JSR	:CLEAR ALL REQUEST QUEUES
5914	033306	012701	MOV	:FIRST ADDRESS TO BE CLEARED
5915	033312	012702	MOV	:LAST ADDRESS TO BE CLEARED
5916	033316	005021	1\$: CLR	:CLEAR
5917	033320	020102	CMP	:ARE WE DONE?
5918	033322	101775	BLOS	:BRANCH IF NO
5919	033324	012702	MOV	:LAST ADDRESS
5920	033330	012721	2\$: MOV	:INITIALIZE
5921	033334	020102	CMP	:DONE?
5922	033336	101774	BLOS	:LOOP IF NO
5923	033340	005037	CLR	:SET ALL DRIVES TO OFFLINE
5924	033344	005037	CLR	
5925	033350	005037	CLR	
5926	033354	005037	CLR	
5927	033360	013703	MOV	:SETUP THE RH11/RPO4/5/6 VECTOR
5928	033364	012723	MOV	
5929	033370	013713	MOV	
5930	033374	013704	MOV	:FIRST ADDRESS OF RH11/RPO4


```

5985 033602 022705 020020      CMP      #20020,R5      ;IS IT A SINGLE PORT RPO4?
5986 033606 001431      BEQ      2$          ;BRANCH IF YES
5987 033610 022705 024020      CMP      #24020,R5      ;IS IT A DUAL PORT RPO4?
5988 033614 001426      BEQ      2$          ;BR IF YES
5989 033616 112761 000002 033132    MOVB     #2,DRVTP(R1) ;SET RPO5 INDICATOR
5990 033624 022705 020021      CMP      #20021,R5      ;SINGLE PORT RPO5 ?
5991 033630 001420      BEQ      2$          ;BR IF YES
5992 033632 022705 024021      CMP      #24021,R5      ;DUAL PORT RPO5 ?
5993 033636 001415      BEQ      2$          ;BR IF YES
5994 033640 112761 000004 033132    MOVB     #4,DRVTP(R1) ;SET RPO6 INDICATOR
5995 033646 022705 020022      CMP      #20022,R5      ;SINGLE PORT RPO6 ?
5996 033652 001407      BEQ      2$          ;BR IF YES
5997 033654 022705 024022      CMP      #24022,R5      ;DUAL PORT RPO6 ?
5998 033660 001404      BEQ      2$          ;BR IF YES
5999 033662 112761 177777 033132    MOVB     #-1,DRVTP(R1);SET INDICATOR TO 'OTHER'
6000 033670 000446      BR       6$          ;EXIT
6001 033672 012746 000121      2$:     MOV      #121,-(SP) ;DO A "READ-IN PRESET"
6002 033676 004037 040500      JSR      RO,WRT.RP
6003 033702 000000      RPCS1
6004 033704 034032      8$
6005 033706 012746 010000      MOV      #BIT12,-(SP) ;SET FMT22=1
6006 033712 004037 040500      JSR      RO,WRT.RP
6007 033716 000032      RPOF
6008 033720 034032      8$
6009 033722 004037 040324      JSR      RO,RO.RP ;READ RPDS1
6010 033726 000012      RPDS1
6011 033730 034032      8$
6012 033732 012605      MOV      (SP)+,R5 ;AND SAVE IT IN R5
6013 033734 100015      BPL      4$          ;BRANCH IF ATA=0
6014 033736 116164 033236 000016    MOVB     ATABIT(R1),RPAS(R4);CLEAR ATTENTION BIT
6015 033744 004037 040324      JSR      RO,RO.RP ;FIND OUT WHY ATA=1
6016 033750 000014      RPER1
6017 033752 034032      8$
6018 033754 006126      ROL      (SP)+ ;IS IT UNSAFE?
6019 033756 100004      BPL      4$          ;BR IF NOT
6020 033760 112761 177777 033122    MOVB     #-1,DRVSTA(R1);SET UNSAFE INDICATOR
6021 033766 000407      BR       6$          ;EXIT
6022 033770 005105      4$:     COM      R5 ;CHECK MOL, DPR, DRY, AND VV
6023 033772 042705 167077      BIC      #<BIT12!BIT08!BIT07!BIT06>,R5
6024 033776 001003      BNE      6$          ;BRANCH IF MOL, DPR, DRY, OR VV IS CLEAR
6025 034000 112761 000001 033122    MOVB     #1,DRVSTA(R1);SET DRIVE STATUS TO ONLINE
6026 034006 005720      6$:     TST      (RO)+ ;STEP OVER THE ERROR RETURN
6027 034010 000410      BR       8$          ;EXIT
6028 034012 006301      7$:     ASL      R1 ;CHANGE INDEX TO ADDRESS WORDS
6029 034014 012761 003720 033214    MOV      #2000.,TIMER(R1);START 2 SEC TIMER
6030 034022 006201      ASR      R1 ;RESTORE R1
6031 034024 112761 177777 033142    MOVB     #-1,DPINT(R1);SET PORT INITIALIZE INIDICATOR
6032 034032 012605      8$:     MOV      (SP)+,R5 ;RESTORE R5
6033 034034 000200      RTS      RO ;EXIT
6034
6035 ;REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
6036 ;CALL
6037 ;
6038 ;
    
```



```

6039      :      JSR      RO,#RPO4      ;CALL THE RPO4/5/6 DRIVER
6040      :      PNTADR     ;ADDRESS OF POINTER OF DRIVES PARAMETER BLOCK
6041      :      RETURN1    ;RETURN HERE IF QUEUE IS FULL
6042      :      RETURN2    ;RETURN HERE IF REQUEST IS IN QUEUE OR THERE
6043      :      :          ;IS AN ERROR CONDITION
6044
6045 034036 013746 177776 RPO4: MOV      #PS,-(SP)      ;SAVE THE CALLING STATUS
6046 034042 013737 033254 177776 MOV      RPVEC+2,#PS    ;DON'T ALLOW ANY RPO4/5/6 INTERRUPTS
6047 034050 112737 000001 033166 MOVVB   #1,ACTDRV     ;SET "ACTIVE DRIVER" FLAG
6048 034056 104412 SAVREG   ;SAVE R0 - R5
6049 034060 011002 MOV      (R0),R2      ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
6050 034062 005062 000016 CLR      16(R2)      ;CLEAR THE STATUS/ERROR INDICATOR
6051 034066 111201 MOVVB   (R2),R1      ;PICKUP THE DRIVE NUMBER
6052 034070 013704 033250 MOV      RPADR,R4     ;UNIBUS ADDRESS OF RPCS1
6053 034074 105761 033122 TSTB   DRVSTA(R1)    ;CHECK DRIVES STATUS
6054 034100 003014 BGT     1$          ;BRANCH IF ONLINE
6055 034102 105761 033170 TSTB   ULDFLG(R1)    ;UNLOAD COMMAND IN QUEUE?
6056 034106 001036 BNE     3$          ;BRANCH IF YES
6057 034110 105761 033142 TSTB   DPINT(R1)     ;TRYING TO INIT THE DRIVE
6058 034114 001042 BNE     5$          ;BR IF YES
6059 034116 004037 033500 JSR     RO,DRVINT    ;GO INIT. THE DRIVE
6060 034122 000434 BR      4$          ;ERROR RETURN
6061 034124 105761 033122 TSTB   DRVSTA(R1)    ;IS DRIVE STATUS ONLINE?
6062 034130 003445 BLE     6$          ;BR IF NOT
6063 034132 105761 033152 1$: TSTB   DPRQS(R1)     ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
6064 034136 001031 BNE     5$          ;BR IF YES
6065 034140 010164 000010 MOV      R1,RPCS2(R4) ;SELECT THE DRIVE
6066 034144 004037 041446 JSR     RO,DRVQUE     ;PUT THIS REQUEST IN QUEUE
6067 034150 000460 BR      9$          ;QUEUE IS FULL
6068 034152 122762 000103 000002 CMPB   #103,2(R2)    ;IS THIS REQ. FOR AN UNLOAD?
6069 034160 001003 BNE     2$          ;BR IF NO
6070 034162 112761 177777 033170 MOVVB   #-1,ULDFLG(R1) ;SET THE "UNLOAD IN QUEUE" FLAG
6071 034170 105761 033112 2$: TSTB   DRVACT(R1)    ;IS THIS DRIVE ACTIVE?
6072 034174 001043 BNE     8$          ;BR IF YES
6073 034176 004737 034330 JSR     PC,OPT        ;CALL THE OPTIMIZER
6074 034202 000440 BR      8$
6075 034204 012762 120000 000016 3$: MOV      #BIT15:BIT13,16(R2) ;SET THE "UNLOAD IN QUEUE" ERROR FLAG
6076 034212 000434 BR      8$          ;EXIT
6077 034214 004737 035440 4$: JSR     PC,CI7        ;GO HANDLE THE PARITY ERROR
6078 034220 000431 BR      8$
6079 034222 004037 041446 5$: JSR     RO,DRVQUE     ;PUT REQUEST IN QUEUE
6080 034226 000431 BR      9$          ;QUEUE IS FULL
6081 034230 032714 000100 BIT     #BIT06,(R4)  ;IS 'IE' SET ALREADY ?
6082 034234 001023 BNE     8$          ;BR IF IT IS
6083 034236 004737 041004 JSR     PC,SET.IE    ;SET INTERRUPT
6084 034242 000420 BR      8$          ;RETURN, REQUEST IN QUEUE
6085 034244 105761 033122 6$: TSTB   DRVSTA(R1)    ;SEE IF DRIVE OFFLINE OR UNSAFE
6086 034250 002412 BLT     7$          ;BR IF UNSAFE
6087 034252 012762 140000 000016 MOV      #BIT15:BIT14,16(R2) ;SET OFFLINE ERROR INDICATOR
6088 034260 105761 033132 TSTB   DRVSTYP(R1)  ;SEE IF OFFLINE OR NONEXISTENT
6089 034264 001007 BNE     8$          ;BR IF OFFLINE
6090 034266 012762 100002 000016 MOV      #BIT15:BIT01,16(R2) ;REPORT DRIVE NONEXISTENT
6091 034274 000403 BR      8$          ;GO TO EXIT
6092 034276 012762 110000 000016 7$: MOV      #BIT15:BIT12,16(R2) ;DRIVE IS UNSAFE

```

```

6093 034304 104413          8$: RESREG          ;RESTORE R0 - R5
6094 034306 005720          TST      (R0)+      ;SETUP FOR NORMAL RETURN
6095 034310 000401          BR       10$        ;FINISH UP, THEN EXIT
6096 034312 104413          9$: RESREG          ;RESTORE R0 - R5
6097 034314 005720          10$: TST      (R0)+  ;CORRECT THE RETURN ADDRESS
6098 034316 105037 033166      CLR      ACTDRV     ;CLEAR "ACTIVE DRIVER" FLAG
6099 034322 012637 177776      MOV      (SP)+,2#PS ;RETURN "PS" TO USER LEVEL
6100 034326 000200          RTS       R0        ;RETURN TO CALLER

;OPTIMIZER-CALLED FOR A PARTICULAR DRIVE
CALL
MOV      #DRVNUM,R1    ;DRIVE NUMBER TO R1
JSR      PC,OPT        ;SETUP A COMMAND
OPT: SAVREG          ;SAVE R0 - R5
MOV      2#PS, -(SP)   ;SAVE PROC. STATUS
BIC      ATABIT(R1),SRCHWT ;CLEAR "SEARCH WAIT" KEY
JSR      PC,GETREQ     ;GET "DPB" POINTER OF REQUEST
TST      R2            ;IS THERE A REQUEST IN QUEUE?
BEQ      7$           ;NO--BRANCH TO EXIT
BIT      #BIT12,RPDS1(R4) ;IS MOL STILL SET ?
BEQ      9$           ;BR IF NOT
BIT      #BIT6,RPDS1(R4) ;IS VV SET ?
BNE      10$          ;BR IF IT IS
9$: JSR      R0,DRVINT ;SEE IF DRIVE STILL ONLINE ?
BR       6$           ;PARITY OR 'DVA' NOT SET
10$: TST      DRVSTA(R1) ;IS DRIVE ONLINE?
BGT      1$           ;YES--BRANCH
JSR      PC,POPQUE    ;NO--REMOVE REQUEST FROM QUEUE
MOV      #BIT15:BIT14,16(R2) ;SET OFFLINE STATUS/ERROR INDICATOR
TST      DRVSTA(R1)  ;IS DRIVE UNSAFE ?
BPL      8$           ;BR TO EXIT IF NOT
MOV      #BIT15:BIT12,16(R2) ;SET UNSAFE STATUS/ERROR INDICATOR
BR       8$           ;BRANCH TO EXIT
1$: MOV      #111, -(SP) ;LOAD COMMAND ONTO THE STACK
JSR      R0,WRT.RP    ;LOAD THE REGISTER
RPCS1   ;REGISTER INCREMENT
6$      ;ERROR RETURN ADDRESS
BIT      #BIT11,(R4)  ;DRIVE AVAILABLE ?
BEQ      5$           ;BR IF NOT
CMP      #150,2(R2)   ;IS THE REQUEST FOR I/O?
BLT      2$           ;YES--BRANCH
JSR      PC,C14       ;CALL THE COMMAND INITIATOR
BR       8$           ;BRANCH TO EXIT
2$: TST      DTUW      ;DATA TRANSFER UNDERWAY?
BGE      4$           ;YES--GO START A SEARCH
TST      SEEKFG       ;DO IMPLIED SEEKS?
BMI      3$           ;YES---BRANCH
JSR      R0,LA        ;NO--DO LOOK AHEAD
BR       8$           ;RETURN HERE ON A PARITY ERROR
BR       4$           ;GO START A SEARCH
3$: JSR      PC,C11    ;START A DATA TRANSFER
BR       8$

```



```

6147 034532 004737 034716      4$: JSR    PC,CI3      ;START A SEARCH
6148 034536 000420                BR      8$          ;GO TO THE EXIT
6149 034540 112761 177777 033152 5$: MOVB   #-1,DPRQS(R1) ;SET PORT REQUEST INDICATOR
6150 034546 010103                MOV     R1,R3      ;SET UP TO ADDRESS WORDS
6151 034550 006303                ASL     R3          ;CONVERT TO WORD INDEX
6152 034552 012763 023420 033214  MOV     #10000.,TIMER(R3) ;START 10 SEC TIMER
6153 034560 000402                BR      7$          ;EXIT
6154 034562 004737 035440      6$: JSR    PC,CI7      ;PROCESS THE PARITY ERROR
6155 034566 032714 000100      7$: BIT    #BIT06,(R4) ;SEE IF 'IE' ALREADY SET
6156 034572 001002                BNE     8$          ;BR IF SET
6157 034574 004737 041004      JSR    PC,SET.IE    ;SET "IE" WITHOUT A "TRE"
6158 034600 012637 177776      8$: MOV     (SP)+,@#PS ;RESTORE PROC. STATUS
6159 034604 104413                RESREG ;RESTORE RO - R5
6160 034606 000207                RTS     PC

6161
6162                ;COMMAND INITIATOR
6163
6164                ;CALL
6165                MOV     #DRVNUM,R1 ;DRIVE NUMBER
6166                MOV     #DPB,R2    ;ADDRESS OF DPB
6167                JSR    PC,CI?     ;CI?= CI1,CI3, OR CI4
6168                ;WHERE:
6169                ;CI1=DATA TRANSFER
6170                ;CI2=SEARCH REQUESTED BY DATA XFER
6171                ;CI4=NOT DATA TRANSFER
6172
6173 034610 004737 041544      CI1: JSR    PC,POPQUE   ;REMOVE REQUEST FROM "DRIVES WAIT" QUEUE
6174 034614 010237 033162      MOV     R2,TRNSWT  ;PUT REQ. IN TRANSFER WAIT QUEUE
6175 034620 010203                MOV     R2,R3      ;DPB ADDRESS TO R3
6176 034622 013704 033250      MOV     RPADR,R4   ;RPCS1 ADDRESS
6177 034626 010164 000010      MOV     R1,RPCS2(R4) ;SELECT DRIVE
6178 034632 062703 000004      ADD     #4,R3      ;DESIRED WORD COUNT
6179 034636 062704 000002      ADD     #2,R4      ;RPWC ADDRESS
6180 034642 012324                MOV     (R3)+,(R4)+ ;LOAD WORD COUNT
6181 034644 012324                MOV     (R3)+,(R4)+ ;LOAD BUFFER ADDRESS
6182 034646 012346                MOV     (R3)+,-(SP) ;LOAD SECTOR AND TRACK
6183 034650 004037 040500      JSR    RO,WRT.RP   ;CALL THE LOAD(WRITE) ROUTINE
6184 034654 000006                RPDA                ;INDEX OF REGISTER TO LOAD
6185 034656 035440                CI7                ;ERROR RETURN ADDRESS
6186 034660 012346                MOV     (R3)+,-(SP) ;LOAD CYLINDER ADDRESS
6187 034662 004037 040500      JSR    RO,WRT.RP
6188 034666 000034                RPCA
6189 034670 035440                CI7
6190 034672 016246 000002      MOV     2(R2),-(SP) ;LOAD "COMMAND+GO", "A17&A16", AND "PSEL"
6191 034676 004037 040500      JSR    RO,WRT.RP
6192 034702 000000                RPCS1
6193 034704 035440                CI7
6194 034706 010137 033234      MOV     R1,DTUW    ;SET "DATA TRANSFER UNDERWAY"
6195 034712 000137 035402      JMP     CI5
6196 034716 013704 033250      CI3: MOV     RPADR,R4   ;RPCS1 ADDRESS
6197 034722 010164 000010      MOV     R1,RPCS2(R4) ;SELECT DRIVE
6198 034726 016246 000012      MOV     12(R2),-(SP) ;DESIRED CYLINDER ADDRESS
6199 034732 004037 040500      JSR    RO,WRT.RP
6200 034736 000034                RPCA
    
```

6201	034740	035440			CI7		
6202	034742	116203	000010		MOVB	10(R2),R3	;PICKUP SECTOR ADDRESS
6203	034746	163703	033264		SUB	MXWINDW,R3	;BACKUP BY MAX. SEARCH FOR I/O WINDOW
6204	034752	002002			BGE	1\$	
6205	034754	062703	000026		ADD	#22.,R3	
6206	034760	010346		1\$:	MOV	R3,-(SP)	;COMBINE THE ADJUSTED SECTOR WITH
6207	034762	116266	000011	000001	MOVB	11(R2),1(SP)	;THE DESIRED TRACK
6208	034770	004037	040500		JSR	RO,WRT.RP	;LOAD DESIRED TRACK & SECTOR
6209	034774	000006			RPDA		
6210	034776	035440			CI7		
6211	035000	012746	000131		MOV	#131,-(SP)	;START A SEARCH
6212	035004	004037	040500		JSR	RO,WRT.RP	
6213	035010	000000			RPCS1		
6214	035012	035440			CI7		
6215	035014	156137	033236	033164	BISB	ATABIT(R1),SRCHWT	;SET "SEARCH WAIT" KEY
6216	035022	000567			BR	CI5	
6217	035024	013704	033250	CI4:	MOV	RPADR,R4	;RPCS1 ADDRESS
6218	035030	010164	000010		MOV	R1,RPCS2(R4)	;SELECT DRIVE
6219	035034	116203	000002		MOVB	2(R2),R3	;PICKUP THE REQUESTED COMMAND
6220	035040	122703	000131		CMPB	#131,R3	;IS IT A SEARCH COMMAND?
6221	035044	001007			BNE	1\$;BRANCH IF NO
6222	035046	016246	000010		MOV	10(R2),-(SP)	;LOAD DESIRED TRACK & SECTOR
6223	035052	004037	040500		JSR	RO,WRT.RP	
6224	035056	000006			RPDA		
6225	035060	035440			CI7		
6226	035062	000403			BR	2\$;GO LOAD CYLINDER
6227	035064	122703	000105	1\$:	CMPB	#105,R3	;IS IT A SEEK COMMAND
6228	035070	001007			BNE	3\$;BRANCH IF NO
6229	035072	016246	000012	2\$:	MOV	12(R2),-(SP)	;LOAD DESIRED CYLINDER
6230	035076	004037	040500		JSR	RO,WRT.RP	
6231	035102	000034			RPCA		
6232	035104	035440			CI7		
6233	035106	000546			BR	CI6	
6234	035110	122703	000115	3\$:	CMPB	#115,R3	;IS IT AN "OFFSET" COMMAND?
6235	035114	001013			BNE	4\$;BR IF NO
6236	035116	004037	040324		JSR	RO,RO.RP	;MERGE THE OFFSET VALUE INTO RPOF
6237	035122	000032			RPOF		;BUT DON'T CHANGE THE UPPER
6238	035124	035440			CI7		
6239	035126	116216	000001		MOVB	1(R2),(SP)	;BYTE WHEN LOADING THE
6240	035132	004037	040500		JSR	RO,WRT.RP	;REGISTER (RPOF)
6241	035136	000032			RPOF		
6242	035140	035440			CI7		
6243	035142	000530			BR	CI6	;GO START THE COMMAND
6244	035144	122703	000107	4\$:	CMPB	#107,R3	;IS IT A "RECALIBRATE" COMMAND?
6245	035150	001525			BEQ	CI6	;BRANCH IF YES
6246	035152	122703	000117		CMPB	#117,R3	;IS IT A RETURN TO CENTER?
6247	035156	001522			BEQ	CI6	;BRANCH IF YES
6248	035160	122703	000103		CMPB	#103,R3	;IS IT AN "UNLOAD" COMMAND?
6249	035164	001016			BNE	5\$;BRANCH IF NO
6250	035166	112761	000001	033112	MOVB	#1,DRVACT(R1)	;SET THE DRIVE ACTIVE INDICATOR
6251	035174	105061	033122		CLRB	DRVSTA(R1)	;PUT DRIVE STATUS TO OFFLINE
6252	035200	112761	000001	033170	MOVB	#1,ULDFLG(R1)	;SET "UNLOAD IN PROGRESS" FLAG
6253	035206	010346			MOV	R3,-(SP)	;START THE "UNLOAD" COMMAND
6254	035210	004037	040500		JSR	RO,WRT.RP	

6255	035214	000000				RPCS1			
6256	035216	035440				CI7			
6257	035220	000207				RTS	PC		; RETURN TO USER
6258	035222	122703	000143		5\$:	CMPB	#143,R3		; IS IT A "SET FORMAT" COMMAND?
6259	035226	001014				BNE	6\$; BRANCH IF NO
6260	035230	004037	040324			JSR	RO, RD.RP		; READ THE OFFSET REGISTER
6261	035234	000032				RPOF			
6262	035236	035440				CI7			
6263	035240	116266	000001	000001		MOVB	1(R2), 1(SP)		; COMBINE "FMT22", "ECI", AND "HCI"
6264	035246	004037	040500			JSR	RO, WRT.RP		; LOAD "FMT22", "ECI", AND/OR "HCI".
6265	035252	000032				RPOF			
6266	035254	035440				CI7			
6267	035256	000436				BR	12\$		
6268	035260	122703	000141		6\$:	CMPB	#141,R3		; IS IT A "GET REGISTER" COMMAND?
6269	035264	001023				BNE	10\$; BRANCH IF NO
6270	035266	016203	000006		7\$:	MOV	6(R2), R3		; POINTS TO 1ST ADDRESS OF WHERE
6271									; TO PUT THE REGISTER(S)
6272	035272	116237	000010	035310		MOVB	10(R2), 9\$; INIT. THE INDEX FOR THE FIRST REG.
6273	035300	116205	000011			MOVB	11(R2), R5		; INDEX OF LAST REG. TO MOVE
6274	035304	004037	040324		8\$:	JSR	RO, RD.RP		; READ RPO4/5/6 REGISTER
6275	035310	000000			9\$:	RPCS1			; INDEX OF REG. TO READ
6276	035312	035440				CI7			
6277	035314	012623				MOV	(SP)+, (R3)+		; GET THE CONTENTS OF RH11/RPO4/5/6 REG.
6278	035316	023705	035310			CMP	9\$, R5		; LAST REG. BEEN READ?
6279	035322	001414				BEQ	12\$; GET OUT IF YES
6280	035324	062737	000002	035310		ADD	#2, 9\$; INCREASE THE INDEX BY 2
6281	035332	000764				BR	8\$; LOOP--MORE TO READ
6282	035334	122703	000145		10\$:	CMPB	#145,R3		; IS IT A "SELECT DRIVE" COMMAND?
6283	035340	001405				BEQ	12\$; BRANCH IF YES
6284	035342	010346			11\$:	MOV	R3, -(SP)		; LOAD THE COMMAND
6285	035344	004037	040500			JSR	RO, WRT.RP		
6286	035350	000000				RPCS1			
6287	035352	035440				CI7			
6288	035354	004737	041544		12\$:	JSR	PC, POPQUE		; REMOVE REG. FROM QUEUE
6289	035360	052762	000200	000016		BIS	#BIT07, 16(R2)		; SET THE "DONE" BIT
6290	035366	005737	033210			TST	SAVEFG		; SAVE THE RH11/RPO4/5/6 REGISTERS?
6291	035372	100002				BPL	13\$; BRANCH IF NO
6292	035374	004737	040666			JSR	PC, SVRH11		; YES--GO SAVE THE REGISTERS
6293	035400	000207			13\$:	RTS	PC		; RETURN TO USER
6294	035402	006301			CI5:	ASL	R1		
6295	035404	012761	001750	033214		MOV	#1000., TIMER(R1)		; SET A ONE SECOND TIMER
6296	035412	006201				ASR	R1		
6297	035414	112761	000001	033112		MOVB	#1, DRVACT(R1)		; SET THE DRIVE ACTIVE
6298	035422	000207				RTS	PC		; RETURN TO THE USER
6299	035424	010346			CI6:	MOV	R3, -(SP)		; LOAD THE COMMAND
6300	035426	004037	040500			JSR	RO, WRT.RP		
6301	035432	000000				RPCS1			
6302	035434	035440				CI7			
6303	035436	000761				BR	CI5		
6304	035440	032764	010000	000010	CI7:	BIT	#BIT12, RPCS2(R4)		; DRIVE NON-EXISTENT ?
6305	035446	001034				BNE	CI8		; BR IF YES
6306	035450	005702			1\$:	TST	R2		; ANYTHING IN QUEUE ?
6307	035452	001405				BEQ	CI7B		; BR IF NOT
6308	035454	012762	104000	000016		MOV	#BIT15!BIT11, 16(R2)		; SET "PARITY" ERROR INDICATOR

6309	035462	004737	040666			JSR	PC,SVRH11	:GO SAVE THE RH11/RPO4/5/6 REGISTERS
6310	035466	012746	000111			MOV	#11,-(SP)	:DO A "DRIVE CLEAR"
6311	035472	004037	040500			JSR	RO,WRT.RP	
6312	035476	000000				RPCS1		
6313	035500	035540				CIB		
6314	035502	004737	041426			JSR	PC,EMPTYQ	:EMPTY THE QUEUE
6315	035506	105061	033170			CLRB	ULDFLG(R1)	:CLEAR THE UNLOAD IN QUEUE FLAG
6316	035512	105061	033112			CLRB	DRVACT(R1)	:DRIVE IS IDLE
6317	035516	020137	033234			CMP	R1,DTUW	:IF THIS DRIVE HAD AN I/O REQUEST
6318	035522	001005				BNE	IS	:IN PROGRESS CLEAR ALL OF THE FLAGS
6319	035524	005037	033162			CLR	TRNSWT	
6320	035530	012737	177777	033234		MOV	#-1,DTUW	
6321	035536	000207				RTS	PC	
6322	035540	104412				SAVREG		:SAVE R0 - R5
6323	035542	032764	010000	000010		BIT	#BIT12,RPCS2(R4)	:IS 'NED' SET ?
6324	035550	001002				BNE	IS	:BR IF YES
6325	035552	005001				CLR	R1	
6326	035554	005003				CLR	R3	
6327	035556	105761	033112			TSTB	DRVACT(R1)	:DRIVE ACTIVE?
6328	035562	001443				BEQ	5\$:BRANCH IF NO
6329	035564	013702	033162			MOV	TRNSWT,R2	:GET THE "TRANSFER WAIT" QUEUE
6330	035570	020137	033234			CMP	R1,DTUW	:DID THIS DRIVE HAVE AN I/O IN PROGRESS?
6331	035574	001402				BEQ	2\$:BRANCH IF YES
6332	035576	004737	041522			JSR	PC,GETREQ	:GET THE DPB POINTER
6333	035602	005702				TST	R2	:QUEUE ENTRY FOR DRIVE ?
6334	035604	001415				BEQ	4\$:BR IF NOT
6335	035606	032764	010000	000010		BIT	#BIT12,RPCS2(R4)	: 'NED' SET ?
6336	035614	001404				BEQ	3\$:BR IF NOT
6337	035616	012762	100002	000016		MOV	#BIT15:BIT01,16(R2)	:SET 'DRIVE NON-EXISTENT' INDICATOR
6338	035624	000405				BR	4\$:CONTINUE
6339	035626	012762	102000	000016		MOV	#BIT15:BIT10,16(R2)	:SET "NON-CLEARABLE PARITY" ERROR INDICATOR
6340	035634	004737	040666			JSR	PC,SVRH11	:SAVE RH11/RPO4/5/6 REGISTERS
6341	035640	012763	177777	033214		MOV	#-1,TIMER(R3)	:STOP THE TIMER
6342	035646	105061	033112			CLRB	DRVACT(R1)	:SET "DRIVE ACTIVE" TO IDLE
6343	035652	020137	033234			CMP	R1,DTUW	:IS THIS DRIVE SETUP FOR A TRANSFER
6344	035656	001005				BNE	5\$:BR IF NOT
6345	035660	012737	177777	033234		MOV	#-1,DTUW	:RESET THE INDICATOR
6346	035666	005037	033162			CLR	TRNSWT	:CLEAR THE TRANSFER QUEUE
6347	035672	105061	033170			CLRB	ULDFLG(R1)	:CLEAR UNLOAD FLAG
6348	035676	032764	010000	000010		BIT	#BIT12,RPCS2(R4)	: 'NED' SET ?
6349	035704	001021				BNE	6\$:BR IF YES
6350	035706	005201				INC	R1	:MOVE TO THE NEXT DRIVE
6351	035710	062703	000002			ADD	#2,R3	
6352	035714	042701	177770			BIC	#7,R1	
6353	035720	001316				BNE	IS	:BRANCH IF MORE DRIVES
6354	035722	012737	177777	033234		MOV	#-1,DTUW	:NO DATA TRANSFERS UNDERWAY
6355	035730	005037	033162			CLR	TRNSWT	:CLEAR THE 'TRANSFER WAIT' QUEUE
6356	035734	004737	041350			JSR	PC,CLRQUE	:CLEAR ALL OF THE REQUEST QUEUES
6357	035740	012764	000040	000010		MOV	#BIT05,RPCS2(R4)	:DO A MASSBUS INIT.
6358	035746	000406				BR	7\$:CONTINUE
6359	035750	004737	041426			JSR	PC,EMPTYQ	:CLEAR THE DRIVE'S QUEUE
6360	035754	105061	033122			CLRB	DRVSTA(R1)	:SET DRIVE TO OFFLINE
6361	035760	105061	033132			CLRB	DRVTYP(R1)	:CLEAR THE DRIVE TYPE INDICATOR
6362	035764	004737	041004			JSR	PC,SET.IE	:SET "IE" WITHOUT "TRE"


```

6363 035770 104413 RESREG PC ;RESTORE R0 - R5
6364 035772 000207 RTS PC ;RETURN
6365
6366 ;LOOK AHEAD ROUTINE
6367
6368 ;CALL
6369
6370 MOV #DRVNUM,R1 ;DRIVE NUMBER
6371 MOV #DPB,R2 ;POINT TO DPB
6372 JSR RO,LA ;GO CHECK THE WINDOW
6373 RETURN1 ;ERROR RETURN
6374 RETURN2 ;START A SEARCH
6375 RETURN3 ;START A DATA TRANSFER
6376
6376 035774 013704 033250 LA: MOV RPADR,R4 ;GET RPCS1'S ADDRESS
6377 036000 010164 000010 MOV R1,RPCS2(R4) ;SELECT DRIVE
6378 036004 004037 040324 JSR RO,RO.RP ;READ CURRENT CYLINDER
6379 036010 000036 RPCC
6380 036012 036124 4$ ;ERROR RETURN ADDRESS
6381 036014 022662 000012 CMP (SP)+,12(R2) ;IS CURRENT CYLINDER=DESIRED
6382 ;CYLINDER?
6383 036020 001037 BNE 3$ ;EXIT IF NO
6384 036022 105261 033200 INCB LACNT(R1) ;INCREMENT THE LOOK AHEAD COUNT
6385 036026 126137 033200 033256 CMPB LACNT(R1),MXLACT ;EXCEED MAX?
6386 036034 003026 BGT 2$ ;BRANCH IF YES
6387 036036 116203 000010 MOVB 10(R2),R3 ;GET DESIRED SECTOR ADDRESS AND
6388 036042 000303 SWAB R3 ;MULT. BY 64--ALIGN WITH
6389 036044 006203 ASR R3 ;LOOK AHEAD REGISTER
6390 036046 006203 ASR R3
6391 036050 012737 000340 177776 MOV #340,3#PS ;PRIORITY LEVEL "7"
6392 036056 004037 040324 JSR RO,RO.RP ;READ LOOK AHEAD REGISTER
6393 036062 000020 RPLA
6394 036064 036124 4$
6395 036066 162603 SUB (SP)+,R3 ;CALCULATE THE DELTA
6396 036070 002002 BGE 1$
6397 036072 062703 002600 ADD #(<22.*64.>),R3 ;MAKE THE DELTA POSITIVE
6398 036076 023703 033260 1$: CMP MXDLTA,R3 ;CHECK THE DELTA TO SEE
6399 036102 002406 BLT 3$ ;IF IT IS WITHIN THE
6400 036104 023703 033262 CMP MNDLTA,R3 ;WINDOW---IF YES, ZERO
6401 036110 002003 BGE 3$ ;THE LOOK AHEAD COUNT
6402 036112 105061 033200 2$: CLRB LACNT(R1) ;AND TAKE THE I/O EXIT
6403 036116 005720 TST (RO)+
6404 036120 005720 3$: TST (RO)+ ;ADJUST THE RETURN ADDRESS
6405 036122 000402 BR 5$ ;EXIT
6406 036124 004737 035440 4$: JSR PC,C17 ;PROCESS THE ERROR
6407 036130 000200 5$: RTS RO ;RETURN
6408
6409 ;INTERRUPT SERVICE ROUTINE
6410
6411 036132 112737 000001 033166 ISR: MOVB #1,ACTDRV ;SET "ACTIVE DRIVER" FLAG
6412 036140 104412 SAVREG ;SAVE R0 - R5
6413 036142 013704 033250 MOV RPADR,R4 ;ADDRESS OF RHSCS1
6414 036146 013701 033234 MOV DTUW,R1 ;GET "DATA TRANSFER UNDERWAY" INDICATOR
6415 036152 002403 BLT 1$ ;BRANCH IF NO DATA TRANSFER UNDERWAY
6416 036154 004737 036176 JSR PC,TD ;CALL TRANSFER DONE
    
```



```

6417 036160 000402          BR      2$          ;EXIT
6418 036162 004737 036460 1$:      JSR      PC,SC    ;CALL SPECIAL CONDITIONS
6419 036166 104413          2$:      RESREG          ;RESTORE R0 - R5
6420 036170 105037 033166          CLR      ACTDRV     ;CLEAR "ACTIVE DRIVER" FLAG
6421 036174 000002          RTI              ;RETURN
6422
6423          ;TRANSFER DONE ROUTINE
6424
6425 036176 105061 033112  TD:      CLR      DRVACT(R1) ;SET DRIVE ACTIVE INDICATOR TO IDLE
6426 036202 012737 177777 033234      MOV      #-1,DTUW    ;NO DATA TRANSFERS UNDERWAY
6427 036210 006301          ASL      R1
6428 036212 012761 177777 033214      MOV      #-1,TIMER(R1) ;CANCEL TIMEOUT
6429 036220 006201          ASR      R1
6430 036222 013702 033162          MOV      TRANSW,R2    ;GET "DPB" ADDRESS FROM THE
6431 036226 005037 033162          CLR      TRANSW      ;TRANSFER WAIT QUEUE--CLEAR QUEUE
6432 036232 052762 000200 000016      BIS      #BIT07,16(R2) ;SET DONE
6433 036240 010164 000010          MOV      R1,RPCS2(R4) ;SELECT THE DRIVE
6434 036244 004037 040324          JSR      R0,RD.RP     ;TRANSFER ERROR(TRE=1)?
6435 036250 000000          RPCS1
6436 036252 035440          CI7
6437 036254 006126          ROL      (SP)+
6438 036256 100421          BMI      3$          ;BR IF YES
6439 036260 005737 033210          TST      SAVEFG      ;SAVE THE RH11/RPO4/5/6 REGISTERS?
6440 036264 100002          BPL      1$          ;BRANCH IF NO
6441 036266 004737 040666          JSR      PC,SVRH11    ;YES--SAVE THE REGISTERS
6442 036272 004737 036352 1$:      JSR      PC,WC        ;SEE IF WRITE CHECK TO BE PUT IN QUEUE
6443 036276 004737 041522          JSR      PC,GETREQ    ;GET DPB POINTER
6444 036302 005702          TST      R2          ;ENTRY FOR DRIVE ?
6445 036304 001403          BEQ      2$          ;BR IF NOT
6446 036306 004737 034330          JSR      PC,OPT       ;CALL OPTIMIZER
6447 036312 000462          BR       SC          ;CHECK OTHER DRIVES
6448 036314 012714 000113 2$:      MOV      #113,(R4)   ;RELEASE THE DRIVE
6449 036320 000457          BR       SC          ;CHECK FOR OTHER DRIVES
6450 036322 052762 100100 000016 3$:      BIS      #BIT15!BIT06,16(R2) ;SET DATA ERROR FLAG
6451 036330 004737 041426          JSR      PC,EMPTYQ    ;EMPTY THE "DRIVE'S WAIT" QUEUE
6452 036334 004737 040666          JSR      PC,SVRH11    ;SAVE THE RH11/RPO4/5/6 REGISTERS
6453 036340 012714 040111          MOV      #40111,(R4) ;ISSUE A "DRIVE CLEAR"
6454 036344 012714 000113          MOV      #113,(R4)   ;ISSUE A RELEASE TO THE DRIVE
6455 036350 000443          BR       SC          ;CHECK FOR OTHER DRIVES
6456
6457          ;FORCED WRITE CHECK ROUTINE
6458
6459 036352 005737 001404  WC:      TST      AUTOCK     ;AUTOMATIC WRITE CHECKS ?
6460 036356 001437          BEQ      2$          ;BR IF NOT
6461 036360 122762 000002 000024      CMPB    #2,$CODE(R2) ;LAST OPERATION WRITE DATA ?
6462 036366 001404          BEQ      1$          ;BR IF IT WAS
6463 036370 122762 000003 000024      CMPB    #3,$CODE(R2) ;LAST OPERATION WRITE HEADER & DATA ?
6464 036376 001027          BNE      2$          ;BR IF NOT
6465 036400 004037 041446 1$:      JSR      R0,DRVQUE   ;PUT THE OPERATION IN THE QUEUE
6466 036404 000424          BR       2$          ;QUEUE IS FULL
6467 036406 005062 000016          CLR      16(R2)      ;CLEAR "DONE" BIT IN DPB
6468 036412 116262 000234 000027      MOV      $RPCS1(R2),$PREV0(R2) ;SAVE WRITE OPERATION CODE
6469 036420 016262 000012 000034      MOV      $CYL(R2),$PREVA+2(R2) ;SAVE CYLINDER
6470 036426 016262 000010 000032      MOV      $SEC(R2),$PREVA(R2) ;SAVE SECTOR AND TRACK ADDRESSES

```



```

6471 036434 142762 000002 000024 BICB #2,$CODE(R2) ;CHANGE WRITE TO CHECK
6472 036442 142762 000020 000002 BICB #20,$COMND(R2) ;CHANGE DRIVER CODE TO WRITE CHECK
6473 036450 152762 000010 000002 BISB #10,$COMND(R2) ;FINISH CHANGING CODE TO WRITE CHECK
6474 036456 000207 2$: RTS PC ;EXIT
6475
6476 ;SPECIAL CONDITION ROUTINE
6477
6478 036460 116403 000016 SC: MOVB RPAS(R4),R3 ;READ "RPAS"
6479 036464 001012 BNE 2$ ;BRANCH IF ANY 'ATA' BITS SET
6480 036466 004037 040324 JSR RD,RD.RP ;READ CONTROL AND STATUS REGISTER
6481 036472 000000 RPCS1
6482 036474 035540 CIB
6483 036476 106126 ROLB (SP)+ ;IS "IE"=1?
6484 036500 100403 BMI 1$ ;YES, NO DRIVES TO CHECK
6485 036502 104001 ERROR 1 ;REPORT AN ILLEGAL INTERRUPT
6486 036504 004737 041004 JSR PC,SET.IE ;SET INTERRUPT ENABLE
6487 036510 000207 1$: RTS PC ;RETURN
6488 036512 005046 2$: CLR -(SP) ;PROCESS ALL DRIVES THAT HAVE
6489 036514 110316 MOVB R3,(SP) ;AN "ATA"=1
6490 036516 012703 000001 MOV #1,R3
6491 036522 005001 CLR R1
6492 036524 030316 SC3: BIT R3,(SP) ;ATA=1?
6493 036526 001005 BNE SC5 ;YES--BRANCH
6494 036530 005201 SC4: INC R1 ;MOVE TO THE NEXT DRIVE
6495 036532 106303 ASLB R3
6496 036534 001373 BNE SC3 ;BRANCH IF MORE TO CHECK?
6497 036536 005726 TST (SP)+ ;CLEAN OFF THE STACK
6498 036540 000207 RTS PC ;RETURN TO USER
6499 036542 105761 033142 SC5: TSTB DPINT(R1) ;INITIALIZING THE DRIVE ?
6500 036546 001402 BEQ 1$ ;BR IF NOT
6501 036550 000137 037436 JMP SC13 ;PROCESS THE DRIVE
6502 036554 105761 033152 1$: TSTB DPRQS(R1) ;PORT REQUEST OUTSTANDING ?
6503 036560 001402 BEQ 2$ ;BR IF NOT
6504 036562 000137 037436 JMP SC13 ;START THE OUTSTANDING COMMAND
6505 036566 105761 033122 2$: TSTB DRVSTA(R1) ;CHECK THE DRIVE STATUS
6506 036572 003025 BGT 5$ ;BRANCH IF ONLINE
6507 036574 105761 033170 TSTB ULDFLG(R1) ;UNLOAD IN PROGRESS?
6508 036600 003422 BLE 5$ ;BRANCH IF NOT
6509 036602 004737 041522 JSR PC,GETREQ ;GET DPB POINTER
6510 036606 004737 040566 JSR PC,SVFH11 ;SAVE THE RH11/RPO4/5/6 REGISTERS
6511 036612 004737 037366 JSR PC,SC12 ;SAVE RPDS1, RPER1, RPER2, AND RPER3
6512 ;ALSO DO A DRIVE INIT (DRVINT)
6513 036616 105761 033122 TSTB DRVSTA(R1) ;DID DRIVE COME ONLINE?
6514 036622 003416 BLE 6$ ;NO---BRANCH
6515 036624 032737 040000 033102 BIT #BIT14,RPERRS ;WAS THERE AN ERROR?
6516 036632 001002 BNE 3$ ;BR IF ERROR
6517 036634 000137 037276 JMP SC11 ;NO ERROR
6518 036640 013705 033104 3$: MOV RPERRS+2,R5 ;YES -- PICKUP RPER1 AND
6519 036644 000476 BR SC6A ;GO PROCESS THE ERROR
6520 036646 105761 033112 5$: TSTB DRVACT(R1) ;DRIVE ACTIVE WITH COMMAND OR ERROR RECOVERY ?
6521 036652 001027 BNE SC6 ;BR IF EITHER
6522 036654 004737 037366 JSR PC,SC12 ;SAVE RPDS1, RPER1, RPER2, AND RPER3
6523 ;ALSO DO A DRVINT
6524 036660 105761 033142 6$: TSTB DPINT(R1) ;TRYING TO INIT THE DRIVE ?
    
```

6525	036664	001321			BNE	SC4		;BR IF YES, CHECK ON MORE DRIVES
6526	036666	105761	033122		TSTB	DRVSTA(R1)		;CHECK ON DRIVE'S STATUS
6527	036672	100412			BMI	7\$;BR IF UNSAFE
6528	036674	032737	020000	033110	BIT	#BIT13,RPERRS+6		;ADDRESS PLUG CHANGED ?
6529	036702	001011			BNE	8\$;BR IF YES
6530	036704	012746	000113		MOV	#113,-(SP)		;RELEASE COMMAND
6531	036710	004037	040500		JSR	RD,WRT.RP		;WRITE THE COMMAND INTO RPCS1
6532	036714	000000			RPCS1			;REGISTER INDEX
6533	036716	037246			SC8			;PARITY EXIT ADDRESS
6534	036720	011605		7\$:	MOV	(SP),R5		;PICKUP (RPAS) BEFORE THE ERROR CALL
6535	036722	104002			ERROR	2		;REPORT THE UNEXPECTED ATTENTION
6536	036724	000701			BR	SC4		;GO CHECK FOR MORE ATA'S
6537	036726			8\$:				
6538	036726	104005			ERROR	5		;REPORT THE ADDRESS PLUG CHANGE
6539	036730	000677			BR	SC4		;CHECK FOR MORE DRIVES
6540	036732	006301			ASL	R1		;SETUP TO ADDRESS WORDS
6541	036734	012761	177777	033214	MOV	#-1,TIMER(R1)		;STOP THE TIMER
6542	036742	006201			ASR	R1		;RESTORE THE DRIVE ADDRESS
6543	036744	004737	041522		JSR	PC,GETREQ		;GET THE DPB POINTER FROM THE QUEUE
6544	036750	010164	000010		MOV	R1,RPCS2(R4)		;SELECT DRIVE
6545	036754	004037	040324		JSR	RD,RD.RP		;READ THE RPO4'S STATUS REG.
6546	036760	000012			RPDS1			
6547	036762	037246			SC8			
6548	036764	011605			MOV	(SP),R5		;AND PUT IT IN R5
6549	036766	006126			ROL	(SP)+		;WAS THERE AN ERROR?
6550	036770	100407			BMI	1\$;BR IF ERROR
6551	036772	105761	033112		TSTB	DRVACT(R1)		;CHECK DRIVE'S STATE
6552	036774	003137			BGT	SC11		;BR IF DRIVE ACTIVE WITH ORDER
6553	037000	052762	100210	000016	BIS	#BIT15!BIT07!BIT03,16(R2)		;INFORM USER OF ERROR RECOVER COMPLETION
6554	037006	000470			BR	SC7		
6555	037010	004037	040324		JSR	RD,RD.RP		;READ ERROR REGISTER #1
6556	037014	000014		1\$:	RPERR1			
6557	037016	037246			SC8			
6558	037020	012605			MOV	(SP)+,R5		;AND SAVE IT IN R5
6559	037022	004737	040666		JSR	PC,SVRH11		;SAVE RH11/RPO4/5/6 REGISTERS
6560	037026	012746	000111		MOV	#111,-(SP)		;ISSUE A DRIVE CLEAR
6561	037032	004037	040500		JSR	RD,WRT.RP		
6562	037036	000000			RPCS1			
6563	037040	037246			SC8			
6564	037042	006105		SC6A:	ROL	R5		;WAS "UNSAFE" CONDITION =1?
6565	037044	100406			BMI	1\$;BRANCH IF YES
6566	037046	005702			TST	R2		;ANYTHING IN QUEUE ?
6567	037050	001447			BEQ	SC7		;BR IF NOT
6568	037052	052762	100240	000016	BIS	#BIT15!BIT07!BIT05,16(R2)		;INFORM USER OF ERROR
6569	037060	000443			BR	SC7		
6570	037062	004037	040324		JSR	RD,RD.RP		;READ DRIVE STATUS REG. #1
6571	037066	000012		1\$:	RPDS1			
6572	037070	037246			SC8			
6573	037072	011605			MOV	(SP),R5		;SAVE RPDS1 IN R5
6574	037074	006126			ROL	(SP)+		; "ERR"=1?
6575	037076	100011			BPL	2\$;BR IF NO--UNSAFE CLEARED
6576	037100	112761	177777	033122	MOVB	#-1,DRVSTA(R1)		;DRIVE IS UNSAFE
6577	037106	004737	040666		JSR	PC,SVRH11		;SAVE RH11/RPO4/5/6 REGISTERS
6578	037112	052762	110000	000016	BIS	#BIT15!BIT12,16(R2)		;INFORM USER OF UNSAFE ERROR

6579	037120	000423			BR	SC7		
6580	037122	032705	010000	2\$:	BIT	#BIT12,R5	;"MOL" = 1 ?	
6581	037126	001015			BNE	3\$;BR IF YES	
6582	037130	112761	177777	033112	MOVB	#-1,DRVACT(R1)	;ACTIVE ERROR RECOVER	
6583	037136	112761	000001	033122	MOVB	#1,DRVSTA(R1)	;ONLINE	
6584	037144	006301			ASL	R1		
6585	037146	012761	072460	033214	MOV	#30000.,TIMER(R1)	;START 30 SECOND TIMER	
6586	037154	006201			ASR	R1		
6587	037156	000137	036530		JMP	SC4		
6588	037162	052762	100220	000016	3\$:	BIS	#BIT15!BIT07!BIT04,16(R2)	;INFORM USER OF ERROR
6589	037170	105061	033112	SC7:	CLRB	DRVACT(R1)	;DRIVE IS IDLE	
6590	037174	004737	041426		JSR	PC,EMPTYQ	;DUMP THE QUEUE	
6591	037200	105761	033170		TSTB	ULDFLG(R1)	;UNLOAD IN PROGRESS OR QUEUE?	
6592	037204	003002			BGT	1\$;BR IF NOT	
6593	037206	105061	033170		CLRB	ULDFLG(R1)	;CLEAR UNLOAD FLAG	
6594	037212	116164	033236	000016	1\$:	MOVB	ATABIT(R1),RPAS(R4)	;CLEAR ATTENTION BIT
6595	037220	105761	033122		TSTB	DRVSTA(R1)	;IS THE DRIVE UNSAFE ?	
6596	037224	100406			BMI	2\$;BR IF IT IS	
6597	037226	012746	000113		MOV	#113,-(SP)	;RELEASE COMMAND	
6598	037232	004037	040500		JSR	RD,WRT.RP	;WRITE THE COMMAND INTO RPCS1	
6599	037236	000000			RPCS1		;REGISTER INDEX	
6600	037240	037246			SC8		;PARITY EXIT ADDRESS	
6601	037242	000137	036530	2\$:	JMP	SC4	;CHECK FOR MORE DRIVES	
6602	037246	105761	033112	SC8:	TSTB	DRVACT(R1)	;IS DRIVE IDLE?	
6603	037252	001405			BEQ	1\$;YES--BRANCH	
6604	037254	004737	041522		JSR	PC,GETREQ	;GET DPB POINTER	
6605	037260	004737	035440		JSR	PC,CI7	;PROCESS THE PARITY ERROR	
6606	037264	000402			BR	2\$;CONTINUE	
6607	037266	004737	035466	1\$:	JSR	PC,CI7B	;PROCESS THE UNCORRECTABLE PARITY ERROR	
6608	037272	000137	036530	2\$:	JMP	SC4	;CHECK MORE DRIVES	
6609	037276	105761	033170	SC11:	TSTB	ULDFLG(R1)	;"UNLOAD IN PROGRESS"?	
6610	037302	003402			BLE	1\$;BRANCH IF NO	
6611	037304	105061	033170		CLRB	ULDFLG(R1)	;CLEAR UNLOAD FLAG	
6612	037310	105061	033112	1\$:	CLRB	DRVACT(R1)	;SET DRIVE IDLE	
6613	037314	136137	033236	033164	BITB	ATABIT(R1),SRCHWT	;DOING A SEARCH OPERATION FOR	
6614							;AN I/O COMMAND?	
6615	037322	001012			BNE	2\$;BRANCH IF YES	
6616	037324	004737	041544		JSR	PC,POPQUE	;REMOVE REQUEST FROM QUEUE	
6617	037330	052762	000200	000016	BIS	#BIT07,16(R2)	;SET "DONE" BIT	
6618	037336	005737	033210		TST	SAVEFG	;SAVE THE REGISTERS?	
6619	037342	100002			BPL	2\$;BRANCH IF NO	
6620	037344	004737	040666		JSR	PC,SVRH11	;YES--SAVE ALL OF THE RH11/RPO4/5/6 REG'S	
6621	037350	116164	033236	000016	2\$:	MOVB	ATABIT(R1),RPAS(R4)	;CLEAR ATTENTION BIT
6622	037356	004737	034330		JSR	PC,OPT	;START A REQUEST	
6623	037362	000137	036530		JMP	SC4	;CHECK FOR MORE DRIVES	
6624	037366	010164	000010	SC12:	MOV	R1,RPCS2(R4)	;SELECT DRIVE	
6625	037372	016437	000012	033102	MOV	RPDS1(R4),RPERRS	;SAVE THE FOUR REGISTERS THAT	
6626	037400	016437	000014	033104	MOV	RPER1(R4),RPERRS+2	;WILL TELL US SOMETHING	
6627	037406	016437	000040	033106	MOV	RPER2(R4),RPERRS+4		
6628	037414	016437	000042	033110	MOV	RPER3(R4),RPERRS+6		
6629	037422	004037	033500		JSR	RD,DRVINT	;INIT. THE STATE OF THE DRIVE	
6630	037426	000401			BR	1\$;TAKE ERROR EXIT	
6631	037430	000207			RTS	PC	;RETURN	
6632	037432	005726		1\$:	TST	(SP)+	;POP PC OFF OF THE STACK	

```

6633 037434 000704          BR      SC8      ;PROCESS THE PARITY ERROR
6634 037436 006301          ASL     R1      ;SETUP TO ADDRESS WORDS
6635 037440 012761 177777 033214 SC13:  MOV     #-1,TIMER(R1) ;STOP THE TIMER
6636 037446 006201          ASR     R1      ;
6637 037450 010164 000010        MOV     R1,RPCS2(R4) ;SELECT THE DRIVE
6638 037454 116164 033236 000016        MOVVB  ATABIT(R1),RPAS(R4) ;CLEAR THE ATTENTION BIT
6639 037462 032714 004000        BIT     #BIT11,(R4) ;DRIVE AVAILABLE ?
6640 037466 001006          BNE     1$      ;BR IF AVAILABLE
6641 037470 006301          ASL     R1      ;
6642 037472 012761 023420 033214        MOV     #10000.,TIMER(R1) ;START 10 SEC TIMER AGAIN
6643 037500 006201          ASR     R1      ;
6644 037502 000433          BR      3$      ;EXIT
6645 037504 105761 033142          1$:    TSTB   DPINT(R1) ;INITIALIZING THE DRIVE ?
6646 037510 001424          BEQ     2$      ;BR IF NOT
6647 037512 105061 033142          CLRB   DPINT(R1) ;CLEAR THE INIT INDICATOR
6648 037516 004037 033500          JSR    RO,DRVINT ;GO INIT THE DRIVE
6649 037522 000240          NOP     ;DUMMY PARITY ERROR RETURN
6650 037524 105761 033122          TSTB   DRVSTA(R1) ;DRIVE ONLINE ?
6651 037530 003014          BGT     2$      ;BR IF YES -- START ORDER
6652 037532 005702          TST    R2      ;QUEUE ENTRY FOR THE DRIVE
6653 037534 001416          BEQ     3$      ;BR IF NOT
6654 037536 004737 041522          JSR    PC,GETREQ ;GET DPB ADDRESS
6655 037542 052762 140000 000016        BIS    #BIT15:BIT14,16(R2) ;INFORM USER THAT DRIVE OFFLINE
6656 037550 004737 040666          JSR    PC,SVRH11 ;SAVE THE REGISTERS
6657 037554 004737 041426          JSR    PC,EMPTYQ ;EMPTY THE REQUEST QUEUE
6658 037560 000404          BR      3$      ;
6659 037562 105061 033152          2$:    CLRB   DPRQS(R1) ;CLEAR THE PORT REQUEST INDICATOR
6660 037566 004737 034330          JSR    PC,OPT   ;START THE PENDING REQUEST
6661 037572 000137 036530          3$:    JMP    SC4   ;PROCESS OTHER DRIVES
6662
6663          ;RPO4/5/6 TIMER ROUTINE
6664          ;CALL
6665          ;
6666          ;    MOV     #TIME,-(SP) ;ELAPSED TIME IN MILLISECONDS ON THE STACK
6667          ;    JSR    PC,RPTMR ;CALL RPO4/5/6 TIME ROUTINE
6668 037576 005737 033166          RPTMR: TST     ACTDRV ;CHECK "ACTDRV & ACTSTR"
6669 037602 001030          BNE     4$      ;IF NON ZERO EXIT
6670 037604 112737 000001 033167        MOVVB  #1,ACTSTR ;SET "ACTSTR"
6671 037612 104412          SAVREG ;SAVE R0 - R5
6672 037614 005001          CLR     R1      ;START WITH DRIVE 0
6673 037616 005003          CLR     R3      ;
6674 037620 005763 033214          1$:    TST     TIMER(R3) ;IS THE TIMER RUNNING?
6675 037624 002407          BLT     2$      ;BRANCH IF NO
6676 037626 166663 000002 033214        SUB     2(SP),TIMER(R3) ;COUNT THE INTERVAL
6677 037634 003003          BGT     2$      ;BR IF NO SOFTWARE TIMEOUT
6678 037636 004737 037670          JSR    PC,STC   ;CALL SOFTWARE TIMEOUT ROUTINE
6679 037642 000405          BR      3$      ;GO TO THE EXIT
6680 037644 005201          2$:    INC     R1      ;MOVE TO NEXT DRIVE
6681 037646 005723          TST    (R3)+   ;
6682 037650 022701 000010        CMP     #8.,R1  ;OUT OF DRIVES?
6683 037654 003361          BGT     1$      ;BRANCH IF NO
6684 037656 104413          3$:    RESREG ;RESTORE R0 - R5
6685 037660 105037 033167        CLRB   ACTSTR  ;ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
6686 037664 012616          4$:    MOV     (SP)+,(SP) ;ADJUST THE STACK
    
```



```

6687 037666 000207          RTS      PC          ;RETURN
6688
6689          ;SOFTWARE TIMEOUT ROUTINE
6690
6691          ;NOTE: THIS ROUTINE MUST BE ENTERED AT PRIORITY 6
6692          ;OR GREATER
6693
6694          ;CALL:  STO      #DRVNUM,R1      ;DRIVE NUMBER
6695          ;MOV      PC,STO      ;CALL
6696
6697
6698
6699 037670 010146          STO:    MOV      R1,-(SP)      ;SAVE R1
6700 037672 010346          MOV      R3,-(SP)      ;SAVE R3
6701 037674 013704 033250          MOV      RPADR,R4      ;GET ADDRESS OF "RPCS1"
6702 037700 010164 000010          MOV      R1,RPCS2(R4)  ;SELECT THE DRIVE
6703 037704 004037 040324          JSR      RO,RD.RP      ;READ "DRIVE STATUS REG"
6704 037710 000012          RPDS1
6705 037712 040212          STOS
6706 037714 105726          TSTB    (SP)+          ;IS "DRY"=1?
6707 037716 100477          BMI     ST02          ;BR IF YES
6708 037720 105761 033142          ST01:  TSTB    DPINT(R1)  ;TRYING TO INITIALIZE THE DRIVE ?
6709 037724 001074          BNE     ST02          ;BR IF YES
6710 037726 105761 033152          TSTB    DPRQS(R1)     ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
6711 037732 001071          BNE     ST02          ;BR IF YES
6712 037734 013702 033162          MOV      TRNSWT,R2    ;PICKUP TRANSFER WAIT QUEUE
6713 037740 020137 033234          CMP      R1,DTUW      ;TRANSFER UNDERWAY ON THIS DRIVE?
6714 037744 001402          BEQ     1$            ;BRANCH IF YES
6715 037746 004737 041522          JSR      PC,GETREQ    ;GET DPB ADDRESS
6716 037752 052762 101000 000016 1$:  BIS      #BIT15:BIT09,16(R2) ;SET THE ERROR FLAGS
6717 037760 004737 040666          JSR      PC,SVRH11    ;SAVE RH11/RPO4/5/6 REGISTERS
6718 037764 012764 000040 000010          MOV      #BIT05,RPCS2(R4) ;"INIT" THE MASS BUS
6719 037772 105061 033112          CLRB    DRVACT(R1)   ;DRIVE IS IDLE
6720 037776 105061 033170          CLRB    ULDFLG(R1)   ;CLEAR THE UNLOAD FLAG
6721 040002 005001          CLR     R1            ;START WITH DRIVE 0
6722 040004 005003          CLR     R3
6723 040006 004037 033500          2$:    JSR      RO,DRVINT   ;INIT. THIS DRIVE
6724 040012 000477          BR      ST05          ;PARITY ERROR RETURN
6725 040014 105761 033112          TSTB    DRVACT(R1)   ;DRIVE IDLE BEFORE THE INIT.?
6726 040020 001414          BEQ     4$            ;YES--BRANCH
6727 040022 013702 033162          MOV      TRNSWT,R2    ;GET TRANSFER WAIT QUEUE
6728 040026 023701 033234          CMP      DTUW,R1     ;WAS THERE I/O ON THIS DRIVE?
6729 040032 001402          BEQ     3$            ;YES--BRANCH
6730 040034 004737 041522          JSR      PC,GETREQ    ;GET THE DPB POINTER FROM QUEUE
6731 040040 052762 100400 000016 3$:  BIS      #BIT15:BIT08,16(R2) ;INFORM USER OF INIT.
6732 040046 105061 033112          CLRB    DRVACT(R1)   ;SET DRIVE ACTIVE TO IDLE
6733 040052 105061 033170          CLRB    ULDFLG(R1)   ;NO UNLOAD
6734 040056 012763 177777 033214          MOV      #-1,TIMER(R3) ;STOP THE TIMER
6735 040064 005723          TST     (R3)+        ;UPDATE THE INDEX
6736 040066 005201          INC     R1            ;INCREMENT THE DRIVE NUMBER
6737 040070 022701 000010          CMP      #8.,R1      ;LAST DRIVE BEEN CHECKED?
6738 040074 003344          BGT     2$           ;NO--LOOP
6739 040076 012737 177777 033234          MOV      #-1,DTUW    ;NO DATA TRANSFERS UNDERWAY
6740 040104 005037 033162          CLR     TRNSWT       ;CLEAR TRANSFER WAIT QUEUE

```

```

6741 040110 004737 041350 JSR PC,CLRQUE ;CLEAR ALL REQUEST QUEUES
6742 040114 000500 BR ST09 ;EXIT
6743 040116 116405 000016 ST02: MOVB RPAS(R4),R5 ;READ ATTENTION REG
6744 040122 136105 033236 BITB ATABIT(R1),R5 ;IS ATTENTION FOR THIS DRIVE UP ?
6745 040126 001017 BNE ST03 ;YES--BRANCH
6746 040130 105761 033142 TSTB DPINT(R1) ;TRYING TO INITIALIZE THE DRIVE ?
6747 040134 001031 BNE ST06 ;BR IF YES - DRIVE NOT ONLINE
6748 040136 105761 033152 TSTB DPRQS(R1) ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
6749 040142 001045 BNE ST07 ;BR IF YES - NO RESPONSE TO REQUEST
6750 040144 020137 033234 CMP R1,DTUW ;DATA TRANSFER UNDERWAY FOR THIS DRIVE
6751 040150 001263 BNE ST01 ;BR IF NO
6752 040152 004037 040324 JSR RD,RD.RP ;YES--CHECK "RDY"
6753 040156 000000 RPCS1
6754 040160 040212 ST05
6755 040162 105726 TSTB (SP)+
6756 040164 100255 BPL ST01 ;BR IF "RDY"=0
6757 040166 105761 033142 ST03: TSTB DPINT(R1) ;INITIALIZING THE DRIVE ?
6758 040172 001003 BNE IS ;BR IF INIT PENDING
6759 040174 105761 033152 TSTB DPRQS(R1) ;PORT REQUEST PENDING ?
6760 040200 001446 BEQ ST09 ;BR IF NOT
6761 040202 012763 177777 033214 IS: MOV #-1,TIMER(R3) ;STOP THE TIMER
6762 040210 000442 BR ST09 ;EXIT
6763 040212 004737 035540 ST05: JSR PC,CIB ;GO HANDLE THE PARITY ERROR
6764 040216 000437 BR ST09
6765 040220 105061 033142 ST06: CLRB DPINT(R1) ;CLEAR THE INITIALIZE INDICATOR
6766 040224 105061 033122 CLRB DRVSTA(R1) ;SET UNIT OFFLINE
6767 040230 012763 177777 033214 MOV #-1,TIMER(R3) ;STOP THE TIMER
6768 040236 004737 041522 JSR PC,GETREQ ;GET THE DPB ADDRESS
6769 040242 005702 TST R2 ;REQUEST IN QUEUE ?
6770 040244 001424 BEQ ST09 ;BR IF NOT
6771 040246 052762 140000 000016 BIS #BIT15!BIT14,16(R2) ;INFORM THE USER DRIVE NOT AVAILABLE
6772 040254 000414 BR ST08 ;FINISH
6773 040256 012763 177777 033214 ST07: MOV #-1,TIMER(R3) ;STOP THE TIMER
6774 040264 105061 033152 CLRB DPRQS(R1) ;CLEAR PORT REQUEST INDICATOR
6775 040270 004737 041522 JSR PC,GETREQ ;GET DPB ADDRESS
6776 040274 005702 TST R2 ;QUEUE ENTRY FOR DRIVE ?
6777 040276 001407 BEQ ST09 ;BR IF NONE
6778 040300 012762 100004 000016 MOV #BIT15!BIT2,16(R2) ;INFORM USER OF PORT REQUEST ERROR
6779 040306 004737 041426 ST08: JSR PC,EMPTYQ ;CLEAR THE QUEUE FOR THE DRIVE
6780 040312 004737 040666 JSR PC,SVRH11 ;SAVE THE REGISTERS
6781 040316 012603 ST09: MOV (SP)+,R3 ;RESTORE R3
6782 040320 012601 MOV (SP)+,R1 ;RESTORE R1
6783 040322 000207 RTS PC ;RETURN
6784
6785 ;ROUTINE TO READ A RH11/RPO4/5/6 REGISTER
6786
6787 ;CALL
6788 JSR RD,RD.RP ;GO READ A REGISTER
6789 INDEX ;REG. INDEX FROM BASE
6790 ERRADR ;ERROR ADDRESS--PROCESS ERROR STARTING
6791 ;AT THIS ADDRESS
6792 RETURN ;CONTENTS OF REG. IS ON THE STACK
6793
6794 040324 013737 033246 040466 RD.RP: MOV MCPMX,RD.RP2 ;MAX. RETRYS ALLOWED
    
```



```

6795 040332 011646          MOV      (SP), -(SP)      ;SAVE R0 FOR RETURN
6796 040334 013737 033250 040350  MOV      RPADR, RD.ADR   ;FORM THE DESIRED ADDRESS
6797 040342 062037 040350          ADD      (R0)+, RD.ADR  ;USING THE BASE AND THE INDEX
6798 040346 013727          RD.RP1: MOV     2(PC)+, (PC)+ ;READ THE DESIRED REGISTER OF THE RPO4
6799 040350 000000          RD.ADR: .WORD 0         ;ADDRESS IS FORMED HERE
6800 040352 000000          RD.WRD: .WORD 0         ;REG. CONTENTS PUT HERE
6801 040354 013766 040352 000002  MOV      RD.WRD, 2(SP)   ;RETURN IT TO THE USER
6802 040362 013746 033250          MOV      RPADR, -(SP)   ;PUT THE ADDRESS ON THE STACK
6803 040366 062716 000010          ADD      #RPCS2, (SP)   ;FORM THE ADDRESS OF RPCS2
6804 040372 032736 010000          BIT      #BIT12, 2(SP)+ ;CHECK THE 'NED' BIT
6805 040376 001035          BNE      RD.RP3         ;BR IF DRIVE NON-EXISTENT
6806 040400 017746 172644          MOV      2RPADR, -(SP)  ;READ RPCS1
6807 040404 032716 020000          BIT      #BIT13, (SP)   ;DID MCPE SET?
6808 040410 001002          BNE      1$            ;BRANCH IF YES
6809 040412 022620          CMP      (SP)+, (R0)+   ;ADJUST FOR RETURN
6810 040414 000430          BR       RD.RP4        ;EXIT
6811 040416          1$:
6812 040416 104003          ERROR   3              ;REPORT "MCPE" ERROR
6813 040420 005737 033234          TST      DTUW           ;DATA TRANSFER UNDERWAY?
6814 040424 100405          BMI     2$             ;NO--BRANCH
6815 040426 032716 040000          BIT      #BIT14, (SP)   ;NO--"TRE"=1?
6816 040432 001402          BEQ     2$             ;NO--BRANCH
6817 040434 005726          TST      (SP)+         ;YES--CLEAN OFF THE STACK AND
6818 040436 000415          BR      RD.RP3         ;TAKE THE FATAL ERROR EXIT
6819 040440 052716 040000          2$:  BIS      #BIT14, (SP) ;CLEAR "MCPE" BY SENDING A "1" TO "TRE"
6820 040444 000316          SWAB    (SP)           ;POSITION BEFORE WRITING
6821 040446 013737 033250 040462  MOV      RPADR, 3$      ;FORM ADDRESS OF HIGH BYTE
6822 040454 005237 040462          INC     3$             ;
6823 040460 112637          MOVB   (SP)+, 2(PC)+   ;WRITE THE HIGH BYTE OF RPCS1
6824 040462 000000          3$:  .WORD 0              ;ADDRESS STORAGE
6825 040464 005327          DEC     (PC)+         ;EXCEEDED MAX. RETRYS
6826 040466 000003          RD.RP2: .WORD 3         ;
6827 040470 002326          BGE     RD.RP1         ;BRANCH IF NO
6828 040472 011000          RD.RP3: MOV     (R0), R0 ;FATAL ERROR EXIT
6829 040474 012616          MOV     (SP)+, (SP)   ;
6830 040476 000200          RD.RP4: RTS      R0    ;
6831
6832          ;ROUTINE TO WRITE A REGISTER
6833          ;CALL
6834          ;
6835          ;CALL      MOV     DATA, -(SP)      ;DATA TO BE LOADED ON THE STACK
6836          ;          JSR     R0, WRT.RP        ;CALL THE ROUTINE TO LOAD(WRITE) THE REG.
6837          ;          INDEX  ;INDEX OF THE REGISTER TO BE LOADED
6838          ;          ERRADR  ;ADDRESS TO RETURN TO ON AN ERROR
6839          ;          RETURN ;ERROR FREE RETURN
6840
6841 040500 013737 033246 040650 WRT.RP: MOV     MCPEMX, WRT.R2 ;MAX RETRYS ALLOWED
6842 040506 016637 000002 040566  MOV     2(SP), WRT.WD  ;SAVE THE WORD TO WRITE
6843 040514 012616          MOV     (SP)+, (SP)   ;ADJUST THE STACK
6844 040516 012037 040570          MOV     (R0)+, WRT.AD ;GET INDEX OF REGISTER TO BE WRITTEN
6845 040522 001015          BNE     1$            ;BRANCH IF NOT RPCS1
6846 040524 122737 000150 040566  CMPB   #150, WRT.WD   ;IS THE COMMAND FOR DATA TRANSFERS?
6847 040532 002411          BLT     1$            ;YES--DON'T GET THE OLD A16 & A17, & PSEL
6848 040534 004037 040324          JSR     R0, RD.RP     ;NO---COMBINE A16&A17, & PSEL WITH
    
```

```

6849 040540 000000 RPCS1 ;THE COMMAND BEFORE SENDING IT TO
6850 040542 040656 WRT.R3 ;THE RH11/RPO4
6851 040544 000316 SWAB (SP)
6852 040546 042716 177770 BIC #1C7,(SP)
6853 040552 112637 040567 MOVB (SP)+,WRT.WD+1
6854 040556 063737 033250 040570 1$: ADD RPADR,WRT.AD ;FORM THE ADDRESS OF THE DISK REG.
6855 040564 012737 WRT.R1: MOV (PC)+,3(PC)+ ;LOAD THE DESIRED REG.
6856 040566 000000 WRT.WD: .WORD 0 ;WORD TO WRITE GOES HERE
6857 040570 000000 WRT.AD: .WORD 0 ;ADDRESS IS FORMED HERE
6858 040572 013746 033250 MOV RPADR,-(SP) ;PUT THE ADDRESS ON THE STACK
6859 040576 062716 000010 ADD #RPCS2,(SP) ;FORM THE ADDRESS OF RPCS2
6860 040602 032736 010000 BIT #BIT12,3(SP)+ ;CHECK THE 'NED' BIT
6861 040606 001023 BNE WRT.R3 ;BR IF DRIVE NON-EXISTENT
6862 040610 004037 040324 JSR RO,RO.RP ;CHECK FOR PARITY ERROR ON WRITE
6863 040614 000014 RPER1
6864 040616 040656 WRT.R3
6865 040620 032726 000010 BIT #BIT03,(SP)+
6866 040624 001416 BEQ WRT.R4 ;BRANCH IF "PAR=0"
6867 040626 016037 177776 040640 MOV -2(RO),1$ ;PICKUP THE INDEX
6868 040634 004037 040324 JSR RO,RO.RP ;READ THE REG.
6869 040640 000000 1$: .WORD 0 ;REG. INDEX
6870 040642 040656 WRT.R3 ;RETURN TO THIS ADDRESS ON ERROR
6871 040644 104004 ERROR 4 ;REPORT THE PARITY ON WRITE ERROR
6872 040646 005327 DEC (PC)+ ;DECREMENT THE ERROR COUNT
6873 040650 000003 WRT.R2: .WORD 3 ;RETRY COUNTER
6874 040652 002344 BGE WRT.R1 ;TRY AGAIN IF NOT FINISHED
6875 040654 005726 TST (SP)+ ;CLEAN OFF THE STACK
6876 040656 011000 WRT.R3: MOV (RO),RO ;TAKE THE "PARITY ON WRITE" ERROR EXIT
6877 040660 000401 BR WRT.R5 ;EXIT
6878 040662 005720 WRT.R4: TST (RO)+ ;ADJUST FOR ERROR FREE EXIT
6879 040664 000200 WRT.R5: RTS RO
6880
6881 ;ROUTINE TO SAVE THE RH11/RPO4/5/6 REGISTERS AS PER DPB+14
6882 ;CALL
6883 ;
6884 ; MOV #DPBNUM,R2 ;DPB POINTER TO R2
6885 ; JSR PC,SVRH1 ;SAVE THE DRIVES REG'S
6886
6887 040666 104412 SVRH1: SAVREG ;SAVE RO - R5
6888 040670 005702 TST R2 ;QUEUE ENTRY FOR THE DRIVE ?
6889 040672 001430 BEQ 4$ ;BR IF NONE
6890 040674 013704 033250 MOV RPADR,R4
6891 040700 111264 000010 MOVB (R2),RPCS2(R4) ;SELECT DRIVE
6892 040704 016203 000014 MOV 14(R2),R3 ;GET THE ERROR TABLE POINTER
6893 040710 001433 BEQ 6$ ;EXIT IF NO ADDRESS
6894 040712 005037 040746 CLR 3$ ;COUNTER & POINTER
6895 040716 023727 040746 000022 1$: CMP 3$,#RPDB ;REACHED THE BUFFER REGISTER ?
6896 040724 001006 BNE 2$ ;BR IF NOT
6897 040726 032764 000200 000010 BIT #BIT07,RPCS2(R4) ;'OR' SET ?
6898 040734 001002 BNE 2$ ;BR IF SET
6899 040736 005023 CLR (R3)+ ;STORE RPDB AS ZEROES
6900 040740 000405 BR 4$ ;CONTINUE
6901 040742 004037 040324 2$: JSR RO,RO.RP ;READ THE SELECTED REGISTER
6902 040746 000000 3$: .WORD 0 ;REGISTER INDEX

```



```

6903 040750 040774          5$:          ;ERROR RETURN ADDRESS
6904 040752 012623          MOV      (SP)+,(R3)+      ;STORE THE REGISTER CONTENTS
6905 040754 023727 040746 000046 4$:      CMP      3$,#RPEC2      ;REACHED THE END ?
6906 040762 001406          BEQ      6$              ;BR IF YES
6907 040764 062737 000002 040746          ADD      #2,3$          ;INCREMENT THE REGISTER INDEX
6908 040772 000751          BR       1$              ;CONTINUE READING THE REGISTERS
6909 040774 004737 035440          5$:      JSR      PC,C17      ;PROCESS THE UNCORRECTABLE PARITY ERROR
6910 041000 104413          6$:      RESREG          ;RESTORE R0 - R5
6911 041002 000207          RTS      PC              ;RETURN
6912
6913          ;ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A "TRE"
6914          ;CALL
6915          ;
6916          ;
6917          ;
6918          ;
6919 041004 010446          SET.IE: MOV      R4,-(SP)      ;SAVE R4
6920 041006 013704 033250          MOV      RPADR,R4      ;PICKUP ADDRESS OF RPCS1
6921 041012 010164 000010          MOV      R1,RPCS2(R4)  ;SELECT DRIVE
6922 041016 011446          MOV      (R4),-(SP)    ;READ RPCS1
6923 041020 052716 040000          BIS      #BIT14,(SP)   ;SET THE "TRE" BIT OF THE WORD READ
6924 041024 009316          SWAB    (SP)           ;ADJUST FOR DATO
6925 041026 112714 000100          MOVB    #BIT06,(R4)    ;SET "IE"
6926 041032 032764 010000 000010          BIT     #BIT12,RPCS2(R4) ;IS "NED"=1?
6927 041040 001002          BNE     1$             ;YES--CLEAR "TRE"
6928 041042 005726          TST     (SP)+          ;CLEAN OFF THE STACK
6929 041044 000402          BR      2$             ;
6930 041046 112664 000001          1$:     MOVB    (SP)+,1(R4) ;CLEAR "TRE"
6931 041052 012604          2$:     MOV      (SP)+,R4   ;RESTORE R4
6932 041054 000207          RTS      PC              ;RETURN TO CALLER
6933
6934          ;QUEUE COUNT
6935 041056          000          QCNT:   .BYTE   0           ;DRIVE 0
6936 041057          000          .BYTE   0           ;DRIVE 1
6937 041060          000          .BYTE   0           ;DRIVE 2
6938 041061          000          .BYTE   0           ;DRIVE 3
6939 041062          000          .BYTE   0           ;DRIVE 4
6940 041063          000          .BYTE   0           ;DRIVE 5
6941 041064          000          .BYTE   0           ;DRIVE 6
6942 041065          000          .BYTE   0           ;DRIVE 7
6943
6944          ;QUEUE INPUT POINTERS
6945
6946 041066 041150          QINPT: .WORD   QDRV0        ;DRIVE 0
6947 041070 041170          .WORD   QDRV1        ;DRIVE 1
6948 041072 041210          .WORD   QDRV2        ;DRIVE 2
6949 041074 041230          .WORD   QDRV3        ;DRIVE 3
6950 041076 041250          .WORD   QDRV4        ;DRIVE 4
6951 041100 041270          .WORD   QDRV5        ;DRIVE 5
6952 041102 041310          .WORD   QDRV6        ;DRIVE 6
6953 041104 041330          .WORD   QDRV7        ;DRIVE 7
6954
6955          ;QUEUE OUTPUT POINTERS
6956
    
```

```

6957 041106 041150          QOUTPT: .WORD  QDRV0          ;DRIVE 0
6958 041110 041170          .WORD  QDRV1          ;DRIVE 1
6959 041112 041210          .WORD  QDRV2          ;DRIVE 2
6960 041114 041230          .WORD  QDRV3          ;DRIVE 3
6961 041116 041250          .WORD  QDRV4          ;DRIVE 4
6962 041120 041270          .WORD  QDRV5          ;DRIVE 5
6963 041122 041310          .WORD  QDRV6          ;DRIVE 6
6964 041124 041330          .WORD  QDRV7          ;DRIVE 7
6965
6966 041126 041150          QSTART: .WORD  QDRV0          ;DRIVE 0 START ADDRESS
6967 041130 041170          QSTOP:  .WORD  QDRV1          ;DRIVE 0 STOP ADDRESS & DRIVE 1 START ADDRESS
6968 041132 041210          .WORD  QDRV2          ;STOP DRIVE 1--START DRIVE 2
6969 041134 041230          .WORD  QDRV3          ;STOP DRIVE 2--START DRIVE 3
6970 041136 041250          .WORD  QDRV4          ;STOP DRIVE 3--START DRIVE 4
6971 041140 041270          .WORD  QDRV5          ;STOP DRIVE 4--START DRIVE 5
6972 041142 041310          .WORD  QDRV6          ;STOP DRIVE 5--START DRIVE 6
6973 041144 041330          .WORD  QDRV7          ;STOP DRIVE 6--START DRIVE 7
6974 041146 041350          .WORD  QTERM          ;STOP DRIVE 7
6975
6976          ;DRIVE REQUEST QUEUES
6977
6978 041150 000010          QDRV0:  .BLKW  10
6979 041170 000010          QDRV1:  .BLKW  10
6980 041210 000010          QDRV2:  .BLKW  10
6981 041230 000010          QDRV3:  .BLKW  10
6982 041250 000010          QDRV4:  .BLKW  10
6983 041270 000010          QDRV5:  .BLKW  10
6984 041310 000010          QDRV6:  .BLKW  10
6985 041330 000010          QDRV7:  .BLKW  10
6986          QTERM=.
6987
6988          ;ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
6989
6990          ;CALL
6991          ;      JSR      PC,CLRQUE
6992
6993 041350 104412          CLRQUE: SAVREG          ;SAVE R0 - R5
6994 041352 012702 041056          MOV      #QCNT,R2          ;ZERO THE QUEUE COUNTS
6995 041356 005022          CLR      (R2)+          ;DRIVES 0 & 1
6996 041360 005022          CLR      (R2)+          ;DRIVES 2 & 3
6997 041362 005022          CLR      (R2)+          ;DRIVES 4 & 5
6998 041364 005022          CLR      (R2)+          ;DRIVES 6 & 7
6999 041366 012703 000010          MOV      #8,R3          ;MOVE THE STARTING
7000 041372 012701 041126          MOV      #QSTART,R1          ;ADDRESS OF THE QUEUE INTO
7001 041376 012122          1$: MOV      (R1)+,(R2)+          ;THE QUEUE INPUT POINTER
7002 041400 005303          DEC      R3
7003 041402 001375          BNE      1$
7004 041404 012703 000010          MOV      #8,R3          ;MOVE THE STARTING ADDRESS
7005 041410 012701 041126          MOV      #QSTART,R1          ;OF THE QUEUE INTO THE
7006 041414 012122          2$: MOV      (R1)+,(R2)+          ;QUEUE OUTPUT POINTER
7007 041416 005303          DEC      R3
7008 041420 001375          BNE      2$
7009 041422 104413          RESREG
7010 041424 000207          RTS      PC          ;RESTORE R0 - R5
    
```



```

7011
7012
7013
7014
7015
7016
7017
7018
7019
7020
7021
7022
7023
7024
7025
7026
7027
7028
7029
7030
7031
7032
7033
7034
7035
7036
7037
7038
7039
7040
7041
7042
7043
7044
7045
7046
7047
7048
7049
7050
7051
7052
7053
7054
7055
7056
7057
7058
7059
7060
7061
7062
7063
7064

:EMPTY THE QUEUE SPECIFIED BY R1
:CALL
:   MOV   DRVNUM,R1   ;DRIVE NUMBER TO R1
:   JSR   PC,EMPTYQ
EMPTYQ: CLR   QCNT(R1) ;CLEAR NUMBER OF ITEMS IN QUEUE
:   ASL   R1
:   MOV   QINPT(R1),QOUTPT(R1) ;SET OUTPUT QUEUE POINTER=INPUT POINTER
:   ASR   R1
:   RTS   PC

:ROUTINE TO PUT A REQUEST IN QUEUE
:CALL
:   MOV   #DRVNUM,R1 ;DRIVE NUMBER
:   MOV   #DPB,R2    ;ADDRESS OF PARAMETER BLOCK
:   JSR   RO,DRVQUE ;GO PUT REQUEST IN QUEUE
:   RETURN1 ;RETURN HERE IF QUEUE IS FULL
:   RETURN2 ;RETURN HERE IF REQUEST IS IN QUEUE

DRVQUE: CMP   #10,QCNT(R1) ;IS QUEUE FULL?
:   BEQ   2$         ;BR IF YES-TAKE RETURN1
:   INCB  QCNT(R1)   ;INCREMENT QUEUE COUNT
:   ASL   R1
:   MOV   R2,QINPT(R1) ;PUT THIS REQUEST IN QUEUE
:   ADD   #2,QINPT(R1) ;UPDATE THE QUEUE POINTER
:   CMP   QINPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER
:   BNE   1$         ;BRANCH IF NO
:   MOV   QSTART(R1),QINPT(R1) ;YES--RESET POINTER
1$:   ASR   R1
:   TST   (R0)+      ;TAKE RETURN 2
2$:   RTS   R0        ;RETURN TO USER

:ROUTINE TO GET THE "DPB" ADDRESS OF NEXT REQUEST IN QUEUE
:CALL
:   MOV   #DRVNUM,R1 ;DRIVE NUMBER TO R1
:   JSR   PC,GETREQ ;GO GET THE REQUEST
:   RETURN ;R2="DPB" ADDRESS OF THE REQUEST
:         ;R2=0 IF NO REQUEST IN QUEUE

GETREQ: CLR   R2
:   TSTB  QCNT(R1)   ;IS THERE ANY REQUEST IN QUEUE?
:   BEQ   2$         ;NO---BRANCH
1$:   ASL   R1
:   MOV   QOUTPT(R1),R2 ;PICKUP "DPB" POINTER FOR THIS DRIVE
:   ASR   R1
2$:   RTS   PC        ;RETURN TO USER

:ROUTINE TO "POP" THE REQUEST FROM QUEUE
:CALL
    
```



```

7065      :      MOV      #DRYNUM,R1      ;DRIVE NUMBER TO R
7066      :      JSR      PC,POPQUE      ;CALL TO REMOVE REQUEST
7067      :      RETURN      ;R2=ADDRESS OF DPB REMOVED
7068
7069      041544 105361 041056      POPQUE: DECB      QCNT(R1)      ;DECREMENT QUEUE COUNT
7070      041550 006301      ASL      R1
7071      041552 017102 041106      MOV      @GOUTPT(R1),R2      ;GET THE "DPB" POINTER
7072      041556 062761 000002 041106      ADD      #2,GOUTPT(R1)      ;UPDATE THE QUEUE POINTER
7073      041564 026161 041106 041130      CMP      GOUTPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER?
7074      041572 001003      BNE      IS
7075      041574 016161 041126 041106      MOV      QSTART(R1),GOUTPT(R1) ;YES--RESET THE POINTER
7076      041572 006201      IS:      ASR      R1
7077      041604 000207      RTS      PC      ;RETURN TO USER

```

::*****

.SBTTL DATA, CONTROL, & STATUS BLOCKS

;BLOCK LOCATION EQUATE STATEMENTS

```

7088      000001      $FMT      =      1      ;FMT,HCI,ECI OR OFFSET CODE
7089      000002      $COMND      =      $FMT+1      ;OPERATION CODE
7090      000003      $PSEL      =      $FMT+2      ;PORT SELECT & BITS A16, A17
7091      000004      $WRDM      =      $FMT+3      ;WORD COUNT (2'S COMP)
7092      000006      $BUF      =      $FMT+5      ;BUFFER ADDR OR REGISTER TABLE POINTER
7093      000010      $SEC      =      $FMT+7      ;SECTOR ADDRESS OR 1ST REG ADDR
7094      000011      $TRK      =      $FMT+10      ;TRACK ADDRESS OF LAST REG ADDR
7095      000012      $CYL      =      $FMT+11      ;CYLINDER ADDR
7096      000014      $REG      =      $FMT+13      ;REGISTER STORAGE (IF ERROR)
7097      000016      $STATUS      =      $FMT+15      ;STATUS WORD (SET BY DRIVER)

```

;DRIVE'S HISTORY AND CURRENT INDICATOR STORAGE EQUATES

```

7101      000020      $WRDL      =      $FMT+17      ;WORD COUNT (NOT 2'S COMP)
7102      000022      $SSEC      =      $WRDL+2      ;SECTOR SIZE FOR CURRENT OPERATION
7103      000024      $CODE      =      $WRDL+4      ;PRESENT COMMAND SELECTION CODE
7104      000026      $PACK      =      $WRDL+6      ;WRITE DATA PACK INDICATOR
7105      000027      $PREVO      =      $WRDL+7      ;PREVIOUS COMMAND SELECTION CODE
7106      000030      $PATTC      =      $WRDL+10      ;PATTERN CODE
7107      000032      $PREVA      =      $WRDL+12      ;PREVIOUS ADDRESS - TRACK, SECTOR, CYLINDER
7108      000036      $OPERC      =      $WRDL+16      ;OPERATION COUNT
7109      000042      $POSIT      =      $WRDL+22      ;SEEK COUNT
7110      000046      $STRANS      =      $WRDL+26      ;TOTAL BITS XFERED COUNT (R & W)
7111      000052      $READ      =      $WRDL+32      ;TOTAL BITS READ COUNT
7112      000056      $TOTAL      =      $WRDL+36      ;TOTAL ERRORS (ALL TYPES) COUNT
7113      000060      $SOFT      =      $WRDL+40      ;'SOFT' ERROR COUNT
7114      000062      $HARD      =      $WRDL+42      ;'HARD' ERROR COUNT
7115      000064      $SKI      =      $WRDL+44      ;'SKI' OR 'OCYL' ERROR COUNT
7116      000066      $MISPO      =      $WRDL+46      ;PROG DETECTED MISPOSITIONING ERROR S COUNT
7117      000070      $PASSC      =      $WRDL+50      ;PASS COUNTER
7118      000072      $FAIR      =      $WRDL+52      ;OPERATION QUEUE 'FAIRNESS' COUNT

```



```

7119
7120
7121
7122      000074      $NCODE = $WRDL+54      ;NEXT OPERATION CODE
7123      000075      $NPATC = $NCODE+1      ;NEXT PATTERN
7124      000076      $NSEC = $NCODE+2      ;NEXT SECTOR
7125      000077      $NTRK = $NCODE+3      ;NEXT TRACK
7126      000100      $NICYL = $NCODE+4      ;NEXT CYLINDER
7127      000102      $NWRDL = $NCODE+6      ;NEXT BUFFER SIZE
7128      000104      $NEXT = $NCODE+10      ;PARAMETER SELECTION INDICATOR
7129
7130      ;INDEX EQUATES FOR MAXIMUM/MINIMUM ADDRESSES
7131
7132      000106      MAXCYL = $NCODE+12      ;MAXIMUM CYLINDER ADDRESS
7133      000110      MINCYL = MAXCYL+2      ;MINIMUM CYLINDER ADDRESS
7134      000112      MAXTRK = MAXCYL+4      ;MAXIMUM TRACK ADDRESS
7135      000114      MINTRK = MAXCYL+6      ;MINIMUM TRACK ADDRESS
7136      000116      MAXSEC = MAXCYL+10     ;MAXIMUM SECTOR ADDRESS
7137      000120      MINSEC = MAXCYL+12     ;MINIMUM SECTOR ADDRESS
7138      000122      $FIRST = MAXCYL+14     ;FIRST OPERATION INDICATOR
7139
7140      ;BAD SECTOR/TRACK ADDRESS STORAGE AREA INDEX EQUATE
7141
7142      000124      $BDSEC = MAXCYL+16      ;BAD SECTOR STORAGE TABLE
7143
7144      ;DRIVE ID AREA INDEX EQUATE
7145
7146      000224      $DRVID = $BDSEC+100     ;DRIVE ID
7147
7148      ;RH11/RP04/5/6 REGISTER EQUATES
7149
7150      000234      $RPCS1 = $DRVID+10      ;RP04 REGISTER STORAGE
7151      000236      $RPWC = $RPCS1+2
7152      000240      $RPBA = $RPCS1+4
7153      000242      $RPDA = $RPCS1+6
7154      000244      $RPDS2 = $RPCS1+10
7155      000246      $RPDS1 = $RPCS1+12
7156      000250      $RPER1 = $RPCS1+14
7157      000252      $RPAS = $RPCS1+16
7158      000254      $RPLA = $RPCS1+20
7159      000256      $RPDB = $RPCS1+22
7160      000260      $RPMR = $RPCS1+24
7161      000262      $RPDT = $RPCS1+26
7162      000264      $RPSN = $RPCS1+30
7163      000266      $RPOF = $RPCS1+32
7164      000270      $RPCA = $RPCS1+34
7165      000272      $RPCC = $RPCS1+36
7166      000274      $RPER2 = $RPCS1+40
7167      000276      $RPER3 = $RPCS1+42
7168      000300      $RPEC1 = $RPCS1+44
7169      000302      $RPEC2 = $RPCS1+46
7170
7171
7172      ;BLOCK FOR DRIVE 0
    
```

```

7173
7174 041606 000 000 DRIVED: .BYTE 0,0 ;DRIVE NUMBER
7175 041610 000005 .BLKW 5
7176 041622 042042 .WORD +$RPCS1-$REG
7177 041624 000266 .BLKB $RPEC2-$REG
7178
7179
7180 ;BLOCK FOR DRIVE 1
7181
7182 042112 001 000 DRIVE1: .BYTE 1,0 ;DRIVE NUMBER
7183 042114 000005 .BLKW 5
7184 042126 042346 .WORD +$RPCS1-$REG
7185 042130 000266 .BLKB $RPEC2-$REG
7186
7187
7188 ;BLOCK FOR DRIVE 2
7189
7190 042416 002 000 DRIVE2: .BYTE 2,0 ;DRIVE NUMBER
7191 042420 000005 .BLKW 5
7192 042432 042652 .WORD +$RPCS1-$REG
7193 042434 000266 .BLKB $RPEC2-$REG
7194
7195
7196 ;BLOCK FOR DRIVE 3
7197
7198 042722 003 000 DRIVE3: .BYTE 3,0 ;DRIVE NUMBER
7199 042724 000005 .BLKW 5
7200 042736 043156 .WORD +$RPCS1-$REG
7201 042740 000266 .BLKB $RPEC2-$REG
7202
7203
7204 ;BLOCK FOR DRIVE 4
7205
7206 043226 004 000 DRIVE4: .BYTE 4,0 ;DRIVE NUMBER
7207 043230 000005 .BLKW 5
7208 043242 043462 .WORD +$RPCS1-$REG
7209 043244 000266 .BLKB $RPEC2-$REG
7210
7211
7212 ;BLOCK FOR DRIVE 5
7213
7214 043532 005 000 DRIVES: .BYTE 5,0 ;DRIVE NUMBER
7215 043534 000005 .BLKW 5
7216 043546 043766 .WORD +$RPCS1-$REG
7217 043550 000266 .BLKB $RPEC2-$REG
7218
7219
7220 ;BLOCK FOR DRIVE 6
7221
7222 044036 006 000 DRIVE6: .BYTE 6,0 ;DRIVE NUMBER
7223 044040 000005 .BLKW 5
7224 044052 044272 .WORD +$RPCS1-$REG
7225 044054 000266 .BLKB $RPEC2-$REG
7226

```



```

7227
7228 ;BLOCK FOR DRIVE 7
7229
7230 044342 007 000 DRIVE7: .BYTE 7,0 ;DRIVE NUMBER
7231 044344 000005 .BLKW 5
7232 044356 044576 .WORD +$RPCS1-$REG
7233 044350 000266 .BLKB $RPEC2-$REG
7234
7235
7236 ;GENERAL PURPOSE DPB - USED BY 'READHD', 'RECALT', 'OFFSET', & 'RTNCTR'
7237
7238 044646 000000 000000 177774 GENDPE: .WORD 0,0,-4,CYLDER
7239 044654 054374
7240 044656 000000 000000 044666 .WORD 0,0,GENREG,0
7241 044664 000000
7242 044666 000024 GENREG: .BLKW 24 ;REGISTER STORAGE IF ERROR
7243
7244 ;*****
7245 .SBTTL ERROR MESSAGES
7246
7247 ;*****
7248
7249
7250 044736 044122 030461 044440 EM1: .ASCIZ /RH11 INTERRUPT OCCURRED (RPAS = 0)/
7251 044744 052116 051105 052522
7252 044752 052120 047440 041503
7253 044760 051125 042522 020104
7254 044766 051050 040520 020123
7255 044774 020075 024460 000
7256
7257 045001 125 042516 050130 EM2: .ASCIZ /UNEXPECTED ATTENTION OCCURRED/
7258 045006 041505 042524 020104
7259 045014 052101 042524 052116
7260 045022 047511 020116 041517
7261 045030 052503 051122 042105
7262 045036 000
7263
7264 045037 115 051501 041123 EM3: .ASCIZ /MASSBUS PARITY ERROR (MCPE=1)/
7265 045044 051525 050040 051101
7266 045052 052111 020131 051105
7267 045060 047522 020122 046450
7268 045066 050103 036505 024461
7269 045074 000
7270
7271 045075 115 051501 041123 EM4: .ASCIZ /MASSBUS PARITY ERROR (PAR=1)/
7272 045102 051525 050040 051101
7273 045110 052111 020131 051105
7274 045116 047522 020122 050050
7275 045124 051101 030475 000051
7276
7277 045132 042101 051104 051505 EM5: .ASCIZ /ADDRESS PLUG CHANGE BIT SET/
7278 045140 020123 046120 043525
7279 045146 041440 040510 043516
7280 045154 020105 044502 020124

```

7281	045162	042523	000124		
7282					
7283	045166	044122	030461	042040	EM6: .ASCIZ /RH11 DIDN'T RESPOND TO ADDRESSING/
7284	045174	042111	023516	020124	
7285	045202	042522	050123	047117	
7286	045210	020104	047524	040440	
7287	045216	042104	042522	051523	
7288	045224	047111	000107		
7289					
7290	045230	047125	047503	051122	EM10: .ASCIZ /UNCORRECTABLE MASSBUS PARITY ERROR/
7291	045236	041505	040524	046102	
7292	045244	020105	040515	051523	
7293	045252	052502	020123	040520	
7294	045260	044522	054524	042440	
7295	045266	051122	051117	000	
7296					
7297	045273	106	052101	046101	EM11: .ASCIZ /FATAL MASSBUS PARITY ERROR/
7298	045300	046440	051501	041123	
7299	045306	051525	050040	051101	
7300	045314	052111	020131	051105	
7301	045322	047522	000122		
7302					
7303	045326	042520	051522	051511	EM12: .ASCIZ /PERSISTENT DEVICE UNSAFE/
7304	045334	042524	052116	042040	
7305	045342	053105	041511	020105	
7306	045350	047125	040523	042506	
7307	045356	000			
7308					
7309	045357	117	042520	040522	EM13: .ASCIZ /OPERATION NOT COMPLETED WITHIN TIME LIMIT/
7310	045364	044524	047117	047040	
7311	045372	052117	041440	046517	
7312	045400	046120	052105	042105	
7313	045406	053440	052111	044510	
7314	045414	020116	044524	042515	
7315	045422	046040	046511	052111	
7316	045430	000			
7317					
7318	045431	104	044522	042526	EM14: .ASCIZ /DRIVE WENT OFFLINE/
7319	045436	053440	047105	020124	
7320	045444	043117	046106	047111	
7321	045452	000105			
7322					
7323	045454	047516	051040	051505	EM15: .ASCIZ /NO RESPONSE TO PORT REQUEST/
7324	045462	047520	051516	020105	
7325	045470	047524	050040	051117	
7326	045476	020124	042522	052521	
7327	045504	051505	000124		
7328					
7329	045510	042510	042101	051105	EM20: .ASCIZ /HEADER CRC ERROR/
7330	045516	041440	041522	042440	
7331	045524	051122	051117	000	
7332					
7333	045531	104	052101	020101	EM21: .ASCIZ /DATA CHECK ('DCK') ERROR/
7334	045536	044103	041505	020113	

DZRJD.019

ERROR MESSAGES

7335	045544	023450	041504	023513	
7336	045552	020051	051105	047522	
7337	045560	000122			
7338					
7339	045562	051127	052111	020105	EM22: .ASCIZ /WRITE CHECK ERROR - DATA CHECK ('DCK') SET/
7340	045570	044103	041505	020113	
7341	045576	051105	047522	020122	
7342	045604	020055	040504	040524	
7343	045612	041440	042510	045503	
7344	045620	024040	042047	045503	
7345	045626	024447	051440	052105	
7346	045634	000			
7347					
7348	045635	127	044522	042524	EM23: .ASCIZ /WRITE CHECK ERROR - DATA CHECK ('DCK') NOT SET/
7349	045642	041440	042510	045503	
7350	045650	042440	051122	051117	
7351	045656	026440	042040	052101	
7352	045664	020101	044103	041505	
7353	045672	020113	023450	041504	
7354	045700	023513	020051	047516	
7355	045706	020124	042523	000124	
7356					
7357	045714	042510	042101	051105	EM24: .ASCIZ /HEADER READ ERROR - 'FMT' BIT DROPPED/
7358	045722	051040	040505	020104	
7359	045730	051105	047522	020122	
7360	045736	020055	043047	052115	
7361	045744	020047	044502	020124	
7362	045752	051104	050117	042520	
7363	045760	000104			
7364					
7365	045762	042510	042101	051105	EM25: .ASCIZ /HEADER READ ERROR - HEADER COMPARE ('HCE') ERROR/
7366	045770	051040	040505	020104	
7367	045776	051105	047522	020122	
7368	046004	020055	042510	042101	
7369	046012	051105	041440	046517	
7370	046020	040520	042522	024040	
7371	046026	044047	042503	024447	
7372	046034	042440	051122	051117	
7373	046042	000			
7374					
7375	046043	106	051117	040515	EM26: .ASCIZ /FORMAT ERROR ('FER')/
7376	046050	020124	051105	047522	
7377	046056	020122	023450	042506	
7378	046064	023522	000051		
7379					
7380	046070	042510	042101	051105	EM27: .ASCIZ /HEADER COMPARE ('HCE') ERROR/
7381	046076	041440	046517	040520	
7382	046104	042522	024040	044047	
7383	046112	042503	024447	042440	
7384	046120	051122	051117	000	
7385					
7386	046125	115	051511	042503	EM30: .ASCIZ /MISCELLANEOUS DRIVE ERROR/
7387	046132	046114	047101	047505	
7388	046140	051525	042040	044522	

7389	046146	042526	042440	051122		
7390	046154	051117	000			
7391						
7392	046157	117	042520	040522	EM31:	.ASCIZ /OPERATION INCOMPLETE ('OPI') ERROR/
7393	046164	044524	047117	044440		
7394	046172	041516	046517	046120		
7395	046200	052105	020105	023450		
7396	046206	050117	023511	020051		
7397	046214	051105	047522	000122		
7398						
7399	046222	051104	053111	020105	EM32:	.ASCIZ /DRIVE TIMING ('DTE') ERROR/
7400	046230	044524	044515	043516		
7401	046236	024040	042047	042524		
7402	046244	024447	042440	051122		
7403	046252	051117	000			
7404						
7405	046255	120	051101	052111	EM33:	.ASCIZ /PARITY ('PAR') ERROR AFTER OPERATION STARTED/
7406	046262	020131	023450	040520		
7407	046270	023522	020051	051105		
7408	046276	047522	020122	043101		
7409	046304	042524	020122	050117		
7410	046312	051105	052101	047511		
7411	046320	020116	052123	051101		
7412	046326	042524	000104			
7413						
7414	046332	051127	052111	020105	EM34:	.ASCIZ /WRITE CLOCK FAILURE ('WCF') ERROR/
7415	046340	046103	041517	020113		
7416	046346	040506	046111	051125		
7417	046354	020105	023450	041527		
7418	046362	023506	020051	051105		
7419	046370	047522	000122			
7420						
7421	046374	047111	040526	044514	EM35:	.ASCIZ /INVALID ADDRESS ('IAE') ERROR/
7422	046402	020104	042101	051104		
7423	046410	051505	020123	023450		
7424	046416	040511	023505	020051		
7425	046424	051105	047522	000122		
7426						
7427	046432	051127	052111	020105	EM36:	.ASCIZ /WRITE LOCK ('WLE') ERROR/
7428	046440	047514	045503	024040		
7429	046446	053447	042514	024447		
7430	046454	042440	051122	051117		
7431	046462	000				
7432						
7433	046463	104	052101	020101	EM37:	.ASCIZ /DATA CHECK ('DCK') SET DURING WRITE CHECK COMMAND/
7434	046470	044103	041505	020113		
7435	046476	023450	041504	023513		
7436	046504	020051	042523	020124		
7437	046512	052504	044522	043516		
7438	046520	053440	044522	042524		
7439	046526	041440	042510	045503		
7440	046534	041440	046517	040515		
7441	046542	042116	000			
7442						

7443	046545	122	030510	020061	EM40:	.ASCIZ /RH11 OR UNIBUS TRANSFER ERROR/
7444	046552	051117	052440	044516		
7445	046560	052502	020123	051124		
7446	046566	047101	043123	051105		
7447	046574	042440	051122	051117		
7448	046602	000				
7449						
7450	046603	102	051525	040440	EM41:	.ASCIZ /BUS ADDRESS OR WORD COUNT INCORRECT/
7451	046610	042104	042522	051523		
7452	046616	047440	020122	047527		
7453	046624	042122	041440	052517		
7454	046632	052116	044440	041516		
7455	046640	051117	042522	052103		
7456	046646	000				
7457						
7458	046647	104	052101	020101	EM42:	.ASCIZ /DATA COMPARE ERRORS - NO OTHER ERROR(S) DETECTED/
7459	046654	047503	050115	051101		
7460	046662	020105	051105	047522		
7461	046670	051522	026440	047040		
7462	046676	020117	052117	042510		
7463	046704	020122	051105	047522		
7464	046712	024122	024523	042040		
7465	046720	052105	041505	042524		
7466	046726	000104				
7467						
7468	046730	040503	023516	020124	EM43:	.ASCIZ /CAN'T MATCH DATA READ WITH A PATTERN/
7469	046736	040515	041524	020110		
7470	046744	040504	040524	051040		
7471	046752	040505	020104	044527		
7472	046760	044124	040440	050040		
7473	046766	052101	042524	047122		
7474	046774	000				
7475						
7476	046775	105	051122	051117	EM44:	.ASCIZ /ERROR BIT(S) SET, BUT NO ERROR SIGNALLED BY THE RH11/
7477	047002	041040	052111	051450		
7478	047010	020051	042523	026124		
7479	047016	041040	052125	047040		
7480	047024	020117	051105	047522		
7481	047032	020122	044523	047107		
7482	047040	046101	042105	041040		
7483	047046	020131	044124	020105		
7484	047054	044122	030461	000		
7485						
7486	047061	105	041503	046040	EM45:	.ASCIZ /ECC LOGIC FAILURE - POSITION REGISTER VALUE TOO LARGE/
7487	047066	043517	041511	043040		
7488	047074	044501	052514	042522		
7489	047102	026440	050040	051517		
7490	047110	052111	047511	020116		
7491	047116	042522	044507	052123		
7492	047124	051105	053040	046101		
7493	047132	042525	052040	047517		
7494	047140	046040	051101	042507		
7495	047146	000				
7496						

7497	047147	102	051525	040440	EM46:	.ASCIZ	/BUS ADDRESS AND WORD COUNT NOT CONSISTENT/
7498	047154	042104	042522	051523			
7499	047162	040440	042116	053440			
7500	047170	051117	020104	047503			
7501	047176	047125	020124	047516			
7502	047204	020124	047503	051516			
7503	047212	051511	042524	052116			
7504	047220	000					
7505							
7506	047221	123	042505	020113	EM50:	.ASCIZ	/SEEK INCOMPLETE ('SKI') OR OFF CYLINDER ('OCYL') ERROR/
7507	047226	047111	047503	050115			
7508	047234	042514	042524	024040			
7509	047242	051447	044513	024447			
7510	047250	047440	020122	043117			
7511	047256	020106	054503	044514			
7512	047264	042116	051105	024040			
7513	047272	047447	054503	023514			
7514	047300	020051	051105	047522			
7515	047306	000122					
7516							
7517	047310	051120	043517	040522	EM51:	.ASCIZ	/PROGRAM DETECTED POSITIONING ERROR/
7518	047316	020115	042504	042524			
7519	047324	052103	042105	050040			
7520	047332	051517	052111	047511			
7521	047340	044516	043516	042440			
7522	047346	051122	051117	000			
7523							
7524	047353	104	044522	042526	EM60:	.ASCIZ	/DRIVE UNSAFE ERROR/
7525	047360	052440	051516	043101			
7526	047366	020105	051105	047522			
7527	047374	000122					
7528							
7529	047376	050122	051501	000	DH1:	.ASCIZ	/RPAS/
7530							
7531	047403	104	044522	042526	DH2:	.ASCIZ	/DRIVE RPDS1 RPER1 RPER2 RPER3 RPAS/
7532	047410	020040	051040	042120			
7533	047416	030523	020040	051040			
7534	047424	042520	030522	020040			
7535	047432	051040	042520	031122			
7536	047440	020040	051040	042520			
7537	047446	031522	020040	051040			
7538	047454	040520	000123				
7539							
7540	047460	051104	053111	020105	DH3:	.ASCIZ	/DRIVE REG ADR DATA/
7541	047466	020040	042522	020107			
7542	047474	042101	020122	042040			
7543	047502	052101	000101				
7544							
7545	047506	051104	053111	020105	DH4:	.ASCIZ	/DRIVE REG ADR GOOD BAD/
7546	047514	020040	042522	020107			
7547	047522	042101	020122	020040			
7548	047530	043440	047517	020104			
7549	047536	020040	041040	042101			
7550	047544	000					

7605	050106	000250	000274	000276			
7606	050114	000300	000302	000000			
7607							
7608	050122	000236	000240	000242	DT15:	.WORD	\$RPWC,\$RPBA,\$RPDA,\$RPAS,\$RPLA,\$RPDB,\$RPMR,\$RPDT,0
7609	050130	000252	000254	000256			
7610	050136	000260	000262	000000			
7611							
7612	050144	000264	000266	000270	DT16:	.WORD	\$RPSN,\$RPOF,\$RPCA,\$RPCC,\$STATUS,0
7613	050152	000272	000016	000000			
7614							
7615	050160	051120	051505	047105	LIN2C:	.ASCIZ	/PRESENT ORDER = /
7616	050166	020124	051117	042504			
7617	050174	020122	020075	000			
7618	050201	040	050040	042522	LIN2P:	.ASCIZ	/ PREVIOUS ORDER = /
7619	050206	044526	052517	020123			
7620	050214	051117	042504	020122			
7621	050222	020075	000				
7622	050225	052	042440	051122	LIN2S:	.ASCIZ	Q* ERROR AT BAD TRACK/SECTORQ
7623	050232	051117	040440	020124			
7624	050240	040502	020104	051124			
7625	050246	041501	027513	042523			
7626	050254	052103	051117	000			
7627	050261	105	051122	051117	LINM3:	.ASCIZ	/ERROR AT C/
7628	050266	040440	020124	000103			
7629	050274	052040	000		T:	.ASCIZ	/ T/
7630	050277	120	042522	042523	LINN3:	.ASCIZ	/PRESENT ADDR = C/
7631	050304	052116	040440	042104			
7632	050312	020122	020075	000103			
7633	050320	051440	000		S:	.ASCIZ	/ S/
7634	050323	040	020040	051120	LINP3:	.ASCIZ	/ PREV ADDR = C/
7635	050330	053105	040440	042104			
7636	050336	020122	020075	000103			
7637	050344	052123	051101	020124	LINS3:	.ASCIZ	/START CYL = /
7638	050352	054503	020114	020075			
7639	050360	000					
7640	050361	040	042440	042116	LINEN3:	.ASCIZ	/ END CYL = /
7641	050366	041440	046131	036440			
7642	050374	000040					
7643	050376	020040	041501	052524	LINA3:	.ASCIZ	/ ACTUAL CYL = /
7644	050404	046101	041440	046131			
7645	050412	036440	000040				
7646	050416	020040	051124	020113	LINT3:	.ASCIZ	/ TRK = /
7647	050424	020075	000				
7648	050427	040	050122	040503	LINCA3:	.ASCIZ	/ RPCA = /
7649	050434	036440	000040				
7650	050440	050122	040504	036440	LINDA3:	.ASCIZ	/RPDA = /
7651	050446	000040					
7652	050450	050122	040502	036440	LINB3:	.ASCIZ	/RPBA = /
7653	050456	000040					
7654	050460	020040	050122	041527	LINW3:	.ASCIZ	/ RPWC = /
7655	050466	036440	000040				
7656	050472	052123	051101	020124	LINST3:	.ASCIZ	/START TRK = /
7657	050500	051124	020113	020075			
7658	050506	000					

7659	050507	123	040524	052122	LINSS3: .ASCIZ /START SEC = /
7660	050514	051440	041505	036440	
7661	050522	000040			
7662	050524	052502	043106	051105	LINM4: .ASCIZ /BUFFER ADDR = /
7663	050532	040440	042104	020122	
7664	050540	020075	000		
7665	050543	040	051440	055111	LINS4: .ASCIZ / SIZE = /
7666	050550	020105	020075	000	
7667	050555	040	040440	052103	LINX4: .ASCIZ / ACTUAL NMBR WRDS XFRD = /
7668	050562	040525	020114	046516	
7669	050570	051102	053440	042122	
7670	050576	020123	043130	042122	
7671	050604	036440	000040		
7672	050610	047507	042117	042040	LIND5: .ASCIZ /GOOD DATA = /
7673	050616	052101	020101	020075	
7674	050624	000			
7675	050625	040	041040	042101	LINB5: .ASCIZ / BAD DATA = /
7676	050632	042040	052101	020101	
7677	050640	020075	000		
7678	050643	040	051440	041505	LINP5: .ASCIZ / SECT POS = /
7679	050650	020124	047520	020123	
7680	050656	020075	000		
7681	050661	110	040505	042504	LINS5: .ASCIZ /HEADER FROM ERROR SECTOR = /
7682	050666	020122	051106	046517	
7683	050674	042440	051122	051117	
7684	050702	051440	041505	047524	
7685	050710	020122	020075	000	
7686	050715	122	042520	030503	LINEP5: .ASCIZ /RPEC1 = /
7687	050722	036440	000040		
7688	050726	051040	042520	031103	LINEO5: .ASCIZ / RPEC2 = /
7689	050734	036440	000040		
7690	050740	042523	052103	051117	LINB6: .ASCIZ /SECTOR IS ECC CORRECTABLE /
7691	050746	044440	020123	041505	
7692	050754	020103	047503	051122	
7693	050762	041505	040524	046102	
7694	050770	020105	000		
7695	050773	123	041505	047524	LINC6: .ASCIZ /SECTOR READ CORRECTLY /
7696	051000	020122	042522	042101	
7697	051006	041440	051117	042522	
7698	051014	052103	054514	000040	
7699	051022	047503	051122	041505	LING6: .ASCIZ /CORRECTED ON /
7700	051030	042524	020104	047117	
7701	051036	000040			
7702	051040	051040	052105	044522	LINR6: .ASCIZ / RETRIES/
7703	051046	051505	000		
7704	051051	125	041516	051117	LINU06: .ASCIZ /UNCORRECTABLE AFTER /
7705	051056	042522	052103	041101	
7706	051064	042514	040440	052106	
7707	051072	051105	000040		
7708	051076	020040	047524	040524	LIN7M: .ASCIZ / TOTAL MISPOS ERR = /
7709	051104	020114	044515	050123	
7710	051112	051517	042440	051122	
7711	051120	036440	000040		
7712	051124	051117	042504	051522	LIN70: .ASCIZ /ORDERS:/

7713	051132	000072								
7714	051134	052040	052117	046101	LIN7P:	.ASCIZ	/	TOTAL	SEEKS = /	
7715	051142	051440	042505	051513						
7716	051150	036440	000040							
7717	051154	052040	052117	046101	LIN7S:	.ASCIZ	/	TOTAL	SKI,OCYL ERR = /	
7718	051162	051440	044513	047454						
7719	051170	054503	020114	051105						
7720	051176	020122	020075	000						
7721	051203	040	042440	051122	LIN7T:	.ASCIZ	/	ERRORS:/		
7722	051210	051117	035123	000						
7723	051215	040	053440	042122	LIN7X:	.ASCIZ	/	WRDS	XFR:/	
7724	051222	020123	043130	035122						
7725	051230	000								
7726	051231	040	053440	042122	LIN7R:	.ASCIZ	/	WRDS	READ:/	
7727	051236	020123	042522	042101						
7728	051244	000072								
7729	051246	044504	043106	051105	LIN8M:	.ASCIZ	/	DIFFERENT	ERROR DURING RETRY/	
7730	051254	047105	020124	051105						
7731	051262	047522	020122	052504						
7732	051270	044522	043516	051040						
7733	051276	052105	054522	000						
7734	051303	104	052101	020101	LIN9B:	.ASCIZ	/	DATA	COMPARISON ERRORS/	
7735	051310	047503	050115	051101						
7736	051316	051511	047117	042440						
7737	051324	051122	051117	000123						
7738	051332	020040	020040	020040	LIN9H:	.ASCII	/	GOOD	BAD/<CR><LF>	
7739	051340	020040	043440	047517						
7740	051346	020104	020040	041040						
7741	051354	042101	005015							
7742	051360	047514	020103	020040		.ASCIZ	/	LOC	DATA	DATA/<CR><LF>
7743	051366	020040	042040	052101						
7744	051374	020101	020040	042040						
7745	051402	052101	006501	000012						
7746	051410	047514	020103	020040	LIN9I:	.ASCIZ	/	LOC	DATA/<CR><LF>	
7747	051416	020040	042040	052101						
7748	051424	006501	000012							
7749	051430	047524	040524	020114	LIN9E:	.ASCIZ	/	TOTAL	COMPARE ERRORS = /	
7750	051436	047503	050115	051101						
7751	051444	020105	051105	047522						
7752	051452	051522	036440	000040						
7753	051460	044124	020105	040504	LIN9G:	.ASCIZ	/	THE	DATA COMPARED OK/<CR><LF>	
7754	051466	040524	041440	046517						
7755	051474	040520	042522	020104						
7756	051502	045517	005015	000						
7757	051507	105	051122	051117	LIN10A:	.ASCIZ	/	ERROR	BURST BEGINS AT WORD /	
7758	051514	041040	051125	052123						
7759	051522	041040	043505	047111						
7760	051530	020123	052101	053440						
7761	051536	051117	020104	000						
7762	051543	040	047111	042040	LIN10B:	.ASCIZ	/	IN	DATA FIELD OF ERROR SECTOR/<CR><LF>	
7763	051550	052101	020101	044506						
7764	051556	046105	020104	043117						
7765	051564	042440	051122	051117						
7766	051572	051440	041505	047524						

DZRJD.019

ERROR MESSAGES

7767	051600	006522	000012		
7768	051604	051105	047522	020122	LIN10C: .ASCII /ERROR WAS NOT IN THE DATA READ - /<CR><LF>
7769	051612	040527	020123	047516	
7770	051620	020124	047111	052040	
7771	051626	042510	042040	052101	
7772	051634	020101	042522	042101	
7773	051642	026440	006440	012	
7774	051647	105	041503	041440	.ASCIZ /ECC CORRECTION CAN'T BE PERFORMED/
7775	051654	051117	042522	052103	
7776	051662	047511	020116	040503	
7777	051670	023516	020124	042502	
7778	051676	050040	051105	047506	
7779	051704	046522	042105	000	
7780	051711	105	041503	041440	LIN10H: .ASCII /ECC CORRECTION RESULTS/<CR><LF>
7781	051716	051117	042522	052103	
7782	051724	047511	020116	042522	
7783	051732	052523	052114	006523	
7784	051740	012			
7785	051741	101	042104	020122	.ASCIZ /ADDR BAD CORRECTED /<CR><LF>
7786	051746	020040	041040	042101	
7787	051754	020040	020040	041440	
7788	051762	051117	042522	052103	
7789	051770	042105	006440	000012	
7790	051776	047503	052116	047105	LIN11H: .ASCIZ /CONTENTS OF ERROR SECTOR (REPORTED ABOVE)/<CR><LF>
7791	052004	051524	047440	020106	
7792	052012	051105	047522	020122	
7793	052020	042523	052103	051117	
7794	052026	024040	042522	047520	
7795	052034	052122	042105	040440	
7796	052042	047502	042526	006451	
7797	052050	000012			
7798	052052	042101	051104	020040	.ASCIZ /ADDR DATA/<CR><LF>
7799	052060	020040	040504	040524	
7800	052066	005015	000		
7801	052071	040	040		LIN4SP: .ASCII //
7802	052073	040			LINSP: .ASCII //
7803	052074	000040			LINSP0: .ASCIZ //
7804					
7805					.SBTTL TELETYPE MESSAGES
7806					
7807	052076	051104	053111	000105	UNTMSG: .ASCIZ /DRIVE/
7808	052104	047440	043106	044514	UNTOFF: .ASCIZ / OFFLINE/
7809	052112	042516	000		
7810	052115	040	047117	044514	UNTON: .ASCIZ / ONLINE/
7811	052122	042516	000		
7812	052125	040	047516	020124	UNTNOT: .ASCIZ / NOT BEING TESTED/
7813	052132	042502	047111	020107	
7814	052140	042524	052123	042105	
7815	052146	000			
7816	052147	040	046101	042522	UNTASN: .ASCIZ / ALREADY BEING TESTED/
7817	052154	042101	020131	042502	
7818	052162	047111	020107	042524	
7819	052170	052123	042105	000	
7820	052175	040	047516	020124	NOTRP: ASCIZ @ NOT AN RPO4/5/6@

7821	052202	047101	051040	030120	
7822	052210	027464	027465	000066	
7823	052216	047040	052117	050040	NOTPRS: .ASCIZ / NOT PRESENT/
7824	052224	042522	042523	052116	
7825	052232	000			
7826	052233	040	047516	020124	NOTAVL: .ASCIZ / NOT AVAILABLE/
7827	052240	053101	044501	040514	
7828	052246	046102	000105		
7829	052252	052440	051516	043101	NOTSAF: .ASCIZ / UNSAFE/
7830	052260	000105			
7831	052262	047125	052111	051440	SYSTAT: .ASCIZ /UNIT STATUS: /<CR><LF><LF>
7832	052270	040524	052524	035123	
7833	052276	005015	000012		
7834	052302	050122	032060	000	RPO4B: .ASCIZ /RPO4/
7835	052307	122	030120	000065	RPO5: .ASCIZ /RPO5/
7836	052314	050122	033060	000	RPO6: .ASCIZ /RPO6/
7837	052321	104	044522	042526	STATHD: .ASCII /DRIVE PERFORMANCE SUMMARY/<CR><LF>
7838	052326	050040	051105	047506	
7839	052334	046522	047101	042503	
7840	052342	051440	046525	040515	
7841	052350	054522	005015		
7842	052354	051104	020126	040520	.ASCII /DRV PASS ORDERS SEEKS WRDS XFER WRDS READ /
7843	052362	051523	047440	042122	
7844	052370	051105	020123	020040	
7845	052376	042523	045505	020123	
7846	052404	020040	051127	051504	
7847	052412	054040	042506	020122	
7848	052420	020040	051127	051504	
7849	052426	051040	040505	020104	
7850	052434	047523	052106	044040	.ASCIZ /SOFT HARD SKI MISP OTHER/<CR><LF>
7851	052442	051101	020104	051440	
7852	052450	044513	046440	051511	
7853	052456	020120	052117	042510	
7854	052464	006522	000012		
7855	052470	047504	042516	005015	PDONE: .ASCIZ /DONE/<CR><LF><LF>
7856	052476	000012			
7857	052500	037407	040506	040524	DROPNG: .ASCIZ <07>/?FATAL OR EXCESSIVE ERRORS/
7858	052506	020114	051117	042440	
7859	052514	041530	051505	044523	
7860	052522	042526	042440	051122	
7861	052530	051117	000123		
7862	052534	047105	020104	043117	ENDPAS: .ASCIZ /END OF PASS/
7863	052542	050040	051501	000123	
7864	052550	005015	047105	020104	ENDTST: .ASCIZ <CR><LF>/END OF TEST/
7865	052556	043117	052040	051505	
7866	052564	000124			
7867	052566	005015	051104	053111	DEASSG: .ASCIZ <CR><LF>/DRIVE DEASSIGNED/
7868	052574	020105	042504	051501	
7869	052602	044523	047107	042105	
7870	052610	000			
7871	052611	015	025012	025052	DRNUM: .ASCIZ <CR><LF>/***** DRIVE #/
7872	052616	025052	025052	025052	
7873	052624	020052	051104	053111	
7874	052632	020105	000043		

7875	052636	051440	040524	052122	ASGND: .ASCIZ / STARTED/<CR><LF>
7876	052644	042105	005015	000	
7877	052651	015	037412	023440	NEDCLK: .ASCIZ <CR><LF>/? 'L' OR 'P' CLOCK REQUIRED ON SYSTEM/<CR><LF>
7878	052656	023514	047440	020122	
7879	052664	050047	020047	046103	
7880	052672	041517	020113	042522	
7881	052700	052521	051111	042105	
7882	052706	047440	020116	054523	
7883	052714	052123	046505	005015	
7884	052722	000			
7885	052723	056	000		PERIOD: .ASCIZ /:/
7886	052725	077	000		QUES: .ASCIZ /?/
7887	052727	111	053116	046101	INVLD: .ASCIZ /INVALID COMMAND/<CR><LF>
7888	052734	042111	041440	046517	
7889	052742	040515	042116	005015	
7890	052750	000			
7891	052751	015	042412	052116	ENTCOM: .ASCIZ <CR><LF>/ENTER COMMANDS: /<CR><LF>
7892	052756	051105	041440	046517	
7893	052764	040515	042116	035123	
7894	052772	006440	000012		
7895	052776	047105	042524	020122	ENTDRV: .ASCIZ /ENTER I.D. FOR DRV #/
7896	053004	027111	027104	043040	
7897	053012	051117	042040	053122	
7898	053020	021440	000		
7899	053023	015	042412	052116	ENTLMT: .ASCIZ <CR><LF>/ENTER ADDRESS LIMITS FOR DRV #/
7900	053030	051105	040440	042104	
7901	053036	042522	051523	046040	
7902	053044	046511	052111	020123	
7903	053052	047506	020122	051104	
7904	053060	020126	000043		
7905	053064	047105	042524	020122	ENTADR: .ASCIZ @ENTER BAD TRK/SEC ADRS FOR DRV #@
7906	053072	040502	020104	051124	
7907	053100	027513	042523	020103	
7908	053106	042101	051522	043040	
7909	053114	051117	042040	053122	
7910	053122	021440	000		
7911	053125	072	000		COLON: .ASCIZ /:/
7912	053127	015	042012	052101	DATEIS: .ASCIZ <CR><LF>/DATE: /
7913	053134	035105	000040		
7914	053140	005015	050117	051105	IDIS: .ASCIZ <CR><LF>/OPERATOR I.D.: /
7915	053146	052101	051117	044440	
7916	053154	042056	035056	000040	
7917	053162	005015	042012	053122	HEDLIN: .ASCIZ <CR><LF><LF>/DRV DRV I.D./<CR><LF>
7918	053170	020040	051104	020126	
7919	053176	027111	027104	005015	
7920	053204	000			
7921	053205	116	047117	006505	NONE: .ASCIZ /NONE/<CR><LF>
7922	053212	000012			
7923	053214	020077	047111	040526	BADENT: .ASCIZ /? INVALID ENTRY/<CR><LF>
7924	053222	044514	020104	047105	
7925	053230	051124	006531	000012	
7926	053236	054523	052123	046505	BUSY: .ASCIZ /SYSTEM BUSY.../<CR><LF>
7927	053244	041040	051525	027131	
7928	053252	027056	005015	000	


```

7929 053257 015 050012 047522 INTDON: .ASCII <CR><LF>/PROGRAM INITIALIZATION COMPLETE/
7930 053264 051107 046501 044440
7931 053272 044516 044524 046101
7932 053300 055111 052101 047511
7933 053306 020116 047503 050115
7934 053314 042514 042524
7935 053320 005015 054524 042520 .ASCIZ <CR><LF>/TYPE A 'CONTROL C' TO ENTER COMMANDS/<CR><LF><LF>
7936 053326 040440 023440 047503
7937 053334 052116 047522 020114
7938 053342 023503 052040 020117
7939 053350 047105 042524 020122
7940 053356 047503 046515 047101
7941 053364 051504 005015 000012
7942
7943 .EVEN
7944 ;PARAMETER ENTRY TABLE
7945
7946
7947 053372 053520 000000 001364 PARLST: .WORD PAR1,0,MAXDL
7948 053400 053526 177777 001370 .WORD PAR2,-1,INTRVL
7949 053406 053656 177777 001362 .WORD PAR19,-1,PASCNT
7950 053414 053535 177777 001374 .WORD PAR3,-1,CMPLMT
7951 053422 053625 000001 001400 .WORD PAR11,1,WCSEL
7952 053430 053633 000007 001402 .WORD PAR14,7,RATIO
7953 053436 053650 000001 001410 .WORD PAR16,1,ENDET
7954 053444 053616 000001 001376 .WORD PAR10,1,FORMAT
7955 053452 053641 000001 001404 .WORD PAR15,1,AUTOCK
7956 053460 053665 000001 001406 .WORD PAR20,1,NOTPRT
7957 053466 000000 .WORD 0 ;TABLE TERMINATOR
7958
7959 053470 047105 042524 020122 ASKPAR: .ASCIZ /FILTER PARAMETERS: /
7960 053476 040520 040522 042515
7961 053504 042524 051522 020072
7962 053512 000040
7963 053514 027440 000040 SLASH: .ASCIZ @ / @
7964
7965 053520 040515 042130 000114 PAR1: .ASCIZ /MAXDL/
7966 053526 047111 051124 046126 PAR2: .ASCIZ /INTRVL/
7967 053534 000
7968 053535 103 050115 046514 PAR3: .ASCIZ /CMPLMT/
7969 053542 000124
7970 053544 040515 041530 046131 PAR4: .ASCIZ /MAXCYL/
7971 053552 000
7972 053553 115 047111 054503 PAR5: .ASCIZ /MINCYL/
7973 053560 000114
7974 053562 040515 052130 045522 PAR6: .ASCIZ /MAXTRK/
7975 053570 000
7976 053571 115 047111 051124 PAR7: .ASCIZ /MINTRK/
7977 053576 000113
7978 053600 040515 051530 041505 PAR8: .ASCIZ /MAXSEC/
7979 053606 000
7980 053607 115 047111 042523 PAR9: .ASCIZ /MINSEC/
7981 053614 000103
7982 053616 047506 046522 052101 PAR10: .ASCIZ /FORMAT/

```



```

7983 053624 000
7984 053625 127 051503 046105 PAR11: .ASCIZ /WCSEL/
7985 053632 000
7986 053633 122 052101 047511 PAR14: .ASCIZ /RATIO/
7987 053640 000
7988 053641 101 052125 041517 PAR15: .ASCIZ /AUTOCK/
7989 053646 000113
7990 053650 047105 042504 000124 PAR16: .ASCIZ /ENDET/
7991 053656 040520 041523 052116 PAR19: .ASCIZ /PASCNT/
7992 053664 000
7993 053665 116 052117 051120 PAR20: .ASCIZ /NOTPRT/
7994 053672 000124

```

.EVEN

;PARAMETER TABLE POINTERS FOR ADDRESS LIMITS

```

8000 053674 053714 TABLE: .WORD TABLE0 ;PARAMETER TABLE FOR DRIVE 0
8001 053676 053762 .WORD TABLE1 ;PARAMETER TABLE FOR DRIVE 1
8002 053700 054030 .WORD TABLE2 ;PARAMETER TABLE FOR DRIVE 2
8003 053702 054076 .WORD TABLE3 ;PARAMETER TABLE FOR DRIVE 3
8004 053704 054144 .WORD TABLE4 ;PARAMETER TABLE FOR DRIVE 4
8005 053706 054212 .WORD TABLE5 ;PARAMETER TABLE FOR DRIVE 5
8006 053710 054260 .WORD TABLE6 ;PARAMETER TABLE FOR DRIVE 6
8007 053712 054326 .WORD TABLE7 ;PARAMETER TABLE FOR DRIVE 7

```

;PARAMETER TABLE FOR ADDRESS LIMITS

```

8008
8009
8010
8011 053714 053553 000000 041716 TABLE0: .WORD PAR5,0,MINCYL+DRIVE0
8012 053722 053544 000000 041714 .WORD PAR4,0,MAXCYL+DRIVE0
8013 053730 053571 000022 041722 .WORD PAR7,18.,MINTRK+DRIVE0
8014 053736 053562 000022 041720 .WORD PAR6,18.,MAXTRK+DRIVE0
8015 053744 053607 000025 041726 .WORD PAR9,21.,MINSEC+DRIVE0
8016 053752 053600 000025 041724 .WORD PAR8,21.,MAXSEC+DRIVE0,0
8017 053760 000000

```

```

8018
8019 053762 053553 000000 042222 TABLE1: .WORD PAR5,0,MINCYL+DRIVE1
8020 053770 053544 000000 042220 .WORD PAR4,0,MAXCYL+DRIVE1
8021 053776 053571 000022 042226 .WORD PAR7,18.,MINTRK+DRIVE1
8022 054004 053562 000022 042224 .WORD PAR6,18.,MAXTRK+DRIVE1
8023 054012 053607 000025 042232 .WORD PAR9,21.,MINSEC+DRIVE1
8024 054020 053600 000025 042230 .WORD PAR8,21.,MAXSEC+DRIVE1,0
8025 054026 000000

```

```

8026
8027 054030 053553 000000 042526 TABLE2: .WORD PAR5,0,MINCYL+DRIVE2
8028 054036 053544 000000 042524 .WORD PAR4,0,MAXCYL+DRIVE2
8029 054044 053571 000022 042532 .WORD PAR7,18.,MINTRK+DRIVE2
8030 054052 053562 000022 042530 .WORD PAR6,18.,MAXTRK+DRIVE2
8031 054060 053607 000025 042536 .WORD PAR9,21.,MINSEC+DRIVE2
8032 054066 053600 000025 042534 .WORD PAR8,21.,MAXSEC+DRIVE2,0
8033 054074 000000

```

```

8034
8035 054076 053553 000000 043032 TABLE3: .WORD PAR5,0,MINCYL+DRIVE3
8036 054104 053544 000000 043030 .WORD PAR4,0,MAXCYL+DRIVE3

```



```

8037 054112 053571 000022 043036 .WORD PAR7,18.,MINTRK+DRIVE3
8038 054120 053562 000022 043034 .WORD PAR6,18.,MAXTRK+DRIVE3
8039 054126 053607 000025 043042 .WORD PAR9,21.,MINSEC+DRIVE3
8040 054134 053600 000025 043040 .WORD PAR8,21.,MAXSEC+DRIVE3,0
8041 054142 000000
8042
8043 054144 053553 000000 043336 TABLE4: .WORD PAR5,0,MINCYL+DRIVE4
8044 054152 053544 000000 043334 .WORD PAR4,0,MAXCYL+DRIVE4
8045 054160 053571 000022 043342 .WORD PAR7,18.,MINTRK+DRIVE4
8046 054166 053562 000022 043340 .WORD PAR6,18.,MAXTRK+DRIVE4
8047 054174 053607 000025 043346 .WORD PAR9,21.,MINSEC+DRIVE4
8048 054202 053600 000025 043344 .WORD PAR8,21.,MAXSEC+DRIVE4,0
8049 054210 000000
8050
8051 054212 053553 000000 043642 TABLE5: .WORD PAR5,0,MINCYL+DRIVES
8052 054220 053544 000000 043640 .WORD PAR4,0,MAXCYL+DRIVES
8053 054226 053571 000022 043646 .WORD PAR7,18.,MINTRK+DRIVES
8054 054234 053562 000022 043644 .WORD PAR6,18.,MAXTRK+DRIVES
8055 054242 053607 000025 043652 .WORD PAR9,21.,MINSEC+DRIVES
8056 054250 053600 000025 043650 .WORD PAR8,21.,MAXSEC+DRIVES,0
8057 054256 000000
8058
8059 054260 053553 000000 044146 TABLE6: .WORD PAR5,0,MINCYL+DRIVE6
8060 054266 053544 000000 044144 .WORD PAR4,0,MAXCYL+DRIVE6
8061 054274 053571 000022 044152 .WORD PAR7,18.,MINTRK+DRIVE6
8062 054302 053562 000022 044150 .WORD PAR6,18.,MAXTRK+DRIVE6
8063 054310 053607 000025 044156 .WORD PAR9,21.,MINSEC+DRIVE6
8064 054316 053600 000025 044154 .WORD PAR8,21.,MAXSEC+DRIVE6,0
8065 054324 000000
8066
8067 054326 053553 000000 044452 TABLE7: .WORD PAR5,0,MINCYL+DRIVE7
8068 054334 053544 000000 044450 .WORD PAR4,0,MAXCYL+DRIVE7
8069 054342 053571 000022 044456 .WORD PAR7,18.,MINTRK+DRIVE7
8070 054350 053562 000022 044454 .WORD PAR6,18.,MAXTRK+DRIVE7
8071 054356 053607 000025 044462 .WORD PAR9,21.,MINSEC+DRIVE7
8072 054364 053600 000025 044460 .WORD PAR8,21.,MAXSEC+DRIVE7,0
8073 054372 000000
8074
8075
8076 054374 000000 000000 000000 CYLDER: .WORD 0,0,0,0 ;HEADER BUFFER FOR 'READHD' ROUTINE
8077 054402 000000
8078
8079 054404 ENDPGM = . ;LAST LOCATION OF PROG + 2
8080
8081 ;*****
8082
8083 ;ROUTINE TO GET THE DATE AND THE OPERATOR FROM THE OPERATOR
8084 ;CALL:
8085 ; JSR PC,OPRDAT
8086 ; RETURN
8087
8088 ;NOTE: THIS ROUTINE IS ENTERED ONLY AT INITIAL START
8089
8090 054404 104401 054516 OPRDAT: TYPE ,ENTDAT ;'ENTER DATE'

```



```

8091 054410 104411 RDLIN ;READ THE ENTRY
8092 054412 012605 MOV (SP)+,R5 ;PUT THE ENTRY ADDRESS INTO R5
8093 054414 112537 001220 MOV (R5)+,DATE ;STORE THE DATE
8094 054420 112537 001221 MOV (R5)+,DATE+1 ;STORE THE DATE
8095 054424 112537 001222 MOV (R5)+,DATE+2 ;STORE THE DATE
8096 054430 112537 001223 MOV (R5)+,DATE+3 ;STORE THE DATE
8097 054434 112537 001224 MOV (R5)+,DATE+4 ;STORE THE DATE
8098 054440 112537 001225 MOV (R5)+,DATE+5 ;STORE THE DATE
8099 054444 112537 001226 MOV (R5)+,DATE+6 ;STORE THE DATE
8100 054450 112537 001227 MOV (R5)+,DATE+7 ;STORE THE DATE
8101 054454 104401 054535 TYPE ,ENTID ;'ENTER OPERATOR I.D.'
8102 054460 104411 RDLIN ;READ THE ENTRY
8103 054462 012605 MOV (SP)+,R5 ;ENTRY ADDRESS
8104 054464 112537 001232 MOV (R5)+,OPERID ;STORE THE I.D.
8105 054470 112537 001233 MOV (R5)+,OPERID+1 ;STORE THE I.D.
8106 054474 112537 001234 MOV (R5)+,OPERID+2 ;STORE THE I.D.
8107 054500 112537 001235 MOV (R5)+,OPERID+3 ;STORE THE I.D.
8108 054504 112537 001236 MOV (R5)+,OPERID+4 ;STORE THE I.D.
8109 054510 112537 001237 MOV (R5)+,OPERID+5 ;STORE THE I.D.
8110 054514 000207 RTS PC ;RETURN
8111
8112 054516 005015 047105 042524 ENTDAT: .ASCIZ <CR><LF>/ENTER DATE: /
8113 054524 020122 040504 042524
8114 054532 020072 000
8115 054535 105 052116 051105 ENTDAT: .ASCIZ /ENTER OPERATOR I.D.: /
8116 054542 047440 042520 040522
8117 054550 047524 020122 027111
8118 054556 027104 020072 000
8119
8120 .EVEN
8121 .SBTTL ROUTINE TO SIZE MEMORY
8122
8123 ;*****
8124 ;*CALL:
8125 ;* JSR PC,$SIZE
8126 ;* RETURN
8127 ;*$LSTAD WILL CONTAIN THE LAST AVAILABLE MEMORY LOCATION
8128
8129 $SIZE: MOV RO,-(SP) ;SAVE RO ON THE STACK
8130 MOV R1,-(SP) ;SAVE R1 ON THE STACK
8131 MOV @#ERRVEC,-(SP) ;SAVE PRESENT ERROR VECTOR PS & PC
8132 MOV @#ERRVEC+2,-(SP)
8133 MOV SP,RO ;SAVE THE STACK POINTER
8134 ;;SET THE ERRVEC PS TO THE PRESENT PS
8135 TRAP ;PUSH OLD PSW AND PC ON STACK
8136 MOV (SP)+,@#ERRVEC+2 ;SAVE THE PSW IN @#ERRVEC+2
8137 MOV #2,@#ERRVEC ;SET FOR TIMEOUT
8138 MOV #20000,R1 ;FIRST ADDRESS
8139 1$: TST (R1) ;TEST THIS ADDRESS
8140 TST (R1)+ ;STEP TO NEXT ADDRESS
8141 BR 1$ ;TRY ANOTHER
8142 2$: SUB #2,R1 ;DROP BACK
8143 MOV RO,SP ;RESTORE THE STACK
8144 MOV (SP)+,@#ERRVEC+2 ;RESTORE ERROR VECTOR

```



```

8145 054642 012637 000004      MOV      (SP)+, @#ERRVEC
8146 054646 010137 054660      MOV      R1, $LSTAD      ;; LAST ADDRESS
8147 054652 012601          MOV      (SP)+, R1      ;; RESTORE R1
8148 054654 012600          MOV      (SP)+, R0      ;; RESTORE R0
8149 054656 000207          RTS      PC
8150 054660 000000      $LSTAD: .WORD 0          ;; CONTAINS THE LAST ADDRESS
8151
8152      .SBTTL  BUSADR - GET BUS ADDRESS AND VECTOR ADDRESS FOR RH11
8153      ; THIS ROUTINE IS USED TO INSURE THE BUS ADDRESS
8154      ; OF THE RH11 IS SETUP FOR THE PROPER ADDRESS.
8155      ; IT WILL ALSO READ THE ADDRESS FROM THE TTY IF
8156      ; REQUIRED.
8157      ; NOTE: THIS ROUTINE DESTROYS R0-R4
8158      ; CALL
8159
8160      ;
8161      ;
8162      ;
8163 054662 005737 001260      BUSADR: TST      CHGADR      ; INPUT FROM TTY REQUESTED?
8164 054666 001450          BEQ      7$              ; NO--BRANCH
8165 054670 005037 001260      CLR      CHGADR      ; YES--CLEAR THE REQUEST FLAG
8166 054674 012700 001170      1$:  MOV      #SRPADR, R0  ; FIRST ADDRESS
8167 054700 104401 055072      TYPE    MRPCS1        ; "RPCS1="
8168 054704 012046          MOV      (R0)+, -(SP)   ; PRESENT RPCS1 ADDRESS
8169 054706 104402          TYPOC          ; TYPE IT
8170 054710 104401 052073      TYPE    ,LINSF        ; 2 SPACES
8171 054714 104411          RDLIN          ; GET THE ENTRY
8172 054716 012601          MOV      (SP)+, R1      ; ADDRESS OF ASCII TEXT
8173 054720 004537 055114      JSR      R5, CK.NUM     ; CHECK THE NUMBER
8174 054724 054744          3$              ; CARRIAGE RETURN ONLY ENTERED
8175 054726 055010          7$              ; PERIOD ONLY ENTERED
8176 054730 054674          1$              ; ILLEGAL INPUT
8177 054732 054740          2$              ; TERMINATED WITH A CARRIAGE RETURN
8178 054734 054674          1$              ; TERMINATED WITH A "."
8179 054736 055004          4$              ; TERMINATED WITH A ":"
8180 054740 010260 177776      2$:  MOV      R2, -2(R0)   ; SAVE NEW RPCS1
8181 054744 104401 055103      3$:  TYPE    MRHVEC        ; "RHVEC="
8182 054750 012046          MOV      (R0)+, -(SP)   ; PRESENT RH11 VECTOR ADDRESS ON THE STACK
8183 054752 104402          TYPOC          ; TYPE IT
8184 054754 104401 052073      TYPE    ,LINSF        ; 2 SPACES
8185 054760 104411          RDLIN          ; READ THE ENTRY
8186 054762 012601          MOV      (SP)+, R1      ; ASCII TEXT ADDRESS
8187 054764 004537 055114      JSR      R5, CK.NUM     ; CHECK THE NUMBER
8188 054770 055010          7$              ; CARRIAGE RETURN ONLY ENTERED
8189 054772 055010          7$              ; PERIOD ONLY ENTERED
8190 054774 054744          3$              ; ILLEGAL INPUT
8191 054776 055004          4$              ; TERMINATED WITH A CARRIAGE RETURN
8192 055000 054744          3$              ; TERMINATED WITH A "."
8193 055002 055004          4$              ; TERMINATED WITH A ":"
8194 055004 010260 177776      4$:  MOV      R2, -2(R0)   ; SAVE INPUT
8195 055010 013701 000004      7$:  MOV      ERRVEC, R1     ; SAVE THE ERROR VECTOR
8196 055014 012737 055050 000004      MOV      #8$, ERRVEC    ; SETUP FOR TRAP
8197 055022 005777 124142      TST      @SRPADR        ; CHECK FOR RH11
8198 055026 010137 000004      MOV      R1, ERRVEC     ; RESTORE ERROR VECTOR

```



```

8199 055032 012700 001170      MOV      #SRPADR,R0      ;FIRST ADDRESS OF NEW PARAMETERS
8200 055036 012701 033250      MOV      #RPADR,R1      ;FIRST ADDRESS OF WHERE TO PUT THEM
8201 055042 012021             MOV      (R0)+,(R1)+    ;BUS ADDRESS
8202 055044 012021             MOV      (R0)+,(R1)+    ;VECTOR ADDRESS
8203 055046 000207             RTS      PC              ;RETURN
8204 055050 010137 000004      8$:     MOV      R1,ERRVEC ;RESTORE ERROR VECTOR
8205 055054 022626             CMP      (SP)+,(SP)+    ;CLEAN OFF THE STACK
8206 055056 104006             ERROR   6              ;REPORT THE ERROR
8207 055060 005737 000042      TST     @#42            ;IS THERE A MONITOR?
8208 055064 001703             BEQ     1$              ;NO--GO ASK FOR ADDRESS
8209 055066 000137 005256      JMP     $GET42          ;GO TO END OF PROGRAM
8210
8211 055072 050122 051503 020061 MRPCS1: .ASCIZ @RPCS1 = @
8212 055100 020075          000
8213 055103          122 053110 041505 MRHVEC: .ASCIZ @RHVEC = @
8214 055110 036440 000040
8215
8216          .SBTTL CK.NUM - CHECK NUMBER (OCTAL)
8217          ;THIS ROUTINE CHECKS AN ASCIZ STRING FOR LEGAL CHARACTERS
8218          ;AND FORMS AN OCTAL NUMBER IN R2
8219          ;CALL:
8220          ;
8221          ;
8222          ;
8223          ;
8224          ;
8225          ;
8226          ;
8227          ;
8228          ;
8229          ;
8230 055114 010446      CK.NUM: MOV      R4,-(SP)      ;SAVE R4
8231 055116 010346      MOV      R3,-(SP)      ;SAVE R3
8232 055120 010246      MOV      R2,-(SP)      ;SAVE R2
8233 055122 005004      CLR     R4              ;RETURN POINTER
8234 055124 005003      CLR     R3              ;START NUMBER AT ZERO
8235 055126 005002      CLR     R2              ;STORE RESULT
8236 055130 004537 027510      JSR     R5,CK.CHR      ;CHECK ONE CHARACTER
8237 055134 055232      6$          ;ILLEGAL CHARACTER
8238 055136 055236      8$          ;CARRIAGE RETURN
8239 055140 055232      6$          ;
8240 055142 055234      7$          ;
8241 055144 055150      1$          ;DIGIT 0-7
8242 055146 055232      6$          ;DIGIT 8-9
8243 055150 062705 000004      1$:     ADD     #4,R5      ;INCREMENT RETURN PAST "CR" AND "PERIOD" ONLY RETURNS
8244 055154 006303      2$:     ASL     R3          ;FOR THE OCTAL NUMBER IN R3
8245 055156 103425      BCS     6$          ;DON'T LET IT GET TO BIG
8246 055160 006303      ASL     R3
8247 055162 103423      BCS     6$
8248 055164 006303      ASL     R3
8249 055166 103421      BCS     6$
8250 055170 060203      ADD     R2,R3
8251 055172 004537 027510      JSR     R5,CK.CHR      ;CHECK ONE CHARACTER
8252 055176 055236      8$          ;ILLEGAL CHARACTER
    
```


M15

MD-11-DZRJD-A, RPO4/5/6 MULTI-DRIVE EXERCISER MACY11 27(655) 28-APR-76 17:04 PAGE 157
DZRJD.019 CK.NUM - CHECK NUMBER (OCTAL)

SEQ 0193

8307	055526	051505	040524	052122
8308	055534	052040	042510	050040
8309	055542	047522	051107	046501
8310	055550	005015	000	
8311	055553	015	051412	051531
8312	055560	042524	020115	040510
8313	055566	020123	033061	020113
8314	055574	042515	047515	054522
8315	055602	020054	054047	042130
8316	055610	023520	046040	040517
8317	055616	042504	020122	044527
8318	055624	046114	041040	020105
8319	055632	053117	051105	051127
8320	055640	052111	042524	006516
8321	055646	000012		
8322				
8323	000001			

NOLOAD: .ASCIZ <CR><LF>/SYSTEM HAS 16K MEMORY, 'XXDP' LOADER WILL BE OVERWRITTEN/<CR><L

.END

ABNRML	026454	ABS	= 000200	ACK	= 000123	ACL	= 000040
ACTDRV	033166	ACTSTR	033167	ACU	= 100000	AOE	= 001000
ASGND	052636	ASGN1	024144	ASGN2	024224	ASGN3	024276
ASGN4	024414	ASGN5	024424	ASGN6	024432	ASGN7	024442
ASKPAR	053470	ASNERR	026346	ASNLST	001442	ASNMSG	026366
ASSIGN	024136	ATA	= 100000	ATABIT	033236	ATTN	001244
ATO	= 000001	AT1	= 000002	AT2	= 000004	AT3	= 000010
AT4	= 000020	AT5	= 000040	AT6	= 000100	AT7	= 000200
AUTOCK	001404	AVAIL	001510	A16	= 000400	A17	= 001000
BADENT	053214	BADSEC	001264	BAI	= 000010	BEGCOD	001414
BEGPAT	001412	BEGSIZ	001416	BIT0	= 000001	BIT00	= 000001
BIT01	= 000002	BIT02	= 000004	BIT03	= 000010	BIT04	= 000020
BIT05	= 000040	BIT06	= 000100	BIT07	= 000200	BIT08	= 000400
BIT09	= 001000	BIT1	= 000002	BIT10	= 002000	BIT11	= 004000
BIT12	= 010000	BIT13	= 020000	BIT14	= 040000	BIT15	= 100000
BIT2	= 000004	BIT3	= 000010	BIT4	= 000020	BITS	= 000040
BIT6	= 000100	BIT7	= 000200	BIT8	= 000400	BIT9	= 001000
BLKADR	001720	BPTVEC	= 000014	BUFTBL	001576	BUSADR	054662
BUSY	053236	CFLAG	001262	CHGADR	001260	CI1	034610
CI3	034716	CI4	035024	CI5	035402	CI6	035424
CI7	035440	CI7B	035466	CI8	035540	CKBUS	013020
CKCLK	022342	CKCLK1	022436	CKCLK2	022504	CKCLK3	022534
CKERR	012720	CKFMT	011022	CKHCE	011216	CKSWR	= 104407
CK.CHR	027510	CK.DEC	027462	CK.DIG	027562	CK.NUM	055114
CK.OCT	027434	CLKFLG	001210	CLOCK	023476	CLR	= 000040
CLRDPB	025114	CLRQUE	041350	CMCNT	001312	CMCYL	001314
CMDAT	013366	CMHED	013272	CMPAR	013104	CMPARD	013122
CMPLMT	001374	CMPRES	017460	CMPT	013632	CMPRX	013624
CMSEC	001316	CMSTR	013214	CMTRK	001317	COLON	053125
COMTBL	001740	CR	= 000015	CRLF	= 000200	CSF	= 000002
CSU	= 000010	CYLDER	054374	CYLIMT	001350	DATAPK	025070
DATA0	003006	DATA1	003046	DATE	001220	DATEIS	053127
DCK	= 100000	DCKER	007520	DCKER1	007652	DCL	= 000100
DCU	= 000001	DDISP	= 177570	DEASGN	024512	DEASSG	052566
DE1	= 000040	DFF20	= 000002	DH1	047376	DH14	047554
DH15	047660	DH16	047757	DH2	047403	DH3	047460
DH4	047506	DH6	047545	DIGB	= 000004	DISPLA	001142
DISPLY	= 104414	DISPRE	000174	DLT	= 100000	DL64	= 000020
DMD	= 000001	DONE	007214	DPINT	033142	DPR	= 000400
DPRQS	033152	DRIVE	001242	DRIVED	041606	DRIVE1	042112
DRIVE2	042416	DRIVE3	042722	DRIVE4	043226	DRIVE5	043532
DRIVE6	044036	DRIVE7	044342	DRNUM	052611	DROP	026376
DROPNG	052500	DRQ	= 004000	DRVACT	033112	DRVCLR	= 000111
DRVER	010770	DRVINT	033500	DRVPRM	025310	DRVQUE	041446
DRVSTA	033122	DRVYTP	033132	DRY	= 000200	DSWR	= 177570
DTE	= 010000	DTEER	011626	DTSY	= 000200	DTUW	033234
DT00	= 000001	DT01	= 000002	DT02	= 000004	DT03	= 000010
DT04	= 000020	DT05	= 000040	DT06	= 000100	DT07	= 000200
DT08	= 000400	DT1	050030	DT14	050100	DT15	050122
DT16	050144	DT2	050034	DT3	050052	DT4	050062
DT6	050074	DUNIT	001444	DVA	= 004000	ECBADO	001334
ECBAD1	001342	ECBIT	001320	ECC	014232	ECCX	014764
ECC1	014630	ECC2	014760	ECGD	001332	ECGD1	001340
ECH	= 000100	ECI	= 004000	ECMSK0	001324	ECMSK1	001326

ECSEC	001322	ECWRD	001330	ECWRD1	001336	EMPTY0	041426
EMTVEC =	000030	EM1	044736	EM10	045230	EM11	045273
EM12	045326	EM13	045357	EM14	045431	EM15	045454
EM2	045001	EM20	045510	EM21	045531	EM22	045562
EM3	045635	EM24	045714	EM25	045762	EM26	046043
EM27	046070	EM3	045037	EM30	046125	EM31	046157
EM32	046222	EM33	046255	EM34	046332	EM35	046374
EM36	046422	EM37	046463	EM4	045075	EM40	046545
EM41	046603	EM42	046647	EM43	046730	EM44	046775
EM45	047061	EM46	047147	EM5	045132	EM50	047221
EM51	047210	EM6	045166	EM60	047353	ENDCMP	014106
ENDCON	001352	ENDDET	001410	ENDPAS	052534	ENDPGM =	054404
ENDSEK	001356	ENDTST	052550	ENTADR	053064	ENTCOM	052751
ENTDAT	054516	ENTDRV	052776	ENTID	054535	ENTLMT	053023
ENTPR	004704	EOP	026502	EOPX	026754	EOP1	026534
EOP2	026556	ERCTR	001306	ERPRC1	006574	ERPROC	006560
ERR	= 040000	ERRVEC =	000004	EXT1 =	000001	EXT10 =	000010
EXT2 =	000002	EXT20 =	000020	EXT4 =	000004	EXT40 =	000040
FACTOR	015620	FAIRNS	001254	FALPAR	006760	FALPR1	006770
FEN =	000200	FER =	000020	FILBUF	016172	FMTER	012022
FMT22 =	010000	FORMAT	001376	FRSTER	001300	F1 =	000002
F2 =	000004	F3 =	000010	F4 =	000020	F5 =	000040
GENDPB	044646	GENREG	044666	GETADR	025702	GETBUF	015622
GETID	025572	GETPAR	017252	GETPAT	017224	GETREG =	000141
GETREM	026756	GETREQ	041522	GO =	000001	GODRIV	016320
GRV =	000010	GTSWR =	104406	HCE =	000200	HCEER	012100
HCI =	002000	HCRC =	000400	HRCER	010640	HEDLIN	053162
HOUR	001266	HT =	000011	HZ	001212	IAE =	002000
IAFER	011742	IDIS	053140	IDLE	006132	IDLE1	006230
IF =	000100	ILF =	000001	ILR =	000002	INCHRD	023260
INCMIS	023330	INCSKI	023304	INCSOF	023234	INCTOT	023354
INTDON	053257	INTRVL	001370	INVLD	052727	IOTVEC =	000020
IR =	000100	ISR	036132	IXE =	004000	KSR	023640
KSR1	023674	LA	035774	LACNT	033200	LF =	000012
LIMIT	001310	LINA3	050376	LINB3	050450	LINB5	050625
LINB6	050740	LINCA3	050427	LINC6	050773	LINDA3	050440
LINDEC	022224	LIND5	050610	LINEN3	050361	LINE05	050726
LINEP5	050715	LINE1	020130	LINE2	020174	LINE2A	020344
LINE2B	020362	LINE3	020602	LINE3A	020610	LINE3B	020616
LINE3C	020630	LINE3D	020640	LINE3E	020706	LINE3F	020774
LINE4	021256	LINE5	021346	LINE5A	021446	LINE5B	021540
LINE6	021602	LINE6A	021614	LINE6B	021622	LINE6C	021630
LINE6D	021636	LINE7	021712	LINE7A	022040	LINE8	022160
LING6	051022	LINKDV	027004	LINM3	050261	LINM4	050524
LINN3	050277	LINOCT	022172	LINP3	050323	LINP5	050643
LINR6	051040	LINSP	052073	LINSP0	052074	LINSS3	050507
LINST3	050472	LINS3	050344	LINS4	050543	LINS5	050661
LINT3	050416	LINU06	051051	LINW3	050460	LINX4	050555
LIN10A	051507	LIN10B	051543	LIN10C	051604	LIN10H	051711
LIN11H	051776	LIN2C	050160	LIN2P	050201	LIN2S	050225
LIN3.1	021046	LIN3.3	021160	LIN3.4	021212	LIN4SP	052071
LIN6.1	021644	LIN6.3	021666	LIN7M	051076	LIN70	051124
LIN7P	051134	LIN7R	051231	LIN7S	051154	LIN7T	051203
LIN7X	051216	LIN8M	051246	LIN9B	051303	LIN9E	051430

LIN9G 051460
 LOADRV 055354
 MAIN1 005452
 MATCH 014154
 MAXSEC = 000116
 MCPMX 033246
 MINTRK = 000114
 MNTBL 001770
 MPE = 000400
 MSE = 000020
 MXF = 001000
 M.DP40 027110
 M.DP50 027220
 NEM = 004000
 NOLOAD 055553
 NOTPRS 052216
 OCYL = 100000
 OFLIN 007066
 OFMSG2 002343
 OFMSG6 002523
 OFMTBL 002220
 OF25 = 000001
 OFE = 020000
 OPTER1 011562
 OR = 000200
 PARENT 026222
 PAR1 053520
 PAR15 053641
 PAR20 053665
 PAR6 053562
 PASCNT 001362
 PDONE 052470
 PIP = 020000
 POPQUE 041544
 PRTIM 007156
 PR3 = 000140
 PR7 = 000340
 PSW = 177776
 QDRV0 041150
 QDRV4 041250
 QINPT 041066
 QTERM = 041350
 RANSEC 016456
 RANXIT 017110
 RDDAT = 000171
 RD.ADR 040350
 RD.RP3 040472
 READHD 015246
 RECALO 015174
 RELBUF 015756
 RESVEC = 000010
 RPADR 033250
 RPCC = 000036
 RPDB = 000022

LIN9H 051332
 LST = 002000
 MAIN2 005572
 MAXCYL = 000106
 MAXTRK = 000112
 MHS = 001000
 MINUTE 001270
 MOH = 020000
 MRD = 000020
 MSTCK = 000010
 MXLACT 033256
 M.DP41 027144
 NBA = 100000
 NEWASN 024502
 NOMTCH 012476
 NOTPRT 001406
 OFFCOD 002200
 OFMSGA 002704
 OFMSG3 002377
 OFMSG7 002557
 OFREV = 000200
 OF400 = 000020
 OPERID 001232
 OPRDAT 054404
 ORDERQ 001420
 PARER 011650
 PAR10 053616
 PAR16 053650
 PAR3 053535
 PAR7 053571
 PAT = 000020
 PERIOD 052723
 PIRQ = 177772
 POSER 011442
 PRO = 000000
 PR4 = 000200
 PS = 177776
 PUNSAF 006656
 QDRV1 041170
 QDRV5 041270
 QOUTPT 041106
 QUES 052725
 RANSIZ 016744
 RATIO 001402
 RDHD = 000173
 RD.RP 040324
 RD.RP4 040476
 READIN = 000121
 REDAPK 025102
 RELSE = 000113
 RETRY 001252
 RPAS = 000016
 RPCS1 = 000000
 RPDS1 = 000012

LIN9I 051410
 LSTAD 001256
 MAIN3 006026
 MAXDL 001364
 MCLK = 000002
 MINCYL = 000110
 MINX = 000004
 MOL = 010000
 MRHVEC 055103
 MWR = 000040
 MXWIND 033264
 M.DP42 027154
 NED = 010000
 NEWUNT 001466
 NONE 053205
 NOTRP 052175
 OFFSET = 000115
 OFMSG0 002254
 OFMSG4 002433
 OFMSG8 002613
 OF100 = 000004
 OF50 = 000002
 OPI = 020000
 OPT 034330
 PACK 001216
 PARLST 053372
 PAR11 053625
 PAR19 053656
 PAR4 053544
 PAR8 053600
 PC = %000007
 PGE = 002000
 PIRQVE = 000240
 PROCES 006470
 PR1 = 000040
 PR5 = 000240
 PSEL = 002000
 PWRVEC = 000024
 QDRV2 041210
 QDRV6 041310
 QSTART 041126
 RANCYL 016632
 RANTRK 016532
 RAW = 000020
 RDLIN = 104411
 RD.RP1 040346
 RD.WRD 040352
 RECAL = 000107
 REFMT 006324
 REPLZ 027230
 RMR = 000004
 RPBA = 000004
 RPCS2 = 000010
 RPDT = 000026

LKPAR 004660
 MAIN 005324
 MASK 001250
 MAXER 001366
 MCPE = 020000
 MINSEC = 000120
 MNDLTA 033262
 MONTR 005200
 MRPCS1 055072
 MXDLTA 033260
 M.DPID 027052
 M.DP44 027206
 NEDCLK 052651
 NHS = 002000
 NOTAVL 052233
 NOTSAF 052252
 OFFST 015222
 OFMSG1 002307
 OFMSG5 002467
 OFMSG9 002647
 OF200 = 000010
 OF800 = 000040
 OPTER 011516
 OPTBL 001746
 PAR = 000010
 PARQ 001554
 PAR14 053633
 PAR2 053526
 PAR5 053553
 PAR9 053607
 PCLOCK 001206
 PGM = 001000
 PLU = 020000
 PRTBAD 014772
 PR2 = 000100
 PR6 = 000300
 PSU = 000001
 QCNT 041056
 QDRV3 041230
 QDRV7 041330
 QSTOP 041130
 RANPAT 017100
 RANWRT 017120
 RDCHR = 104410
 RDY = 000200
 RD.RP2 040466
 READDR 022244
 RECALT 015170
 REFMTX 006466
 RESREG = 104413
 RNOP = 000101
 RPCA = 000034
 RPDA = 000006
 RPEC1 = 000044

RPEC2 = 000046
RPER3 = 000042
RPOF = 000032
RPWC = 000002
RPO6 = 052314
R1 = %000001
R5 = %000005
SAVEFG 033210
SC 036460
SC11 037276
SC20 = 002000
SCE 036732
SDETAL 022676
SEEK = 000105
SETFMT = 000143
SIXTEE 001274
SLASH 053514
STACK = 001100
STATHD 052321
STKLMT = 177774
ST02 040116
ST07 040256
SWR 001140
SW00 = 000001
SW04 = 000020
SW08 = 000400
SW11 = 004000
SW15 = 100000
SW5 = 000040
SW9 = 001000
TABLED 053714
TABLE4 054144
TAP = 040000
TIMER 033214
TRAPVE = 000034
TRK1 = 004000
TRK4 = 020000
TYPDS = 104405
TYPON = 104404
ULDFLG 033170
UNSAF 012416
UNTOFF 052104
US2 = 000002
VU30 = 010000
WC 036352
WCKD = 000151
WCU = 000001
WRTDAT = 000161
WRTPK2 017706
WRT.RP 040500
WRT.R4 040662
WSU = 000004
\$BDAT 001126
\$CHARC 031166

RPERRS 033102
RPINIT 033266
RPSN = 000030
RPO4 034036
RTC = 000117
R2 = %000002
R6 = %000006
SAVER1 001302
SCMND 024620
SC12 037366
SC3 036524
SC6A 037042
SEARCH = 000131
SEEKFG 033212
SETVEC 004732
SIZMEM 004510
SP = %000006
START 004066
STATIN 001214
STNDAT 002742
ST03 040166
ST08 040306
SWREG 000176
SW01 = 000002
SW05 = 000040
SW09 = 001000
SW12 = 010000
SW2 = 000004
SW6 = 000100
SYSTAT 052262
TABLE1 053762
TABLE5 054212
TBITVE = 000014
TITLE 055254
TRE = 040000
TRK10 = 040000
TRNSWT 033162
TYPE = 104401
TYPOS = 104403
UNIT 001246
UNTASN 052147
UNTON 052115
US4 = 000004
VV = 000100
WCE = 040000
WCKER 010312
WLE = 004000
WRTHD = 000163
WRTPK3 017732
WRT.R1 040564
WRT.R5 040664
ZROIND 001276
\$BDSEC = 000124
\$CKSWR 031644

RPER1 = 000014
RPLA = 000020
RPTMR 037576
RPO4B 052302
RTNCTR 015144
R3 = %000003
R7 = %000007
SAVERS 001304
SC1 = 000100
SC13 037436
SC4 036530
SC7 037170
SECLMT 001344
SELDRV = 000145
SET.IE 041004
SKI = 040000
SPOTCK 017772
START1 004076
STATIS 015456
ST0 037670
ST05 040212
ST09 040316
SWTIM 007030
SW02 = 000004
SW06 = 000100
SW1 = 000002
SW13 = 020000
SW3 = 000010
SW7 = 000200
T 050274
TABLE2 054030
TABLE6 054260
TD 036176
TKVEC = 000060
TRFER 012162
TRK2 = 010000
TRTVEC = 000014
TYPEST 022626
TYPRI4 027360
UNLOAD = 000103
UNTMSG 052076
UPE = 020000
UWR = 000010
WAIT 001532
WCF = 000040
WCKHD = 000153
WLEER 011774
WRTPK 017476
WRTPK4 017764
WRT.R2 040650
WRT.WD 040566
\$AUTOB 001134
\$BELL 001160
\$CMTAG 001100

RPER2 = 000040
RPMR = 000024
RPVEC 033252
RPO5 052307
RO = %000000
R4 = %000004
S 050320
SAVREG = 104412
SC10 = 001000
SC2 = 000200
SC5 036542
SC8 037246
SECOND 001272
SELPAR 016376
SHDTYP 022654
SKIER 012262
SRCHWT 033164
START2 004102
STATPR 022544
ST01 037720
ST06 040220
SVRH11 040666
SW0 = 000001
SW03 = 000010
SW07 = 000200
SW10 = 002000
SW14 = 040000
SW4 = 000020
SW8 = 000400
TABLE 053674
TABLE3 054076
TABLE7 054326
TDF = 000040
TPVEC = 000064
TRKLMT 001346
TRK20 = 100000
TUF = 000100
TYPOC = 104402
UCPAR 006746
UNS = 040000
UNTNOT 052125
US1 = 000001
VUF = 000002
WAO = 000002
WCFER 012320
WCSEL 001400
WRL = 004000
WRTPK1 017526
WRT.AD 040570
WRT.R3 040656
WRU = 000400
\$BDADR 001122
\$BUF = 000006
\$CM3 = 000000

\$CNTLC 030452	\$CNTLG 032253	\$CNTLU 032246	\$CODE = 000024
\$COMND= 000002	\$CRLF 001165	\$CYL = 000012	\$DBLK 031634
\$DB2D 032500	\$DB20 032674	\$DECVL 032660	\$DOAGN 005320
\$DRVID= 000224	\$DSPLY 027406	\$DTBL 031624	\$ENDAD 005310
\$ERFLG 001103	\$ERMAX 001115	\$ERROR 030460	\$ERRPC 001116
\$ERRTB 004006	\$ERRTY 030616	\$ERTTL 001112	\$FAIR = 000072
\$FILLC 001156	\$FILLS 001155	\$FIRST= 000122	\$FMT = 000001
\$GDADR 001120	\$GDDAT 001124	\$GET42 005256	\$GTSWR 031714
\$HARD = 000062	\$HD = 000000	\$HINUM 032400	\$ICNT 001104
\$INTAG 001135	\$ITEMB 001114	\$LF 001166	\$LKCSB 001176
\$LKCSR 001174	\$LKS 001202	\$LLVEC 001204	\$LONUM 032402
\$LPADR 001106	\$LPERR 001110	\$LPVEC 001200	\$LSTAD 054660
\$MISPO= 000066	\$MNEW 032271	\$MSWR 032260	\$NCODE= 000074
\$NCYL = 000100	\$NEXT = 000104	\$NPATC= 000075	\$NSEC = 000076
\$NTRK = 000077	\$NULL 001154	\$NWRDL= 000102	\$OCNT 031414
\$OCTVL 032776	\$OMODE 031416	\$OPERC= 000036	\$PACK = 000026
\$PASS 001100	\$PASSC= 000070	\$PATTC= 000030	\$POSIT= 000042
\$PREVA= 000032	\$PREVO= 000027	\$PSEL = 000003	\$QUES 001164
\$RAND 032302	\$RDCHR 032126	\$RDLIN 030162	\$RDSZ = 000001
\$READ = 000052	\$REG = 000014	\$RESRE 032442	\$RETRY 015330
\$RPADR 001170	\$RPAS = 000252	\$RPBA = 000240	\$RPCA = 000270
\$RPCC = 000272	\$RPCS1= 000234	\$RPCS2= 000244	\$RPDA = 000242
\$RPDB = 000256	\$RPDS1= 000246	\$RPDT = 000262	\$RPEC1= 000300
\$RPEC2= 000302	\$RPER1= 000250	\$RPER2= 000274	\$RPER3= 000276
\$RPLA = 000254	\$RPMR = 000260	\$RPOF = 000266	\$RPSN = 000264
\$RPVEC 001172	\$RPWC = 000236	\$SAVRE 032404	\$SB2D 027720
\$SB2D 027750	\$SEC = 000010	\$SETUP= 000146	\$SIZE 054564
\$SKI = 000064	\$SOFT = 000060	\$SSEC = 000022	\$STUP = 177777
\$SUPRS 027320	\$SVPC = 000210	\$SWR = 122000	\$STATUS= 000016
\$TERM = 000032	\$TIME 023400	\$TKB 001146	\$TKINT 030000
\$TKS 001144	\$TKSRV 030030	\$TN = 000001	\$TNPWR 032610
\$TOTAL= 000056	\$TPB 001152	\$TPFLG 001157	\$TPS 001150
\$TRANS= 000046	\$TRAP 033014	\$TRAP2 033036	\$TRK = 000011
\$TRP = 000015	\$TRPAD 033050	\$TSTNM 001102	\$TTYIN 030440
\$TYPDS 031420	\$TYPE 030752	\$TYPEC 031122	\$TYPEX 031170
\$TYPOC 031216	\$TYPON 031232	\$TYPOS 031172	\$WRDL = 000020
\$WRDM = 000004	\$OFILL 031415	. = 055650	

ERRORS DETECTED: 0

*DZRJDA, DZRJD/SOL/LI:ME/NL:MC:MD:CND=RPO456.010, DZRJD.019
 RUN-TIME: 105 88 1 SECONDS
 CORE USED: 39K

