

RP04/5/6

MULTIDRIVE LOGIC
CZRJDC0

AH-9197C-MC

COPYRIGHT © 74-77

FICHE 1 OF 1

JAN 1978

digital

MADE IN USA

The microfiche card displays a grid of 14 columns and 24 rows of data frames. Each frame contains technical information, likely logic diagrams or data tables, for the 'MULTIDRIVE LOGIC CZRJDC0' component. The frames are arranged in a regular grid pattern across the card's surface.

801

EJF1CZRP0818Q-11
CZRJDC.F11 15-DEC-77 10:58
.REM 2

00010000MLT-DR780005 MACY11 30(10789)1019+DEC-77 11K08DRP088JBCSEG

00010000

780105
SEG 0001

I D E N T I F I C A T I O N

PRODUCT CODE: AC 9195C-MC
PRODUCT NAME: CZRJDCO MLT-DR LGC
DATE CREATED: APRIL, 1977
MAINTAINER: DIAGNOSTIC ENGINEERING
AUTHOR: C. HESS

COPYRIGHT (C) 1974, 1977 DIGITAL EQUIPMENT CORP., MAYNARD, MASS.

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE
WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A
COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.
DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY
FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT.
THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED
UNDER A LICENSE AND MAY ONLY BE USED OR COPIED IN
ACCORDANCE WITH THE TERMS OF SUCH LICENSE.
DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY
FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT
THAT IS NOT SUPPLIED BY DIGITAL.

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 'Y' 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 EQUIPMENT

REQUIRED

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 COMPARISONS), 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.

NOTE: ON THE FIRST PASS (QUICK VERIFY) THE TIMES ARE APPROXIMATELY ONE EIGHTH OF THE ABOVE VALUES.

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 SUCCESSIVE 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

DEFAULT VALUE

NAME	BASE	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 COMPARISON IF PARAMETER = 0, THE DATA TRANSFER WORD COUNT IS RANDOMLY SELECTED BETWEEN 4 (10) AND THE VALUE IN 'MAXDL'. IF PARAMETER = 1, THE DATA TRANSFER WORD COUNT WILL BE THE VALUE IN 'MAXDL'
ENDET	8	000001	0 OR 1	IF PARAMETER = 1, END OF PASS DETERMINED BY THE 'WORDS READ' COUNT. IF PARAMETER = 0, END OF PASS IS DETERMINED BY THE NUMBER OF SEKS.
FORMAT	8	000001	0 OR 1	IF PARAMETER = 0; DO NOT PERFORM WRITE HEADER & DATA ORDERS; IF PARAMETER > 0 PERFORM WRITE HEADER & DATA ORDERS
RATIO	8	000003	0 - 7	CONTROLS THE APPROXIMATE RATIO OF READ TO WRITE 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
AUTOCK	8	000001	0 OR 1	IF PARAMETER = 1, THE PROGRAM PERFORM WRITE CHECKS AFTER EACH WRITE COMMAND. IF PARAMETER = 0, THE PROGRAM WILL PERFORM WRITE CHECKS RANDOMLY.
NGTPRT	8	000001	0 OR 1	IF PARAMETER = 1, DO NOT PRINT ERROR MESSAGES FOR DATA ERRORS OCCURING 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	STKS	177560	TTY KEYBOARD STATUS REGISTER
1146	STKB	177562	TTY KEYBOARD BUFFER REGISTER
1150	STPS	177564	TTY PRINTER STATUS REGISTER
1152	STPB	177566	TTY PRINTER BUFFER REGISTER
1174	SLKCSR	172540	ADDRESS OF KW11-P STATUS REGISTER
1176	SLKCSB	172542	ADDRESS OF KW11-P COUNTER BUFFER
1200	SLPVEC	104	KW11-P VECTOR ADDRESS
1202	SLKS	177546	ADDRESS OF KW11-L STATUS REGISTER
1204	SLLVEC	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).

LOC	TAG	INITIAL VALUE	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

1416 BEGSIZ 404 4 - MAXDL
 2 = WRITE DATA
 3 = WRITE HEADER & DATA
 4 = READ DATA
 5 = READ HEADER & DATA
 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 TIMEOUT
- 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 TIMEOUT DURING STARTUP. INHIBIT PERFORMANCE SUMMARY TIMEOUTS.
- 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 REASSIGN 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 / RPO4' (OR RPO5 OR RPO6)

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 - 10	THE MINIMUM TRACK ADDRESS
MAXTRK	10	10	0 - 10	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 FOLLOWS:

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 TERMINIATED BY A CARRIAGE RETURN <CR>.

EXAMPLE: S0<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.
WO<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 PAC. 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.
RC<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

RESPONSE	COMMAND(S)
-----	-----
?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, R
?UNIT N UNSAFE	T, W, R
?UNIT N NOT AN RPD4/5/6	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:

RP04/5	RP06
-----	-----
+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	+800 MICRO-INCHES
-1200 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 'RDATA' (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, EM6, EM10, & EM11 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', 'UPE', '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 RH11/RP04/5/6 REGISTERS IN TWO GROUPS:
RPCS1, RPCS2, RPS1, 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 'NGTPRT' MUST BE 0 FOR THE ERROR TO BE REPORTED.

A WORD CALLED 'STATUS' IS DISPLAYED WITH THE RPO4/5/6 REGISTERS. THE CONTENTS OF THIS WORD IDENTIFY HOW THE ERROR WAS PROCESSED BY THE RPO4/5/6 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 CXXY 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 XFRO = 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 BJJFER 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 SUCESSFUL 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 ERROR 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 FOLLOWS 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.

E. PROGRAM DESCRIPTION

E.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

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 NOT 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 TRANSFERRED 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.

B.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.
a)

PROGRAM LISTING

1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784

```

.TITLE CZRJDCO, RPO4/5.6 MLT-DR LGC
.*COPYRIGHT (C) 1975, 1977
.*DIGITAL EQUIPMENT CORP.
.*MAYNARD, MASS. 01754
.*
.*PROGRAM BY C. HESS
.*
.*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
.*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
.*
.SBTTL OPERATIONAL SWITCH SETTINGS
.*
.* SWITCH USE
.* -----

```

170000
170001
170002
170003
170004
170005
170006
170007
170008
170009
170010
170011
170012
170013
170014
170015
170016
170017
170018
170019
170020
170021
170022
170023
170024
170025
170026
170027
170028
170029
170030
170031
170032
170033
170034
170035
170036
170037
170038
170039
170040
170041
170042
170043
170044
170045
170046
170047
170048
170049
170050
170051
170052
170053
170054
170055
170056
170057
170058
170059
170060
170061
170062
170063
170064
170065
170066
170067
170068
170069
170070
170071
170072
170073
170074
170075
170076
170077
170078
170079
170080
170081
170082
170083
170084
170085
170086
170087
170088
170089
170090
170091
170092
170093
170094
170095
170096
170097
170098
170099
170100

```

: *           15      HALT ON ERROR
: *           13      INHIBIT ERROR TIMEOUTS
: *           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
: *           4       B. NO ECC CORRECTION RESULTS DISPLAYED IF ERROR
: *           4       A. DO NOT CHECK FOR MAXIMUM ERROR COUNTS
: *           3       B. DO NOT DROP DRIVE WHEN NORMAL END OF TEST REACHED
: *           3       A. DISPLAY ERROR SECTOR IF 'DCK', 'DTE', OR 'WCF' ERROR
: *           3       B. DISPLAY SECTOR IF 'DCK' ERR UNCORRECTABLE AFTER
: *           3       28TH RETRY
: *           3       C. IF DATA COMPARE ERROR & SW7 SET, DISPLAY
: *           3       REMAINDER OF BUFFER
: *           2       A. DON'T TYPE SUBSYSTEM STATUS WHEN PROGRAM STARTED
: *           2       B. DON'T TYPE PERFORMANCE SUMMARY
: *           1       INHIBIT DATA COMPARSION AFTER READ ORDERS
: *           0       READ ONLY MODE

```

.SBTTL BASIC DEFINITIONS

.*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***

001100

```

STACK= 1100
.EQUIV EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
.EQUIV IOT,SCOPE      ;;BASIC DEFINITION OF SCOPE CALL

```

.*MISCELLANEOUS DEFINITIONS

000011
000012
000015
000200
177776
177774
177772
177570
177570

```

MT= 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

000000
000001
000002
000003
000004
000005
000006
000007
00000E
000007

```

R0= %0      ;;GENERAL REGISTER
R1= %1      ;;GENERAL REGISTER
R2= %2      ;;GENERAL REGISTER
R3= %3      ;;GENERAL REGISTER
R4= %4      ;;GENERAL REGISTER
R5= %5      ;;GENERAL REGISTER
R6= %6      ;;GENERAL REGISTER
R7= %7      ;;GENERAL REGISTER
SP= %6      ;;STACK POINTER
PC= %7      ;;PROGRAM COUNTER

```

.*PRIORITY LEVEL DEFINITIONS

000000
000040
000100
000140
000200
000240

```

PR0= 0      ;;PRIORITY LEVEL 0
PR1= 40     ;;PRIORITY LEVEL 1
PR2= 100    ;;PRIORITY LEVEL 2
PR3= 140    ;;PRIORITY LEVEL 3
PR4= 200    ;;PRIORITY LEVEL 4
PR5= 240    ;;PRIORITY LEVEL 5

```

1841 000300
 1842 000340
 1843
 1844
 1845 100000
 1846 040000
 1847 020000
 1848 010000
 1849 004000
 1850 002000
 1851 001000
 1852 000400
 1853 000200
 1854 000100
 1855 000040
 1856 000020
 1857 000010
 1858 000004
 1859 000002
 1860 000001

PR6= 300 ::PRIORITY LEVEL 6
 PR7= 340 ::PRIORITY LEVEL 7

.*"SWITCH REGISTER" SWITCH DEFINITIONS

SW15= 100000
 SW14= 40000
 SW13= 20000
 SW12= 10000
 SW11= 4000
 SW10= 2000
 SW09= 1000
 SW08= 400
 SW07= 200
 SW06= 100
 SW05= 40
 SW04= 20
 SW03= 10
 SW02= 4
 SW01= 2
 SW00= 1
 .EQUIV SW09,SW9
 .EQUIV SW08,SW8
 .EQUIV SW07,SW7
 .EQUIV SW06,SW6
 .EQUIV SW05,SW5
 .EQUIV SW04,SW4
 .EQUIV SW03,SW3
 .EQUIV SW02,SW2
 .EQUIV SW01,SW1
 .EQUIV SW00,SW0

1871
 1872
 1873 100000
 1874 040000
 1875 020000
 1876 010000
 1877 004000
 1878 002000
 1879 001000
 1880 000400
 1881 000200
 1882 000100
 1883 000040
 1884 000020
 1885 000010
 1886 000004
 1887 000002
 1888 000001

.*DATA BIT DEFINITIONS (BIT00 TO BIT15)

BIT15= 100000
 BIT14= 40000
 BIT13= 20000
 BIT12= 10000
 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

1889
 1890
 1891
 1892
 1893
 1894
 1895
 1896

```
1897 .EQUIV BIT01,BIT1
1898 .EQUIV BIT00,BIT0
1899
1900 :*BASIC "CPU" TRAP VECTOR ADDRESSES
1901 000004 ERRVEC= 4 ;: TIME OUT AND OTHER ERRORS
1902 000010 RESVEC= 10 ;: RESERVED AND ILLEGAL INSTRUCTIONS
1903 000014 TBITVEC=14 ;: "T" BIT
1904 000014 TRIVEC= 14 ;: TRACE TRAP
1905 000014 BPTVEC= 14 ;: BREAKPOINT TRAP (BPT)
1906 000020 IOTVEC= 20 ;: INPUT/OUTPUT TRAP (IOT) **SCOPE**
1907 000024 PWRVEC= 24 ;: POWER FAIL
1908 000030 EMTVEC= 30 ;: EMULATOR TRAP (EMT) **ERROR**
1909 000034 TRAPVEC=34 ;: "TRAP" TRAP
1910 000060 TKVEC= 60 ;: TTY KEYBOARD VECTOR
1911 000064 TPVEC= 64 ;: TTY PRINTER VECTOR
1912 000240 PIQVEC=240 ;: PROGRAM INTERRUPT REQUEST VECTOR
1913
1914 ;:*****
1915
1916 .SBTTL RH11 REGISTERS
1917
1918 ;:*****
1919
1920 :CONTROL AND STATUS REGISTER 1 (RPCS1)
1921
1922 000100 IE= 100 ;: INTERRUPT ENABLE (BIT #6)
1923 000200 RDY= 200 ;: READY (BIT #7)
1924 000400 A16= 400 ;: HIGH ORDER BUS ADDRESS BIT (BIT #8)
1925 001000 A17= 1000 ;: HIGH ORDER BUS ADDRESS BIT (BIT #9)
1926 002000 PSEL= 2000 ;: PORT SELECT (BIT #10)
1927 020000 MCPE= 20000 ;: MASSBUS PARITY ERROR (BIT #13)
1928 040000 TRE= 40000 ;: TRANSFER ERROR (BIT #14)
1929 ;SC= 100000 ;: SPECIAL CONDITION (BIT #15)
1930
1931 :WORD COUNT REGISTER (RPWC)
1932 ;(EACH BIT IS CALLED BY BIT NUMBER)
1933
1934 :BUS ADDRESS REGISTER (RPBA)
1935 ;(EACH BIT IS CALLED BY BIT NUMBER)
1936
1937 :CONTROL AND STATUS REGISTER 2 (RPCS2)
1938
1939 000001 US1= 1 ;: UNIT SELECT (BIT #0)
1940 000002 US2= 2 ;: UNIT SELECT (BIT #1)
1941 000004 US4= 4 ;: UNIT SELECT (BIT #2)
1942 000010 BAI= 10 ;: BUS ADDRESS INCREMENT INHIBIT (BIT #3)
1943 000020 PAT= 20 ;: MASSBUS PARITY TEST (BIT #4)
1944 000040 CLR= 40 ;: CLEAR (BIT #5)
1945 000100 IR= 100 ;: INPUT READY (BIT #6)
1946 000200 OR= 200 ;: OUTPUT READY (BIT #7)
1947 000400 MPE= 400 ;: MASS BUS PARITY ERROR (BIT #8)
1948 001000 MXF= 1000 ;: MISSED TRANSFER ERROR (BIT #9)
1949 002000 PGE= 2000 ;: PROGRAM ERROR (BIT #10)
1950 004000 NEM= 4000 ;: NON EXISTENT MEMORY (BIT #11)
1951 010000 NED= 10000 ;: NON EXISTENT DRIVE (BIT #12)
1952 020000 UFE= 20000 ;: UNIBUS PARITY ERROR (BIT #13)
```

```

1953      040000      WCE=    40000      ;WRITE CHECK ERROR (BIT #14)
1954      100000      DLT=    100000     ;DATA LATE (BIT #15)
1955
1956      ;DATA BUFFER REGISTER (RPDB)
1957      ;(EACH BIT IS CALLED BY BIT NUMBER)
1958
1959
1960      ;*****
1961      .SBTTL RPO4/5/6 REGISTERS
1962
1963      ;*****
1964
1965      ;CONTROL AND STATUS 1 REGISTER. (#00)
1966
1967
1968      000001      GO=      1      ;GO BIT (BIT #0)
1969      000002      F1=      2      ;FUNCTION CODE BIT #1
1970      000004      F2=      4      ;FUNCTION CODE BIT #2
1971      000010      F3=     10      ;FUNCTION CODE BIT #3
1972      000020      F4=     20      ;FUNCTION CODE BIT #4
1973      000040      F5=     40      ;FUNCTION CODE BIT #5
1974      004000      DVA=    4000     ;DEVICE AVAILABLE (BIT #11)
1975
1976      ;DRIVE STATUS REGISTER (RPDS1) (#01)
1977
1978      ;DFS=      1      DRIVE FORWARD 5"/SEC. (BIT #0)
1979      000002      DFF20=  2      ;DRIVE FORWARD 20"/SEC. (BIT #1)
1980      000004      DIG8=   4      ;DRIVE TO INNER GUARD BAND (BIT #2)
1981      000010      GRV=   10      ;GO REVERSE (BIT #3)
1982      000020      DL64=  20      ;DIFFERENCE LESS THAN 64 (BIT #4)
1983      000040      DE1=   40      ;DIFFERENCE EQUALS 1 (BIT #5)
1984      000100      VV=   100     ;VOLUME VALID (BIT #6)
1985      000200      DRY=   200     ;DRIVE READY (BIT #7)
1986      000400      DPR=   400     ;DRIVE PRESENT (BIT #8)
1987      001000      PGM=  1000     ;PROGRAMABLE (BIT #9)
1988      002000      LST=  2000     ;LAST SECTOR TRANSFERRED (BIT #10)
1989      004000      WRL=  4000     ;WRITE LOCK (BIT #11)
1990      010000      MOL= 10000     ;MEDIUM ON-LINE (BIT #12)
1991      020000      PIP= 20000     ;POSITIONING OPERATION IN PROGRESS (BIT #13)
1992      040000      ERR= 40000     ;COMPOSITE ERROR (BIT #14)
1993      100000      ATA=100000     ;ATTENTION ACTIVE (BIT #15)
1994
1995      ;ERROR REGISTER #01 (RPER1) (#02)
1996
1997      000001      ILF=      1      ;ILLEGAL FUNCTION (BIT #0)
1998      000002      ILR=      2      ;ILLEGAL REGISTER (BIT #1)
1999      000004      RMR=      4      ;REGISTER MODIFICATION REFUSED (BIT #2)
2000      000010      PAR=     10      ;PARITY ERROR (BIT #3)
2001      000020      FER=     20      ;FORMAT ERROR (BIT #4)
2002      000040      WCF=     40      ;WRITE CLOCK FAIL (BIT #5)
2003      000100      ECH=    100     ;ECC HARD ERROR (BIT #6)
2004      000200      HCE=    200     ;HEADER COMPARE ERROR (BIT #7)
2005      000400      HCRC=   400     ;HEADER CRC ERROR (BIT #8)
2006      001000      ACE=  1000     ;ADDRESS OVERFLOW ERROR (BIT #9)
2007      002000      IAE=  2000     ;INVALID ADDRESS ERROR (BIT #10)
2008      004000      WLE=  4000     ;WRITE LOCK ERROR (BIT #11)

```

```

2009      010000      DTE=      10000      ;DRIVE TIMING ERROR (BIT #12)
2010      020000      OPI=      20000      ;OPERATION INCOMPLETE (BIT #13)
2011      040000      UNS=      40000      ;DRIVE UNSAFE (BIT #14)
2012      100000      DCK=      100000     ;DATA CHECK ERROR (BIT 15)
2013
2014      ;MAINTAINABILITY REGISTER (RPMR) (#03)
2015
2016      000001      DMD=       1      ;DIAGINOSTIC MODE (BIT #0)
2017      000002      MCLK=      2      ;MAINTAINABILITY CLOCK (BIT #1)
2018      000004      MINX=      4      ;MAINTAINABILITY INDEX (BIT #2)
2019      000010      MSTCK=     10     ;MAINTAINABILITY SECTOR CLOCK (BIT #3)
2020      000020      MRD=      20     ;MAINTAINABILITY READ (BIT #4)
2021      000040      MWR=      40     ;MAINTAINABILITY WRITE (BIT #5)
2022      000200      DTSY=     200     ;MAINTAINABILITY SYNC DETECTED (BIT #7)
2023
2024      ;ATTENTION SUMMARY PSEUDO-REGISTER (RPAS) (#04)
2025
2026      000001      ATO=       1      ;DEVICE 0 (BIT #0)
2027      000002      AT1=      2      ;DEVICE 1 (BIT #1)
2028      000004      AT2=      4      ;DEVICE 2 (BIT #2)
2029      000010      AT3=     10      ;DEVICE 3 (BIT #3)
2030      000020      AT4=     20      ;DEVICE 4 (BIT #4)
2031      000040      AT5=     40      ;DEVICE 5 (BIT #5)
2032      000100      AT6=    100      ;DEVICE 6 (BIT #6)
2033      000200      AT7=    200      ;DEVICE 7 (BIT #7)
2034
2035      ;DESIRED SECTOR/TRACK ADDRESS REGISTER (RPDA) (#05)
2036      ;(EACH BIT IS CALLED BY BIT NUMBER)
2037
2038      ;DRIVE TYPE REGISTER (RPDT) (#06)
2039
2040      000001      DT00=      1      ;DRIVE TYPE NUMBER BIT 1
2041      000002      DT01=      2      ;DRIVE TYPE NUMBER BIT 2
2042      000004      DT02=      4      ;DRIVE TYPE NUMBER BIT 3
2043      000010      DT03=     10      ;DRIVE TYPE NUMBER BIT 4
2044      000020      DT04=     20      ;DRIVE TYPE NUMBER BIT 5
2045      000040      DT05=     40      ;DRIVE TYPE NUMBER BIT 6
2046      000100      DT06=    100      ;DRIVE TYPE NUMBER BIT 7
2047      000200      DT07=    200      ;DRIVE TYPE NUMBER BIT 8
2048      000400      DT08=    400      ;DRIVE TYPE NUMBER BIT 9
2049      004000      DRQ=     4000     ;DRIVE REQUEST REQUIRED (BIT #11)
2050      020000      MOH=     2000     ;MOVING HEAD (BIT #13)
2051      040000      TAP=     4000     ;TAPE DRIVE (BIT #14)
2052      100000      NBA=    10000     ;NOT BLOCK ADDRESSED (BIT #15)
2053
2054      ;LOOK-AHEAD REGISTER (RPLA) (#07)
2055
2056      000001      EXT1=      1      ;EXTENSION 1 (BIT #0)
2057      000002      EXT2=      2      ;EXTENSION 2 (BIT #1)
2058      000004      EXT4=      4      ;EXTENSION 3 (BIT #2)
2059      000010      EXT10=     10     ;EXTENSION 4 (BIT #3)
2060      000020      EXT20=     20     ;EXTENSION 5 (BIT #4)
2061      000040      EXT40=     40     ;EXTENSION 6 (BIT #5)
2062      000100      SC1=     100     ;SECTOR COUNT FIELD 0 (BIT #6)
2063      000200      SC2=     200     ;SECTOR COUNT FIELD 1 (BIT #7)
2064      ;SC4=     400     ;SECTOR COUNT FIELD 2 (BIT #8)

```

2065	001000	SC10= 1000	:SECTOR COUNT FIELD 3 (BIT #9)
2066	002000	SC20= 2000	:SECTOR COUNT FIELD 4 (BIT #10)
2067	004000	TRK1= 4000	:TRACK FIELD 1 (BIT #11)
2068	010000	TRK2= 10000	:TRACK FIELD 2 (BIT #12)
2069	020000	TRK4= 20000	:TRACK FIELD 3 (BIT #13)
2070	040000	TRK10= 40000	:TRACK FIELD 4 (BIT #14)
2071	100000	TRK20= 100000	:TRACK FIELD 5 (BIT #15)
2072			
2073		:RPO4 ERROR REGISTER #2 (RPER2) (#10)	
2074			
2075	000001	WCU= 1	:WRITE CURRENT UNSAFE (BIT #0)
2076	000002	CSF= 2	:CURRENT SINK FAILURE (BIT #1)
2077	000004	WSU= 4	:WRITE SELECT UNSAFE (BIT #2)
2078	000010	CSU= 10	:CURRENT SWITCH UNSAFE (BIT #3)
2079	000020	MSE= 20	:MOTOR SEQUENCE ERROR (BIT #4)
2080	000040	TDF= 40	:TRANSITIONS DETECTOR FAILURE (BIT #5)
2081	000100	TUF= 100	:TRANSITIONS UNSAFE (BIT #6)
2082	000200	FEN= 200	:FAILSAFE ENABLED (BIT #7)
2083	000400	WRU= 400	:WRITE READY UNSAFE (BIT #8)
2084	001000	MHS= 1000	:MULTIPLE HEAD SELECT (BIT #9)
2085	002000	NHS= 2000	:NO HEAD SELECTION (BIT #10)
2086	004000	IXE= 4000	:INDEX ERROR (BIT #11)
2087	010000	VU30= 10000	:30VOLT UNSAFE (BIT #12)
2088	020000	PLU= 20000	:PLO UNSAFE (BIT #13)
2089	100000	ACU= 100000	:AC UNSAFE (BIT #15)
2090			
2091		:RPO5/6 ERROR REGISTER #02 (RPER2) (#10)	
2092			
2093	000001	WCU= 1	:WRITE CURRENT UNSAFE (BIT #0)
2094	000002	CSF= 2	:CURRENT SINK FAILURE (BIT #1)
2095	000004	WSU= 4	:WRITE SELECT UNSAFE (BIT #2)
2096	000010	CSU= 10	:CURRENT SWITCH UNSAFE (BIT #3)
2097	000020	RAW= 20	:READ AND WRITE (BIT #4)
2098	000040	TDF= 40	:TRANSITIONS DETECTOR FAILURE (BIT #5)
2099	000100	TUF= 100	:TRANSITIONS UNSAFE (BIT #6)
2100	000200	ABS= 200	:ABNORMAL STOP (BIT #7)
2101	000400	WRU= 400	:WRITE READY UNSAFE (BIT #8)
2102	001000	MHS= 1000	:MULTIPLE HEAD SELECT (BIT #9)
2103	002000	NHS= 2000	:NO HEAD SELECTION (BIT #10)
2104	004000	IXE= 4000	:INDEX ERROR (BIT #11)
2105	020000	PLU= 20000	:PLO UNSAFE (BIT #12)
2106			
2107		:OFFSET REGISTER (RPOF) (#11)	
2108			
2109	000001	OF25= 1	:OFFSET 25 MICRO INCHES (BIT #0)
2110	000002	OF50= 2	:OFFSET 50 MICRO INCHES (BIT #1)
2111	000004	OF100= 4	:OFFSET 100 MICRO INCHES (BIT #2)
2112	000010	OF200= 10	:OFFSET 200 MICRO INCHES (BIT #3)
2113	000020	OF400= 20	:OFFSET 400 MICRO INCHES (BIT #4)
2114	000040	OF800= 40	:OFFSET 800 MICRO INCHES (BIT #5)
2115	000200	OFREV= 200	:OFFSET NEGATIVE (REVERSE) (BIT #5)
2116	002000	HCI= 2000	:HEADER COMPARE INHIBIT (BIT #10)
2117	004000	ECI= 4000	:ERROR CORRECTION CODE INHIBIT (BIT #11)
2118	010000	FMT22= 10000	:FORMAT BIT (BIT #12)
2119			
2120		:DESIRED CYLINDER ADDRESS (RPCA) (#12)	

121
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
164
165
166
167
168
169
170
171
172
173
174
175
176

000001
000002
000010
000040
000100
040000
100000

000001
000002
000040
000100
020000
040000
100000

000101
000103
000105
000107
000111
000113
000115
000117
000121
000123
000131
000141
000143
000145
000151
000153

;(EACH BIT IS CALLED BY BIT NUMBER)

;CURRENT CYLINDER ADDRESS (RPCC) (#13)
;(EACH BIT IS CALLED BY BIT NUMBER)

;SERIAL NUMBER REGISTER (RPSN) (#14)
;(EACH IS CALLED BY BIT NUMBER)

;RPO4 ERROR REGISTER #03 (RPER3) (#15)

PSU= 1 ;PACK SPEED UNSAFE (BIT #0)
VUF= 2 ;VELOCITY UNSAFE (BIT #1)
UWR= 10 ;ANY UNSAFE EXCEPT READ/WRITE (BIT #3)
ACL= 40 ;AC LOW (BIT #5)
DCL= 100 ;DC LOW (BIT #6)
SKI= 40000 ;SEEK INCOMPLETE (BIT #14)
OCYL= 100000 ;OFF CYLINDER (BIT #15)

;RPO5/6 ERROR REGISTER #03 (RPER3) (#15)

DCU= 1 ;DC UNSAFE (BIT #0)
WAO= 2 ;WRITE AND OFFSET (BIT #1)
ACL= 40 ;AC LOW (BIT #5)
DCL= 100 ;DC LOW (BIT #6)
OPE= 20000 ;OPERATOR PLUG ERROR (BIT #13)
SKI= 40000 ;SEEK INCOMPLETE (BIT #14)
OCYL= 100000 ;OFF CYLINDER ERROR (BIT #15)

;ECC POSITION REGISTER (RPEC1) (#16)
;(EACH BIT IS CALLED BY BIT NUMBER)

;ECC PATTERN REGISTER (RPEC2) (#17)
;(EACH BIT IS CALLED BY BIT NUMBER)

;;*****

SBTTL RPO4/5/6 DRIVER COMMANDS

;;*****

RNOP = 101 ;NO OPERATION
UNLOAD = 103 ;UNLOAD
SEEK = 105 ;SEEK
RECAL = 107 ;RECALIBRATE
DRVCLR = 111 ;DRIVE CLEAR
RELSE = 113 ;RELEASE
OFFSET = 115 ;OFFSET
RTC = 117 ;RETURN TO CENTER LINE
READIN = 121 ;READ IN PRESET
ACK = 123 ;PACK ACKNOWLEDGE
SEARCH = 131 ;SEARCH
GETREG = 141 ;GET REGISTERS
SETFMT = 143 ;SET FORMAT (& ECI OR HCI)
SELDRV = 145 ;SELECT DRIVE
WCKD = 151 ;WRITE CHECK DATA
WCKHD = 153 ;WRITE CHECK HEADER & DATA

```

000161      WRTDAT =      161      ;WRITE DATA
000163      WRTHD  =      163      ;WRITE HEADER & DATA
000171      RDDAT  =      171      ;READ DATA
000173      RDHD   =      173      ;READ HEADER & DATA

.SBTTL TRAP CATCHER

      .=0
; *ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
; *SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
; *LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
      .=174
DISPREG: .WORD 0      ;: SOFTWARE DISPLAY REGISTER
SWREG:   .WORD 0      ;: SOFTWARE SWITCH REGISTER
.SBTTL STARTING ADDRESS(ES)
      JMP @#START1    ;: JUMP TO STARTING ADDRESS OF PROGRAM
      JMP @#START      ;: CHANGE THE RH11 UNIBUS ADDRESS
;: AFTER INITIAL START

.SBTTL ACT11 HOOKS

;: *****
;: HOOKS REQUIRED BY ACT11
      $SVPC=.          ;: SAVE PC
      .=46
      $ENDAD          ;: 1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
      .=52
      .WORD 40000     ;: 2)SET LOC.52 TO 40000
      .=$SVPC         ;: RESTORE PC

```

2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261

.SBTTL COMMON TAGS

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

001100
000000
000
000
000000
000000
000000
000000
000000
000
001
000000
000000
000000
000000
000000
000000
000000
000
000
000000
177570
177570
177560
177562
177564
177566
000
002
012
000
177607
077
015
000012

000377

. = 1100
SCMTAG: .WORD 0
\$PASS: .BYTE 0
\$STNM: .BYTE 0
\$ERFLG: .BYTE 0
\$ICNT: .WORD 0
\$LPADR: .WORD 0
\$LPERR: .WORD 0
\$ERTTL: .WORD 0
\$ITEMB: .BYTE 0
\$ERMAX: .BYTE 1
\$ERRPC: .WORD 0
\$GDADR: .WORD 0
\$BDADR: .WORD 0
\$GDDAT: .WORD 0
\$BDDAT: .WORD 0
\$AUTOB: .BYTE 0
\$INTAG: .BYTE 0
\$SWR: .WORD DSWR
DISPLAY: .WORD DDISP
\$TKS: 177560
\$TKB: 177562
\$TPS: 177564
\$TPB: 177566
\$NULL: .BYTE 0
\$FILLS: .BYTE 2
\$FILLC: .BYTE 12
\$TPFLG: .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
\$STATIN: .WORD 0
PACK: .WORD 0

: 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
: 'TYPE STATISTICS' INDICATOR
: 'W' COMMAND INDICATOR

```

2262 001220 000000 000000 000000 DATE: .WORD 0,0,0,0,0 ; OPERATOR ENTERED DATE
2263 001226 000000 000000 000000
2264 001232 000000 000000 000000 OPERID: .WORD 0,0,0,0 ; OPERATOR ID
2265 001240 000000
2266 001242 000000 DRIVE: .WORD 0 ; DRIVE # STORAGE: ERRORS 1-5 & 10
2267 001244 000000 ATTN: .WORD 0 ; ATTN REG STORAGE: ERRORS 1-5 & 10
2268 001246 000000 UNIT: .WORD 0 ; DRIVE # STORAGE FOR PRINTOUT
2269 001250 000000 MASK: .WORD 0 ; ERROR RETRY REGISTER MASK
2270 001252 000 000 RETRY: .BYTE 0,0 ; ERROR RETRY LIMIT IN THE LOWER BYTE
2271 ; RETRY COUNT IN THE UPPER BYTE
2272 001254 000003 FAIRNS: .WORD 3 ; MAXIMUM TIME IN QUEUE VALUE
2273 001256 000000 LSTAD: .WORD 0 ; STORE LAST MEMORY ADDRESS HERE
2274 001260 000000 CHGADR: .WORD 0 ; CHANGE RH11 UNIBUS ADDRESS FLAG
2275 001262 000000 CFLAG: .WORD 0 ; 'CONTROL C' FLAG
2276 001264 000000 BADSEC: .WORD 0 ; BAD SECTOR/TRACK FLAG
2277 001266 000000 HOUR: .WORD 0 ; HOUR COUNT STORED HERE (MAXIMUM - 999.)
2278 001270 000000 MINUTE: .WORD 0 ; MINUTE'S COUNT STORED HERE
2279 001272 000000 SECOND: .WORD 0 ; SECOND'S COUNT STORED HERE
2280 001274 000000 SIXTEE: .WORD 0 ; TIMER ROUTINE COUNTER (FOR ONE SECOND)
2281 001276 177777 ZROIND: .WORD -1 ; ZERO INDICATOR FOR THE DATA COMPARE ROUTINE
2282 001300 000 FRSTER: .BYTE 0 ; DATA COMPARE ERROR FLAG
2283 ; IF > 0, PROCESSING 'DCKER' OR CAN'T MATCH PATTERN
2284 ; IF < 0, MISCOMPARISON FOUND
2285 001301 000 .BYTE 0 ; MISCOMPARISON OR CAN'T MATCH PATTERN FLAG
2286 ; IF < 0, ERROR IN BUFFER
2287 001302 000000 SAVER1: .WORD 0 ; SAVE R1 HERE
2288 001304 000000 SAVER5: .WORD 0 ; SAVE R5 HERE
2289 001306 000000 ERCTR: .WORD 0 ; NUMBER OF ERRORS
2290 001310 000000 LIMIT: .WORD 0 ; DISPLAY LIMIT
2291 001312 000000 CMCNT: .WORD 0 ; WORD COUNT
2292 001314 000000 CMCYL: .WORD 0 ; CYLINDER ADDRESS
2293 001316 000 CMSEC: .BYTE 0 ; SECTOR ADDRESS
2294 001317 000 CMTRK: .BYTE 0 ; TRACK ADDRESS
2295 001320 000000 ECBIT: .WORD 0 ; ERROR BURST BIT OFFSET
2296 001322 000000 ECSEC: .WORD 0 ; ERROR BURST WORD OFFSET (RELATIVE TO SECTOR)
2297 001324 000000 ECMSK0: .WORD 0 ; CORRECTION MASK FOR FIRST ERROR WORD
2298 001326 000000 ECMSK1: .WORD 0 ; CORRECTION MASK FOR SECOND ERROR WORD
2299 001330 000000 ECWRD: .WORD 0 ; LOCATION OF FIRST ERROR WORD
2300 001332 000000 ECGD: .WORD 0 ; GOOD DATA, FIRST WORD
2301 001334 000000 ECBAD0: .WORD 0 ; BAD DATA, FIRST WORD
2302 001336 000000 ECWRD1: .WORD 0 ; LOCATION OF SECOND ERROR WORD
2303 001340 000000 ECGD1: .WORD 0 ; GOOD DATA, SECOND WORD
2304 001342 000000 ECBAD1: .WORD 0 ; BAD DATA, SECOND WORD
2305 001344 000025 SECLMT: .WORD 21 ; SECTOR ADDRESS LIMIT
2306 001346 000022 TRKLMT: .WORD 18 ; TRACK ADDRESS LIMIT
2307 001350 000632 CYLIMT: .WORD 410 ; CYLINDER ADDRESS LIMIT FOR RP04'S
2308 ; (CHANGED TO 814. FOR RP06)
2309
2310 ; *****
2311 ; *****
2312 .SBTTL COMMON PARAMETERS
2313 ; *****
2314 ; *****
2315 ; *****
2316 ; PROGRAM USES THESE PARAMETERS TO DETERMINE REGULAR END OF PASS
2317 001352 002740 ENDCN: .WORD 002740 ; 1.875X10+8 WORDS (10 [3X10+9 BITS])

```

2318 001354 005455
2319 001356 143300
2320 001360 000055
2321
2322 001362 120274
2323 001364 000005
2324 001366 134330
2325 001370 000005
2326
2327
2328
2329 001372 000000
2330 001374 000000
2331 001376 000000
2332 001400 000000
2333
2334 001402 000001
2335 001404 000000
2336
2337
2338 001406 000144
2339 001410 000005 000000
2340
2341
2342
2343 001414 000004
2344 001416 000001
2345
2346 001420 000000
2347
2348
2349 001422 000003
2350
2351
2352
2353
2354
2355
2356
2357
2358 001424 000001
2359
2360
2361
2362 001426 000001
2363
2364
2365
2366
2367
2368 001430 000001
2369
2370
2371
2372

ENDSK: .WORD 005455 ;MSW
;3 X 10+6 SEEKS (LSW)
;MSW
;PROGRAM USES THESE PARAMETERS TO DETERMINE Q.V. END OF PASS
QVCON: .WORD 120274 ;2.3437X10+7 WORDS (10)
;MSW
QVSEK: .WORD 134330 ;3.75 X 10+5 SEEKS (10)
;MSW
;THE NUMBERS TO DETERMINE END OF PASS ARE LOADED IN HERE BY THE PROGRAM.
;THE FIRST TIME THROUGH, THE QV PARAMETERS ARE USED, AFTER THAT THE
;REGULAR PARAMETERS ARE USED.
ENDCON: .WORD 0
;MSW
ENDSEK: .WORD 0
;MSW
PASCNT: .WORD 1 ;NUMBER OF PASSES TO END OF TEST
MAXDL: .WORD 0 ;MAXIMUM DATA TRANSFER SIZE IN WORDS
;(FILLED BY PROGRAM AT STARTUP OR BY OPERATOR
;DURING PARAMETER ENTRY DIALOG.)
MAXER: .WORD 100. ;MAXIMUM ERRORS - 100(10)
INTRVL: .WORD 5,0 ;FIRST WORD IS THE PERFORMANCE TYPEOUT INTERVAL
;(IN MINUTES). SECOND WORD IS THE INTERVAL
;COUNTER
;COUNTER. UPPER BYTE IS VALUE.
CPLMT: .WORD 4 ;NUMBER OF COMPARE ERRORS TYPED OUT
FORMAT: .WORD 1 ;IF NOT EQ 0, ALLOW WRITE HEADER & DATA ORDERS
;IF EQ 0, DO NOT ALLOW WRITE HEADER & DATA ORDERS
WCSEL: .WORD 0 ;IF EQ TO 0, GENERATE A RANDOM WORD COUNT
;FOR THE OPERATION.
;IF NOT EQ TO 0, USE THE VALUE IN 'MAXDL' FOR
;THE WORD COUNT
RATIO: .WORD 3 ;READ/WRITE RATIO (RANGE 0 - 7)
0 - 0/8 (READ/WRITE)
1 - 7/1
2 - 6/2
3 - 5/3
4 - 4/4
5 - 3/5
6 - 2/6
7 - 1/7
AUTOCK: .WORD 1 ;IF NOT EQ 0, DO AN APPROPRIATE WRITE
;CHECK AFTER EACH WRITE ORDER.
NOTPRT: .WORD 1 ;IF EQ 0, SELECT WRITE CHECK ORDERS
;RANDOMLY.
;IF EQ 1, DO NOT PRINT DATA ERROR MESSAGES
;ASSOCIATED WITH OPERATOR SPECIFIED
;BAD PACK AREAS.
ENDET: .WORD 1 ;IF NOT EQ 0, PRINT ERROR MESSAGES RELATING TO
;THESE AREAS.
;IF NOT EQ 0, END OF PASS DETERMINED
;BY THE 'WORDS READ' COUNT.
;IF EQ 0, END OF PASS DETERMINED
;BY THE SEEK COUNT.

.SBTTL VALUES FOR FIRST OPERATION

;;*****

001432 000010
001434 000005

BEGPAT: .WORD 10
BEGCOD: .WORD 5

; STARTING PATTERN CODE [RANGE 1 - 17 (OCTAL)]
; STARTING COMMAND CODE [RANGE 0 - 5]
; 0 = WRITE CHECK DATA ('WCKD')
; 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.

001436 000404

BEGSIZ: .WORD 404

;;*****

.SBTTL TABLES, CONSTANTS, AND VARIABLE LOCATIONS

;;*****

001440 000000 000000 000000
001446 000000 000000 000000
001454 000000 000000 000000

ORDERG: .WORD 0,0,0,0,0,0,0,0,0 ;LIST OF DRIVES PERFORMING COMMANDS

001462 000000

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

001464 000000 000000 000000
001472 000000 000000 000000
001500 000000 000000 000000

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

001506 000000 000000 000000
001514 000000 000000 000000
001522 000000 000000 000000

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

001530 000000 000000 000000
001536 000000 000000 000000
001544 000000 000000 000000

AVAIL: .WORD 0,0,0,0,0,0,0,0,0 ;LIST OF DRIVES WAITING FOR BUFFERS-PARAMETERS

001552 000000 000000 000000
001560 000000 000000 000000
001566 000000 000000 000000

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

001574 000000 000000 000000
001602 000000 000000 000000
001610 000000 000000 000000

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

001616 000000
001620 000000 000000

BUFTBL: .WORD 0 ;BUFFER ALLOCATION TABLE ENTRY COUNT
.WORD 0,0

001624	000000	000000	.WORD	0,0	
001630	000000	000000	.WORD	0,00	
001634	000000	000000	.WORD	0,00	
001640	000000	000000	.WORD	0,00	
001644	000000	000000	.WORD	0,00	
001650	000000	000000	.WORD	0,00	
001654	000000	000000	.WORD	0,00	
001660	000000	000000	.WORD	0,00	
001664	000000	000000	.WORD	0,00	
001670	000000	000000	.WORD	0,00	
001674	000000	000000	.WORD	0,00	
001700	000000	000000	.WORD	0,00	
001704	000000	000000	.WORD	0,00	
001710	000000	000000	.WORD	0,00	
001714	000000	000000	.WORD	0,00	
001720	000000	000000	.WORD	0,00	
001724	000000	000000	.WORD	0,00	
001730	000000	000000	.WORD	0,00	
001734	000000	000000	.WORD	0,00	
001740	041670		BLKADR: .WORD	DRIVE0	: ADDRESS OF THE BLOCK FOR DRIVE 0
001742	042174		.WORD	DRIVE1	: ADDRESS OF THE BLOCK FOR DRIVE 1
001744	042500		.WORD	DRIVE2	: ADDRESS OF THE BLOCK FOR DRIVE 2
001746	043004		.WORD	DRIVE3	: ADDRESS OF THE BLOCK FOR DRIVE 3
001750	043310		.WORD	DRIVE4	: ADDRESS OF THE BLOCK FOR DRIVE 4
001752	043614		.WORD	DRIVE5	: ADDRESS OF THE BLOCK FOR DRIVE 5
001754	044120		.WORD	DRIVE6	: ADDRESS OF THE BLOCK FOR DRIVE 6
001756	044424		.WORD	DRIVE7	: ADDRESS OF THE BLOCK FOR DRIVE 7
001760	151		COMTBL: .BYTE	WCKD	: WRITE CHECK DATA
001761	153		.BYTE	WCKHD	: WRITE CHECK HEADER AND DATA
001762	161		.BYTE	WRTDAT	: WRITE DATA
001763	163		.BYTE	WRTHD	: WRITE HEADER AND DATA
001764	171		.BYTE	RDDAT	: READ DATA
001765	173		.BYTE	RHDH	: READ HEADER AND DATA
001766	002		OPTBL: .BYTE	2	: UNLOAD
001767	004		.BYTE	4	: SEEK
001770	006		.BYTE	6	: RECAL
001771	010		.BYTE	10	: DRIVE CLEAR
001772	012		.BYTE	12	: RELEASE
001773	014		.BYTE	14	: OFFSET
001774	016		.BYTE	16	: RETURN TO CENTERLINE
001775	020		.BYTE	20	: READIN PRESET
001776	022		.BYTE	22	: PACK ACKNOWLEDGE
001777	030		.BYTE	30	: SEARCH
002000	050		.BYTE	50	: WRITE CHECK DATA
002001	052		.BYTE	52	: WRITE CHECK HEADER AND DATA
002002	060		.BYTE	60	: WRITE DATA
002003	062		.BYTE	62	: WRITE HEADER AND DATA
002004	070		.BYTE	70	: READ DATA
002005	072		.BYTE	72	: READ HEADER AND DATA
002006	377		.BYTE	-1	: TERMINATOR

0020:0

.EVEN

2586	002010	047125	047514	042101	MNTBL:	.ASCIZ	/UNLOAD	/
2587	002016	000040						
2588	002020	042523	045505	020040		.ASCIZ	/SEEK	/
2589	002026	000040						
2590	002030	042522	040503	020114		.ASCIZ	/RECAL	/
2591	002036	000040						
2592	002040	051104	041526	051114		.ASCIZ	/DRVCLR	/
2593	002046	000040						
2594	002050	042522	051514	020105		.ASCIZ	/RELSE	/
2595	002056	000040						
2596	002060	043117	051506	052105		.ASCIZ	/OFFSET	/
2597	002066	000040						
2598	002070	052122	020103	020040		.ASCIZ	/RTC	/
2599	002076	000040						
2500	002100	042522	042101	047111		.ASCIZ	/READIN	/
2501	002106	000040						
2502	002110	040520	045503	020040		.ASCIZ	/PACK	/
2503	002116	000040						
2504	002120	042523	051101	044103		.ASCIZ	/SEARCH	/
2505	002126	000040						
2506	002130	041527	042113	020040		.ASCIZ	/WCKD	/
2507	002136	000040						
2508	002140	041527	044113	020104		.ASCIZ	/WCKHD	/
2509	002146	000040						
2510	002150	051127	042124	052101		.ASCIZ	/WRTDAT	/
2511	002156	000040						
2512	002160	051127	044124	020104		.ASCIZ	/WRTHD	/
2513	002166	000040						
2514	002170	042122	040504	020124		.ASCIZ	/RDDAT	/
2515	002176	000040						
2516	002200	042122	042110	020040		.ASCIZ	/RDHD	/
2517	002206	000040						
2518	002210	047516	042516	020040		.ASCIZ	/NONE	/
2519	002216	000040						
2520								
2521	002220	000			OFFCOD:	.BYTE	0	: OFFSET CODE TABLE
2522	002221	010				.BYTE	10	: +200 U INCHES
2523	002222	210				.BYTE	210	: -200 U INCHES
2524	002223	020				.BYTE	20	: +400 U INCHES
2525	002224	220				.BYTE	220	: -400 U INCHES
2526	002225	030				.BYTE	30	: +600 U INCHES
2527	002226	230	000			.BYTE	230,0	: -600 U INCHES, TERMINATOR
2528	002230	020				.BYTE	20	: +400 U INCHES
2529	002231	220				.BYTE	220	: -400 U INCHES
2530	002232	040				.BYTE	40	: +800 U INCHES
2531	002233	240				.BYTE	240	: -800 U INCHES
2532	002234	060				.BYTE	60	: +1200 U INCHES
2533	002235	260	000			.BYTE	260,0	: -1200 U INCHES, TERMINATOR
2534		002240			.EVEN			
2535								
2536	002240	002274			OFMTBL:	.WORD	OFMSG0	: 1ST OFFSET MESSAGE
2537	002242	002327				.WORD	OFMSG1	: 2ND OFFSET MESSAGE
2538	002244	002363				.WORD	OFMSG2	: 3RD OFFSET MESSAGE
2539	002246	002417				.WORD	OFMSG3	: 4TH OFFSET MESSAGE
2540	002250	002453				.WORD	OFMSG4	: 5TH OFFSET MESSAGE
2541	002252	002507				.WORD	OFMSG5	: 6TH OFFSET MESSAGE

J04

CZRJDCO, RPO4 5.6 MLT-DR LGC
CZRJDC.P11 15-DEC-77 10:58

MACY11 30(1046) 15-DEC-77 11:03 PAGE 49
TABLES, CONSTANTS, AND VARIABLE LOCATIONS

SEG 0048

2542	002254	002543			.WORD	OFMSG6	:7TH OFFSET MESSAGE
2543	002256	002274			.WORD	OFMSG0	:1ST OFFSET MESSAGE
2544	002260	002417			.WORD	OFMSG3	:4TH OFFSET MESSAGE
2545	002262	002453			.WORD	OFMSG4	:5TH OFFSET MESSAGE
2546	002264	002577			.WORD	OFMSG7	:8TH OFFSET MESSAGE
2547	002266	002633			.WORD	OFMSG8	:9TH OFFSET MESSAGE
2548	002270	002667			.WORD	OFMSG9	:10TH OFFSET MESSAGE
2549	002272	002724			.WORD	OFMSGA	:11TH OFFSET MESSAGE
2550							
2551	002274	043101	042524	020122	OFMSG0:	.ASCIZ	/AFTER RETRY WITHOUT OFFSET/
2552	002302	042522	051124	020131			
2553	002310	044527	044124	052517			
2554	002316	020124	043117	051506			
2555	002324	052105	000				
2556	002327	101	020124	043117	OFMSG1:	.ASCIZ	/AT OFFSET +200 MICRO-INCHES/
2557	002334	051506	052105	025440			
2558	002342	030062	020060	044515			
2559	002350	051103	026517	047111			
2560	002356	044103	051505	000			
2561	002363	101	020124	043117	OFMSG2:	.ASCIZ	/AT OFFSET -200 MICRO-INCHES/
2562	002370	051506	052105	026440			
2563	002376	030062	020060	044515			
2564	002404	051103	026517	047111			
2565	002412	044103	051505	000			
2566	002417	101	020124	043117	OFMSG3:	.ASCIZ	/AT OFFSET +400 MICRO-INCHES/
2567	002424	051506	052105	025440			
2568	002432	030064	020060	044515			
2569	002440	051103	026517	047111			
2570	002446	044103	051505	000			
2571	002453	101	020124	043117	OFMSG4:	.ASCIZ	/AT OFFSET -400 MICRO-INCHES/
2572	002460	051506	052105	026440			
2573	002466	030064	020060	044515			
2574	002474	051103	026517	047111			
2575	002502	044103	051505	000			
2576	002507	101	020124	043117	OFMSG5:	.ASCIZ	/AT OFFSET +600 MICRO-INCHES/
2577	002514	051506	052105	025440			
2578	002522	030066	020060	044515			
2579	002530	051103	026517	047111			
2580	002536	044103	051505	000			
2581	002543	101	020124	043117	OFMSG6:	.ASCIZ	/AT OFFSET -600 MICRO-INCHES/
2582	002550	051506	052105	026440			
2583	002556	030066	020060	044515			
2584	002564	051103	026517	047111			
2585	002572	044103	051505	000			
2586	002577	101	020124	043117	OFMSG7:	.ASCIZ	/AT OFFSET +800 MICRO-INCHES/
2587	002604	051506	052105	025440			
2588	002612	030070	020060	044515			
2589	002620	051103	026517	047111			
2590	002626	044103	051505	000			
2591	002633	101	020124	043117	OFMSG8:	.ASCIZ	/AT OFFSET -800 MICRO-INCHES/
2592	002640	051506	052105	026440			
2593	002646	030070	020060	044515			
2594	002654	051103	026517	047111			
2595	002662	044103	051505	000			
2596	002667	101	020124	043117	OFMSG9:	.ASCIZ	/AT OFFSET +1200 MICRO-INCHES/
2597	002674	051506	052105	025440			

K04

CZRJDCO, RPO4 5.6 MLT-DR LGC
 CZRJDC.P11 15-DEC-77 10:58

MACY11 30(1046) 15-DEC-77 11:03 PAGE 50
 TABLES, CONSTANTS, AND VARIABLE LOCATIONS

SEQ 0049

```

2598 002702 031061 030060 046440
2599 002710 041511 047522 044455
2600 002716 041516 042510 000123
2601 002724 052101 047440 043106
2602 002732 042523 020124 030455
2603 002740 030062 020060 044515
2604 002746 051103 026517 047111
2605 002754 044103 051505 000
2606
2607      002762
2608
2609
2610
2611
2612
2613
2614
2615 002762 000000
2616 002764 003066
2617 002766 003126
2618 002770 003166
2619 002772 003226
2620 002774 003266
2621 002776 003326
2622 003000 003366
2623 003002 003426
2624 003004 003466
2625 003006 003526
2626 003010 003566
2627 003012 003626
2628 003014 003666
2629 003016 003726
2630 003020 003766
2631 003022 003826
2632 003024 003866
2633
2634 003026 000000
2635 003030 000000
2636 003032 000000
2637 003034 000000
2638 003036 000000
2639 003040 000000
2640 003042 000000
2641 003044 000000
2642 003046 000000
2643 003050 000000
2644 003052 000000
2645 003054 000000
2646 003056 000000
2647 003060 000000
2648 003062 000000
2649 003064 000000
2650
2651 003066 000001
2652 003070 000003
2653 003072 000007
  
```

OFMSGA: .ASCIZ /AT OFFSET -1200 MICRO-INCHES/

.EVEN

;*****

.SBTTL DATA PATTERNS

;*****

```

STNDAT: .WORD      0                ;STANDARD DATA PATTERN POINTER TABLE
        .WORD     DATA1
        .WORD     DATA1+40
        .WORD     DATA1+100
        .WORD     DATA1+140
        .WORD     DATA1+200
        .WORD     DATA1+240
        .WORD     DATA1+300
        .WORD     DATA1+340
        .WORD     DATA1+400
        .WORD     DATA1+440
        .WORD     DATA1+500
        .WORD     DATA1+540
        .WORD     DATA1+600
        .WORD     DATA1+640
        .WORD     DATA1+700
        .WORD     DATA0                ;ZEROS
        .WORD     DATA1+642          ;ONES

DATA0:  .WORD     0                ;DUMMY DATA PATTERN
        .WORD     0
        .WORD     0
        .WORD     0
        .WORD     0
        .WORD     0
        .WORD     0
        .WORD     0
        .WORD     0
        .WORD     0
        .WORD     0
        .WORD     0
        .WORD     0
        .WORD     0
        .WORD     0
        .WORD     0
        .WORD     0
        .WORD     0
        .WORD     0
        .WORD     0
        .WORD     0
        .WORD     0
        .WORD     0
        .WORD     0

DATA1:  .WORD     000001           ;STANDARD PATTERN 1
        .WORD     000003
        .WORD     000007
  
```

2654	003074	000017	.WORD	000017	
2655	003076	000037	.WORD	000037	
2656	003100	000077	.WORD	000077	
2657	003102	000177	.WORD	000177	
2658	003104	000377	.WORD	000377	
2659	003106	000777	.WORD	000777	
2660	003110	001777	.WORD	001777	
2661	003112	003777	.WORD	003777	
2662	003114	007777	.WORD	007777	
2663	003116	017777	.WORD	017777	
2664	003120	037777	.WORD	037777	
2665	003122	077777	.WORD	077777	
2666	003124	177777	.WORD	177777	
2667					
2668	003126	177776	.WORD	177776	; STANDARD PATTERN 2
2669	003130	177774	.WORD	177774	
2670	003132	177770	.WORD	177770	
2671	003134	177760	.WORD	177760	
2672	003136	177740	.WORD	177740	
2673	003140	177700	.WORD	177700	
2674	003142	177600	.WORD	177600	
2675	003144	177400	.WORD	177400	
2676	003146	177000	.WORD	177000	
2677	003150	176000	.WORD	176000	
2678	003152	174000	.WORD	174000	
2679	003154	170000	.WORD	170000	
2680	003156	160000	.WORD	160000	
2681	003160	140000	.WORD	140000	
2682	003162	100000	.WORD	100000	
2683	003164	000000	.WORD	000000	
2684					
2685	003166	000000	.WORD	000000	; STANDARD PATTERN 3
2686	003170	000000	.WORD	000000	
2687	003172	000000	.WORD	000000	
2688	003174	177777	.WORD	177777	
2689	003176	177777	.WORD	177777	
2690	003200	177777	.WORD	177777	
2691	003202	000000	.WORD	000000	
2692	003204	000000	.WORD	000000	
2693	003206	177777	.WORD	177777	
2694	003210	177777	.WORD	177777	
2695	003212	000000	.WORD	000000	
2696	003214	177777	.WORD	177777	
2697	003216	000000	.WORD	000000	
2698	003220	177777	.WORD	177777	
2699	003222	000000	.WORD	000000	
2700	003224	177777	.WORD	177777	
2701					
2702	003226	000000	.WORD	000000	; STANDARD PATTERN 4
2703	003230	010421	.WORD	010421	
2704	003232	021042	.WORD	021042	
2705	003234	031463	.WORD	031463	
2706	003236	042104	.WORD	042104	
2707	003240	052525	.WORD	052525	
2708	003242	063146	.WORD	063146	
2709	003244	073567	.WORD	073567	

2710	003246	104210	.WORD	104210	
2711	003250	114631	.WORD	114631	
2712	003252	125252	.WORD	125252	
2713	003254	135673	.WORD	135673	
2714	003256	146314	.WORD	146314	
2715	003260	156735	.WORD	156735	
2716	003262	167356	.WORD	167356	
2717	003264	177777	.WORD	177777	
2718					
2719	003266	052525	.WORD	052525	; STANDARD PATTERN 5
2720	003270	052525	.WORD	052525	
2721	003272	052525	.WORD	052525	
2722	003274	125252	.WORD	125252	
2723	003276	125252	.WORD	125252	
2724	003300	125252	.WORD	125252	
2725	003302	052525	.WORD	052525	
2726	003304	052525	.WORD	052525	
2727	003306	125252	.WORD	125252	
2728	003310	125252	.WORD	125252	
2729	003312	052525	.WORD	052525	
2730	003314	125252	.WORD	125252	
2731	003316	052525	.WORD	052525	
2732	003320	125252	.WORD	125252	
2733	003322	052525	.WORD	052525	
2734	003324	125252	.WORD	125252	
2735					
2736	003326	007417	.WORD	007417	; STANDARD PATTERN 6
2737	003330	007417	.WORD	007417	
2738	003332	007417	.WORD	007417	
2739	003334	170360	.WORD	170360	
2740	003336	170360	.WORD	170360	
2741	003340	170360	.WORD	170360	
2742	003342	007417	.WORD	007417	
2743	003344	007417	.WORD	007417	
2744	003346	170360	.WORD	170360	
2745	003350	170360	.WORD	170360	
2746	003352	007417	.WORD	007417	
2747	003354	170360	.WORD	170360	
2748	003356	007417	.WORD	007417	
2749	003360	170360	.WORD	170360	
2750	003362	007417	.WORD	007417	
2751	003364	170360	.WORD	170360	
2752					
2753	003366	026455	.WORD	026455	; STANDARD PATTERN 7
2754	003370	026455	.WORD	026455	
2755	003372	026455	.WORD	026455	
2756	003374	151322	.WORD	151322	
2757	003376	151322	.WORD	151322	
2758	003400	151322	.WORD	151322	
2759	003402	026455	.WORD	026455	
2760	003404	026455	.WORD	026455	
2761	003406	151322	.WORD	151322	
2762	003410	151322	.WORD	151322	
2763	003412	026455	.WORD	026455	
2764	003414	151322	.WORD	151322	
2765	003416	026455	.WORD	026455	

2766	003420	151322	.WORD	151322	
2767	003422	026455	.WORD	026455	
2768	003424	151322	.WORD	151322	
2769					
2770	003426	165555	.WORD	165555	;STANDARD PATTERN 8
2771	003430	133333	.WORD	133333	
2772	003432	165555	.WORD	165555	
2773	003434	133333	.WORD	133333	
2774	003436	165555	.WORD	165555	
2775	003440	133333	.WORD	133333	
2776	003442	165555	.WORD	165555	
2777	003444	133333	.WORD	133333	
2778	003446	165555	.WORD	165555	
2779	003450	133333	.WORD	133333	
2780	003452	165555	.WORD	165555	
2781	003454	133333	.WORD	133333	
2782	003456	165555	.WORD	165555	
2783	003460	133333	.WORD	133333	
2784	003462	165555	.WORD	165555	
2785	003464	133333	.WORD	133333	
2786					
2787	003466	000001	.WORD	000001	;STANDARD PATTERN 9
2788	003470	000002	.WORD	000002	
2789	003472	000004	.WORD	000004	
2790	003474	000010	.WORD	000010	
2791	003476	000020	.WORD	000020	
2792	003500	000040	.WORD	000040	
2793	003502	000100	.WORD	000100	
2794	003504	000200	.WORD	000200	
2795	003506	000400	.WORD	000400	
2796	003510	001000	.WORD	001000	
2797	003512	002000	.WORD	002000	
2798	003514	004000	.WORD	004000	
2799	003516	010000	.WORD	010000	
2800	003520	020000	.WORD	020000	
2801	003522	040000	.WORD	040000	
2802	003524	100000	.WORD	100000	
2803					
2804	003526	177776	.WORD	177776	;STANDARD PATTERN 10
2805	003530	177775	.WORD	177775	
2806	003532	177773	.WORD	177773	
2807	003534	177767	.WORD	177767	
2808	003536	177757	.WORD	177757	
2809	003540	177737	.WORD	177737	
2810	003542	177677	.WORD	177677	
2811	003544	177577	.WORD	177577	
2812	003546	177377	.WORD	177377	
2813	003550	176777	.WORD	176777	
2814	003552	175777	.WORD	175777	
2815	003554	173777	.WORD	173777	
2816	003556	167777	.WORD	167777	
2817	003560	157777	.WORD	157777	
2818	003562	137777	.WORD	137777	
2819	003564	077777	.WORD	077777	
2820					
2821	003566	172666	.WORD	172666	;STANDARD PATTERN 11

2822	003570	155555	.WORD	155555	
2823	003572	172666	.WORD	172666	
2824	003574	155555	.WORD	155555	
2825	003576	172666	.WORD	172666	
2826	003600	155555	.WORD	155555	
2827	003602	172666	.WORD	172666	
2828	003604	155555	.WORD	155555	
2829	003606	172666	.WORD	172666	
2830	003610	155555	.WORD	155555	
2831	003612	172666	.WORD	172666	
2832	003614	155555	.WORD	155555	
2833	003616	172666	.WORD	172666	
2834	003620	155555	.WORD	155555	
2835	003622	172666	.WORD	172666	
2836	003624	155555	.WORD	155555	
2837					
2838	003626	077777	.WORD	077777	; STANDARD PATTERN 12
2839	003630	137777	.WORD	137777	
2840	003632	157777	.WORD	157777	
2841	003634	167777	.WORD	167777	
2842	003636	173777	.WORD	173777	
2843	003640	175777	.WORD	175777	
2844	003642	176777	.WORD	176777	
2845	003644	177377	.WORD	177377	
2846	003646	177577	.WORD	177577	
2847	003650	177677	.WORD	177677	
2848	003652	177737	.WORD	177737	
2849	003654	177757	.WORD	177757	
2850	003656	177767	.WORD	177767	
2851	003660	177773	.WORD	177773	
2852	003662	177775	.WORD	177775	
2853	003664	177776	.WORD	177776	
2854					
2855	003666	153333	.WORD	153333	; STANDARD PATTERN 13
2856	003670	066667	.WORD	066667	
2857	003672	153333	.WORD	153333	
2858	003674	066667	.WORD	066667	
2859	003676	153333	.WORD	153333	
2860	003700	066667	.WORD	066667	
2861	003702	153333	.WORD	153333	
2862	003704	066667	.WORD	066667	
2863	003706	153333	.WORD	153333	
2864	003710	066667	.WORD	066667	
2865	003712	153333	.WORD	153333	
2866	003714	066667	.WORD	066667	
2867	003716	153333	.WORD	153333	
2868	003720	066667	.WORD	066667	
2869	003722	153333	.WORD	153333	
2870	003724	066667	.WORD	066667	
2871					
2872	003726	000000	.WORD	000000	; STANDARD PATTERN 14
2873	003730	177777	.WORD	177777	
2874	003732	177777	.WORD	177777	
2875	003734	177777	.WORD	177777	
2876	003736	177777	.WORD	177777	
2877	003740	177777	.WORD	177777	

2878	003742	177777	.WORD	177777
2879	003744	177777	.WORD	177777
2880	003746	177777	.WORD	177777
2881	003750	177777	.WORD	177777
2882	003752	177777	.WORD	177777
2883	003754	177777	.WORD	177777
2884	003756	177777	.WORD	177777
2885	003760	177777	.WORD	177777
2886	003762	177777	.WORD	177777
2887	003764	177777	.WORD	177777
2888				
2889	003766	177777	.WORD	177777
2890	003770	000000	.WORD	000000
2891	003772	000000	.WORD	000000
2892	003774	000000	.WORD	000000
2893	003776	000000	.WORD	000000
2894	004000	000000	.WORD	000000
2895	004002	000000	.WORD	000000
2896	004004	000000	.WORD	000000
2897	004006	000000	.WORD	000000
2898	004010	000000	.WORD	000000
2899	004012	000000	.WORD	000000
2900	004014	000000	.WORD	000000
2901	004016	000000	.WORD	000000
2902	004020	000000	.WORD	000000
2903	004022	000000	.WORD	000000
2904	004024	000000	.WORD	000000
2905				

; STANDARD PATTERN 15

.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

2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961

004026

\$ERRTB:
;ERROR 1

;ERROR 1

004026 045020
004030 047460
004032 050112
004034 000000

EM1
DH1
DT1
0

;RH11 INTERRUPT OCCURRED (RPAS = 0)

;ERROR 2

004036 045063
004040 047465
004042 050116
004044 000000

EM2
DH2
DT2
0

;UNEXPECTED ATTENTION OCCURRED

;ERROR 3

004046 045121
004050 047542
004052 050134
004054 000000

EM3
DH3
DT3
0

;MASSBUS PARITY ERROR (MCPE=1)

;ERROR 4

004056 045157
004060 047570
004062 050144
004064 000000

EM4
DH4
DT4
0

;MASSBUS PARITY ERROR (PAR=1)

;ERROR 5

004066 045214
004070 047465
004072 050116
004074 000000

EM5
DH2
DT2
0

;ADDRESS PLUG BIT CHANGED

;ERROR 6

004076 045250
004100 047627

EM6
DH6

;RH11 DIDN'T RESPOND TO ADDRESSING


```

2962 004102 050156 DT6
2963 004104 000000 0
2964
2965 ;;*****
2966 .SBTTL SETUP AND INITIALIZATION ROUTINE
2967
2968 : START ADDRESS = 200
2969 : ADDRESS TO CHANGE RH11 UNIBUS ADDRESS = 204
2970
2971 ;;*****
2972
2973
2974 004106 012737 177777 001260 START: MOV #-1,CHGADR ;SET RH11 ADDRESS CHANGE FLAG
2975 004114 000402 BR START2 ;START THE PROGRAM
2976 004116 005037 001260 START1: CLR CHGADR ;CLEAR THE RH11 ADDRESS CHANGE FLAG
2977 004122 000005 START2: RESET ;CLEAR THE BUS
2978 004124 013737 001352 001372 MOV ENDCN,ENDCON ;SET UP FOR NORMAL PASS
2979 004132 013737 001354 001374 MOV ENDCN+2,ENDCON+2
2980 004140 013737 001356 001376 MOV ENDSK,ENDSEK
2981 004146 013737 001360 001400 MOV ENDSK+2,ENDSEK+2
2982
2983 SETUP:
2984 .SBTTL INITIALIZE THE COMMON TAGS
2985 004154 012706 001100 ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
2986 004160 005026 MOV $CMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
2987 004162 022706 001140 CLR (R6)+ ;;CLEAR MEMORY LOCATION
2988 004166 001374 CMP $SWR,R6 ;;DONE?
2989 004170 012706 001100 BNE -6 ;;LOOP BACK IF NO
2990 ;;INITIALIZE A FEW VECTORS
2991 004174 012737 030530 000030 MOV $ERRVEC,$EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
2992 004202 012737 000340 000032 MOV #340,$EMTVEC+2 ;;LEVEL 7
2993 004210 012737 033076 000034 MOV $TRAP,$TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
2994 004216 012737 000340 000036 MOV #340,$TRAPVEC+2 ;;LEVEL 7
2995 004224 012737 176543 032462 MOV #176543,$HINUM ;;PRIME THE RANDOM NUMBER GENERATOR
2996 004232 012737 123456 032464 MOV #123456,$LONUM ;;BOTH HIGH AND LOW WORDS
2997 ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
2998 ;;EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
2999 004240 013746 000004 MOV $ERRVEC-(SP) ;;SAVE ERROR VECTOR
3000 004244 012737 004300 000004 MOV #64,$ERRVEC ;;SET UP ERROR VECTOR
3001 004252 012737 177570 001140 MOV $DSWR,$SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
3002 004260 012737 177570 001142 MOV $DDISP,$DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
3003 004266 022777 177777 174644 CMP #-1,$SWR ;;TRY TO REFERENCE HARDWARE SWR
3004 004274 001012 BNE 66$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
3005 ;;AND THE HARDWARE SWR IS NOT = -1
3006 004276 000403 BR 65$ ;;BRANCH IF NO TIMEOUT
3007 004300 012716 004306 64$: MOV #65$,(SP) ;;SET UP FOR TRAP RETURN
3008 004304 000002 RTI
3009 004306 012737 000176 001140 65$: MOV $SWREG,$SWR ;;POINT TO SOFTWARE SWR
3010 004314 012737 000174 001142 MOV $DISPRREG,$DISPLAY
3011 004322 012637 000004 66$: MOV (SP)+,$ERRVEC ;;RESTORE ERROR VECTOR
3012
3013 004326 012737 000240 000032 MOV #240,$EMTVEC+2 ;CHANGE EMT PRIORITY TO 5
3014 004334 012737 000240 00003E MOV #240,$TRAPVEC+2 ;CHANGE TRAP PRIORITY TO 5
3015 004342 005227 177777 INC #-1 ;FIRST START
3016 004346 001017 BNE 1$ ;BR IF NOT
3017 004350 023737 000042 00004E CMP #42,$46 ;ARE WE IN ACT-11 AUTO MODE?

```

```

3018 004356 001413 BEQ 1$ ; BRANCH IF YES
3019 004360 104401 055362 TYPE TITLE ; NAME AND MAINDEC NUMBER
3020 004364 005737 000042 TST 42 ; AUTO ACCEPT OR CHAIN MODE ?
3021 004370 001006 BNE 1$ ; BR IF EITHER
3022 004372 122737 000011 000041 CMPB #11,41 ; LOADED FROM AN RPO4/5/6 ?
3023 004400 001002 BNE 1$ ; BR IF NOT
3024 004402 104401 055464 TYPE LOADRV ; INSTRUCT THE OPERATOR ON HOW TO TEST DRIVE C
3025 004406 004737 030050 1$: JSR PC,$TKINT ; TURN ON THE KEYBOARD INTERRUPT
3026 .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
3027 004412 005737 000042 TST 0#42 ; ARE WE RUNNING UNDER XXDP/ACT?
3028 004416 001006 BNE 67$ ; BRANCH IF YES
3029 004420 023727 001140 000176 CMP SWR,#SWREG ; SOFTWARE SWITCH REG SELECTED?
3030 004426 001005 BNE 68$ ; BRANCH IF NO
3031 004430 104406 GTSWR ; GET SOFT-SWR SETTINGS
3032 004432 000403 BR 68$
3033 004434 112737 000001 001134 67$: MOVB #1,$AUTOB ; SET AUTO-MODE INDICATOR
3034 004442 68$:
3035 004442 005227 177777 INC #-1 ; FIRST START ?
3036 004446 001015 BNE 2$ ; BR IF NOT
3037 004450 004737 054752 JSR PC,BUSADR ; CHECK RH11 BUS ADDRESS
3038 004454 013737 001170 033332 $RADR,$PADR ; RH11 ADDRESS
3039 004462 013737 001172 033334 $RVEC,$PVEC ; RH11 VECTOR ADDRESS
3040 004470 005737 000042 TST 0#42 ; ACT-11 AUTO OR CHAIN MODE?
3041 004474 001002 BNE 2$ ; BRANCH IF EITHER, SKIP
3042 DATE & OPERATOR ID INPUT
3043 004476 004737 054466 JSR PC,OPRDAT ; GET THE DATE AND OPERATOR ID
3044 004502 005037 001214 2$: CLR STATIN ; CLEAR PERFORMANCE SUMMARY TYPEOUT FLAG
3045 004506 012705 001440 3$: MOV #ORDERQ,$R5 ; START OF AREA TO CLEAR
3046 004512 005025 CLR ($R5)+
3047 004514 022705 001740 CMP #BLKADR,$R5 ; LOOK FOR END OF CLEAR AREA
3048 004520 001374 BNE 3$ ; BR IF NOT FINISHED
3049 004522 012706 001100 MOV #STACK,$SP ; SETUP THE STACK POINTER
3050 004526 005037 177776 CLR PS ; CLEAR THE PROCESSOR STATUS WORD
3051 004532 013737 001212 001274 MOV HZ,SIXTEE ; 1/60 TH OR 1/50 TH SECOND COUNTER VALUE
3052 004540 005037 001266 CLR HOUR ; CLEAR THE HOUR'S COUNTER
3053 004544 005037 001270 CLR MINUTE ; CLEAR THE MINUTE'S COUNTER
3054 004550 005037 001272 CLR SECOND ; CLEAR THE SECOND'S COUNTER
3055 004554 005037 001412 CLR INTRVL+2 ; CLEAR INTERVAL COUNTER
3056 004560 005037 001216 CLR PACK ; CLEAR THE 'R' OR 'W' COMMAND FLAG
3057 004564 005037 001262 CLR CFLAG ; CLEAR THE 'CONTROL C' FLAG
3058 004570 042737 170000 001406 BIC #170000,$MAXER ; MAKE SURE ERROR LIMITS ARE NOT TOO HIGH
3059
3060 ;ROUTINE TO DETERMINE BUFFER AREA SIZE
3061
3062 004576 005227 177777 SIZMEM: INC #-1 ; SEE IF TIME TO SIZE MEMORY
3063 004602 001005 BNE 1$ ; BR IF NOT
3064 004604 004737 054646 JSR PC,$SIZE ; SEE HOW MUCH MEMORY ON SYSTEM
3065 004610 013737 054742 001256 MOV $LSTAD,$LSTAD ; SAVE THE LAST ADDRESS
3066 004616 012737 000001 001616 1$: MOV #1,$BUFTBL ; LOAD NUMBER OF BUFFERS
3067 004624 012737 054466 001620 MOV #ENDPGM,$BUFTBL+2 ; STARTING ADDRESS OF BUFFER
3068 004632 013737 001256 001622 MOV $LSTAD,$BUFTBL+4 ; LAST ADDR TO BUFFER ALLOCATION TABLE
3069 004640 162737 054466 001622 SUB #ENDPGM,$BUFTBL+4 ; SUBTRACT PROGRAM SPACE
3070 004646 000241 CLC ; CLEAR THE 'C' BIT
3071 004650 006037 001622 ROR $BUFTBL+4 ; CONVERT TO WORD COUNT
3072 004654 162737 000144 001622 SUB #100,$BUFTBL+4 ; SAVE ROOM FOR THE 'ABS' LOADER
3073 004662 023727 001256 100000 CMP $LSTAD,$#100000 ; 16K ON THE SYSTEM ?

```

G05

CRJDCO RPO4 5 6 MLT-DR LGC
 CRJDC.P11 15-DEC-77 10:58

MACY11 30(1046) 15-DEC-77 11:03 PAGE 59
 GET VALUE FOR SOFTWARE SWITCH REGISTER

SEG C058

```

3074 004670 103406          BLO      3$          ;BR IF YES
3075 004672 105737 000041  TSTB     41          ;SEE WHO LOADED THE PROGRAM
3076 004676 001403          BEQ      3$          ;BR IF LOADED BY PAPER TAPE
3077 004700 162737 002570 001622  SUB      #1400.,BUFTBL+4 ;SUBTRACT 'XXDP' LOADER SIZE
3078 004706 005737 001404 3$:  TST      MAXDL      ;VALUE IN 'MAXDL' ?
3079 004712 001012          BNE      4$          ;BR IF VALUE IS
3080 004714 012737 013534 001404  MOV      #5980.,MAXDL ;ASSUME FULL TRACK + 1 SEC MAXIMUM
3081 004722 023737 001404 001622  CMP      MAXDL,BUFTBL+4 ;IS THAT TOO LARGE ?
3082 004730 103403          BLO      4$          ;BR IF NOT
3083 004732 013737 001622 001404  MOV      BUFTBL+4,MAXDL ;USE MAX AVAIL MEMORY AS MAX BUFFER SIZE
3084 004740 013737 001622 053456 4$:  MOV      BUFTBL+4,PARLST+2 ;VALUE FOR THE PARAMETER TABLE
3095
3086          ;SEE IF THE OPERATOR WANTS TO CHANGE ANY PARAMETERS
3087
3088 004746 005737 000042  LKPAR:  TST      #42          ;'XXDP' CHAIN MODE OR 'ACT11' OPERATION ?
3089 004752 001022          BNE      SETVEC      ;BR IF YES
3090 004754 104401 053552          TYPE     ,ASKPAR      ;ASK FOR PARAMETERS
3091 004760 104411          RDLIN    ;READ THE ENTRY
3092 004762 012605          MOV      (SP)+,R5     ;ADDRESS OF ENTRY TO R5
3093 004764 122715 000131          CMPB    #'Y',(R5)    ;WAS ENTRY A 'Y' (YES)
3094 004770 001013          BNE      SETVEC      ;BR IF NOT 'Y'
3095
3096 004772 012703 053454          ENTOR:  MOV      #PARLST,R3 ;PARAMETER TABLE ADDRESS
3097 004776 004737 026272          JSR     PC,PARENT    ;GET THE PARAMETER ENTRY
3098 005002 023727 001404 000004  CMP      MAXDL,#4     ;IS THE 'MAXDL' VALUE OK ?
3099 005010 103003          BHIS    SETVEC      ;BR IF IT IS
3100 005012 012737 000004 001404  MOV      #4,MAXDL    ;SET 'MAXDL' TO THE MINIMUM VALUE
3101
3102          ;DISPLAY DRIVE STATUS AND SET UP THE OTHER SYSTEM DEVICES THAT
3103          ; THE PROGRAM WILL USE
3104
3105 005020 004737 022412          SETVEC: JSR     PC,CKCLK ;START THE CLOCK
3106 005024 004737 033350          JSR     PC,RPINIT    ;INITIALIZE THE RPO4/S/6 DRIVER
3107 005030 012737 177777 033272  MOV      #-1,SAVEFG  ;SET THE SAVE REGISTERS FLAG
3108 005036 062727 177777 000000  ADD      #-1,#0      ;CHECK FOR FIRST START
3109 005044 103004          BCC     11$          ;BR IF FIRST START
3110 005046 032777 000004 174064  BIT      #SW02,#SWP  ;TYPEOUT THE DRIVE STATUS TABLE ?
3111 005054 001076          BNE     10$          ;BR IF NOT
3112 005056 012737 000340 177776 11$:  MOV      #PR7,PS     ;SET PRIORITY TO 7
3113 005064 005004          CLR     R4           ;DRIVE TABLE POINTER
3114 005066 104401 001165          TYPE   ,$CRLF       ;CR-LF
3115 005072 104401 052344          TYPE   ,SYSTAT      ;TYPE STATUS HEADING
3116 005076
3117 005076 010446          1$:  MOV      R4,-(SP)    ;SAVE R4 FOR TYPEOUT
3118
3119 005100 104403          ;TYPE DRIVE NUMBER
3120 005102 002          ;GO TYPE--OCTAL ASCII
3121 005103 000          ;TYPE 2 DIGIT(S)
3122 005104 104401 052153          ;SUPPRESS LEADING ZEROS
3123 005110 105764 033204          TYPE   ,LIN4SP      ;SPACES
3124 005114 100416          TSTB   DRVSTA(R4)   ;CHECK DRIVE'S STATUS
3125 005116 001020          BMI    4$           ;BR IF UNSAFE
3126 005120 105764 033214          BNE    5$           ;BR IF ONLINE
3127 005124 001404          TSTB   DRVTP(R4)   ;SEE IF OFFLINE OR NONEXISTENT
3128 005126 100006          BEQ    2$           ;BR IF NONEXISTENT
3129 005130 104401 052257          BPL    3$           ;BR IF OFFLINE
3129          TYPE   ,NOTRP  ;DRIVE NOT AN RPO4 5 6

```

H05

CZRJDCO. RPO4 5 6 MLT-DR LGC
CZRJDC.P11 15-DEC-77 10:58

MACY11 30(1046) 15-DEC-77 11:03 PAGE 60
GET VALUE FOR SOFTWARE SWITCH REGISTER

SEG 0059

```

3130 005134 000440          BR      9$      ;CHECK NEXT DRIVE
3131 005136 104401 052300 2$:      TYPE      NOTPRS ;DRIVE NOT PRESENT
3132 005142 000435          BR      9$      ;CHECK NEXT DRIVE
3133 005144 104401 052166 3$:      TYPE      UNTOFF ;DRIVE OFFLINE
3134 005150 000405          BR      6$      ;PRINT DRIVE TYPE
3135 005152 104401 052334 4$:      TYPE      NOTSAF ;DRIVE UNSAFE
3136 005156 000402          BR      6$      ;PRINT DRIVE TYPE
3137 005160 104401 052177 5$:      TYPE      ,UNTON  ;DRIVE ONLINE
3138 005164 104401 052155 6$:      TYPE      ,LINSR  ;SPACES
3139 005170 012737 052364 005234  MOV     #RPO4B,B$ ;ADDRESS OF RPO4 MESSAGE
3140 005176 132764 000001 033214  BITB   #BIT00,DRV TYP(R4) ;RPO4 ?
3141 005204 001012          BNE     7$      ;BR IF YES
3142 005206 012737 052371 005234  MOV     #RPO5,B$ ;ADDRESS OF RPO5 MESSAGE
3143 005214 132764 000002 033214  BITB   #BIT01,DRV TYP(R4) ;RPO5 ?
3144 005222 001003          BNE     7$      ;BR IF YES
3145 005224 012737 052376 005234  MOV     #RPO6,B$ ;ADDRESS OF RPO6 MESSAGE
3146 005232 104401 7$:      TYPE      ;TYPE THE DRIVE TYPE MESSAGE
3147 005234 000000 8$:      .WORD    0      ;MESSAGE ADDRESS HERE
3148 005236 104401 001165 9$:      TYPE      $CRLF  ;CR-LF
3149 005242 005204          INC     R4      ;INCREMENT DRIVE NUMBER TABLE POINTER
3150 005244 020427 000010  CMP     R4,#8.  ;FINISHED ?
3151 005250 001312          BNE     1$      ;BR IF NOT
3152 005252 104401 001165 10$:     TYPE      $CRLF  ;CR-LF
3153 005256 005037 177776  CLR     PS      ;SET PRIORITY BACK TO '0'
3154 005262 000137 005266  JMP     MONTR   ;CHECK FOR 'XXDP' OR 'ACT11' MONITOR

;SETUP IF 'XXDP' OR 'ACT11' OPERATION
3158 005266 005737 000042  MONTR:  TST     42      ;'XXDP' CHAIN MODE OR 'ACT11' AUTO ACCEPT
3159 005272 001402          BEQ     1$      ;BR IF NEITHER
3160 005274 004737 024274  JSR     PC,ASGN2 ;ASSIGN DRIVES
3161 005300 005227 177777 1$:      INC     #-1    ;FIRST START ?
3162 005304 001011          BNE     2$      ;BR IF NOT
3163 005306 105737 000041  TSTB   2#41    ;LOADED FROM PAPER TAPE ?
3164 005312 001406          BEQ     2$      ;BR IF YES
3165 005314 023727 001256 100000  CMP     LSTAD,#100000 ;MORE THAN 16K ON THE SYSTEM ?
3166 005322 103002          BHIS   2$      ;BR IF YES
3167 005324 104401 055663  TYPE    ,NOLOAD ;TELL THE OPERATOR THAT THE 'XXDP' LOADER
3168 005330 004737 030050 2$:      JSR     PC,$TKINT ;WILL BE OVERWRITTEN
3169 005334 104401 053341  TYPE    ,INTDON ;INITIALIZE THE KEYBOARD INTERRUPT HANDLER
3170 005340 000137 005522  JMP     MAIN1  ;TYPE 'INITIALIZE COMPLETE'
3171 005340 000137 005522  ;START THE PROGRAM

;'XXDP' OR 'ACT11' END OF TEST ROUTINE
3175 005344 013700 000042  $GET42: MOV     42,R0  ;MONITOR ADDRESS
3176 005350 001002          BNE     1$      ;BR IF MONITOR
3177 005352 000137 005522  JMP     MAIN1  ;NONE CONTINUE
3178 005356 000005 1$:      RESET   ;CLEAR EVERYTHING
3179 005360 004710  SENDAD: JSR     PC,(R0) ;GO TO THE MONITOR
3180 005362 000240          NOP     ;SAVE ROOM
3181 005364 000240          NOP     ;FOR
3182 005366 000240          NOP     ;ACT11
3183 005370 000137 004116  $DOAGN: JMP     START1 ;START AGAIN

;*****

```

3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3240
3241

005374 012703 000010
005400 012705 001464
005404 005715
005406 001011
005410 062705 000002
005414 005303
005416 001372
005420 005737 001462
005424 001036
005426 000137 005344
005432 012701 001530
005436 005711
005440 001405
005442 021115
005444 001414
005446 062701 000002
005452 000771
005454 012701 001552
005460 005711
005462 001752
005464 021115
005466 001403
005470 062701 000002
005474 000771
005476 011100
005500 104401 052650
005504 004737 022676
005510 005015
005512 005011
005514 004737 017530
005520 000733

005522 012703 000010
005526 005002
005530 005004
005532 005005
005534 005765 001506
005540 001006
005542 062705 000002
005546 005204
005550 005303
005552 001370
005554 000432
005556 104401 052160
005562 010446

005564 104403
005566 002
005567 00C

.SBTTL MAIN PROGRAM

::*****

MAIN: MOV #8,R3 ;DRIVE COUNTER
MOV #DUNIT,R5 ;ADDRESS OF 'DROP DRIVE' TABLE
1\$: TST (R5) ;SEE IF ENTRY AT PRESENT POSITION
BNE 3\$;BR IF THERE IS ONE
2\$: ADD #2,R5 ;INCREMENT TO NEXT TABLE POSITION
DEC R3 ;DECREMENT DRIVE COUNTER
BNE 1\$;BR IF MORE TO CHECK
TST ASNLST ;ANY DRIVES ACTIVE ?
BNE MAIN1 ;BR IF YES
JMP \$GET42 ;CHECK FOR MONITOR RETURN
3\$: MOV #AVAIL,R1 ;ADDRESS OF 'AVAILABLE DRIVES' TABLE
4\$: TST (R1) ;SEE IF AT END OF TABLE
BEQ 5\$;BR IF AT END: GO CHECK 'WAIT' TABLE
CMP (R1),(R5) ;IS DRIVE IN 'AVAIL' THE ONE TO BE DROPPED
BEQ 7\$;BR IF YES
ADD #2,R1 ;INCREMENT 'AVAIL' TABLE ADDRESS
BR 4\$;CONTINUE LOOKING
5\$: MOV #WAIT,R1 ;MOVE THE ADDRESS OF THE BUFFER WAIT TABLE
6\$: TST (R1) ;AT THE END OF THE 'WAIT' TABLE ?
BEQ 2\$;BR IF YES: SEE IF ANY MORE 'DROP' REQUESTS
CMP (R1),(R5) ;DRIVE IN THE 'WAIT' TABLE ?
BEQ 7\$;BR IF IT IS
ADD #2,R1 ;INCREMENT 'WAIT' TABLE ADDRESS
BR 6\$;CONTINUE LOOK THROUGH THE 'WAIT' TABLE
7\$: MOV (R1),R0 ;PUT THE DRIVE'S BLOCK ADDRESS IN R0
TYPE DEASSG ;TYPE 'DRIVE DEASSIGNED'
JSR PC,TYPEST ;TYPE THE DRIVE'S PERFORMANCE SUMMARY
CLR (R5) ;CLEAR THE 'DROP DRIVE' TABLE ENTRY
CLR (R1) ;REMOVE THE DRIVE FROM THE 'AVAIL' OR 'WAIT' TABLE
JSR PC,CMPRES ;COMPRESS THE RESPECTIVE TABLE
BR 2\$;SEE IF ANY MORE DRIVES

:LOOK FOR DRIVES TO BE ASSIGNED

MAIN1: MOV #8,R3 ;DRIVE COUNT
CLR R2 ;'AVAIL' INDEX
CLR R4 ;ASSIGN LIST INDEX
CLR R5 ;NEW DRIVE INDEX
1\$: TST NEWUNT(R5) ;NEW DRIVE IN THIS POSITION
BNE 3\$;BR IF THERE IS
2\$: ADD #2,R5 ;INCREMENT R5
INC R4 ;INCREMENT ASSIGN INDEX
DEC R3 ;DECREMENT DRIVE COUNT
BNE 1\$;BR IF MORE DRIVES
BR MAIN2 ;START OPERATIONS FOR THE AVAILABLE DRIVES
3\$: TYPE UNMSG ;'DRIVE'
MOV R4,-(SP) ;SAVE R4 FOR TYPEOUT
;TYPE DRIVE NUMBER
;GO TYPE--OCTAL ASCII
;TYPE 2 DIGIT(S)
;SUPPRESS LEADING ZEROS

JOS

CZRJDCO, RPO4 5/6 MLT-DR LGC
CZRJDCO.P11 15-DEC-77 10:58

MACY11 30(1046) 15-DEC-77 11:03 PAGE 62
MAIN PROGRAM

SEG 0061

```

3242 005570 104401 052720
3243 005574 005762 001530
3244 005600 001403
3245 005602 062702 000002
3246 005606 000772
3247 005610 016562 001506 001530
3248 005616 005065 001506
3249 005622 156437 033320 001462
3250 005630 016200 001530
3251 005634 062702 000002
3252 005640 000740
3253
3254
3255
3256
3257 005642 005737 001552
3258 005646 001113
3259 005650 005002
3260 005652 005762 001530
3261 005656 001551
3262 005660 016200 001530
3263 005664 005760 000104
3264 005670 001021
3265 005672 105760 000026
3266 005676 001403
3267 005700 004737 017546
3268 005704 000415
3269 005706 012701 001574
3270 005712 020011
3271 005714 001403
3272 005716 005721
3273 005720 001403
3274 005722 000773
3275 005724 004737 017530
3276 005730 004737 016446
3277 005734 004737 017322
3278 005740 005046
3279 005742 004737 015672
3280 005746 012660 000006
3281 005752 001424
3282 005754 004737 016242
3283 005760 005060 000072
3284 005764 004737 016370
3285 005770 012705 001440
3286 005774 005725
3287 005776 001376
3288 006000 010045
3289 006002 105760 000026
3290 006006 001025
3291 006010 012705 001574
3292 006014 005725
3293 006016 001376
3294 006020 010045
3295 006022 000417
3296 006024 026037 000072 001254
3297 006032 001405

TYPE ASGND
4$: TST AVAIL(R2) ;'ASSIGNED'
    BEQ 5$ ;AT END OF AVAILABLE TABLE
    ADD #2,R2 ;BR IF YES
    BR 4$ ;INCREMENT AVAILABLE TABLE INDEX
5$: MOV NEWUNT(R5),AVAIL(R2) ;CONTINUE LOOKING FOR END OF TABLE
    CLR NEWUNT(R5) ;MOVE ADDR OF DRIVE INTO AVAIL LST
    BISB ATABIT(R4),ASNLST ;TAKE DRIVE OUT OF NEW DRIVE TABLE
    MOV AVAIL(R2),R0 ;SET DRIVE ASSIGNED INDICATOR
    ADD #2,R2 ;PUT STARTING ADDRESS OF BLOCK IN R0
    BR 2$ ;INCREMENT AVAILABLE TABLE POINTER
    ;LOOK FOR MORE DRIVES

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

MAIN2: TST WAIT ;OUTSTANDING BUFFER REQUESTS
       BNE MAIN3 ;BR IF THERE ARE
       CLR R2 ;CLEAR DRIVE TABLE POINTER
1$: TST AVAIL(R2) ;ANY DRIVES WAITING FOR PARAMETERS
    BEQ IDLE ;BRANCH IF NONE
    MOV AVAIL(R2),R0 ;CONTROL BLOCK ADDR IN R0
    TST $NEXT(R0) ;PARAMETERS BEEN SELECTED ?
    BNE 6$ ;BR IF THEY HAVE
    TSTB $PACK(R0) ;'R' OR 'W' COMMAND FOR THE DRIVE ?
    BEQ 2$ ;BR IF NOT
    JSR PC,WRTPK ;GET DATA PACK PARAMETERS
    BR 7$ ;GET THE BUFFER
2$: MOV #PARQ,R1 ;ADDRESS OF THE PARAMETER QUEUE
3$: CMP R0,(R1) ;IS CURRENT DRIVE IN THE QUEUE ?
    BEQ 4$ ;BR IF IT IS
    TST (R1)+ ;AT END OF THE QUEUE
    BEQ 5$ ;BR IF AT END
    BR 3$ ;CONTINUE LOOKING
4$: JSR PC,CMPRES ;COMPRESS THE TABLE
5$: JSR PC,SELPAR ;SELECT THE PARAMETERS
6$: JSR PC,GETPAR ;LOAD NEW PARAMETERS
7$: CLR -(SP) ;MAKE ROOM ON THE STACK FOR THE BUFFER ADDR
    JSR PC,GETBUF ;GET BUFFER
    MOV (SP)+,$BUF(R0) ;MOVE BUFFER ADDR TO DPB
    BEQ 8$ ;BR IF '0' ADDR (NO BUFFER)
    JSR PC,FILBUF ;FILL THE BUFFER
    CLR $FAIR(R0) ;CLEAR THE 'FAIRNESS' COUNT
    JSR PC,GODRIV ;PUT CURRENT DPB IN DRIVER
    MOV #ORDERQ,R5 ;ADDRESS OF ORDER QUEUE IN R5
    TST (R5)+ ;END OF QUEUE ?
    BNE -2 ;BR IF NOT
    MOV R0,-(R5) ;PUT BLOCK ADDRESS INTO QUEUE
    TSTB $PACK(R0) ;'R' OR 'W' COMMAND FOR DRIVE ?
    BNE 10$ ;BR IF EITHER
    MOV #PARQ,R5 ;PUT BLOCK INTO THE PARAMETER QUEUE
    TST (R5)+ ;FIND THE END OF THE QUEUE
    BNE -2 ;BR IF NOT AT END OF QUEUE
    MOV R0,-(R5) ;PUT BLOCK ADDRESS INTO THE QUEUE
    BR 10$ ;CONTINUE LOOKING
8$: CMP $FAIR(R0),FAIRNS ;ENTRY BEEN IN THE QUEUE LONG ENOUGH ?
    BEQ 9$ ;BR IF YES

```

```

3298 006034 005260 000072          INC    $FAIR(R0)      ; INCREMENT THE ENTRY COUNT
3299 006040 062702 000002          ADD    #2,R2         ; INCREMENT THE POINTER
3300 006044 000702                   BR     1$           ; LOOK FOR SOME MORE DRIVES
3301 006046 012705 001552          9$:   MOV    #WAIT,R5   ; 'WAIT' QUEUE ADDRESS
3302 006052 005725                   TST    (R5)+        ; LOOK FOR AN OPENING
3303 006054 001376                   BNE   #-2           ; BR IF NONE YET
3304 006056 016245 001530          MOV    AVAIL(R2),-(R5) ; MOVE DRIVE'S BLOCK ADDRESS TO QUEUE
3305 006062 012701 001530          10$:  MOV    #AVAIL,R1   ; 'AVAILABLE' TABLE ADDRESS
3306 006066 060201                   ADD    R2,R1        ; FORM ADDRESS OF LAST ENTRY
3307 006070 004737 017530          JSR   PC,COMPRES    ; COMPRESS THE TABLE
3308 006074 000666                   BR     1$           ; CONTINUE LOOKING
3309
3310          ;GET BUFFER ASSIGNMENTS FOR DRIVES IN THE 'BUFFER WAIT' QUEUE
3311
3312 006076 013700 001552          MAIN3: MOV   WAIT,R0   ; MOVE THE 'WAIT' ENTRY TO R0
3313 006102 005046                   CLR   -(SP)         ; MAKE ROOM ON THE STACK FOR THE BUFFER ADDR
3314 006104 004737 015672          JSR   PC,GETBUF     ; TRY TO GET A BUFFER
3315 006110 012660 000006          MOV   (SP)+,$BUF(R0) ; MOVE THE BUFFER ADDR TO THE CPB
3316 006114 001002                   BNE   1$           ; BR IF A BUFFER WAS ASSIGNED
3317 006116 000137 006202          JMP   IDLE          ; NO BUFFER AVAILABLE YET
3318 006122 004737 016242          1$:   JSR   PC,FILBUF   ; FILL THE BUFFER
3319 006126 004737 016370          JSR   PC,GODRIV     ; PUT THE ENTRY IN THE DRIVER
3320 006132 005060 000072          CLR   $FAIR(R0)    ; CLEAR THE 'FAIRNESS' COUNT
3321 006136 012705 001440          MOV   #ORDERQ,R5   ; ADDRESS OF ORDER QUEUE IN R5
3322 006142 005725                   TST   (R5)+        ; AT END OF THE QUEUE
3323 006144 001376                   BNE   #-2           ; BR IF NOT
3324 006146 010045                   MOV   R0,-(R5)     ; PUT BLOCK ADDRESS IN QUEUE
3325 006150 105760 000026          TSTB  $PACK(R0)    ; 'R' OR 'W' COMMAND FOR DRIVE ?
3326 006154 001005                   BNE   2$           ; BR IF YES, DON'T PUT BLOCK INTO 'PARQ'
3327 006156 012705 001574          MOV   #PARQ,R5     ; FIND THE END OF THE PARAMETER QUEUE
3328 006162 005725                   TST   (R5)+        ; OPEN SLOT IN THE QUEUE ?
3329 006164 001376                   BNE   #-2           ; BR IF NOT
3330 006166 010045                   MOV   R0,-(R5)     ; PUT BLOCK ADDRESS INTO THE QUEUE
3331 006170 012701 001552          2$:   MOV   #WAIT,R1   ; ADDRESS OF TABLE TO COMPRESS
3332 006174 004737 017530          JSR   PC,COMPRES    ; COMPRESS THE WAIT TABLE
3333 006200 000620                   BR     MAIN2        ; LOOK FOR MORE ENTRIES
3334
3335          ;WAIT FOR AN ORDER TO FINISH
3336
3337 006202 012701 001440          IDLE:  MOV   #ORDERQ,R1 ; ADDRESS OF THE ORDER QUEUE IN R1
3338 006206 012100                   1$:   MOV   (R1)+,R0    ; PUT BLOCK ADDRESS INTO R0
3339 006210 001433                   BEQ   IDLE1        ; BR IF END OF QUEUE
3340 006212 005760 000016          TST   $STATUS(R0)  ; SEE IF DRIVE FINISHED
3341 006216 001773                   BEQ   1$           ; BR IF DRIVE NOT FINISHED
3342 006220 162701 000002          SUB   #2,R1        ; CORRECT THE QUEUE POINTER
3343 006224 010146                   MOV   R1,-(SP)     ; SAVE THE QUEUE ADDRESS
3344 006226 004737 015526          JSR   PC,STATIS    ; ACCUMULATE STATISTICS FOR DRIVE IN R0
3345 006232 000240                   NOP                ; DEBUGGING AID
3346 006234 004737 006540          JSR   PC,PROCES    ; PROCESS END OF ORDER
3347 006240 005037 001264          CLR   BADSEC       ; CLEAR THE BAD TRK/SEC ERROR INDICATOR
3348 006244 004737 026524          JSR   PC,ABNRML    ; SEE IF ANY DRIVES HAVE TOO MANY ERRORS
3349 006250 004737 026552          JSR   PC,EOP       ; SEE IF ANY DRIVE HAS XFERED 3x10^9 BITS
3350 006254 012601                   MOV   (SP)+,R1     ; RESTORE THE ORDER TABLE INDEX
3351 006256 012705 001530          MOV   #AVAIL,R5    ; FIND THE END OF THE 'AVAILABLE' TABLE
3352 006262 005725                   2$:   TST   (R5)+        ; END OF THE TABLE ?
3353 006264 001376                   BNE   2$           ; BR IF NOT AT END OF LIST

```

```

3354 006266 011145          MOV      (R1), -(R5)      ; MOVE THE BLOCK ADDRESS INTO THE TABLE
3355 006270 004737 017530    JSR      PC, CMPRES      ; COMPRESS THE ORDER QUEUE
3356 006274 004737 016026    JSR      PC, RELBUF      ; RESTORE BUFFER
3357 006300 005737 001262    IDLE1:  TST      CFLAG    ; 'CONTROL C' FLAG ENTERED ?
3358 006304 001403          BEQ      1$             ; BR IF IT WAS
3359 006306 004737 023710    JSR      PC, KSR        ; SERVICE THE KEYBOARD
3360 006312 000733          BR       IDLE          ; SYSTEM WAS BUSY
3361 006314 032777 000004 172616 1$:  BIT      #SW02, #SWR    ; TYPE PERFORMANCE SUMMARY
3362 006322 001007          BNE      2$             ; BR IF NOT
3363 006324 005737 001214    TST      STATIN        ; TIME TO TYPE THE PERFORMANCE SUMMARY ?
3364 006330 001404          BEQ      2$             ; BR IF NOT
3365 006332 005037 001214    CLR      STATIN        ; CLEAR THE INDICATOR
3366 006336 004737 022614    JSR      PC, STATPR     ; TYPE THE SUMMARY
3367 006342 005737 001574    2$:  TST      PARQ        ; ENTRY IN THE PARAMETER QUEUE ?
3368 006346 001410          BEQ      3$             ; BR IF NOT
3369 006350 013700 001574    MOV      PARQ, R0      ; PUT THE BLOCK ADDRESS INTO R0
3370 006354 004737 016446    JSR      PC, SELPAR     ; GET THE PARAMETERS FOR NEXT OPERATION
3371 006360 012701 001574    MOV      #PARQ, R1     ; SETUP TO COMPRESS THE TABLE
3372 006364 004737 017530    JSR      PC, CMPRES     ; COMPRESS THE PARAMETER QUEUE
3373 006370 000137 005374    3$:  JMP      MAIN        ; CONTINUE THE LOOP
3374
3375          ; SETUP TO REFORMAT AN ERROR SECTOR
3376
3377 006374 032777 000001 172536 REFMT:  BIT      #SW0, #SWR    ; READ ONLY SWITCH SET ?
3378 006402 001055          BNE      REFMTX        ; BR IF IT IS
3379 006404 032777 000200 172526    BIT      #SW7, #SWR    ; SWITCH 7 SET ?
3380 006412 001051          BNE      REFMTX        ; BR IF IT IS
3381 006414 005737 001416    TST      FORMAT        ; WRITE HEADER & DATA ORDERS ALLOWED ?
3382 006420 001446          BEQ      REFMTX        ; BR IF NOT
3383 006422 016060 000272 000100    MOV      $RPCC(R0), $NCRYL(R0) ; USE PRESENT CYLINDER
3384 006430 004737 022314    JSR      PC, READDR     ; GET CORRECTED SECTOR-TRACK ADDRESSES
3385 006434 112660 000077    MOV      (SP)+, $NTRK(R0) ; TRACK ADDR TO DPB
3386 006440 112660 000076    MOV      (SP)+, $NSEC(R0) ; SECTOR ADDR TO DPB
3387 006444 012760 000404 000102    MOV      #260, $NWRDL(R0) ; WORD COUNT FOR FORMAT
3388 006452 023727 001404 000404    CMP      MAXDL, #260.  ; CAN A FULL SECTOR BE WRITTEN ?
3389 006460 103003          BHS     1$             ; BR IF IT CAN
3390 006462 013760 001404 000102    MOV      MAXDL, $NWRDL(R0) ; PUT TRANSFER SIZE INTO THE DPB
3391 006470 112760 000003 000074 1$:  MOV      #3, $NCODE(R0) ; COMMAND CODE
3392 006476 004737 017274    JSR      PC, GETPAT     ; GET A PATTERN
3393 006502 110560 000075    MOV      R5, $NPATC(R0) ; PATTERN CODE TO CONTROL BLOCK
3394 006506 012760 177777 000104    MOV      #-1, $NEXTR(R0) ; SET PARAMETERS SELECTED INDICATOR
3395 006514 012701 001574    MOV      #PARQ, R1     ; SET UP TO SEE IF BLOCK IN THE PARAMETER QUEUE
3396 006520 005711          2$:  TST      (R1)         ; SEE IF AT END OF TABLE
3397 006522 001405          BEQ      REFMTX        ; BR IF AT END
3398 006524 020021          CMP      R0, (R1)+     ; SEE IF BLOCK AT PRESENT POSITION
3399 006526 001374          BNE     2$             ; BR IF NOT
3400 006530 005041          CLR      -(R1)         ; CLEAR THE ENTRY
3401 006532 004737 017530    JSR      PC, CMPRES     ; COMPRESS THE TABLE
3402 006536 000207    REFMTX: RTS      PC    ; RETURN
3403
3404          ; PROCESS THE ORDER TERMINATION
3405

```


MOS

C2RJDC0, RP04/5.6 MLT-DR LGC
C2RJDC.P11 15-DEC-77 10:58

MACY11 30(1046) 15-DEC-77 11:03 PAGE 65
MAIN PROGRAM

SEQ 0064

```

3406 006540 111037 001246 PROCES: MOVB (RO),UNIT ;DRIVE NUMBER FOR ANY ERROR MESSAGES
3407 006544 005760 000016 TST STATUS(RO) ;SEE IF DRIVER SIGNALLED AN ERROR
3408 006550 100427 BMI ERPROC ;BR IF ERROR
3409 006552 032760 100000 000234 BIT #BIT15,$RPCS1(RO) ;SEE IF 'SC' SET
3410 006560 001410 BEQ 1$ ;BR IF NOT SET
3411 006562 032760 040000 000234 BIT #BIT14,$RPCS1(RO) ;SEE IF 'TRE' SET
3412 006570 001017 BNE ERPROC ;BR IF SET
3413 006572 032760 040000 000246 BIT #BIT14,$RPDS1(RO) ;SEE IF 'ERR' SET
3414 006600 001013 BNE ERPROC ;BR IF SET
3415 006602 004737 012770 1$: JSR PC,CKERR ;NO ERROR, CHECK ERROR BITS ANYWAY
3416 006606 004737 013070 JSR PC,CKBUS ;NO ERROR, CHECK BUS ADDR & WC
3417 006612 032777 000002 172320 BIT #SW01,$SWR ;DATA COMPARE ?
3418 006620 001002 BNE 2$ ;BR IF NOT
3419 006622 004737 013154 JSR PC,CMPAR ;NO ERROR, COMPARE DATA
3420 006626 000207 2$: RTS ;RETURN

;ORDER TERMINATED WITH AN ERKOR - PROCESS THE ERROR
3421
3422
3423
3424 006630 032760 000200 000016 ERPROC: BIT #BIT07,$STATUS(RO) ;DONE BIT SET ?
3425 006636 001402 BEQ ERPRC1 ;BR IF ORDER DIDN'T COMPLETE NORMALLY
3426 006640 000137 007264 JMP DONE ;PROCESS ERROR WITH 'DONE' BIT SET

;PROCESS ORDER COMPLETION WITH 'ERROR' & 'DONE NOT' BITS
3427
3428
3429
3430 006644 032760 010000 000016 ERPRC1: BIT #BIT12,$STATUS(RO) ;SEE IF DRIVE WAS UNSAFE
3431 006652 001025 BNE PUNSAF ;BR IF YES
3432 006654 032760 004000 000016 BIT #BIT11,$STATUS(RO) ;PARITY ERROR OCCURRED
3433 006662 001055 BNE UCPAR ;BR IF IT DID
3434 006664 032760 002000 000016 BIT #BIT10,$STATUS(RO) ;FATAL PARITY ERROR?
3435 006672 001056 BNE FALPAR ;BR IF THERE IS ONE
3436 006674 032760 001000 000016 BIT #BIT09,$STATUS(RO) ;TIMEOUT?
3437 006702 001076 BNE SWTIM ;BR IF YES
3438 006704 032760 040002 000016 BIT #BIT14!BIT01,$STATUS(RO) ;DRIVE WENT OFFLINE ?
3439 006712 001111 BNE OFLIN ;BR IF IT DID
3440 006714 032760 000004 000016 BIT #BIT2,$STATUS(RO) ;PORT REQUEST TIME OUT ?
3441 006722 001141 BNE PRTIM ;BR IF IT DID
3442 006724 000207 RTS ;ERROR. RETURN

;DRIVE IS PERSISTENTLY UNSAFE
3443
3444
3445
3446 006726 104401 001165 PUNSAF: TYPE ,$CRLF ;CR-LF
3447 006732 104401 045410 TYPE ,EM12 ;'DRIVE UNSAFE' MESSAGE
3448 006736 104401 052673 TYPE ,DRNUM ;DRIVE NUMBER
3449 006742 013746 001246 MOV UNIT,-(SP) ;SAVE UNIT FOR TYPEOUT
3450 ;TYPE DRIVE NUMBER
3451 ;GO TYPE--OCTAL ASCII
3452 006746 104403 TYPES ;TYPE 2 DIGIT(S)
3453 006750 002 .BYTE 2 ;SUPPRESS LEADING ZEROS
3454 006751 000 .BYTE 0
3455 006752 104401 001165 TYPE , $CRLF ;CR-LF
3456 006756 004737 020200 JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
3457 006762 104414 045410 DISPLY ,EM12 ;PERSISTENT DEVICE UNSAFE MESSAGE
3458 006766 004737 020244 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
3459 006772 004737 020652 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
3460 006776 004737 021326 JSR PC,LINE4 ;PRINT LINE 4 OF THE ERROR MESSAGE
3461 007002 004737 023424 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
3462 007006 004737 021762 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE

```

NOS

CZRJDCO RPO4 5.6 MLT-DR LGC
CZRJDC.P11 15-DEC-77 10:58

MACY11 30(1046) 15-DEC-77 11:03 PAGE 66
MAIN PROGRAM

SEG 0065

```

3462 007012 000137 026446          JMP      DROP          ;DROP THE DRIVE
3463
3464          ;UNCORRECTABLE MASSBUS PARITY ERROR OCCURRED
3465
3466 007016 104401 001165          UCPAR:  TYPE      ,SCLRF          ;CR-LF
3467 007022 104401 045312          TYPE      ,EM10          ;'UNCORRECTABLE PARITY ERROR' MESSAGE
3468 007026 000404          BR        FALPR1       ;FINISH PROCESSING THE ERROR
3469
3470          ;'FATAL' MASSBUS PARITY ERROR OCCURRED
3471
3472 007030 104401 001165          FALPAR: TYPE      ,SCLRF          ;CR-LF
3473 007034 104401 045355          TYPE      ,EM11          ;'FATAL PARITY ERROR' MESSAGE
3474 007040 104401 052673          FALPR1: TYPE      ,DRNUM          ;DRIVE NUMBER
3475 007044 013746 001246          MOV      UNIT, -(SP)  ;SAVE UNIT FOR TYPEOUT
3476          ;TYPE DRIVE NUMBER
3477 007050 104403          TYPOS          ;GO TYPE--OCTAL ASCII
3478 007052          .BYTE      2          ;TYPE 2 DIGIT(S)
3479 007053          .BYTE      0          ;SUPPRESS LEADING ZEROS
3480 007054 104401 001165          TYPE      ,SCLRF          ;CR-LF
3481 007060 004737 023424          JSR      PC, INCTOT    ;INCREMENT TOTAL ERROR COUNT
3482 007064 032777 100000          BIT      #SW15, #SWR  ;HALT ON ERROR ?
3483 007072 001401          BEQ      1$          ;BR IF NOT
3484 007074 000000          HALT          ;ERROR HALT
3485 007076 000207          1$:      RTS      PC
3486
3487          ;SOFTWARE TIMEOUT OCCURRED
3488
3489 007100 004737 020200          SWTIM:  JSR      PC, LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
3490 007104 104414 045441          DISPLY   EM13          ;PRINT THE TIME OUT MESSAGE
3491 007110 004737 020244          JSR      PC, LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
3492 007114 004737 020652          JSR      PC, LINE3      ;PRINT LINE 3 OF ERROR MESSAGE
3493 007120 004737 021326          JSR      PC, LINE4      ;PRINT LINE 4 OF ERROR MESSAGE
3494 007124 004737 023424          JSR      PC, INCTOT    ;INCREMENT TOTAL ERROR COUNT
3495 007130 004737 021762          JSP      PC, LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
3496 007134 000207          RTS      PC          ;RETURN
3497
3498          ;DRIVE WENT OFFLINE
3499
3500 007136 104401 001165          OFLIN:  TYPE      ,SCLRF          ;CR-LF
3501 007142 104401 045513          TYPE      ,EM14          ;'DRIVE WENT OFFLINE' MESSAGE
3502 007146 104401 052673          TYPE      ,DRNUM          ;DRIVE NUMBER
3503 007152 013746 001246          MOV      UNIT, -(SP)  ;SAVE UNIT FOR TYPEOUT
3504          ;TYPE DRIVE NUMBER
3505 007156 104403          TYPOS          ;GO TYPE--OCTAL ASCII
3506 007160          .BYTE      2          ;TYPE 2 DIGIT(S)
3507 007161          .BYTE      0          ;SUPPRESS LEADING ZEROS
3508 007162 104401 001165          TYPE      ,SCLRF          ;CR-LF
3509 007166 004737 020200          JSR      PC, LINE1      ;PRINT LINE 1 OF THE ERROR MESSAGE
3510 007172 104414 045513          DISPLY   EM14          ;PRINT OFFLINE MESSAGE
3511 007176 004737 020244          JSR      PC, LINE2      ;PRINT LINE 2 OF THE ERROR MESSAGE
3512 007202 004737 020652          JSR      PC, LINE3      ;PRINT LINE 3 OF THE ERROR MESSAGE
3513 007206 004737 021326          JSR      PC, LINE4      ;PRINT LINE 4 OF THE ERROR MESSAGE
3514 007212 004737 023424          JSR      PC, INCTOT    ;INCREMENT TOTAL ERROR COUNT
3515 007216 004737 021762          JSR      PC, LINE7      ;PRINT LINE 7 OF THE ERROR MESSAGE
3516 007222 000137 026446          JMP      DROP          ;DROP THE DRIVE
3517

```

```

3518 ;PORT REQUEST TIMEOUT ERROR
3519
3520 007226 004737 020200 PRTIM: JSR PC,LINE1 ;TYPE LINE 1 OF THE ERROR MESSAGE
3521 007232 104414 045536 DISPLY EM15 ;PRINT PORT TIME OUT MESSAGE
3522 007236 004737 020244 JSR PC,LINE2 ;TYPE LINE 2 OF THE ERROR MESSAGE
3523 007242 004737 020652 JSR PC,LINE3 ;TYPE LINE 3 OF THE ERROR MESSAGE
3524 007246 004737 021326 JSR PC,LINE4 ;TYPE LINE 4 OF THE ERROR MESSAGE
3525 007252 004737 023424 JSR PC,INCR'OT ;INCREMENT TOTAL ERROR COUNT
3526 007256 004737 021762 JSR PC,LINE7 ;TYPE LINE 7 OF THE ERROR MESSAGE
3527 007262 000207 RTS PC ;RETURN
3528
3529 ;PROCESS ORDER COMPLETION WITH 'ERROR' & 'DONE' BITS SET
3530
3531 007264 032760 000030 000016 DONE: BIT #BIT04!BIT03,STATUS(RO) ;UNSAFE OCCURRED
3532 007272 001402 BEQ .+6 ;BR IF NOT
3533 007274 000137 012466 JMP UNSAF ;REPORT UNSAFE
3534 007300 032760 04000C 000244 BIT #BIT14,$RPCS2(RO) ;IS 'WCE' SET?
3535 007306 001006 BNE 1$ ;BR IF SET
3536 007310 032760 040000 000246 BIT #BIT14,$RPDS1(RO) ;CHECK 'ERR'
3537 007316 001002 BNE 1$ ;BR IF SET
3538 007320 000137 012232 JMP TRFER ;PROCESS 'TRE'
3539 007324 032760 000400 000250 1$: BIT #BIT08,$RPER1(RO) ;'HCRC' SET?
3540 007332 001402 BEQ .+6 ;BR IF NOT
3541 007334 000137 010710 JMP HCRCER ;PROCESS 'HCRC'
3542 007340 032760 000020 000250 BIT #BIT04,$RPER1(RO) ;'FMT' SET?
3543 007346 001402 BEQ .+6 ;BR IF NOT SET
3544 007350 000137 011072 JMP CKFMT ;CHECK FORMAT ERROR
3545 007354 032760 000200 000250 BIT #BIT07,$RPER1(RO) ;'HCE' SET?
3546 007362 001402 BEQ .+6 ;BR IF NOT SET
3547 007364 000137 011266 JMP CKHCE ;CHECK 'HCE' ERROR
3548 007370 032760 020000 000250 BIT #BIT13,$RPER1(RO) ;'OPI' SET?
3549 007376 001402 BEQ .+6 ;BR IF NOT SET
3550 007400 000137 011566 JMP OPIER ;REPORT 'OPI'
3551 007404 032760 000010 000250 BIT #BIT3,$RPER1(RO) ;'PAR' SET?
3552 007412 001402 BEQ .+6 ;BR IF NOT SET
3553 007414 000137 011720 JMP PARER ;REPORT 'PAR'
3554 007420 032760 000040 000250 BIT #BITS,$RPER1(RO) ;'WCF' SET?
3555 007426 001402 BEQ .+6 ;BR IF NOT SET
3556 007430 000137 012370 JMP WCFER ;REPORT 'WCF'
3557 007434 032760 002000 000250 BIT #BIT10,$RPER1(RO) ;'IAE' SET?
3558 007442 001402 BEQ .+6 ;BR IF NOT SET
3559 007444 000137 012012 JMP IAEER ;REPORT 'IAE'
3560 007450 032760 004000 000250 BIT #BIT11,$RPER1(RO) ;'WLE' SET?
3561 007456 001402 BEQ .+6 ;BR IF NOT SET
3562 007460 000137 012044 JMP WLEER ;REPORT 'WLE'
3563 007464 032760 001000 000250 BIT #BIT9,$RPER1(RO) ;'AOE' SET?
3564 007472 001405 BEQ 2$ ;BR IF NOT SET
3565 007474 032760 002000 000246 BIT #BIT10,$RPDS1(RO) ;'LST' SET?
3566 007502 001401 BEQ 2$ ;BR IF NOT SET
3567 007504 000207 RTS PC ;'AOE' & 'LST' SET, EXIT
3568 007506 032760 010000 000250 2$: BIT #BIT12,$RPER1(RO) ;SEE IF 'DTE' SET
3569 007514 001402 BEQ .+6 ;BR IF NOT
3570 007516 000137 011676 JMP DTEER ;REPORT 'DTE' ERROR
3571 007522 032760 040000 000244 BIT #BIT14,$RPCS2(RO) ;SEE IF 'WCK' SET
3572 007530 001402 BEQ .+6 ;BR IF NOT SET
3573 007532 000137 010362 JMP WCKER ;REPORT 'WCK'

```

```

3574 007536 005760 000250          TST      $RPER1(R0)          ;SEE IF 'DCK' SET
3575 007542 100002          BPL      .+6                ;BR IF NOT
3576 007544 000137 007570          JMP      DCKER              ;PROCESS 'DCK'
3577 007550 032760 140000 000276  BIT      #BIT15!BIT14,$RPER3(R0) ;'SKI' OR 'OCYL' SET
3578 007556 001402          BEQ      .+6                ;BR IF NOT SET
3579 007560 000137 012332          JMP      SKIER              ;REPORT ERROR
3580 007564 000137 011040          JMP      DRIVER            ;REPORT DRIVE ERROR
3581
3582          ;PROCESS DATA ('DCK') CHECK ERROR
3583
3584 007570 022760 010042 000300  DCKER:  CMP      #10042,$RPEC1(R0) ;VALID POSITION COUNT ?
3585 007576 101406          BLOS    1$                 ;BR IF NOT VALID
3586 007600 005760 000300          TST      $RPEC1(R0)        ;POSITION COUNT 0 ?
3587 007604 001403          BEQ      1$                 ;BR IF 0'S
3588 007606 005760 000302          TST      $RPEC2(R0)        ;VALUE IN PATTERN REGISTER ?
3589 007612 001026          BNE     4$                 ;BR IF YES
3590 007614 004737 020200          JSR     PC,LINE1           ;TYPE FIRST LINE OF ERROR MESSAGE
3591 007620 104414 047143          DISPLY  EM45              ;TYPE 'ECC LOGIC ERROR'
3592 007624 004737 020244          JSR     PC,LINE2           ;TYPE LINE 2 OF ERROR MESSAGE
3593 007630 004737 023424          JSR     PC,INCTOT         ;INCREMENT TOTAL ERROR COUNT
3594 007634 012737 000003 001252  MOV     #3,$RETRY          ;RETRY COUNT
3595 007642 004737 015400          JSR     PC,$RETRY         ;RETRY THE ORDER
3596 007646 000403          BR      2$                 ;RETRY WAS NOT SUCCESSFUL
3597 007650 004737 021700          JSR     PC,LINE6C         ;TYPE LINE 6C OF ERROR MESSAGE
3598 007654 000402          BR      3$                 ;FINISH THE ERROR REPORT
3599 007656 004737 021706 2$:      JSR     PC,LINE6D         ;TYPE LINE 6D OF ERROR MESSAGE
3600 007662 004737 021762 3$:      JSR     PC,LINE7         ;TYPE LINE 7 OF ERROR MESSAGE
3601 007666 000402          BR      5$                 ;EXIT
3602 007670 004737 020042 4$:      JSR     PC,SPOTCK         ;SEE IF ERROR AT A BAD SPOT ON THE PACK
3603 007674 000207 000001 5$:      RTS     PC                ;IT IS, DON'T REPORT IT
3604 007676 126027 000024 6$:      CMPB   $CODE(R0),#1      ;IS ORDER A WRITE CHECK ?
3605 007704 101002          BHI     6$                 ;BR IF NOT
3606 007706 000137 010362          WCKER  JMP              ;REPORT ERROR UNDER WRITE CHECK PROCESSING
3607 007712 004737 020200 6$:      JSR     PC,LINE1         ;PRINT LINE 1 OF ERROR MESSAGE
3608 007716 104414 045613          DISPLY  EM21              ;DATA CHECK ERROR
3609 007722 004737 020244  DCKER1: JSR     PC,LINE2         ;PRINT LINE 2 OF ERROR MESSAGE
3610 007726 004737 020652          JSR     PC,LINE3         ;PRINT LINE 3 OF ERROR MESSAGE
3611 007732 004737 021326          JSR     PC,LINE4         ;PRINT LINE 4 OF ERROR MESSAGE
3612 007736 004737 015042          JSR     PC,PRTBAD         ;SEE IF BAD SECTOR TO BE PRINTED
3613 007742 012737 110100 001250  MOV     #BIT15!BIT12!BIT06,MASK ;LOAD ERROR MASK
3614 007750 032760 010100 000250  BIT     #BIT12!BIT06,$RPER1(R0) ;CHECK 'DTE' & 'ECH'
3615 007756 001003          BNE     1$                 ;BR IF SET
3616 007760 004737 021652          JSR     PC,LINE6         ;PRINT LINE 6 OF ERROR MESSAGE
3617 007764 000541          BR      8$                 ;FINISH THE ERROR REPORT
3618 007766 012737 000020 001252 1$:      MOV     #16,$RETRY        ;RETRY COUNT
3619 007774 005001          CLR     R1                 ;R1 IS OFFSET CODE POINTER
3620 007776 032760 000002 000262  BIT     #BIT01,$RPDT(R0)   ;IS DRIVE AN RPO6 ?
3621 010004 001002          BNE     16$                ;BR IF IT IS
3622 010006 012701 000007          MOV     #7,R1             ;INCREMENT PAST RPO6 OFFSET CODES
3623 010012 004737 016026 16$:      JSR     PC,RELBUF        ;RELEASE THE BUFFER
3624 010016 004737 022314          JSR     PC,READDR        ;GET THE ADDRESS OF THE ERROR SECTOR
3625 010022 112660 000011          MOV     (SP)+,$TRK(R0)    ;TRACK ADDRESS OF ERROR SECTOR
3626 010026 112660 000010          MOV     (SP)+,$SEC(R0)   ;SECTOR ADDRESS OF ERROR SECTOR
3627 010032 016060 000272 000012  MOV     $RPCC(R0),$CYL(R0) ;PRESENT CYLINDER
3628 010040 026060 000022 000020  CMP     $SSEC(R0),$WRDL(R0) ;SEE IF TRANSFER LENGTH LESS THAN 1 SECTOR
3629 01004E 103C10          BHIS   15$                ;BR IF IT IS; USE PRESENT TRANSFER LENGTH

```

```

3630 010050 016060 000022 000020      MOV      $SSEC(RO), $SWDL(RO) ; CHANGE TRANSFER SIZE TO 1 SECTOR
3631 010056 016060 000020 000004      MOV      $SWDL(RO), $SWDM(RO) ; SETUP WORD COUNT FOR OPERATION
3632 010064 005460 000004      NEG      $SWDM(RO) ; CHANGE COUNT TO 2'S COMP
3633 010070 005046      CLR      -(SP) ; SPACE FOR NEW BUFFER ADDRESS
3634 010072 004737 015672      JSR      PC, GETBUF ; GET A BUFFER
3635 010076 012660 000006      MOV      (SP)+, $BUF(RO) ; NEW BUFFER ADDRESS TO DPB
3636 010102 004737 016370      JSR      PC, GODRIV ; RETRY
3637 010106 005760 000016      TST      $STATUS(RO) ; TEST FOR DONE
3638 010112 001775      BEQ      3$ ; BR IF NOT DONE
3639 010114 100075      BPL      10$ ; BR IF NOT ERROR
3640 010116 032760 000200 000016      BIT      #BIT7, $STATUS(RO) ; SEE IF ORDER TERMINATED NORMALLY
3641 010124 001006      BNE      14$ ; BR IF NOT
3642 010126 004737 023424      JSR      PC, INCTOT ; INCREMENT TOTAL ERROR COUNT
3643 010132 104414 051330      DISPLY  LINBM ; 'DIFFERENT ERROR DURING RETRY'
3644 010136 000137 006644      JMP      ERPRC1 ; SEE WHICH ERROR
3645 010142 033760 001250 000250 14$:      BIT      MASK, $RPER1(RO) ; LOOK AT CURRENT ERROR
3646 010150 001430      BEQ      5$ ; BR IF DIFFERENT ERROR
3647 010152 032760 010100 000250      BIT      #BIT12:BIT6, $RPER1(RO) ; 'ECH' OR 'DTE' STILL SET ?
3648 010160 001437      BEQ      7$ ; BR IF NEITHER SET
3649 010162 105237 001253      INCB    RETRY+1 ; INCREMENT RETRY COUNT
3650 010166 123737 001252 001253      CMPB    RETRY, RETRY+1 ; DONE ?
3651 010174 001342      BNE      2$ ; BR IF NOT
3652 010176 005201      INC     R1 ; INCREMENT TABLE INDEX
3653 010200 116137 002220 044731      MOVB    OFFCOD(R1), GENDPB+$FMT ; OFFSET CODE
3654 010206 001435      BEQ      9$ ; BR IF END OF OFFSET TABLE
3655 010210 062737 000002 001252      ADD     #2, RETRY ; NEW RETRY LIMIT
3656 010216 004737 015272      JSR      PC, OFFST ; OFFSET
3657 010222 005737 044746      TST     GENDPB+$STATUS ; SEE IF FINISHED WITH OFFSET
3658 010226 001775      BEQ      4$ ; BR IF NOT
3659 010230 100324      BPL      2$ ; BR IF NO ERROR PERFORMING OFFSET
3660 010232 004737 022230      JSR      PC, LINE8 ; PRINT LINE 8 OF ERROR MESSAGE
3661 010236 004737 023330      JSR      PC, INCHRD ; INCREMENT 'HARD' ERROR COUNT
3662 010242 004737 023424      JSR      PC, INCTOT ; INCREMENT TOTAL ERROR COUNT
3663 010246 004737 021762      JSR      PC, LINE7 ; PRINT LINE 7 OF ERROR MESSAGE
3664 010252 004737 015042      JSR      PC, PRBAD ; PRINT THE BAD SECTOR
3665 010256 000436      BR      13$ ; CLEAN UP AND RETURN
3666 010260 004737 021672      JSR      PC, LINE6B ; PRINT LINE 6B OF ERROR MESSAGE
3667 010264 004737 021610      JSR      PC, LINE5B ; PRINT LINE 5B OF THE ERROR MESSAGE
3668 010270 004737 023304      JSR      PC, INCSOF ; INCREMENT 'SOFT' ERROR COUNT
3669 010274 004737 014302      JSR      PC, ECC ; CORRECT THE ERROR USING ECC AND CHECK IT
3670 010300 000407      BR      11$ ; COMPARE THE BUFFER
3671 010302 004737 021706      JSR      PC, LINE6D ; PRINT LINE 6D OF ERROR MESSAGE
3672 010306 000753      BR      6$ ; INCREMENT ERROR COUNT
3673 010310 004737 021664      JSR      PC, LINE6A ; PRINT LINE 6A OF ERROR MESSAGE
3674 010314 004737 023304      JSR      PC, INCSOF ; INCREMENT 'SOFT' ERROR COUNT
3675 010320 012737 000001 001300 11$:      MOV     #1, FRSTER ; SET PROCESSING 'DCKER' INDICATOR
3676 010326 004737 013172      JSR      PC, CMPARD ; COMPARE THE BUFFER
3677 010332 105737 001301      TSTB   FRSTER+1 ; ERROR IN COMPARE ?
3678 010336 100406      BMI     13$ ; BRANCH IF ERROR
3679 010340 004737 023424      JSR      PC, INCTOT ; INCREMENT TOTAL ERROR COUNT
3680 010344 104414 051542      DISPLY  LIN9G ; 'DATA COMPARE OK' MESSAGE
3681 010350 004737 021762      JSR      PC, LINE7 ; PRINT LINE 7 OF ERROR MESSAGE
3682 010354 004737 006374      JSR      PC, REFMT ; REFORMAT THE ERROR SECTOR
3683 010360 000207      RTS     PC ; RETURN
3684
3685

```

:WRITE CHECK ERROR PROCESSING

```

3686
3687 010362 032760 100000 000250 WCKER: BIT #BIT15,$RPER1(RO) ;SEE IF 'DCK' SET ALSO
3688 010370 001034 BNE 2$ ;BR IF IT IS
3689 010372 004737 020200 JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
3690 010376 104414 045717 DISPLY ,EM23 ;PRINT WCE & DCK NOT
3691 010402 005037 001250 CLR MASK ;CLEAR ERROR MASK
3692 010406 004737 020244 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
3693 010412 004737 020652 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
3694 010416 004737 021326 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
3695 010422 004737 021416 JSR PC,LINES ;PRINT LINE 5 OF ERROR MESSAGE
3696 010426 004737 023424 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
3697 010432 012737 000003 001252 MOV #3,RETRY ;RETRY LIMIT
3698 010440 004737 015400 JSR PC,$RETRY ;RETRY THE OPERATION
3699 010444 000403 BR 1$ ;RETRY UNSUCCESSFUL
3700 010446 004737 021700 JSR PC,LINE6C ;PRINT LINE 6C OF ERROR MESSAGE
3701 010452 000502 BR 8$ ;FINISH PROCESSING THE ERROR
3702 010454 004737 021706 1$: JSR PC,LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
3703 010460 000506 BR 10$ ;FINISH PROCESSING THE ERROR
3704 010462 004737 020042 2$: JSR PC,SPOTCK ;SEE IF ERROR AT BAD SPOT ON THE PACK
3705 010466 000507 BR 11$ ;EXIT IF AT BAD SPOT ON PACK
3706 010470 004737 020200 JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
3707 010474 012737 045644 010522 MOV #EM22,13$ ;ASSUME THAT EM22 WILL BE PRINTED
3708 010502 032760 040000 000244 BIT #BIT14,$RPCS2(RO) ;DID 'WCK' ALSO SET ?
3709 010510 001003 BNE 12$ ;BR IF IT DID
3710 010512 012737 046545 010522 MOV #EM37,13$ ;MESSAGE FOR 'DCK' AND 'WCK' NOT DURING
3711 ;WRITE CHECK
3712 010520 104414 12$: DISPLY ;TYPE THE ERROR MESSAGE
3713 010522 000000 13$: .WORD 0 ;MESSAGE ADDRESS GOES HERE
3714 010524 004737 020244 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
3715 010530 004737 020652 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
3716 010534 004737 021326 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
3717 010540 004737 021416 JSR PC,LINES ;PRINT LINE 5 OF ERROR MESSAGE
3718 010544 032760 000100 000250 BIT #BIT06,$RPER1(RO) ;'ECH' SET ALSO ?
3719 010552 001442 BEQ 8$ ;FINISH PROCESSING THE ERROR
3720 010554 012737 000020 001252 3$: MOV #16,RETRY ;RETRY LIMIT - 16 (10)
3721 010562 004737 016370 4$: JSR PC,GODRIV ;RETRY THE ORDER
3722 010566 005760 000016 5$: TST $STATUS(RO) ;ORDER FINISHED ?
3723 010572 001775 BEQ 5$ ;BR IF NOT
3724 010574 100405 BMI 6$ ;BR IF ERROR ON ORDER
3725 010576 105237 001253 INCB RETRY+1 ;INCREMENT RETRY COUNT
3726 010602 004737 021700 JSR PC,LINE6C ;PRINT LINE 6C OF ERROR MESSAGE
3727 010606 000431 BR 9$ ;FINISH ERROR PROCESSING
3728 010610 105237 001253 6$: INCB RETRY+1 ;INCREMENT RETRY COUNT
3729 010614 123737 001252 001253 CMPB RETRY,RETRY+1 ;DONE ?
3730 010622 001714 BEQ 1$ ;BR IF AT RETRY LIMIT
3731 010624 032760 100000 000250 BIT #BIT15,$RPER1(RO) ;'DCK' SET
3732 010632 001407 BEQ 7$ ;BR IF NOT - DIFFERENT ERROR
3733 010634 032760 000100 000250 BIT #BIT06,$RPER1(RO) ;'ECH' ALSO SET ?
3734 010642 001347 BNE 4$ ;BR IF IT IS, RETRY ORDER
3735 010644 004737 021700 JSR PC,LINE6C ;PRINT LINE 6C OF ERROR MESSAGE
3736 010650 000403 BR 8$ ;FINISH PROCESSING ERROR
3737 010652 004737 022230 7$: JSR PC,LINE8 ;PRINT LINE 8 - 'DIFFERENT ERROR'
3738 010656 000405 BR 9$ ;FINISH PROCESSING ERROR
3739 010660 004737 023424 8$: JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
3740 010664 004737 021762 JSR PC,LINE7 ;FINISH THE ERROR MESSAGE
3741 010670 000406 BR 11$ ;EXIT

```

```

3742 010672 004737 023424 9$: JSR PC,INCTOT ; INCREMENT TOTAL ERROR COUNT
3743 010676 004737 021762 10$: JSR PC,LINE7 ; FINISH THE ERROR MESSAGE
3744 010702 004737 006374 JSR PC,REFMT ; REFORMAT THE SECTOR IN ERROR
3745 010706 000207 11$: RTS PC ; RETURN
3746
3747 ;REPORT 'HCRC' ERROR
3748
3749 010710 004737 020042 HRCRER: JSR PC,SPOTCK ; SEE IF ERROR AT PACK BAD SPOT
3750 010714 000450 BR 3$ ; EXIT IF IT IS
3751 010716 004737 020200 JSR PC,LINE1 ; PRINT LINE 1 OF ERROR MESSAGE
3752 010722 104414 045572 DISPLY EM20 ; REPORT 'HCRC'
3753 010726 004737 020244 JSR PC,LINE2 ; PRINT LINE 2 OF ERROR MESSAGE
3754 010732 004737 020652 JSR PC,LINE3 ; PRINT LINE 3 OF ERROR MESSAGE
3755 010736 004737 021326 JSR PC,LINE4 ; PRINT LINE 4 OF ERROR MESSAGE
3756 010742 032760 040000 000244 BIT #BIT14,$RPCS2(RO) ; 'WCE' ERROR ALSO ?
3757 010750 001402 BEQ 1$ ; BR IF NOT
3758 010752 004737 021416 JSR PC,LINES ; DISPLAY WORDS WHICH CAUSED 'WCE'
3759 010756 004737 023304 1$: JSR PC,INCSOF ; INCREMENT 'SOFT' ERROR COUNT
3760 010762 004737 023424 JSR PC,INCTOT ; INCREMENT TOTAL ERROR COUNT
3761 010766 012737 000400 001250 MOV #BIT8,MASK ; SET ERROR MASK
3762 010774 012737 000003 001252 MOV #3,RETRY ; RETRY LIMIT
3763 011002 004737 015400 JSR PC,$RETRY ; RETRY ORDER
3764 011006 000405 BR 2$ ; RETRY NOT SUCCESSFUL
3765 011010 004737 021700 JSR PC,LINE6C ; PRINT LINE 6C OF ERROR MESSAGE
3766 011014 004737 021762 JSR PC,LINE7 ; PRINT LINE 7 OF ERROR MESSAGE
3767 011020 000406 BR 3$ ; EXIT
3768 011022 004737 021706 2$: JSR PC,LINE6D ; PRINT LINE 6D OF ERROR MESSAGE
3769 011026 004737 021762 JSR PC,LINE7 ; PRINT LINE 7 OF ERROR MESSAGE
3770 011032 004737 006374 JSR PC,REFMT ; REFORMAT THE ERROR SECTOR
3771 011036 000207 3$: RTS PC ; RETURN
3772
3773 ;REPORT DRIVE ERROR
3774
3775 011040 004737 020200 DRIVER: JSR PC,LINE1 ; PRINT LINE 1 OF ERROR MESSAGE
3776 011044 104414 046207 DISPLY EM30 ; REPORT DRIVE ERROR
3777 011050 004737 020244 JSR PC,LINE2 ; PRINT LINE 2 OF ERROR MESSAGE
3778 011054 004737 020652 JSR PC,LINE3 ; PRINT LINE 3 OF ERROR MESSAGE
3779 011060 004737 023424 JSR PC,INCTOT ; INCREMENT TOTAL ERROR COUNT
3780 011064 004737 021762 JSR PC,LINE7 ; PRINT LINE 7 OF ERROR MESSAGE
3781 011070 000207 RTS PC ; RETURN
3782
3783 ;PROCESS FORMAT ('FER') ERROR
3784
3785 011072 032760 000400 000250 CKFMT: BIT #BIT8,$RPER!(RO) ; 'HCRC' SET ON ORIGINAL ERROR ?
3786 011100 001402 BEQ 1$ ; BR IF NOT SET
3787 011102 000137 010710 JMP HRCRER ; REPORT HCRC ERROR
3788 011106 004737 022314 1$: JSR PC,READDR ; GET CORRECTED TRACK & SECTOR ADDRSSES
3789 011112 004737 015316 JSR PC,READHD ; READ HEADER
3790 011116 032737 000400 044764 BIT #BIT8,GENREG+RPER1 ; 'HCRC' SET WHEN HEADER READ?
3791 011124 001002 BNE 2$ ; BR IF 'HCRC' SET
3792 011126 000137 012072 JMP FMTER ; NO ERROR IS 'FMT' ONLY
3793 011132 004737 020042 2$: JSR PC,SPOTCK ; SEE IF ERROR AT BAD SPOT ON THE PACK
3794 011136 000452 BR 3$ ; EXIT IF IT IS
3795 011140 004737 020200 JSR PC,LINE1 ; PRINT LINE 1 OF ERROR MESSAGE
3796 011144 104414 045776 DISPLY EM24 ; HEADER READ ERROR - FMT BIT DROPPED UP
3797 011150 004737 020244 JSR PC,LINE2 ; PRINT LINE 2 OF ERROR MESSAGE

```

3799	011154	004737	020652			JSR	PC,LINE3	:PRINT LINE 3 OF ERROR MESSAGE
3799	011160	004737	02132C			JSR	PC,LINE4	:PRINT LINE 4 OF ERROR MESSAGE
3800	011164	032760	040000	000244		BIT	#BIT14,\$RPCS2(RO)	: 'WCE' ERROR ALSO ?
3801	011172	001402				BEG	3\$:BR IF NOT
3802	011174	004737	021416			JSR	PC,LINE5	:DISPLAY WORDS WHICH CAUSED 'WCE'
3803	011200	004737	021516		3\$:	JSR	PC,LINE5A	:DISPLAY HEADER
3804	011204	004737	023304			JSR	PC,INCSOF	:INCREMENT SOFT ERROR COUNT
3805	011210	004737	023424			JSR	PC,INCTOT	:INCREMENT TOTAL ERROR COUNT
3806	011214	012737	000020	001250		MOV	#BIT4,MASK	:SET ERROR MASK
3807	011222	012737	000003	001252		MOV	#3,RETRY	:RETRY LIMIT
3808	011230	004737	015400			JSR	PC,\$RETRY	:RETRY THE ORDER
3809	011234	000405				BR	4\$:RETRY NOT SUCESSFUL
3810	011236	004737	021700			JSR	PC,LINE6C	:PRINT LINE 6C OF ERROR MESSAGE
3811	011242	004737	021762			JSR	PC,LINE7	:PRINT LINE 7 OF ERROR MESSAGE
3812	011246	000406				BR	5\$:EXIT
3813	011250	004737	021706		4\$:	JSR	PC,LINE6D	:PRINT LINE 6D OF ERROR MESSAGE
3814	011254	004737	021762			JSR	PC,LINE7	:PRINT LINE 7 OF ERROR MESSAGE
3815	011260	004737	006374			JSR	PC,REFMT	:REFORMAT THE ERROR SECTOR
3816	011264	000207			5\$:	RTS	PC	:RETURN
3817								
3818								:PROCESS HEADER COMPARE ('HCE') ERROR
3819								
3820	011266	032760	000400	000250	CKHCE:	BIT	#BIT8,\$RPER1(RO)	:HCRC SET ON ORIGINAL ERROR ?
3821	011274	001402				BEG	1\$:BR IF NOT SET
3822	011276	000137	010710			JMP	HCR CER	:REPORT HEADER CRC ERROR
3823	011302	004737	022314		1\$:	JSR	PC,READDR	:GET CURRENT SECTOR & TRACK ADDRS
3824	011306	004737	015316			JSR	PC,READHD	:READ HEADER OF CURRENT SECTOR
3825	011312	032737	000400	044764		BIT	#BIT8,GENREG+RPER1	: 'HCRC' SET ?
3826	011320	001016				BNE	3\$:BR IF SET
3827	011322	042737	010000	054456		BIC	#BIT12,CYLDER	:CLEAR FORMAT BIT FROM HEADER
3828	011330	026037	000272	054456		CMP	\$RPCC(RO),CYLDER	:CORRECT CYLINDER ?
3829	011336	001402				BEG	2\$:BR IF IT IS
3830	011340	000137	011512			JMP	POSER	:REPORT POSITIONING ERROR
3831	011344	052737	010000	054456	2\$:	BIS	#BIT12,CYLDER	:RESTORE THE FORMAT BIT
3832	011352	000137	012150			JMP	HCEER	:REPORT 'HCE' ERROR
3833	011356	004737	020042		3\$:	JSR	PC,SPOTCK	:SEE IF ERROR AT BAD SPOT
3834	011362	000452				BR	6\$:EXIT IF IT IS
3835	011364	004737	020200			JSR	PC,LINE1	:PRINT LINE 1 OF ERROR MESSAGE
3836	011370	104414	046044			DISPLY	EM25	:HEADER READ ERROR - 'HCE' SET
3837	011374	004737	020244			JSR	PC,LINE2	:PRINT LINE 2 OF ERROR MESSAGE
3838	011400	004737	020652			JSR	PC,LINE3	:PRINT LINE 3 OF ERROR MESSAGE
3839	011404	004737	021326			JSR	PC,LINE4	:PRINT LINE 4 OF ERROR MESSAGE
3840	011410	032760	040000	000244		BIT	#BIT14,\$RPCS2(RO)	: 'WCE' ERROR ALSO ?
3841	011416	001402				BEG	4\$:BR IF NOT
3842	011420	004737	021416			JSR	PC,LINE5	:DISPLAY WORDS WHICH CAUSED 'WCE'
3843	011424	004737	021516		4\$:	JSR	PC,LINE5A	:PRINT LINE 5 OF ERROR MESSAGE
3844	011430	004737	023304			JSR	PC,INCSOF	:INCREMENT SOFT ERROR COUNT
3845	011434	004737	023424			JSR	PC,INCTOT	:INCREMENT TOTAL ERROR COUNT
3846	011440	012737	000200	001250		MOV	#BIT7,MASK	:SET ERROR MASK
3847	011446	012737	000003	001252		MOV	#3,RETRY	:RETRY LIMIT
3848	011454	004737	015400			JSR	PC,\$RETRY	:RETRY THE ORDER
3849	011460	000405				BR	5\$:RETRY NOT SUCESSFUL
3850	011462	004737	021700			JSR	PC,LINE6C	:PRINT LINE 6C OF ERROR MESSAGE
3851	011466	004737	021762			JSR	PC,LINE7	:PRINT LINE 7 OF ERROR MESSAGE
3852	011472	000406				BR	6\$:EXIT
3853	011474	004737	021706		5\$:	JSP	PC,LINE6C	:PRINT LINE 6D OF ERROR MESSAGE


```

3854 011500 004737 021762      JSR      PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
3855 011504 004737 006374      JSR      PC,REFMT      ;REFORMAT THE ERROR SECTOR
3856 011510 000207                RTS      PC              ;RETURN
3857
3858      ;REPORT POSSIBLE POSITIONING ERROR
3859
3860 011512 004737 015240      POSER:  JSR      PC,RECALT      ;RECALIBRATE
3861 011516 004737 020200      JSR      PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
3862 011522 104414 047372      DISPLY  EM51          ;PROGRAM DETECTED POSITIONING ERROR
3863 011526 004737 020244      JSR      PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
3864 011532 004737 020700      JSR      PC,LINE3C     ;PRINT LINE 3C OF ERROR MESSAGE
3865 011536 052737 010000      BIS      #BIT12,CYLDER ;RESTORE THE FORMAT BIT
3866 011544 004737 021516      JSR      PC,LINE5A     ;PRINT LINE 5A OF THE ERROR MESSAGE
3867 011550 004737 023400      JSR      PC,INCMIS     ;INCREMENT MISPOSITIONING COUNT
3868 011554 004737 023424      JSR      PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
3869 011560 004737 022110      JSR      PC,LINE7A    ;PRINT LINE 7A OF ERROR MESSAGE
3870 011564 000207                RTS      PC              ;EXIT
3871
3872      ;REPORT 'OPI' ERROR
3873
3874 011566 004737 020042      OPIER:  JSR      PC,SPOTCK     ;SEE IF ERROR AT BAD SPOT
3875 011572 000207                RTS      PC              ;RETURN IF IT IS
3876 011574 004737 020200      JSR      PC,LINE1     ;PRINT LINE 1 OF ERROR MESSAGE
3877 011600 104414 046241      DISPLY  EM31          ;'OPI' ERROR
3878 011604 004737 020244      JSR      PC,LINE2     ;PRINT LINE 2 OF ERROR MESSAGE
3879 011610 004737 020652      JSR      PC,LINE3     ;PRINT LINE 3 OF ERROR MESSAGE
3880 011614 004737 021326      JSR      PC,LINE4     ;PRINT LINE 4 OF ERROR MESSAGE
3881 011620 004737 023424      JSR      PC,INCTOT    ;INCREMENT TOTAL ERROR COUNT
3882 011624 012737 020000      MOV      #BIT13,MASK  ;ERROR MASK
3883 011632 012737 000003      OPIER1: MOV      #3,RETRY ;RETRY LIMIT
3884 011640 004737 015400      JSR      PC,$RETRY    ;RETRY THE ORDER
3885 011644 000405                BR      IS             ;RETRY UNSUCCESSFUL
3886 011646 004737 021700      JSR      PC,LINE6C    ;PRINT LINE 6C OF ERROR MESSAGE
3887 011652 004737 021762      JSR      PC,LINE7     ;PRINT LINE 7 OF ERROR MESSAGE
3888 011656 000207                RTS      PC              ;EXIT
3889 011660 004737 021706      IS:     JSR      PC,LINE6D    ;PRINT LINE 6D OF ERROR MESSAGE
3890 011664 004737 021762      JSR      PC,LINE7     ;PRINT LINE 7 OF ERROR MESSAGE
3891 011670 004737 006374      JSR      PC,REFMT     ;REFORMAT THE ERROR SECTOR
3892 011674 000207                RTS      PC              ;RETURN
3893
3894      ;REPORT 'DTE' ERROR
3895
3896 011676 004737 020042      DTEER:  JSR      PC,SPOTCK     ;SEE IF ERROR AT BAD SPOT
3897 011702 000207                RTS      PC              ;RETURN IF IT IS
3898 011704 004737 020200      JSR      PC,LINE1     ;PRINT LINE 1 OF ERROR MESSAGE
3899 011710 104414 046304      DISPLY  EM32          ;'DTE' ERROR
3900 011714 000137 007722      JMP      DCKER1       ;FINISH PROCESSING THE 'DTE' ERROR
3901
3902      ;REPORT 'PAR' ERROR
3903
3904 011720 004737 020200      PARER:  JSR      PC,LINE1     ;PRINT LINE 1 OF ERROR MESSAGE
3905 011724 104414 046337      DISPLY  EM33          ;REPORT 'PAR'
3906 011730 004737 020244      JSR      PC,LINE2     ;PRINT LINE 2 OF ERROR MESSAGE
3907 011734 004737 020756      JSR      PC,LINE3E    ;PRINT LINE 3E OF ERROR MESSAGE
3908 011740 004737 021326      JSR      PC,LINE4     ;PRINT LINE 4 OF ERROR MESSAGE
3909 011744 004737 023424      JSR      PC,INCTOT    ;INCREMENT TOTAL ERROR COUNT

```

```

3910 011750 012737 000010 001250      MOV      #BIT03,MASK      ;ERROR MASK
3911 011756 012737 000003 001252      MOV      #3,RETRY       ;RETRY LIMIT
3912 011764 004737 015400      JSR      PC,$RETRY      ;RETRY ORDER
3913 011770 000405                BR       2$             ;RETRY UNSUCCESSFUL
3914 011772 004737 021700      JSR      PC,LINE6C      ;RETRY SUCCESSFUL
3915 011776 004737 021762      1$:     JSR      PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
3916 012002 000207                RTS                ;EXIT
3917 012004 004737 021706      2$:     JSR      PC,LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
3918 012010 000772                BR       1$           ;FINISH ERROR MESSAGE
3919
3920                ;REPORT 'IAE' ERROR
3921
3922 012012 004737 020200      IAEER:  JSR      PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
3923 012016 104414 046456      DISPLY  EM35          ;REPORT 'IAE'
3924 012022 004737 020244      JSR      PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
3925 012026 004737 021044      JSR      PC,LINE3F      ;PRINT LINE 3F OF ERROR MESSAGE
3926 012032 004737 023424      JSR      PC,INCTOT      ;INCREMENT TOTAL ERROR COUNT
3927 012036 004737 021762      JSR      PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
3928 012042 000207                RTS                ;RETURN
3929
3930                ;REPORT 'WLE' ERROR
3931
3932 012044 004737 020200      WLEER:  JSR      PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
3933 012050 104414 046514      DISPLY  EM36          ;REPORT 'WLE'
3934 012054 004737 020244      JSR      PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
3935 012060 004737 023424      JSR      PC,INCTOT      ;INCREMENT TOTAL ERROR COUNT
3936 012064 004737 021762      JSR      PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
3937 012070 000207                RTS                ;RETURN
3938
3939                ;REPORT FORMAT ERROR
3940
3941 012072 004737 020200      FMTER:  JSR      PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
3942 012076 104414 046125      DISPLY  EM26          ;FORMAT ERROR
3943 012102 004737 020244      JSR      PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
3944 012106 004737 020652      JSR      PC,LINE3      ;PRINT LINE 3 OF ERROR MESSAGE
3945 012112 004737 021326      JSR      PC,LINE4      ;PRINT LINE 4 OF ERROR MESSAGE
3946 012116 032760 040000 000244      BIT     #BIT14,$RPCS2(RO) ;'WCE' ERROR ALSO ?
3947 012124 001402                BEQ     1$            ;BR IF NOT
3948 012126 004737 021416      1$:     JSR      PC,LINES ;DISPLAY WORDS WHICH CAUSED 'WCE'
3949 012132 004737 021516      JSR      PC,LINESA      ;PRINT LINE 5A OF ERROR MESSAGE
3950 012136 004737 023424      JSR      PC,INCTOT      ;INCREMENT TOTAL ERROR COUNT
3951 012142 004737 021762      JSR      PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
3952 012146 000207                RTS                ;RETURN
3953
3954                ;REPORT HEADER COMPARE ERROR
3955
3956 012150 004737 020200      HCEER:  JSR      PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
3957 012154 104414 046152      DISPLY  EM27          ;HEADER COMPARE ERROR
3958 012160 004737 020244      JSR      PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
3959 012164 004737 020652      JSR      PC,LINE3      ;PRINT LINE 3 OF ERROR MESSAGE
3960 012170 004737 021326      JSR      PC,LINE4      ;PRINT LINE 4 OF ERROR MESSAGE
3961 012174 032760 040000 000244      BIT     #BIT14,$RPCS2(RO) ;'WCE' ERROR ALSO ?
3962 012202 001402                BEQ     1$            ;BR IF NOT
3963 012204 004737 021416      1$:     JSR      PC,LINES ;DISPLAY WORDS WHICH CAUSED 'WCE'
3964 012210 004737 021516      JSR      PC,LINESA      ;PRINT LINE 5A OF ERROR MESSAGE
3965 012214 004737 023424      JSR      PC,INCTOT      ;INCREMENT TOTAL ERROR COUNT

```

```

3966 012220 004737 021762 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
3967 012224 004737 006374 JSR PC,REFMT ;REFORMAT THE ERROR SECTOR
3968 012230 000207 RTS PC ;RETURN
3969
3970 ;PROCESS CONTROL/INTERFACE TRANSFER ERROR
3971
3972 012232 004737 020200 TRFER: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
3973 012236 104414 046627 DISPLY EM40 ;RH11 OR UNIBUS TRANSFER ERROR
3974 012242 004737 020244 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
3975 012246 004737 020652 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
3976 012252 004737 021326 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
3977 012256 004737 023424 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
3978 012262 032760 121400 000244 BIT #BIT15!BIT13!BIT9!BIT8,$RPCS2(RO) ;'DLT','UPE','MXF','MDPE' SET ?
3979 012270 001415 BEQ 2$ ;BR IF NONE SET
3980 012272 012737 000003 001252 MOV #3,RETRY ;RETRY LIMIT
3981 012300 005037 001250 CLR MASK ;CLEAR ERROR MASK
3982 012304 004737 015400 JSR PC,$RETRY ;RETRY THE OPERATION
3983 012310 000403 BR 1$ ;RETURN HERE IF RETRY UNSUCCESSFUL
3984 012312 004737 021700 JSR PC,LINE6C ;PRINT LINE 6C OF ERROR MESSAGE
3985 012316 000402 BR 2$ ;FINISH THE ERROR REPORT
3986 012320 004737 021706 1$: JSR PC,LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
3987 012324 004737 021762 2$: JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
3988 012330 000207 RTS PC
3989
3990 ;PROCESS 'SKI' OR 'OCYL' ERRORS
3991
3992 012332 004737 020200 SKIER: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
3993 012336 104414 047303 DISPLY EM50 ;'SKI' OR 'OCYL' ERROR
3994 012342 004737 020244 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
3995 012346 004737 020666 JSR PC,LINE3B ;PRINT LINE 3B OF ERROR MESSAGE
3996 012352 004737 023424 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
3997 012356 004737 023354 JSR PC,INCSKI ;INCREMENT 'SKI' OR 'OCYL' ERROR COUNT
3998 012362 004737 022110 JSR PC,LINE7A ;PRINT LINE 7A OF ERROR MESSAGE
3999 012366 000207 RTS PC
4000
4001 ;REPORT WRITE CLOCK FAILURE ('WCF')
4002
4003 012370 004737 020200 WCFER: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
4004 012374 104414 046414 DISPLY EM34 ;REPORT WRITE CLOCK FAILURE
4005 012400 004737 020244 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
4006 012404 004737 020660 JSR PC,LINE3A ;PRINT LINE 3A OF ERROR MESSAGE
4007 012410 004737 021326 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
4008 012414 004737 023424 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
4009 012420 004737 015042 JSR PC,PRTBAD ;SEE IF BAD SECTOR TO BE PRINTED
4010 012424 012737 000003 001252 MOV #3,RETRY ;RETRY COUNT
4011 012432 012737 000040 001250 MOV #BIT05,MASK ;ERROR MASK
4012 012440 004737 015400 JSR PC,$RETRY ;RETRY THE ORDER
4013 012444 000405 BR 2$ ;RETURN HERE IF RETRY UNSUCCESSFUL
4014 012446 004737 021700 JSR PC,LINE6C ;PRINT LINE 6C OF ERROR MESSAGE
4015 012452 004737 021762 1$: JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
4016 012456 000207 RTS PC
4017 012460 004737 021706 2$: JSR PC,LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
4018 012464 000772 BR 1$
4019
4020 ;PROCESS DRIVE UNSAFE ERROR
4021

```

```

4022 012466 004737 020200 UNSAF: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
4023 012472 104414 047435 DISPLY EM60 ;REPORT DRIVE UNSAFE
4024 012476 004737 020244 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
4025 012502 004737 020652 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
4026 012506 004737 023424 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
4027 012512 012737 000003 001252 MOV #3,RETRY ;RETRY COUNT
4028 012520 004737 015400 JSR PC,$RETRY ;RETRY THE ORDER
4029 012524 000403 BR 1$ ;RETRY WAS UNSUCCESSFUL
4030 012526 004737 021700 JSR PC,LINE6C ;PRINT LINE 6C OF ERROR MESSAGE
4031 012532 000402 BR 2$ ;CONTINUE WITH ERROR REPORT
4032 012534 004737 021706 1$: JSR PC,LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
4033 012540 004737 021762 2$: JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
4034 012544 000207 RTS PC ;RETURN
4035
4036 ;REPORT AN 'UNKNOWN' DATA PATTERN
4037
4038 012546 105737 001300 NOMTCH: TSTB FRSTER ;FIRST ERROR IN THE SECTOR ?
4039 012552 001013 BNE 1$ ;BR IF NOT OR IF PROCESSING 'DCKER'
4040 012554 004737 020200 JSR PC,LINE1 ;TYPE LINE 1 OF ERROR MESSAGE
4041 012560 104414 047012 DISPLY EM43 ;'CAN'T MATCH DATA WITH PATTERN'
4042 012564 004737 020244 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
4043 012570 004737 020660 JSR PC,LINE3A ;PRINT LINE 3A OF ERROR MESSAGE
4044 012574 004737 021326 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
4045 012600 000404 BR 2$ ;CONTINUE PROCESSING ERROR
4046 012602 104414 047012 1$: DISPLY EM43 ;'CAN'T MATCH DATA WITH PATTERN'
4047 012606 104414 001165 DISPLY $CRLF ;CR-LF
4048 012612 104414 051472 2$: DISPLY LIN9I ;HEADER FOR DATA PRINTOUT
4049 012616 010146 MOV R1,-(SP) ;ADDRESS OF WORD 1
4050 012620 004737 022242 JSR PC,LINOC ;TYPE WORD 1
4051 012624 104414 052155 DISPLY LINS ;SPACES
4052 012630 012146 MOV (R1)+,-(SP) ;ADDRESS OF WORD 1
4053 012632 004737 022242 JSR PC,LINOC ;TYPE WORD 1
4054 012636 104414 001165 DISPLY $CRLF ;CR-LF
4055 012642 010146 MOV R1,-(SP) ;ADDRESS OF WORD 2
4056 012644 004737 022242 JSR PC,LINOC ;TYPE WORD 2
4057 012650 104414 052155 DISPLY LINS ;SPACES
4058 012654 012146 MOV (R1)+,-(SP) ;ADDRESS OF WORD 2
4059 012656 004737 022242 JSR PC,LINOC ;TYPE WORD 2
4060 012662 104414 001165 DISPLY $CRLF ;CR-LF
4061 012666 010146 MOV R1,-(SP) ;ADDRESS OF WORD 3
4062 012670 004737 022242 JSR PC,LINOC ;TYPE WORD 3
4063 012674 104414 052155 DISPLY LINS ;SPACES
4064 012700 012146 MOV (R1)+,-(SP) ;ADDRESS OF WORD 3
4065 012702 004737 022242 JSR PC,LINOC ;TYPE WORD 3
4066 012706 104414 001165 DISPLY $CRLF ;CR-LF
4067 012712 010146 MOV R1,-(SP) ;ADDRESS OF WORD 4
4068 012714 004737 022242 JSR PC,LINOC ;TYPE WORD 4
4069 012720 104414 052155 DISPLY LINS ;SPACES
4070 012724 012146 MOV (R1)+,-(SP) ;ADDRESS OF WORD 4
4071 012726 004737 022242 JSR PC,LINOC ;TYPE WORD 4
4072 012732 104414 001165 DISPLY $CRLF ;CR-LF
4073 012736 062701 000770 ADD #252.*2.,R1 ;INCREMENT BUFFER POINTER
4074 012742 005002 CLR R2 ;CLEAR 'WORDS TO COMPARE' COUNT IN R2
4075 012744 112737 000001 001300 MOVB #1,FRSTER ;SET 'NOT FIRST ERROR' INDICATOR
4076 012752 112737 177777 001301 MOVB #-1,FRSTER+1 ;SET ERROR FOUND INDICATOR
4077 012760 013737 001414 00131C MOV CMPLMT,LIMIT ;RESET THE COMPARE ERROR TIMEOUT LIMIT

```

```

4078 012766 000207          RTS      PC          ;RETURN
4079
4080          ;CHECK ERROR BITS IN THE RH11 & RPO4/5/6 REGISTERS
4081
4082 012770 032760 060000 000234 CKERR: BIT      #E0000,$RPCS1(RO) ;SEE IF 'TRE' OR 'MCPE' SET
4083 012776 001015          BNE     IS          ;BR IF EITHER SET
4084 013000 032760 177400 000244          BIT      #I77400,$RPCS2(RO) ;SEE IF ERROR BITS IN CS2 SET
4085 013006 001011          BNE     IS          ;BR IF ANY SET
4086 013010 005760 000250          TST     $RPER1(RO) ;ANY BITS SET IN ER1
4087 013014 001006          BNE     IS          ;BR IF ANY SET
4088 013016 005760 000274          TST     $RPER2(RO) ;ANY BITS SET IN ER2 ?
4089 013022 001003          BNE     IS          ;BR IF ANY SET
4090 013024 005760 000276          TST     $RPER3(RO) ;ANY BITS SET IN ER3 ?
4091 013030 001416          BEQ     2$          ;BR IF NONE SET
4092 013032 004737 020200 1$: JSR     PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
4093 013036 104414 047057          DISPLY  EM44 ;ERROR BITS SET, BUT 'SC' OR 'TRE' NOT SET
4094 013042 004737 020244          JSR     PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
4095 013046 004737 020652          JSR     PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
4096 013052 004737 021326          JSR     PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
4097 013056 004737 023424          JSR     PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
4098 013062 004737 021762          JSR     PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
4099 013066 000207 2$: RTS      PC          ;RETURN
4100
4101          ;CHECK BUS ADDRESS REGISTER & WORD COUNT REGISTER
4102
4103 013070 005760 000236 CKBUS: TST     $RPWC(RO) ;CHECK WORD COUNT
4104 013074 001010          BNE     IS          ;BR IF NOT ZERO
4105 013076 016046 000020          MOV     $WRDL(RO),-(SP) ;WORD LENGTH
4106 013102 006316          ASL     (SP) ;CHANGE INTO BYTE CC INT
4107 013104 066016 000006          ADD     $BUF(RO),(SP) ;ADD THE STARTING LOCATION
4108 013110 022660 000240          CMP     (SP)+,$RPA(RO) ;BUFFER ADDRESS PROPER ?
4109 013114 001416          BEQ     2$          ;BR IF OK
4110 013116 004737 020200 1$: JSR     PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
4111 013122 104414 046665          DISPLY  EM41 ;BUS ADDRESS OR WORD COUNT INCORRECT
4112 013126 004737 020244          JSR     PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
4113 013132 004737 020710          JSR     PC,LINE3D ;PRINT LINE 3D OF ERROR MESSAGE
4114 013136 004737 021326          JSR     PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
4115 013142 004737 023424          JSR     PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
4116 013146 004737 021762          JSR     PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
4117 013152 000207 2$: RTS      PC          ;RETURN
4118
4119          ;COMPARE THE BUFFER
4120
4121 013154 132760 000004 000024 CMPAR: BITB   #BIT02,$CODE(RO) ;SEE IF READ ORDER
4122 013162 001001          BNE     IS          ;BR IF IT IS
4123 013164 000207          FTS     PC          ;RETURN
4124 013166 005037 001300 1$: CLR     FASTER ;CLEAR 'FIRST ERROR' INDICATOR
4125 013172 032777 000002 165740 CMPARD: BIT   #SWC!,2SWR ;IS SWITCH 1 SET?
4126 013200 001401          BEQ     IS          ;BR IF NOT
4127 013202 000207          RTS     PC          ;YES, DON'T COMPARE
4128 013204 005037 001306 1$: CLR     ERCTR ;CLEAR THE ERROR COUNTER
4129 013210 016001 000006          MOV     $BUF(RO),R1 ;BUFFER ADDRESS
4130 013214 016037 000020 001312          MOV     $WRDL(RO),CMCNT ;WORD COUNT TO WORKING LOCATION
4131 013222 066037 000236 001312          ADD     $RPWC(RO),CMCNT ;CALCULATE ACTUAL WORDS TRANSFERED
4132 013230 016037 000012 001314          MOV     $CYL(RO),CMCYL ;CYLINDER ADDRESS WORKING LOCATION
4133 013236 052737 010000 001314          BIS     #BIT12,CMCYL ;SET FORMAT BIT

```

4134	013244	016037	000010	001316		MOV	\$\$SEC(R0),CMSEC	:SECTOR & TRACK ADDRESSES TO WORKING LOCNS
4135	013252	013737	001414	001310		MOV	CMPLMT,LIMIT	:DISPLAY LIMIT
4136	013260	005237	001310			INC	LIMIT	:CONVERT PARAMETER INTO LIMIT VALUE
4137	013264	012737	177777	001276	CMSTR:	MOV	#-1,ZROIND	:CLEAR THE 'ZERO'S' INDICATOR
4138	013272	005037	001302			CLR	SAVER1	:CLEAR THE R1 SAVE WORD
4139	013276	005037	001304			CLR	SAVER5	:CLEAR THE R5 SAVE WORD
4140	013302	023760	001312	000022		CMP	CMCNT,\$\$SEC(R0)	:IS BUFFER SIZE GREATER THAN ONE SECTOR ?
4141	013310	101003				BHI	1\$:BR IF IT IS
4142	013312	013702	001312			MOV	CMCNT,R2	:LESS THAN, USE REMAINING BUFFER
4143	013316	000402				BR	2\$	
4144	013320	016002	000022		1\$:	MOV	\$\$SEC(R0),R2	:COMPARE SECTOR
4145	013324	166037	000022	001312	2\$:	SUB	\$\$SEC(R0),CMCNT	:DECREMENT WORD COUNT
4146	013332	126027	000024	000005		CMPB	\$\$CODE(R0),#5	:READ HEADER & DATA?
4147	013340	001036				BNE	CMDAT	:BR IF NOT
4148	013342	023721	001314		CMHED:	CMP	CMCYL,(R1)+	:CHECK CYLINDER
4149	013346	001402				BEQ	1\$:BR IF COMPARE OK
4150	013350	004737	013424			JSR	PC,5\$:REPORT ERROR
4151	013354	023721	001316		1\$:	CMP	CMSEC,(R1)+	:COMPARE SECTOR & TRACK
4152	013360	001402				BEQ	2\$:BR IF EQ
4153	013362	004737	013424			JSR	PC,5\$:REPORT ERROR
4154	013366	005721			2\$:	TST	(R1)+	:1ST KEY WORD ZERO?
4155	013370	001402				BEQ	3\$:BR IF IT IS
4156	013372	004737	013424			JSR	PC,5\$:REPORT ERROR
4157	013376	005721			3\$:	TST	(R1)+	:CHECK 2ND KEY WORD
4158	013400	001402				BEQ	4\$:BR IF ZERO
4159	013402	004737	013424			JSR	PC,5\$:REPORT THE ERROR
4160	013406	162702	000004		4\$:	SUB	#4,R2	:SUBTRACT HEADER LENGTH FROM SIZE
4161	013412	003530				BLE	CMPRX	:BR IF FINISHED
4162	013414	022702	000004			CMP	#4,R2	:SEE IF AT LEAST 4 MORE WORDS TO CHECK
4163	013420	101125				BHI	CMPRX	:BR IF NOT
4164	013422	000405				BR	CMDAT	:COMPARE THE DATA PORTION
4165	013424	005237	001306		5\$:	INC	ERCTR	:INCREMENT THE ERROR COUNT
4166	013430	004737	013702			JSR	PC,CMPT	:REPORT THE COMPARISON ERROR
4167	013434	000207				RTS	PC	:CHECK THE REST OF THE HEADER
4168	013436	004737	014224		CMDAT:	JSR	PC,MATCH	:FIND THE PATTERN
4169	013442	000403				BR	2\$:FOUND A PATTERN
4170	013444	004737	012546			JSR	PC,NOMTCH	:RETURN HERE IF NO MATCH WITH PATTERN MADE
4171	013450	000456				BR	8\$:BYPASS COMPARE ROUTINE
4172	013452	011405			2\$:	MOV	(R4),R5	:ADDRESS OF PATTERN ADDRESS IN R4
4173	013454	012703	000020			MOV	#20,R3	:R3 IS PATTERN POS COUNTER
4174	013460	022125			3\$:	CMP	(R1)+,(R5)+	:COMPARE BUFFER WITH PATTERN
4175	013462	001016				BNE	5\$:BR IF NOT EQUAL
4176	013464	005737	001306			TST	ERCTR	:ERRORS DETECTED ?
4177	013470	001406				BEQ	4\$:BR IF NO ERRORS
4178	013472	032777	000010	165440		BIT	#SW3,2SWP	:SWITCH 3 SET ?
4179	013500	001402				BEQ	4\$:BR IF NOT SET
4180	013502	004737	013702			JSR	PC,CMPT	:DISPLAY THE WORD
4181	013506	005302			4\$:	DEC	R2	:DECREMENT SIZE COUNT
4182	013510	001436				BEQ	8\$:BR WHEN AT END
4183	013512	005303				DEC	R3	:DECREMENT PATT POS COUNT
4184	013514	001361				BNE	3\$:BR IF NOT AT END OF PATT
4185	013516	000755				BR	2\$:RESTART THE PATTERN
4186	013520	005761	177776		5\$:	TST	-2(R1)	:IS MISCOMPARED CHARACTER=0
4187	013524	001410				BEQ	6\$:BR IF YES
4188	013526	012737	177777	001276		MOV	#-1,ZROIND	:SET NON-ZERO MISCOMPARED INDICATOR
4189	013534	005237	001306			INC	ERCTR	:INCREMENT THE ERROR COUNTER

4190	013540	004737	013702		JSR	PC,CMPRT		:REPORT ERROR
4191	013544	000760			BR	4\$:CONTINUE COMPARE
4192	013546	105737	001300	6\$:	TSTB	FRSTER		:FIRST ERROR?
4193	013552	100407			BMI	7\$:BR IF NOT
4194	013554	005037	001276		CLR	ZROIND		:SET THE ZERO INDICATOR
4195	013560	010137	001302		MOV	R1,SAVER1		:SAVE CURRENT R1
4196	013564	010537	001304		MOV	R5,SAVER5		:SAVE CURRENT R5
4197	013570	000746			BR	4\$:CONTINUE COMPARE
4198	013572	005737	001276	7\$:	TST	ZROIND		:ANY MISCOMPARISONS NOT ZEROS ?
4199	013576	001743			BEG	4\$:BR IF NONE-ALL ERRORS=ZERO
4200	013600	004737	013702		JSR	PC,CMPRT		:REPORT ERROR
4201	013604	000740			BR	4\$:CONTINUE COMPARING
4202	013606	005737	001312	8\$:	TST	CMCNT		:AT END OF BUFFER
4203	013612	003430			BLE	CMPRX		:BR IF AT END
4204	013614	126027	000024	000005	CMPB	\$CODE(RO),#5		:SEE IF READ HEADER & DATA
4205	013622	001220			BNE	CMSTR		:BR IF NOT
4206	013624	105237	001316		INCB	CMSEC		:INCREMENT SECTOR
4207	013630	123727	001316	000026	CMPB	CMSEC,#22.		:SECTOR GREATER THAN MAX ?
4208	013636	103612			BLO	CMSTR		:BR IF NOT GREATER THAN MAX
4209	013640	105037	001316		CLRB	CMSEC		:CLEAR SECTOR ADDRESS
4210	013644	105237	001317		INCB	CMTRK		:INCREMENT TRACK
4211	013650	123727	001317	000023	CMPB	CMTRK,#19.		:TRACK GREATER THAN MAX ?
4212	013656	103602			BLO	CMSTR		:BR IF NOT GREATER
4213	013660	105037	001317		CLRB	CMTRK		:RESET TRACK ADDRESS
4214	013664	005237	001314		INC	CMCYL		:INCREMENT CYLINDER ADDRESS
4215	013670	000137	013264		JMP	CMSTR		:CONTINUE WITH COMPARE
4216	013674	004737	014156		JSR	PC,ENOCMP		:PRINT LAST LINE IF ERRORS
4217	013700	000207			RTS	PC		
4218								
4219								
4220								
4221	013702	005737	001302		CMPRT:	TST	SAVER1	:PRINT SAVED VALUES ?
4222	013706	001010				BNE	2\$:BR IF NOT
4223	013710	105737	001300			TSTB	FRSTER	:FIRST ERROR?
4224	013714	100402				BMI	1\$:BR IF NOT
4225	013716	004737	013776			JSR	PC,4\$:PRINT INITIAL MESSAGE INFO
4226	013722	004737	014060	1\$:		JSR	PC,8\$:PRINT REMAINDER OF MESSAGE
4227	013726	000422				BR	3\$:EXIT
4228	013730			2\$:				
4229	013730	010146				MOV	R1,-(SP)	:PUSH R1 ON STACK
4230	013732	010546				MOV	R5,-(SP)	:PUSH R5 ON STACK
4231	013734	013701	001302			MOV	SAVER1,R1	:DISPLAY SAVED R1
4232	013740	013705	001304			MOV	SAVER5,R5	:DISPLAY SAVED R5
4233	013744	004737	013776			JSR	PC,4\$:PRINT INITIAL MESSAGE INFO
4234	013750	004737	014060			JSR	PC,8\$:PRINT SAVED VALUES
4235	013754	005037	001302			CLR	SAVER1	:CLEAR SAVED REGISTER INDICATORS
4236	013760	005037	001304			CLR	SAVER5	:CLEAR THE OTHER ONE
4237	013764	012605				MOV	(SP)+,R5	:POP STACK INTO R5
4238	013766	012601				MOV	(SP)+,R1	:POP STACK INTO R1
4239	013770	004737	014060			JSR	PC,8\$:PRINT REMAINDER OF MESSAGE
4240	013774	000207		3\$:		RTS	PC	:RETURN
4241	013776	105737	001300	4\$:		TSTB	FRSTER	:FIRST ERROR ?
4242	014002	100425				BMI	7\$:BR IF NOT
4243	014004	001013				BNE	5\$:BR IF FIRST ERROR AND PROCESSING 'DCK' ERROR
4244	014006	004737	020200			JSR	PC,LINE1	:PRINT LINE 1 OF ERROR MESSAGE
4245	014012	104414	046731			DISPLY	.EM42	:DATA COMPARE ERROR

```

4246 014016 004737 020244 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
4247 014022 004737 020660 JSR PC,LINE3A ;PRINT LINE 3A OF ERROR MESSAGE
4248 014026 004737 021326 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
4249 014032 000404 BR 6$ ;GO TO TYPE HEADER
4250 014034 104414 051365 5$: DISPLY ,LIN9B ;HEADER MESSAGE OF PROCESSING 'DCK' ERROR
4251 014040 104414 001165 DISPLY ,$CRLF ;CR-LF
4252 014044 104414 051414 6$: DISPLY ,LIN9H ;DISPLAY HEADER
4253 014050 012737 177777 001300 MOV #1,FRSTER ;SET ERROR FLAG
4254 014056 000207 PC ;RETURN
4255 014060 005737 001310 7$: RTS ;TYPEOUT LIMIT REACHED ?
4256 014064 001403 BEQ 9$ ;BR IF IT HAS
4257 014066 005337 001310 DEC LIMIT ;DECREMENT LIMIT COUNTER
4258 014072 001005 BNE 10$ ;BR IF NOT AT LIMIT
4259 014074 032777 000200 165036 9$: BIT #SW07, $SWP ;PRINT ALL DATA COMPARE ERRORS ?
4260 014102 001001 BNE 10$ ;BR IF YES
4261 014104 000207 RTS PC ;RETURN
4262 014106 010146 10$: MOV R1, -(SP) ;BUFFER ADDRESS
4263 014110 162716 000002 SUB #2, (SP) ;ADJUST ADDRESS
4264 014114 004737 022242 JSR PC,LIN0CT ;TYPE IT
4265 014120 104414 052155 DISPLY ,LINSF ;2 SPACES
4266 014124 016546 177776 MOV -2(R5), -(SP) ;PUT GOOD DATA ON THE STACK
4267 014130 004737 022242 JSR PC,LIN0CT ;TYPE IT
4268 014134 104414 052155 DISPLY ,LINSF ;2 SPACES
4269 014140 016146 177776 MOV -2(R1), -(SP) ;BAD DATA
4270 014144 004737 022242 JSR PC,LIN0CT ;TYPE IT
4271 014150 104414 001165 DISPLY , $CRLF ;CR-LF
4272 014154 000207 RTS PC ;RETURN
4273
4274 ;LAST LINE OF COMPARE ERROR REPORTING
4275
4276 014156 105737 001301 ENDCMP: TSTB FRSTER+1 ;ANY COMPARE ERRORS FOUND ?
4277 014162 001417 BEQ 2$ ;BR IF NOT
4278 014164 005737 001306 TST ERCTR ;SEE HOW MANY ERRORS
4279 014170 001410 BEQ 1$ ;BR IF ONLY CAN'T MATCH PATTERN
4280 014172 104414 051512 DISPLY ,LINE ;'NUMBER OF ERRORS='
4281 014176 013746 001306 MOV ERCTR, -(SP) ;NUMBER OF ERRORS
4282 014202 004737 022274 JSR PC,LINDEC ;TYPE IT
4283 014206 104414 001165 DISPLY , $CRLF ;CR-LF
4284 014212 004737 023424 1$: JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
4285 014216 004737 021762 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
4286 014222 000207 2$: RTS PC ;RETURN
4287
4288
4289 ;ROUTINE TO MATCH THE DATA WITH A PATTERN
4290 ;CALL:
4291 ; MOV #BUFFER,R1 ;BUFFER ADDRESS
4292 ; JSR PC,MATCH
4293 ; RETURN1 ;PATTERN ADDRESS IN R4
4294 ; RETURN2 ;COULDN'T MATCH PATTERN
4295
4296 014224 010146 MATCH: MOV R1, -(SP) ;SAVE R1 ON THE STACK
4297 014226 012704 000044 MOV #44,R4 ;PATTERN TABLE INDEX
4298 014232 011601 1$: MOV (SP), R1 ;RELOAD R1
4299 014234 162704 000002 SUB #2,R4 ;DECREMENT INDEX
4300 014240 016405 002762 MOV STNDAT(R4),R5 ;ADDRESS OF PATTERN ADDRESS
4301 014244 001411 BEQ 3$ ;BR IF ALL PATTERNS CHECKED AND NO MATCH

```



```

4302                                     : FOUND
4303 014246 012703 000004                2$: MOV    #4,R3      : NUMBER OF LOCATIONS TO CHECK
4304 014252 022125                      : CMP    (R1)+,(R5)+ : COMPARE THE BUFFER AGAINST THE PATTERN
4305 014254 001366                      : BNE   R3          : BR IF NOT EQUAL TRY NEXT PATTERN
4306 014256 005303                      : DEC   R3         : FINISHED CHECKING?
4307 014260 001374                      : BNE   R3         : BR IF NOT FINISHED
4308 014262 062704 002762                : ADD   #STNDAT,R4 : MAKE PATTERN ADDRESS ABSOLUTE
4309 014266 000403                      : BR    R4         : EXIT
4310 014270 062766 000002 000002        3$: ADD   #2(SP)      : INCREMENT RETURN ADDRESS
4311 014276 012601                      4$: MOV   (SP)+,R1   : RESTORE R1
4312 014300 000207                      : RTS   PC        : RETURN
4313
4314                                     ;USE ECC TO CORRECT THE DATA ERROR
4315
4316 014302 016037 000240 001322        ECC: MOV   $RPBA(RO),ECSEC : ADDRESS OF LAST LOCM XFERED
4317 014310 016046 000236                : MOV   $RPWC(RO),-(SP) : ACT WORDS XFERED (2'S COMP)
4318 014314 066016 000020                : ADD   $WRDL(RO),(SP)  : ADD WORDS REQUESTED
4319 014320 005046                      : CLR   -(SP)         : CLEAR NEXT STACK LOCM
4320 014322 016046 000022                : MOV   $SSEC(RO),-(SP) : SECTOR SIZE
4321 014326 004737 027054                : JSR   PC,LINKDV     : DIVIDE WORDS XFERED BY SECTOR SIZE
4322 014332 005716                      : TST   (SP)         : PARTIAL SECTOR XFERED ?
4323 014334 001413                      : BEQ   R1          : BR IF NOT
4324 014336 006316                      : ASL   (SP)         : CONVERT INTO NUMBER OF BYTES
4325 014340 161637 001322                : SUB   (SP),ECSEC    : SUBTRACT SECTOR RESIDUE
4326 014344 126027 000024 000005        : CMPB  $CODE(RO),#5  : WAS OP READ HEAD & DATA
4327 014352 001007                      : BNE   R3          : BR IF NOT
4328 014354 062737 000010 001322        : ADD   #B.,ECSEC     : ADD HEADER SIZE (IN BYTES) BACK IN
4329 014362 000403                      : BR    R3          : GO ADJUST THE STACK POINTER
4330 014364 162737 001000 001322        1$: SUB   #1000,ECSEC  : SUBTRACT SECTOR DATA FIELD SIZE (IN BYTES)
4331 014372 062706 000004                2$: ADD   #4,SP       : ADJUST THE STACK POINTER
4332 014376 016037 000300 001320        : MOV   $RPEC1(RO),ECBIT : ECC POSITION COUNT
4333 014404 005337 001320                : DEC   ECBIT        : ADJUST THE POSITION COUNT
4334 014410 013737 001320 001330        : MOV   ECBIT,ECWRD   : LOAD THE WORD COUNT LOCATION
4335 014416 042737 177760 001320        : BIC   #17,ECBIT     : SAVE THE BIT OFFSET COUNT
4336 014424 042737 000017 001330        : BIC   #17,ECWRD    : CLEAR THE BIT OFFSET
4337 014432 006237 001330                : ASR   ECWRD        : CHANGE TO BYTE COUNT
4338 014436 006237 001330                : ASR   ECWRD        : CHANGE TO BYTE COUNT
4339 014442 006237 001330                : ASR   ECWRD        : CHANGE TO BYTE COUNT
4340 014446 104414 051571                : DISPLY LIN10A      : 'ERROR BURST BEGINS AT '
4341 014452 013746 001330                : MOV   ECWRD,-(SP)  : PUT THE WORD COUNT ON THE STACK
4342 014456 006216                      : ASR   (SP)         : CONVERT TO WORD COUNT FOR MESSAGE
4343 014460 004737 027770                : JSR   PC,$SB2D     : CONVERT THE WORD COUNT
4344 014464 004737 027370                : JSR   PC,$SUPRS    : PRINT IT
4345 014470 104414 051625                : DISPLY LIN10B      : ' IN DATA FIELD OF ERROR SECTOR'
4346 014474 063737 001322 001330        : ADD   ECSEC,ECWRD  : FIND THE BEGINNING OF THE ERROR BURST
4347 014502 026037 000240 001330        : CMP   $RPBA(RO),ECWRD : SEE IF BURST WAS IN DATA READ
4348 014510 101002                      : BHI   #+6         : BR IF IN DATA READ
4349 014512 000137 015030                : JMP   ECC2         : NOT IN DATA READ - REPORT IT
4350 014516 016037 000302 001324        : MOV   $RPEC2(RO),ECMSK0 : GET THE ERROR MASK
4351 014524 005037 001326                : CLR   ECMSK1       : CLEAR THE UPPER MASK WORD
4352 014530 005737 001320                3$: TST   ECBIT     : BIT OFFSET EQUAL ZERO
4353 014534 001407                      : BEQ   R4          : BR IF IT IS
4354 014536 005337 001320                : DEC   ECBIT       : DECREMENT THE BIT OFFSET COUNT
4355 014542 006337 001324                : ASL   ECMSK0      : SHIFT THE ERROR MASK
4356 014546 006137 001326                : ROL   ECMSK1      : SHIFT THE LOWER INTO THE UPPER
4357 014552 000766                      : BR    R3          : CONTINUE THE SHIFT

```

```

4358 014554 017737 164550 001334 4$: MOV @ECWRD, ECBAD0 ;SAVE THE INCORRECT WORD
4359 014562 005037 001336 CLR ECWRD1 ;CLEAR SECOND INCORRECT WORD ADDRESS
4360 014566 013746 001324 MOV ECMSK0, -(SP) ;PUT LOWER MASK ON STACK
4361 014572 047716 164532 BIC @ECWRD, (SP) ;CLEAR ERRONEOUS ONE BITS FROM MASK
4362 014576 043777 001324 164524 BIC ECMSK0, @ECWRD ;CLEAR ERRONEOUS ONE BITS FROM BAD WORD
4363 014604 052677 164520 BIS (SP)+, @ECWRD ;SET DROPPED BITS
4364 014610 005737 001326 TST ECMSK1 ;DOES BURST GO INTO NEXT WORD ?
4365 014614 001431 BEQ ECC1 ;BR IF BURST ONLY IN ONE WORD
4366 014616 013737 001330 001336 MOV ECWRD, ECWRD1 ;DUPLICATE ADDRESS
4367 014624 062737 000002 001336 ADD #2, ECWRD1 ;INCREMENT ERROR ADDRESS
4368 014632 026037 000240 001336 CMP $RPBA(RO), ECWRD1 ;IS NEXT WORD IN THE BUFFER
4369 014640 101003 BHI $S ;BR IF IT IS
4370 014642 005037 001336 CLR ECWRD1 ;CLEAR 2ND WORD ADDRESS
4371 014646 000414 BR ECC1 ;PRINT WORD CORRECTED
4372 014650 017737 164462 001342 5$: MOV @ECWRD1, ECBAD1 ;SAVE THE SECOND BAD WORD
4373 014656 013746 001326 MOV ECMSK1, -(SP) ;PUT THE UPPER MASK ON THE STACK
4374 014662 047716 164450 BIC @ECWRD1, (SP) ;CLEAR ERRONEOUS ONE BITS FROM UPPER MASK
4375 014666 043777 001326 164442 BIC ECMSK1, @ECWRD1 ;CLEAR ERRONEOUS ONE BITS FROM DATA WORD
4376 014674 052677 164436 BIS (SP)+, @ECWRD1 ;SET DROPPED BITS
4377 014700 104414 051773 ECC1: DISPLY .LIN10H ;HEADER
4378 014704 013746 001330 MOV ECWRD, -(SP) ;PUT ECWRD ON THE STACK
4379 014710 004737 022242 JSR PC, LINOCT ;TYPE ECWRD
4380 014714 104414 052155 DISPLY .LINSF ;SPACES
4381 014720 013746 001334 MOV ECBAD0, -(SP) ;PUT ECBAD0 ON THE STACK
4382 014724 004737 022242 JSR PC, LINOCT ;TYPE ECBAD0
4383 014730 104414 052155 DISPLY .LINSF ;SPACES
4384 014734 017746 164370 MOV @ECWRD, -(SP) ;PUT @ECWRD ON THE STACK
4385 014740 004737 022242 JSR PC, LINOCT ;TYPE @ECWRD
4386 014744 104414 052155 DISPLY .LINSF ;SPACES
4387 014750 005737 001336 TST ECWRD1 ;PRINT THE NEXT WORD ?
4388 014754 001427 BEQ ECCX ;BR IF NOT
4389 014756 104414 001165 DISPLY $CRLF ;CR-LF
4390 014762 013746 001336 MOV ECWRD1, -(SP) ;PUT ECWRD1 ON THE STACK
4391 014766 004737 022242 JSR PC, LINOCT ;TYPE ECWRD1
4392 014772 104414 052155 DISPLY .LINSF ;SPACES
4393 014776 013746 001342 MOV ECBAD1, -(SP) ;PUT ECBAD1 ON THE STACK
4394 015002 004737 022242 JSR PC, LINOCT ;TYPE ECBAD1
4395 015006 104414 052155 DISPLY .LINSF ;SPACES
4396 015012 017746 164320 MOV @ECWRD1, -(SP) ;PUT @ECWRD1 ON THE STACK
4397 015016 004737 022242 JSR PC, LINOCT ;TYPE @ECWRD1
4398 015022 104414 052155 DISPLY .LINSF ;SPACES
4399 015026 000402 BR ECCX ;EXIT
4400 015030 104414 051666 ECC2: DISPLY .LIN10C ;ERROR BURST WAS NOT TRANSFERED TO MEMORY
4401 015034 104414 001165 ECCX: DISPLY $CRLF ;CR-LF
4402 015040 000207 RTS ;RETURN
4403
4404 ;ROUTINE TO DISPLAY THE SECTOR WHICH GAVE THE HARD ERROR
4405
4406 015042 032777 000010 164070 PRTBAD: BIT #SW3, @SWR ;PRINT THE BAD SECTOR ^
4407 015050 001460 BEQ $S ;BR IF NOT
4408 015052 016001 000240 MOV $RPBA(RO), R1 ;PUT THE END ADDRESS INTO R1
4409 015056 016046 000020 MOV $WRDL(RO), -(SP) ;FIND THE BEGINNING OF THE SECTOR
4410 015062 066016 000236 ADD $RPWC(RO), (SP) ;SUBTRACT THE WORDS NOT TRANSFERED
4411 015066 005046 CLR -(SP) ;MAKE THE UPPER DIVIDEND 0
4412 015070 016046 000022 MOV $SSEC(RO), -(SP) ;DIVIDE THE WORDS TRANSFERED BY THE SECTOR SIZE
4413 015074 004737 027054 JSR PC, LINKDV ;DIVIDE

```

```

4414 015100 005716          TST      (SP)          ; REMANDER = 0 ?
4415 015102 001403          BEQ      1$           ; BR IF IT IS - COMPLETE SECTOR TRANSFERED
4416 015104 006316          ASL      (SP)          ; CONVERT THE RESIDUAL SECTOR SIZE INTO BYTE COUNT
4417 015106 161601          SUB      (SP),R1      ; SUBTRACT IT FROM THE END ADDRESS
4418 015110 000410          BR       2$           ; FINISH THE SIZING
4419 015112 162701 001000        SUB      #1000,R1     ; SUBTRACT FULL SECTOR SIZE FROM END ADDR
4420 015116 126027 000024 000005 1$:  CMPB    $CODE(RO),#5 ; WAS OPERATION READ HEADER & DATA ?
4421 015124 001002          BNE      2$           ; BR IF NOT
4422 015126 162701 000C10        SUB      #10,R1      ; SUBTRACT HEADER SIZE FROM ADDR
4423 015132 062706 000004        ADD      #4,SP       ; RESTORE THE STACK POINTER
4424 015136 104414 052060        DISPLY  ,LIN11H      ; PRINT THE HEADER
4425 015142 012702 000007        MOV      #7,R2       ; R2 CONTAINS THE WORDS/LINE COUNT
4426 015146 010146          MOV      R1,-(SP)    ; PUT THE ADDRESS ON THE STACK
4427 015150 004737 022242        JSR     PC,LINOC     ; TYPE THE ADDRESS
4428 015154 020160 000240        CMP     R1,$RPBA(RO) ; PRINTED ALL THE SECTOR ?
4429 015160 001412          BEQ      5$           ; BR IF ALL PRINTED
4430 015162 104414 052155        DISPLY  ,LINSR      ; SPACES
4431 015166 012146          MOV      (R1)+,-(SP) ; PUT THE DATA ON THE STACK
4432 015170 004737 022242        JSR     PC,LINOC     ; TYPE THE DATA
4433 015174 005302          DEC     R2           ; DECREMENT THE HORIZONTAL COUNT
4434 015176 001366          BNE     4$           ; BR IF NOT AT THE END OF THE LINE
4435 015200 104414 001165        DISPLY  $CRLF       ; CR-LF
4436 015204 000756          BR       3$           ; RESTORE THE WORDS/LINE COUNT
4437 015206 104414 001165        5$:    DISPLY  $CRLF  ; PRINT WHAT REMAINS IN THE BUFFER
4438 015212 000207        6$:    RTS     PC     ; RETURN
4439
4440          ; ROUTINE TO DO AN RTC - DRIVE SELECTED IN RO
4441          ; CALL:
4442          ;     MOV      #DPB,RO          ; DPB ADDRESS
4443          ;     JSR     PC,R1NCTR
4444          ;     RETURN
4445
4446
4447
4448
4449
4450

```

```

4446 015214 111037 044730
4447 015220 112737 000117 044732 RTNCTR: MOVB (R0),GENDPB ;MOVE THE DRIVE # TO THE GENERAL DPB
4448 015226 004037 034120 1S: MOVB #RTC,GENDPB+$COMND ;COMMAND CODE
4449 015232 044730 JSR RO,RPO4 ;DRIVER ENTRANCE
4450 015234 000774 GENDPB ;DPB ADDRESS FOR ORDER
4451 015236 000207 BR 1S ;DRIVER DIDN'T ACCEPT ORDER
RTS PC ;RETURN

;ROUTINE TO DO A RECALIBRATE - DRIVE SELECTED IN RO
CALL:
MOV #DPB,RO ;DPB ADDRESS
JSR PC,RECALT
RETURN

OR

MOV #DPB,RO ;DPB ADDRESS
MOVB #DRIVE,GENDPB ;DRIVE ADDRESS
JSR PC,RECALTO
RETURN

4466 015240 111037 044730 044732 RECAL: MOVB (R0),GENDPB ;MOVE THE DRIVE # TO THE GENERAL DPB
4467 015244 112737 000107 RECALO: MOVB #RECAL,GENDPB+$COMND ;RECALIBRATE COMMAND
4468 015252 004037 034120 1S: JSR RO,RPO4 ;DRIVER ENTRANCE
4469 015256 044730 GENDPB ;DPB ADDRESS FOR ORDER
4470 015260 000774 BR 1S ;DRIVER DIDN'T ACCEPT THE ORDER
4471 015262 005737 044746 2S: TST GENDPB+$STATUS ;SEE IF FINISHED
4472 015266 001775 BEQ 2S ;BR IF NOT FINISHED
4473 015270 000207 RTS PC ;RETURN

;OFFSET THE DRIVE IN RO (OFFSET CODE PRELOADED INTO 'RPOF')
CALL:
MOVB #OFFSET,GENDPB+$FMT ;OFFSET CODE
MOV #DPB,RO ;DPB ADDRESS
JSR PC,OFFST
RETURN

4482 015272 111037 044730 044732 OFFST: MOVB (R0),GENDPB ;DRIVE # TO GENERAL DPB
4483 015276 112737 000115 MOVB #OFFSET,GENDPB+$COMND ;COMMAND
4484 015304 004037 034120 1S: JSR RO,RPO4 ;DRIVER ENTRANCE
4485 015310 044730 GENDPB ;DPB ADDRESS FOR ORDER
4486 015312 000774 BR 1S ;DRIVER DIDN'T ACCEPT ORDER
4487 015314 000207 RTS PC

;UTILITY READ HEADER ROUTINE
CALL:
MOV #DPB,RO ;DPB ADDRESS
MOV #SECTOR,-(SP) ;SECTOR ADDRESS
MOV #TRACK,-(SP) ;TRACK ADDRESS
JSR PC,READHD
RETURN

4497 015316 116637 000002 044741 READHD: MOVB 2(SP),GENDPB+$TRK ;TRACK ADDRESS
4498 015324 116637 000004 044740 MOVB 4(SP),GENDPB+$SEC ;SECTOR ADDRESS
4499 015332 111037 044730 MOVB (R0),GENDPB ;DRIVE NUMBER
4500 015336 016037 000272 044742 MOV $RPCO(R0),GENDPB+$CYL ;CYLINDER ADDRESS
4501 015344 112737 000173 044732 MOVB #RDHD,GENDPB+$COMND ;COMMAND

```

```

4502 015352 004037 034120 1$: JSR RO,RPO4 ;DRIVER ENTRANCE
4503 015356 044730 GENDPB ;DPB ADDRESS FOR ORDER
4504 015360 000774 BR 1$ ;DRIVER DIDN'T ACCEPT COMMAND
4505 015362 005737 044746 2$: TST GENDPB+$STATUS ;FINISHED?
4506 015366 001775 BEQ 2$ ;BR IF NOT
4507 015370 012666 000002 MOV (SP)+,2(SP) ;ADJUST STACK FOR RETURN
4508 015374 005726 TST (SP)+
4509 015376 000207 RTS PC ;RETURN
4510
4511 ;RETRY THE PRESENT OPERATION
4512 ;CALL:
4513 ;
4514 ; MOV #COUNT_RETRY ;RETRY COUNT
4515 ; JSR PC,$RETRY
4516 ; RETURN1 ;RETRY UNSUCCESSFUL
4517 ; RETURN2 ;SUCCESSFUL RETRY
4518 ; NOTE: IF A DIFFERENT ERROR OCCURS DURING
4519 ; RETRY, THE ROUTINE EXITS TO 'ERPRC1'
4520 015400 004737 016370 $RETRY: JSR PC,GODRIV ;RE-START ORDER
4521 015404 005760 000016 1$: TST $STATUS(RO) ;ORDER FINISHED?
4522 015410 001775 BEQ 1$ ;BR IF NOT
4523 015412 100405 BMI 2$ ;BR IF ERROR
4524 015414 105237 001253 INCB RETRY+1 ;INCREMENT RETRY COUNT
4525 015420 062716 000002 ADD #2,(SP) ;INCREMENT RETURN
4526 015424 000425 BR 5$ ;GO TO EXIT
4527 015426 032760 000200 000016 2$: BIT #BIT7,$STATUS(RO) ;DID ORDER TERMINATE NORMALLY?
4528 015434 001430 BEQ 7$ ;BR IF NOT
4529 015436 005737 001250 TST MASK ;IS ERROR MASK 0?
4530 015442 001004 BNE 3$ ;BR IF NOT
4531 015444 005760 000250 TST $RPER1(RO) ;MAKE SURE THAT THE DRIVE ERROR REG IS CLEAR
4532 015450 001014 BNE 6$ ;BR IF NOT
4533 015452 000404 BR 4$ ;CONTINUE RETRY
4534 015454 033760 001250 000250 3$: BIT MASK,$RPER1(RO) ;SAME ERROR?
4535 015462 001407 BEQ 6$ ;BR IF NOT
4536 015464 105237 001253 4$: INCB RETRY+1 ;INCREMENT RETRY COUNT
4537 015470 123737 001252 001253 CMPB RETRY,RETRY+1 ;DONE?
4538 015476 001340 BNE $RETRY ;BR IF NOT DONE
4539 015500 000207 5$: RTS PC ;RETURN
4540 015502 004737 022230 6$: JSR PC,LINE8 ;REPORT DIFFERENT ERROR
4541 015506 004737 021762 JSR PC,LINE7 ;PRINT LINE?
4542 015512 005726 TST (SP)+ ;ADJUST STACK POINTER FOR DIRECT RETURN
4543 015514 000207 RTS PC ;RETURN
4544 015516 104414 051330 7$: DISPLY ,LINBM ;'DIFFERENT ERROR DURING RETRY'
4545 015522 000137 006644 JMP ERPRC1 ;REPORT THE ERROR
4546
4547 ;ROUTINE TO UPDATE THE PERFORMANCE SUMMARY STATISTICS
4548 ;CALL:
4549 ;
4550 ; MOV #DPB_RO ;DPB ADDRESS
4551 ; JSR PC,STATIS
4552 ; RETURN
4553 015526 032760 000300 000016 STATIS: BIT #BIT07:BIT06,$STATUS(RO) ;CHECK FOR DATA TERMINATION
4554 015534 001454 BEQ 3$ ;BR IF NOT DATA TERMINATION
4555 015536 016037 000240 015670 MOV $RPBA(RO),FACTOR ;STORE THE FINAL BUFFER ADDRESS
4556 015544 166037 000006 015670 SUB $BUF(RO),FACTOR ;SUBTRACT THE INITIAL ADDRESS
4557 015552 001434 BEQ 2$ ;BR IF NO DATA TRANSFER

```

```

4558 015554 006237 015670          ASR    FACTOR          ;CONVERT TO A WORD COUNT
4559 015560 063760 015670 000046  ADD    FACTOR,$TRANS(RO) ;UPDATE WORD COUNT
4560 015566 005560 000050          ADC    $TRANS+2(RO)     ;ADD ANY CARRY
4561 015572 132760 000002 000024  BITB   #BIT01,$CODE(RO) ;SEE IF ORDER READ OR WRITE
4562 015600 001021          BNE    2$              ;BRANCH IF ORDER WRITE
4563 015602 005737 001424          TST    AUTOCK          ;AUTO WRITE CHECKS BEING PERFORMED
4564 015606 001411          BEQ    1$              ;BR IF NOT
4565 015610 126027 000024 000001  CMPB   $CODE(RO),#1    ;PRESENT OPERATION AN AUTOMATIC WRITE CHECK ?
4566 015616 101005          BHI    1$              ;BR IF NOT
4567 015620 066060 000020 000046  ADD    $WRDL(RO),$TRANS(RO) ;ADD WORDS WRITTEN
4568 015626 005560 000050          ADC    $TRANS+2(RO)     ;ADD A CARRY
4569 015632 063760 015670 000052 1$:    ADD    FACTOR,$READ(RO)  ;UPDATE THE READ WORD COUNT
4570 015640 005560 000054          ADC    $READ+2(RO)     ;ADD ANY CARRY
4571 015644 026060 000012 000270 2$:    CMP    $CYL(RO),$RPCA(RO) ;DID MID-TRANSFER SEEK OCCUR
4572 015652 001405          BEQ    3$              ;BR IF NOT
4573 015654 062760 000001 000042  ADD    #1,$POSIT(RO)    ;INCREMENT SEEK COUNT
4574 015662 005560 000044          ADC    $POSIT+2(RO)    ;ADD CARRY TO UPPER WORD
4575 015666 000207          3$:    RTS    PC
4576 015670 000000          FACTOR: .WORD 0          ;USED FOR WORDS TRANSFERED
4577
4578 ;ROUTINE TO GET A BUFFER
4579 ;CALL:
4580 ;
4581 ;
4582 ;
4583 ;
4584 ;
4585 ;
4586 ;
4587 ;
4588 ;
4589 ;
4590 ;
4591 ;
4592 ;
4593 ;
4594 ;
4595 ;
4596 ;
4597 ;
4598 ;
4599 ;
4600 ;
4601 ;
4602 ;
4603 ;
4604 ;
4605 ;
4606 ;
4607 ;
4608 ;
4609 ;
4610 ;
4611 ;
4612 ;
4613 ;

```

```

4614 016012 005302          DEC      R2          ;DECREMENT TABLE COUNT
4615 016014 001374          BNE      S$          ;CONTINUE IF NOT AT END OF TABLE
4616 016016 012603      6$:  MOV      (SP)+,R3      ;RESTORE R3
4617 016020 012602          MOV      (SP)+,R2      ;RESTORE R2
4618 016022 012601          MOV      (SP)+,R1      ;RESTORE R1
4619 016024 000207          RTS       PC          ;RETURN
4620
4621
4622          ;ROUTINE TO PUT BUFFER BACK IN TABLE
4623          ;CALL:
4624          ;      MOV      #DPB,R0          ;DPB ADDRESS
4625          ;      JSR      PC,RELBUF
4626          ;      RETURN
4627
4628          RELBUF: MOV      R1,-(SP)          ;SAVE R1
4629          MOV      #BUFTBL+2,R1          ;BEGINNING OF TABLE
4630          MOV      BUFTBL,R2          ;ENTRY COUNT
4631          BEQ      2$          ;BR IF EMPTY TABLE
4632          MOV      $WRDL(R0),R3          ;TRIAL ADDRESS
4633          ASL      R3          ;CHANGE TO BYTE COUNT
4634          ADD      $BUF(R0),R3          ;ADDRESS OF HIGHER ADJACENT BLOCK
4635          1$:  CMP      (R1),R3          ;UPPER ADJACENT BLOCK
4636          BEQ      4$          ;BR IF YES
4637          ADD      #4,R1          ;INCREMENT POINTER
4638          DEC      R2          ;DECREMENT ENTRY COUNT
4639          BNE      1$          ;CONTINUE SEARCHING
4640          MOV      $BUF(R0),(R1)          ;PUT THE BUFFER BLOCK INTO THE TABLE
4641          MOV      $WRDL(R0),2(R1)          ;BLOCK SIZE
4642          INC      BUFTBL          ;INCREMENT ENTRY COUNT
4643          INC      R2          ;INCREMENT R2 FOR USE LATER
4644          BR      5$          ;SEE IF A LOWER ADJACENT BLOCK IS IN THE TABLE
4645          2$:  MOV      $BUF(R0),(R1)+          ;BLOCK ADDRESS TO TABLE
4646          MOV      $WRDL(R0),(R1)+          ;SIZE TO TABLE
4647          INC      BUFTBL          ;INCREMENT ENTRY COUNT
4648          BR      10$          ;EXIT
4649          4$:  MOV      $BUF(R0),(R1)          ;RELEASED BUFFER IS LOWER ADJACENT
4650          ADD      $WRDL(R0),2(R1)          ;INCREMENTED SIZE
4651          5$:  MOV      R2,-(SP)          ;SAVE R2
4652          MOV      BUFTBL,R2          ;ENTRY COUNT
4653          MOV      #BUFTBL+2,R5          ;BEGINNING OF TABLE
4654          6$:  MOV      2(R5),R4          ;BLOCK SIZE (IN WORDS)
4655          ASL      R4          ;CHANGE TO BYTE COUNT
4656          ADD      (R5),R4          ;ADD BLOCK BEGINNING ADDRESS
4657          CMP      R4,(R1)          ;R1 STILL POINTS TO INSERTED ENTRY
4658          BEQ      8$          ;LOWER ADJACENT IN TABLE
4659          ADD      #4,R5          ;INCREMENT POINTER
4660          DEC      R2          ;DECREMENT ENTRY COUNT
4661          BNE      6$          ;CONTINUE LOOKING
4662          TST      (SP)+          ;RESTORE STACK POINTER
4663          BR      10$          ;END
4664          8$:  MOV      (SP)+,R2          ;RESTORE R2
4665          ADD      2(R1),2(R5)          ;INCREMENT LOWER BLOCK LENGTH
4666          DEC      BUFTBL          ;DECREMENT ENTRY COUNT
4667          MOV      R1,R5          ;GET READY TO COMPRESS
4668          ADD      #4,R5          ;INCREMENT TO NEXT ENTRY
4669          9$:  MOV      (R5)+,(R1)+          ;COMPRESS TABLE

```

```

4670 016230 012521      MOV      (R5)+,(R1)+    ;MOVE SIZE FIELD DOWN
4671 016232 005302      DEC      R2             ;DECREMENT ENTRY COUNT
4672 016234 001374      BNE     9$             ;BR IF NOT FINISHED
4673 016236 012601      10$:    MOV      (SP)+,R1  ;RESTORE R1
4674 016240 00C207      RTS      PC            ;RETURN
4675
4676
4677      ;FILL THE ASSIGNED BUFFER (IF WRITE OR WRITE CHECK ORDER)
4678      ;CALL:
4679      ;
4680      ;      MOV      #DPB,RO      ;DPB ADDRESS
4681      ;      MOV      #BUFADR,$BUF(RO) ;LOAD BUFFER ADDRESS INTO THE DPB
4682      ;      MOV      #PATTERN,$PATT(RO) ;PATTERN CODE
4683      ;      JSR      PC,FILBUF
4684      ;      RETURN
4685 016242 104412      FILBUF: SAVREG      ;SAVE THE REGISTERS
4686 016244 132760 00C004 000024  BITB     #BIT02,$CODE(RO) ;SEE IF READ ORDER
4687 016252 001044      BNE     4$            ;BR IF READ
4688 016254 016001 000006      1$:    MOV      $BUF(RO),R1  ;BUFFER ADDRESS
4689 016260 016002 000020      MOV      $WRDL(RO),R2  ;POSITIVE WORD COUNT
4690 016264 132760 000001 000024  BITB     #BIT00,$CODE(RO) ;SEE IF WRITE HEADER TYPE ORDER
4691 016272 001413      BEQ     2$            ;BR IF NOT
4692 016274 016011 000012      MOV      $CYL(RO),(R1) ;CYLINDER ADDRESS
4693 016300 052721 010000      BIS      #BIT12,(R1)+  ;SET FMT22 BIT
4694 016304 016021 000010      MOV      $SEC(RO),(R1)+ ;MOVE SECTOR & TRACK
4695 016310 005021      CLR     (R1)+         ;CLEAR FIRST KEY WORD
4696 016312 005021      CLR     (R1)+         ;CLEAR THE SECOND
4697 016314 162702 000004      SUB     #4,R2         ;ADJUST THE WORD COUNT
4698 016320 003421      BLE     4$            ;BR IF END OF PATTERN
4699 016322 005004      2$:    CLR     R4          ;CLEAR R4
4700 016324 116004 000030      MOV      $PATT(RO),R4  ;RELATIVE PATTERN ADDRESS
4701 016330 016405 002762      MOV      $INDAT(R4),R5 ;PATTERN ADDRESS
4702 016334 012703 000020      MOV      #20,R3       ;PATTERN COUNT
4703 016340 012521      3$:    MOV      (R5)+,(R1)+  ;MOVE THE PATTERN INTO THE BUFFER
4704 016342 005302      DEC     R2            ;DECREMENT THE WORD COUNT
4705 016344 001407      BEQ     4$            ;BR IF DONE (WORD COUNT = 0)
4706 016346 005303      DEC     R3            ;DECREMENT THE PATTERN COUNT
4707 016350 001373      BNE     3$            ;BR IF MORE PATTERN
4708 016352 012703 000020      MOV      #20,R3       ;RESTORE PATTERN COUNT
4709 016356 016405 002762      MOV      $INDAT(R4),R5 ;RESTORE THE ADDRESS
4710 016362 000766      BR      3$           ;CONTINUE DISTRIBUTING THE PATTERN
4711 016364 104413      4$:    RESREG      ;RESTORE THE REGISTERS
4712 016366 000207      RTS      PC            ;RETURN
4713
4714      ;START THE ORDER FOR THE DPB IN RO
4715      ;CALL:
4716      ;
4717      ;      MOV      #DPB,RO      ;DPB ADDRESS
4718      ;      JSR      PC,GODRIV
4719      ;      RETURN
4720 016370 010046      GODRIV: MOV      RO,-(SP)  ;SAVE RO
4721 016372 010037 016402      MOV      RO,2$        ;CURRENT DPB ADDRESS
4722 016376 004037 034120      1$:    JSR      RO,RPO4  ;CALL THE DRIVE HANDLER
4723 016402 000000      2$:    .WORD     0       ;DRIVE BLOCK ADDRESS GOES HERE
4724 016404 000000      HALT
4725 016406 012600      MOV      (SP)+,RO    ;RESTORE RO

```



```

4726 016410 062760 000001 000036      ADD      #1,$OPERC(RO) ;INCREMENT THE OPERATION COUNT
4727 016416 005560 000040      ADC      $OPERC+2(RO)
4728 016422 026060 000034 000012      CMP      $PREVA+2(RO),$CYL(RO) ;DID ORDER REQUIRE A CYLINDER CHANGE
4729 016430 001405      BEQ      3$ ;BR IF NOT
4730 016432 062760 000001 000042      ADD      #1,$POSIT(RO) ;INCREMENT SEEK COUNT
4731 016440 005560 000044      ADC      $POSIT+2(RO) ;ADD ANY CARRY
4732 016444 000207      3$:      RTS      PC
4733
4734 ;GENERATE PARAMETERS FOR THE OPERATION
4735 ;CALL:
4736 ;
4737 ;      MOV      #DPB,RO ;DPB ADDRESS
4738 ;      JSR      PC,$ELPAR
4739 ;      RETURN
4740
4741 016446 004737 032364      SELPAR: JSR      PC,$RAND ;CYCLE THE RANDOM NUMBER GENERATOR
4742 016452 032777 000001 162460      BIT      #SWO,$SWR ;SEE IF SWO SET
4743 016460 001012      BNE      2$ ;BR IF SET - READ ONLY
4744 016462 012705 000010      1$:      MOV      #10,R5 ;READ/WRITE SELECTION DIVISOR
4745 016466 004737 027026      JSR      PC,$ETREM ;GET SELECTION VALUE
4746 016472 020537 001422      CMP      R5,$RATIO ;DETERMINE IF READ OR WRITE
4747 016476 103003      BHIS    2$ ;BR IF READ
4748 016500 004737 017170      JSR      PC,$ANWRT ;SELECT A WRITE ORDER
4749 016504 000406      BR      3$ ;CONTINUE WITH THE SELECTION
4750 016506 013705 032464      2$:      MOV      $LONUM,R5 ;SELECT READ OPERATION CODE
4751 016512 042705 177776      BIC      #1,R5 ;MASK OUT ALL BUT BIT 0
4752 016516 062705 000004      ADD      #4,R5 ;TABLE OFFSET FOR READ CODE
4753 016522 110560 000074      3$:      MOVB    R5,$NCODE(RO) ;ORDER SELECTION CODE TO CONTROL BLOCK
4754
4755 ;GENERATE A RANDOM SECTOR ADDRESS BETWEEN VALUES 'MINSEC' & 'MAXSEC'
4756 016526 016005 000116      RANSEC: MOV      MAXSEC(RO),R5 ;GET MAXIMUM SECTOR ADDRESS
4757 016532 026005 000120      CMP      MINSEC(RO),R5 ;'MINSEC' AND 'MAXSEC' THE SAME ?
4758 016536 001417      BEQ      2$ ;BR IF THEY ARE
4759 016540 166005 000120      SUB      MINSEC(RO),R5 ;SUBTRACT MINIMUM SECTOR ADDRESS
4760 016544 100002      BPL     1$ ;BR IF MAX LARGER THAN MIN
4761 016546 062705 000026      ADD      #22.,R5 ;CORRECT THE NUMBER
4762 016552 005205      1$:      INC      R5 ;INCREMENT DIFFERENCE TO USE AS DIVISOR
4763 016554 004737 027026      JSR      PC,$ETREM ;GET THE RANDOM AUGMENT
4764 016560 066005 000120      ADC      MINSEC(RO),R5 ;NEW ADDRESS
4765 016564 020527 000025      CMP      R5,#21. ;IS VALUE TOO LARGE ?
4766 016570 101402      BLOS    2$ ;BR IF NOT
4767 016572 162705 000026      SUB      #22.,R5 ;CORRECT VALUE
4768 016576 110560 000076      2$:      MOVB    R5,$NSEC(RO) ;STORE SECTOR ADDRESS IN DPB
4769
4770 ;GENERATE A RANDOM TRACK ADDRESS BETWEEN VALUES 'MINTRK' & 'MAXTRK'
4771
4772 016602 016005 000112      RANTRK: MOV      MAXTRK(RO),R5 ;GET MAXIMUM TRACK ADDRESS
4773 016606 026005 000114      CMP      MINTRK(RO),R5 ;'MINTRK' AND 'MAXTRK' THE SAME ?
4774 016612 001417      BEQ      2$ ;BR IF THEY ARE
4775 016614 166005 000114      SUB      MINTRK(RO),R5 ;SUBTRACT MINIMUM TRACK ADDRESS
4776 016620 100002      BPL     1$ ;BR IF MAX LARGER THAN MIN
4777 016622 062705 000023      ADD      #19.,R5 ;CORRECT THE NUMBER
4778 016626 005205      1$:      INC      R5 ;INCREMENT DIFFERENCE TO USE AS DIVISOR
4779 016630 004737 027026      JSR      PC,$ETREM ;GET THE RANDOM AUGMENT
4780 016634 066005 000114      ADC      MINTRK(RO),R5 ;NEW TRACK ADDRESS
4781 016640 020527 000022      CMP      R5,#18. ;IS VALUE TOO LARGE ?

```

```

4782 016644 101402          BLOS 2$          ;BR IF NOT
4783 016646 162705 000023  SUB  #19,RS      ;CORRECT VALUE
4784 016652 110560 000077 2$: MOV  RS,$NTRK(RO) ;STORE TRACK ADDRESS IN DPB
4785
4786 ;GENERATE A RANDOM CYLINDER ADDRESS BETWEEN VALUES 'MINCYL' & 'MAXCYL'
4787
4788 016656 012737 000633 001350 MOV  #411,CYLIMT ;ASSUME AN RPO4/5
4789 016664 032760 000002 000262 BIT  #BIT01,$RPDT(PO) ;SEE IF RPO6
4790 016672 001403          BEQ  RANCYL      ;BR IF NOT
4791 016674 012737 001457 001350 MOV  #815,CYLIMT ;CHANGE CYLINDER LIMIT
4792 016702 016005 000106 RANCYL: MOV  MAXCYL(RO),RS ;GET MAXIMUM CYLINDER ADDRESS
4793 016706 026005 000110      CMP  MINCYL(RO),RS ;'MINCYL' AND 'MAXCYL' THE SAME ?
4794 016712 001417          BEQ  2$          ;BR IF THEY ARE
4795 016714 166005 000110      SUB  MINCYL(RO),RS ;SUBTRACT MINIMUM CYLINDER ADDRESS
4796 016720 100002          BPL  1$          ;BR IF MAX LARGER THAN MIN
4797 016722 063705 001350      ADD  CYLIMT,RS    ;CORRECT THE NUMBER
4798 016726 005205 1$: INC  RS        ;INCREMENT DIFFERENCE TO USE AS DIVISOR
4799 016730 004737 027026      JSR  PC,GETREM   ;GET THE RANDOM ARGUMENT
4800 016734 066005 000110      ADD  MINCYL(RO),RS ;NEW CYLINDER ADDRESS
4801 016740 023705 001350      CMP  CYLIMT,RS    ;IS VALUE TOO LARGE ?
4802 016744 101002          BHI  2$          ;BR IF NOT
4803 016746 163705 001350      SJB  CYLIMT,RS    ;CORRECT VALUE
4804 016752 010560 000100 2$: MOV  RS,$NCYL(RO) ;STORE CYLINDER ADDRESS IN DPB
4805 016756 122760 000003 000074 CMPB #3,$NCODE(RO) ;WRITE HEADER & DATA ?
4806 016764 001013          BNE  RANSIZ      ;BR IF NOT
4807 016766 012760 000404 000102 MOV  #260,$NWRDL(PO) ;CHANGE WORD LENGTH TO 260 FOR WRTHD ORDER
4808 016774 023727 001404 000404 CMP  MAXDL,#260.  ;CAN A FULL SECTOR BE WRITTEN ?
4809 017002 103062          BHIS RANPAT      ;BR IF IT CAN
4810 017004 013760 001404 000102 MOV  MAXDL,$NWRDL(RO) ;CHANGE TRANSFER SIZE
4811 017012 000456          BR   RANPAT      ;CONTINUE WITH THE SELECTION
4812
4813 ;GENERATE A RANDOM BUFFER LENGTH BETWEEN 4 & THE VALUE IN 'MAXDL'
4814
4815 017014 013705 001404 RANSIZ: MOV  MAXDL,RS ;GET BUFFER SIZE
4816 017020 005737 001420      TST  WSEL        ;SELECT A RANDOM WORD COUNT ?
4817 017024 001010          BNE  1$          ;BR IF NOT
4818 017026 005205          INC  RS          ;INCREMENT THE MAXIMUM SIZE
4819 017030 004737 027026      JSR  PC,GETREM   ;DIVIDE BY MAX VALUE
4820 017034 005705          TST  RS          ;IS THE REMAINDER 0 ?
4821 017036 001003          BNE  1$          ;NOT 0, CONTINUE
4822 017040 004737 032364      JSR  PC,$RAND    ;CYCLE THE RANDOM NUMBER GENERATOR
4823 017044 000763          BR   RANSIZ      ;TRY AGAIN
4824 017046 010560 000102 1$: MOV  RS,$NWRDL(RO) ;WORD LENGTH TO CONTROL BLOCK
4825 017052 010546          MCV  RS,-(SP)    ;NEW WORD LENGTH ON STACK FOR CHECK
4826 017054 005046          CLR  -(SP)       ;MAKE UPPER DIVIDEND ZERO
4827 017056 012746 000400      MOV  #256,-(SP) ;SECTOR SIZE IS THE DIVISOR
4828 017062 132760 000001 000074 BITB #1,$NCODE(RO) ;SEE IF NEXT ORDER IS A HEADER ORDER
4829 017070 001402          BEQ  2$          ;BR IF NOT
4830 017072 062716 000004      ADD  #4,(SP)     ;ADD HEADER SIZE TO SECTOR SIZE
4831 017076 004737 027054 2$: JSR  PC,LINKDV  ;DIVIDE BUFFER SIZE BY SECTOR SIZE
4832 017102 012616          MOV  (SP)+,(SP) ;MOV REMAINDER UP THE STACK
4833 017104 021627 000004      CMP  (SP),#4 ;SEE IF REMAINDER LESS THAN 4
4834 017110 103012          BHIS 4$         ;BR IF NOT
4835 017112 005737 001420      TST  WSEL        ;SELECTING RANDOM TRANSFER SIZES ?
4836 017116 001403          BEQ  3$         ;BR IF YES
4837 017120 161660 000102      SUB  (SP),$NWRDL(RO) ;ADJUST WORD LENGTH DOWNWARD

```

```

4838 017124 000404          BR      4$          ;CONTINUE
4839 017126 005726          3$:   TST      (SP)+ ;CORRECT THE STACK POINTER
4840 017130 004737 032364   JSR      PC,$RAND ;CYCLE THE RANDOM NUMBER GENERATOR
4841 017134 000727          BR      RANSIZ     ;TRY AGAIN
4842 017136 005726          4$:   TST      (SP)+ ;CORRECT THE STACK POINTER
4843 017140 122760 000002 000074  CMPB    #2,$NCODE(RO) ;SEE IF WRITE DATA
4844 017146 001004          BNE     RANXIT     ;BR IF NOT WRITE DATA
4845
4846          ;GET A RANDOM PATTERN NUMBER
4847
4848 017150 004737 017274   RANPAT: JSR      PC,GETPAT ;GET PATTERN CODE
4849 017154 110560 000075   MOV     R5,$NPATC(RO) ;MOVE PATTERN CODE TO CONTROL BLOCK
4850 017160 012760 177777 000104  RANXIT: MOV     #1,$NEX(RO) ;SET PARAMETERS SELECTED INDICATOR
4851 017166 000207          RTS      PC        ;RETURN
4852
4853          ;ROUTINE TO SELECT A WRITE (OR WRITE CHECK) OPERATION
4854
4855 017170 012705 000004   RANWRT: MOV     #4,R5 ;WRITE OPERATION SELECTION DIVISOR
4856 017174 004737 027026   JSR      PC,GETREM ;GET SELECTION CODE
4857 017200 005737 001424   TST     AUTOCK    ;ARE WRITE CHECK ORDERS TO BE SELECTED
4858          ;RANDOMLY ?
4859 017204 001403          BEQ     1$,R5     ;BR IF THEY ARE
4860 017206 152705 000002   BIS     #2,R5     ;SET CODE TO EXCLUDE WRITE CHECK ORDERS
4861 017212 000420          BR      3$       ;COMPLETE SELECTION
4862 017214 020527 000001   1$:   CMP     R5,#1 ;WRITE CHECK SELECTED ?
4863 017220 101015          BHI     3$       ;BR IF NOT
4864 017222 132760 000002 000024  BITB    #2,$CODE(RO) ;PREVIOUS WRITE OPERATION ?
4865 017230 001407          BEQ     2$,R5    ;BR IF PREVIOUS WAS READ OR WRITE CHECK
4866 017232 116060 000024 000074  MOV     $CODE(RO),$NCODE(RO) ;MOVE CODE TO 'NEXT CODE'
4867 017240 142760 000002 000074  BIC     #2,$NCODE(RO) ;CHANGE WRITE TO WRITE CHECK
4868 017246 000411          BR      5$       ;EXIT
4869 017250 052705 000002   2$:   BIS     #2,R5 ;CHANGE WRITE CHECK TO WRITE
4870 017254 005737 001416   3$:   TST     FORMAT ;WRITE HEADER ORDERS ALLOWED ?
4871 017260 001002          BNE     4$       ;BR IF THEY ARE
4872 017262 042705 000001   BIC     #1,R5    ;ALTER POSSIBLE WRITE HEADER
4873 017266 110560 000074   4$:   MOV     R5,$NCODE(RO) ;SETUP 'NEXT' CODE
4874 017272 000207          5$:   RTS      PC        ;RETURN
4875
4876          ;ROUTINE TO SELECT A PATTERN
4877
4878 017274 012705 000020   GETPAT: MOV     #20,R5 ;SELECT PATTERN
4879 017300 004737 027026   JSR      PC,GETREM ;GET CODE
4880 017304 005705          TST     R5       ;WAS PATTERN ZERO SELECTED ?
4881 017306 001003          BNE     1$,R5    ;BR IF NOT ZERO
4882 017310 004737 032364   JSR      PC,$RAND ;CYCLE THE RANDOM NUMBER GENERATOR
4883 017314 000767          BR      GETPAT   ;TRY AGAIN
4884 017316 006305          1$:   ASL     R5     ;MAKE CODE INTO TABLE INDEX
4885 017320 000207          RTS      PC
4886
4887          ;ROUTINE TO GET THE PREVIOUSLY SELECTED PARAMETER VALUES
4888          ;CALL:
4889          ;
4890          ;   MOV     #DPB,RO ;DPB ADDRESS
4891          ;   JSR      PC,SELPAR ;SELECT THE PARAMETERS
4892          ;
4893          ;   JSR      PC,GETPAR
4894          ;
4895          ;   RETURN

```

```

4894 017322 010546          GETPAR: MOV      R5, -(SP)          ;SAVE R5
4895 017324 116060 000234 000027  MOVB   $RPCS1(R0), $PREV0(R0) ;SAVE CURRENT PARAMETERS
4896 017332 032760 000006 000074  BIT    #6, $NCODE(R0)        ;SEE IF NEXT OPERATION IS READ OR WRITE
4897 017340 001007          BNE    IS                    ;BR IF EITHER
4898 017342 016060 000012 000034  MOV    $CYL(R0), $PREVA+2(R0) ;SAVE STARTING CYLINDER
4899 017350 016060 000010 000032  MOV    $SECC(R0), $PREVA(R0)  ;SAVE STARTING SECTOR AND TRACK
4900 017356 000411          BR     2S
4901 017360 004737 022314 1S:    JSR    PC, READR            ;GET THE DECREMENTED SECTOR AND TRACK ADDRESSES
4902 017364 112660 000033  MOVB   (SP)+, $PREVA+1(R0)    ;TRACK ADDRESS
4903 017370 112660 000032  MOVB   (SP)+, $PREVA(R0)     ;SECTOR ADDRESS
4904 017374 016060 000272 000034  MOV    $RPMC(R0), $PREVA+2(R0) ;CURRENT CYLINDER
4905 017402 032777 000100 161530 2S:    BIT    #SW06, $SWR          ;SWITCH 6 SET ?
4906 017410 001043          BNE    3S                    ;BR IF SET
4907 017412 116060 000074 000024  MOVB   $NCODE(R0), $CODE(R0) ;LOGICAL CODE FOR OPERATION
4908 017420 116005 000074  MOVB   $NCODE(R0), R5        ;LOAD R5 FOR USE AS TABLE INDEX
4909 017424 116560 001760 000002  MOVB   COMBL(R5), $COMND(R0)  ;RPO4 COMMAND CODE
4910 017432 116060 000075 000030  MOVB   $NPATC(R0), $PATTC(R0) ;PATTERN CODE
4911 017440 016060 000076 000010  MOV    $NSEC(R0), $SEC(R0)   ;TRACK AND SECTOR ADDRESSES
4912 017446 016060 000100 000012  MOV    $NCYL(R0), $CYL(R0)   ;CYLINDER ADDRESS
4913 017454 016060 000102 000020  MOV    $NWRDL(R0), $WRDL(R0) ;BUFFER SIZE
4914 017462 016060 000102 000004  MOV    $NWRDL(R0), $WRDL(R0) ;WORD COUNT FOR THE RH11
4915 017470 005460 000004  NEG    $WRDL(R0)             ;COMPLEMENT IT
4916 017474 012760 000400 000022  MOV    #256, $SSEC(R0)       ;INITIAL VALUE OF SECTOR SIZE
4917 017502 032760 000001 000024  BIT    #1, $CODE(R0)        ;HEADER OPERATION ?
4918 017510 001403          BEQ    3S                    ;BR IF NOT
4919 017512 062760 000004 000022  ADD    #4, $SSEC(R0)         ;ADD HEADER SIZE
4920 017520 005060 000104 3S:    CLR    $NEXT(R0)            ;RESET 'PARAMETERS LOADED' INDICATOR
4921 017524 012605          MOV    (SP)+, R5            ;RESTORE R5
4922 017526 000207          RTS    PC                  ;RETURN

;ROUTINE TO COMPRESS A LIST
4923          :CALL:
4924          :
4925          :   MOV    #ADDRS, R1          ;COMPRESS LIST STARTING AT THIS ADDRESS
4926          :   JSR    PC, CMPRES
4927          :   RETURN

4928          :
4929          :
4930 017530 016111 000002  CMPRES: MOV    2(R1), (R1)      ;COMPRESS THE TABLE IN R1
4931 017534 001403          BEQ    IS                    ;BR WHEN ZERO FOUND
4932 017536 062701 000002  ADD    #2, R1                ;INCREMENT R1
4933 017542 000772          BR    CMPRES                ;CONTINUE COMPRESSING TABLE
4934 017544 000207 1S:    RTS    PC                  ;RETURN

;ROUTINE TO SETUP PARAMETEPS FOR A SEQUENTIAL READ OR WRITE OF THE DISK
4935          :CALL:
4936          :
4937          :
4938          :   MOV    #DPB, R0          ;DPB ADDRESS
4939          :   MOV    #-1, $PACK(R0)    ;'WRITE PACK' FLAG
4940          :   OR
4941          :   MOV    #1, $PACK(R0)    ;'READ PACK' FLAG
4942          :   JSR    PC, WRTPK
4943          :   RETURN

4944          :
4945 017546 004737 032364  WRTPK: JSR    PC, $RAND          ;CYCLE THE RANDOM NUMBER GENERATOR
4946 017552 005760 000040  TST    $OPERC+2(R0)         ;SEE IF FIRST OPERATION
4947 017556 001007          BNE    WRTPK1                ;BR IF UPPER WORD OF COUNTER NOT ZERO
4948 017560 005760 000036  TST    $OPERC(R0)           ;LOWER WORD ZERO ?
4949 017564 001004          BNE    WRTPK1                ;BR IF NOT 1ST OPERATION

```

```

4950 017566 105760 000026          TSTB   $SPACK(RO)          ;SEE WHICH - 'R' OR 'W'
4951 017572 100503          BMI    WRTPK3             ;BR IF 'W'
4952 017574 000470          BR     WRTPK2             ;'R' OPERATION
4953 017576 116060 000234 000027 WRTPK1: MOVVB  $RPCS1(RO), $PREV0(RO) ;SAVE CURRENT PARAMETERS
4954 017604 004737 022314          JSR    PC, READDR         ;GET THE DECREMENTED SECTOR AND TRACK ADDRESSES
4955 017610 112660 000033          MOVVB  (SP)+, $PREVA+1(RO) ;TRACK ADDRESS
4956 017614 112660 000032          MOVVB  (SP)+, $PREVA(RO)   ;SECTOR ADDRESS
4957 017620 016060 000272 000034          MOV    $RPCC(RO), $PREVA+2(RO) ;CURRENT CYLINDER
4958 017626 016060 000242 000010          MOV    $RPDA(RO), $SEC(RO)  ;NEW SECTOR & TRACK ADDRESS
4959 017634 016060 000270 000012          MOV    $RPCA(RO), $CYL(RO)  ;NEW CYLINDER ADDRESS
4960 017642 026060 000012 000106          CMP    $CYL(RO), $MAXCYL(RO) ;SEE IF AT END
4961 017650 103427          BLO    2$                ;BR IF LESS THAN 'MAXCYL'
4962 017652 101004          BHI    1$                ;BR IF GREATER THAN 'MAXCYL'
4963 017654 126060 000011 000112          CMPB   $TRK(RO), $MAXTRK(RO) ;SEE IF AT MAX TRACK
4964 017662 101422          BLOS   2$                ;BR IF NOT GREATER
4965 017664 116060 000114 000011 1$          MOVVB  $MINTRK(RO), $TRK(RO) ;RESET TRACK ADDRESS
4966 017672 116060 000120 000010          MOVVB  $MINSEC(RO), $SEC(RO) ;RESET SECTOR ADDRESS
4967 017700 016060 000110 000012          MOV    $MINCYL(RO), $CYL(RO) ;RESET CYLINDER ADDRESS
4968 017706 004737 026626          JSR    PC, EOP2           ;DROP THE DRIVE (NORMAL TERMINATION)
4969 017712 032777 000020 161220          BIT    #SW04, $SWR         ;IS SWITCH 4 SET ?
4970 017720 001003          BNE    2$                ;BR IF SET
4971 017722 005726          TST    (SP)+              ;INCREMENT THE STACK POINTER
4972 017724 000137 005374          JMP    MAIN               ;RETURN DIRECTLY TO 'MAIN'
4973 017730 013760 001404 000020 2$          MOV    $MAXDL, $WRDL(RO)   ;BUFFER SIZE IS MAXIMUM
4974 017736 013760 001404 000004          MOV    $MAXDL, $WRDM(RO)  ;WORD COUNT
4975 017744 005460 000004          NEG    $WRDM(RO)          ;CHANGE WORD COUNT TO 2'S COMPLEMENT
4976 017750 105760 000026          TSTB   $SPACK(RO)         ;READ OR WRITE ?
4977 017754 100412          BMI    WRTPK3             ;BR IF WRITE
4978 017756 012760 000404 000022 WRTPK2: MOV    #260, $SSEC(RO)   ;SECTOR SIZE FOR READ
4979 017764 112760 000005 000024          MOVVB  #5, $CODE(RO)      ;CODE FOR READ HEADER & DATA
4980 017772 112760 000173 000002          MOVVB  #ROHD, $COMND(RO)  ;DRIVE CODE FOR OPERATION
4981 020000 000415          BR     WRTPK4             ;SET UP FOR EXIT
4982 020002 012760 000400 000022 WRTPK3: MOV    #256, $SSEC(RO) ;SECTOR SIZE
4983 020010 112760 000002 000024          MOVVB  #2, $CODE(RO)      ;CODE FOR WRDAT
4984 020016 112760 000161 000002          MOVVB  #WRDAT, $COMND(RO) ;OP CODE
4985 020024 004737 017274          JSR    PC, GETPAT         ;GET PATTERN CODE
4986 020030 110560 000030          MOVVB  R5, $PATT(RO)      ;PATTERN CODE
4987 020034 005060 000104 WRTPK4: CLR    $NEXT(RO)    ;CLEAR 'PARAMETER SELECTED' INDICATOR
4988 020040 000207          RTS                       ;RETURN
4989
4990 ;ROUTINE TO DETERMINE IF ERROR IS AT A LOCATION ON THE PACK DEFINED
4991 ;IN THE BAD TRACK/SECTOR TABLE FOR THE DRIVE.
4992 ;CALL:
4993 ;       JSR    PC, SPOTCK
4994 ;       RETURN1
4995 ;       RETURN2
4996 ;
4997 ;
4998 SPOTCK:
4999 020042 010146          MOV    R1, -(SP)          ;PUSH R1 ON STACK
5000 020044 010246          MOV    R2, -(SP)          ;PUSH R2 ON STACK
5001 020046 012701 000124          MOV    #BADSEC, R1        ;INCREMENT FOR BAD SECTOR TABLE
5002 020052 060001          ADD    R0, R1             ;ADD THE BLOCK'S STARTING ADDRESS
5003 020054 012702 000020          MOV    #16, R2           ;BAD SECTOR TABLE SIZE COUNT
5004 020060 021160 000272 1$          CMP    (R1), $RPCC(RO)    ;IS CYLINDER IN THE TABLE ?
5005 020064 001022          BNE    4$                ;BR IF NOT

```

```

5006 020066 105761 000003          TSTB    3(R1)          ; TRACK ENTRY ?
5007 020072 100426                BMI     5$           ; BR IF NOT
5008 020074 004737 022314          JSR     PC,READDR    ; DECREMENT THE SECTOR/TRACK ADDRESS
5009 020100 122661 000003          CMPB   (SP)+,3(R1)   ; COMPARE THE TRACK ADDRESS
5010 020104 001011                BNE     3$           ; BR IF IT IS NOT EQUAL
5011 020106 105761 000002          TSTB   2(R1)          ; IS A SECTOR ADDRESS IN THE TABLE ?
5012 020112 100002                BPL     2$           ; BR IF ONE IS
5013 020114 005726                TST    (SP)+         ; INCREMENT THE STACK POINTER
5014 020116 000414                BR      5$           ; DISPLAY THE MESSAGE
5015 020120 122661 000002          2$:    CMPB   (SP)+,2(R1) ; COMPARE THE SECTOR ADDRESS
5016 020124 001002                BNE     4$           ; BR IF NOT EQUAL
5017 020126 000410                BR      5$           ; CHECK 'NOTPRT'
5018 020130 005726                3$:    TST    (SP)+         ; INCREMENT THE STACK POINTER
5019 020132 062701 000004          4$:    ADD    #4,R1      ; GO TO THE NEXT LOCATION IN THE TABLE
5020 020136 005711                TST    (R1)          ; PAST THE TABLE ENTRIES ?
5021 020140 100411                BMI     6$           ; BR IF PAST
5022 020142 005302                DEC     R2           ; DECREMENT THE MAXIMUM ENTRY COUNT
5023 020144 001345                BNE     1$           ; BR IF MORE TO CHECK
5024 020146 000406                BR      6$           ; END, EXIT
5025 020150 005737 001426          5$:    TST    NOTPRT     ; PRINT THE ERROR ANYWAY ?
5026 020154 001006                BNE     7$           ; BR IF NOT
5027 020156 012737 177777 001264  MOV     #-1,BADSEC    ; SET THE INDICATOR FOR THE IDENTIFICATION LINE
5028 020164 062766 000002 000004  6$:    ADD    #2,4(SP)   ; INCREMENT THE RETURN
5029 020172                                7$:
5030 020172 012602                MOV     (SP)+,R2     ; POP STACK INTO R2
5031 020174 012601                MOV     (SP)+,R1     ; POP STACK INTO R1
5032 020176 000207                RTS     PC           ; RETURN

```

;;*****

.SBTTL ERROR MESSAGE GENERATION ROUTINES

;;*****

:PRINT LINE 1 OF ERROR MESSAGE:
;'HH:MM:SS'

```

5043 020200 032777 002000 160732 LINE1: BIT     #SW10,JSWR    ; SWITCH 10 SET ?
5044 020206 001402                BEQ     1$           ; BR IF NOT
5045 020210 104401 001160          TYPE   $BELL        ; RING THE BELL
5046 020214 032777 020000 160716  1$:    BIT     #SW13,JSWR    ; INHIBIT TYPEOUT ?
5047 020222 001403                BEQ     2$           ; BR IF NOT
5048 020224 104414 001165          DISPLY $SCRLF       ; CR-LF
5049 020230 000404                BR      3$           ; EXIT
5050 020232 004737 023450          2$:    JSR     PC,$TIME   ; TYPE THE TIME
5051 020236 104414 052156          DISPLY $LINSPO     ; SPACES
5052 020242 000207                3$:    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  RPD'
:'RPSN  RPOF  RPCA  RPCC  STATUS'
:'BUS ADDRESS OR WORD COUNT NOT CONSISTENT'
:'RPBA = XXXXX'  RPWC = XXXXX'

```

5061

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

Line No	Address	Instruction	Comment	Label	Description
5062					
5063					
5064	020244				
5065	020244	010346			
5066	020246	010446			
5067	020250	010546			
5068	020252	104414	001165		
5069	020256	005037	020404		
5070	020262	005004			
5071	020264	012737	050242	020404	
5072	020272	116004	000234		
5073	020276	042704	177701		
5074	020302	004737	020340		
5075	020306	005737	020410		
5076	020312	001440			
5077	020314	012737	050263	020404	
5078	020322	116004	000027		
5079	020326	042704	177701		
5080	020332	004737	020340		
5081	020336	000426			
5082	020340	005005			
5083	020342	126504	001766		
5084	020346	001405			
5085	020350	105765	001766		
5086	020354	100402			
5087	020356	005205			
5088	020360	000770			
5089	020362	006305			
5090	020364	006305			
5091	020366	006305			
5092	020370	012737	002010	020410	
5093	020376	060537	020410		
5094	020402	104414			
5095	020404	000000			
5096	020406	104414			
5097	020410	000000			
5098	020412	000207			
5099	020414	005737	001264		
5100	020420	001404			
5101	020422	104414	001165		
5102	020426	104414	050307		
5103	020432	104414	001165		
5104	020436	104414	047636		
5105	020442	104414	052156		
5106	020446	013746	001246		
5107	020452	004737	022274		
5108	020456	104414	052155		
5109	020462	012705	050162		
5110	020466	004737	020616		
5111	020472	032777	000040	160440	
5112	020500	001014			
5113	020502	104414	047742		
5114	020506	012705	050204		
5115	020512	004737	020616		
5116	020516	104414	050041		
5117	020522	012705	050226		

```

LINE2:
MOV R3,-(SP)      ;: PUSH R3 ON STACK
MOV R4,-(SP)      ;: PUSH R4 ON STACK
MOV R5,-(SP)      ;: PUSH R5 ON STACK
DISPLY ,SCLRF     ;: CR-LF
CLR 4$           ;: CLEAR MESSAGE ADDRESS STORAGE
CLR R4           ;: WORKING REGISTER
MOV #LIN2C,4$     ;: ADDRESS OF 'PRESENT ORDER = ' MSG
MOVB $RPCS1(RO),R4 ;: GET THE OPCODE
BIC #C76,R4      ;: SAVE ONLY SIGNIFICANT BITS
JSR PC,1$        ;: TYPE THE FIRST MNEMONIC
TST 5$          ;: SEE IF MNEMONIC ENTRY FOUND
BEQ LINE2A       ;: BR IF NOT
MOV #LIN2P,4$     ;: ADDRESS OF 'PREVIOUS ORDER = ' MSG
MOVB $PREV0(RO),R4 ;: PREVIOUS OPERATION CODE
BIC #C76,R4      ;: SAVE ONLY SIGNIFICANT BITS
JSR PC,1$        ;: TYPE THE PREVIOUS MNEMONIC
BR LINE2A        ;: CONTINUE
1$: CLR R5        ;: CLEAR THE TABLE INDEX
2$: CMPB OPTBL(R5),R4 ;: LOOK FOR THE OPCODE
BEQ 3$           ;: BR WHEN OPCODE COUNT EQUALS OPCODE
TSTB OPTBL(R5)   ;: LOOK FOR END OF TABLE
BMI 3$           ;: BR IF END
INC R5           ;: INCREMENT THE POINTER
BR 2$           ;: CONTINUE - NOT END OF TABLE
3$: ASL R5        ;: SHIFT INDEX
ASL R5           ;: SHIFT THE INDEX
ASL R5           ;: SHIFT THE INDEX
MOV #MNTBL,5$    ;: ADDRESS OF ASCII TEXT TABLE
ADD R5,5$        ;: ADD THE INDEX
DISPLY ,WORD 0   ;: TYPE IT
4$: .WORD 0      ;: ADDRESS OF 'PRESENT' OR 'PREVIOUS' MESSAGE
DISPLY ,WORD 0   ;: TYPE THE OPERATION MNEMONIC
5$: .WORD 0      ;: ADDRESS OF MESSAGE
RTS PC           ;: RETURN TO MAIN ROUTINE
LINE2A: TST BADSEC ;: PRINT THE BAD SECTOR LINE ^
BEQ LINE2B       ;: BR IF NOT
DISPLY ,SCLRF     ;: CR-LF
DISPLY ,LIN2S     ;: ERROR ADDRESS DEFINED AS BAD AREA
DISPLY ,SCLRF     ;: CR-LF
DISPLY ,DH14      ;: STANDARD RPO4/5/6 REGISTER HEADER
DISPLY ,LINSPO    ;: TYPE A SPACE
MOV UNIT,-(SP)   ;: PUT THE DRIVE NUMBER ON THE STACK
JSR PC,LINDEC    ;: TYPE DRIVE NUMBER
DISPLY ,LINSPL    ;: SPACES
MOV #DT14,R5     ;: REGISTER INDEXES
JSR PC,3$        ;: PRINT THE REGISTERS
BIT #SW05,2SWR   ;: PRINT THE OPTIONAL REGISTERS ^
BNE 1$           ;: BR IF NOT
DISPLY ,DH15      ;:
MOV #DT15,R5     ;: SECOND DATA LINE
JSR PC,3$        ;: PRINT THEM
DISPLY ,DH16      ;:
MOV #DT16,R5     ;: THIRD DATA LINE

```

```

5118 020526 004737 020616      JSR      PC,3$          ;PRINT THE REGISTERS
5119 020532 032760 000100 000016 1$:  BIT      #BIT6,$STATUS(RO) ;DATA ERROR ?
5120 020540 001422          BEQ      2$            ;BR IF NOT
5121 020542 016046 000020      MOV      $WRDL(RO),-(SP) ;TRANSFER SIZE
5122 020546 066016 000236      ADD      $RPWC(RO),(SP) ;ADD REMAINING WORD COUNT
5123 020552 006316          ASL      (SP)          ;CONVERT TO AN BYTE INCREMENT
5124 020554 066016 000006      ADD      $BUF(RO),(SP) ;BUFFER STARTING ADDRESS
5125 020560 022660 000240      CMP      (SP)+,$RPBA(RO) ;CORRECT BUFFER ADDRESS ?
5126 020564 001410          BEQ      2$            ;BR IF YES
5127 020566 104414 047231      DISPLY  ,EM46          ;'BUS ADDRESS AND WORD COUNT ARE NOT CONSISTENT'
5128 020572 104414 001165      DISPLY  ,%CRLF         ;CR-LF
5129 020576 004737 020710      JSR      PC,LINE3D     ;PRINT LINE 3D OF ERROR MESSAGE
5130 020602 004737 021326      JSR      PC,LINE4     ;PRINT LINE 4 OF ERROR MESSAGE
5131 020606          2$:
5132 020606 012605          MOV      (SP)+,R5      ;POP STACK INTO R5
5133 020610 012604          MOV      (SP)+,R4      ;POP STACK INTO R4
5134 020612 012603          MOV      (SP)+,R3      ;POP STACK INTO R3
5135 020614 000207          RTS      PC          ;RETURN TO ERROR PROCESSING ROUTINE
5136 020616 012546          3$:  MOV      (RS)+,-(SP)    ;PUT THE REGISTER INDEX ON THE STACK
5137 020620 060016          ADD      RO,(SP)      ;ADD DRIVE'S TABLE ADDRESS
5138 020622 017646 000000      MOV      0(SP),-(SP)  ;VALUE
5139 020626 004737 022242      JSR      PC,LIN0CT    ;TYPE IT
5140 020632 005726          TST      (SP)+        ;CORRECT THE STACK POINTER
5141 020634 104414 052155      DISPLY  ,LINSP        ;PRINT 2 SPACES
5142 020640 005715          TST      (RS)         ;AT END OF LINE ?
5143 020642 001365          BNE      3$          ;BR IF NOT
5144 020644 104414 001165          4$:  DISPLY  ,%CRLF         ;CR-LF
5145 020650 000207          RTS      PC          ;RETURN
5146
5147          ;PRINT LINE 3 OF ERROR MESSAGE
5148          ;'ERROR AT CCC TT SS  PREVIOLS ADR = CCC TT SS'
5149
5150 020652 104414 050343      LINE3:  DISPLY  ,LINM3 ;LINE 3 ENTRANCE
5151 020656 000517          BR       LIN3.1      ;FINISH PRINTOUT
5152
5153          ;PRINT LINE 3A OF ERROR MESSAGE
5154          ;'START CYL = CCC  END CYL = CCC'
5155
5156 020660 104414 050361      LINE3A:  DISPLY  ,LINM3 ;LINE 3A ENTRANCE
5157 020664 000514          BR       LIN3.1      ;FINISH ERROR LINE
5158
5159          ;PRINT LINE 3B OF ERROR MESSAGE
5160          ;'START CYL = CCC  END CYL = CCC  ACTUAL CYL = CCC'
5161
5162 020666 004737 021230      LINE3B:  JSR      PC,LIN3.3 ;LINE 3B ENTRANCE
5163 020672 104414 001165      DISPLY  ,%CRLF
5164 020676 000207          RTS      PC
5165
5166          ;PRINT LINE 3C OF ERROR MESSAGE
5167          ;'START CYL = CCC  END CYL = CCC  ACTUAL CYL = CCC  TRK = TT'
5168
5169 020700 004737 021230      LINE3C:  JSR      PC,LIN3.3 ;LINE 3C ENTRANCE
5170 020704 000137 021262      JMP      LIN3.4      ;FINISH MESSAGE
5171
5172          ;PRINT LINE 3D OF ERROR MESSAGE
5173          ;'RPBA = XXXXXX  RPWC = XXXXXX'

```


F08

C2RJDCO, RPO4 5 6 MLT-DR LGC
C2RJDC.P11 15-DEC-77 10:58

MACY11 30(1046) 15-DEC-77 11:03 PAGE 97
ERROR MESSAGE GENERATION ROUTINES

SEQ 0096

```

S174
S175 020710 032777 000040 160222 LINE3D: BIT      #SW05,JSWR      ;SWITCH 5 SET ?
S176 020716 001416                BEQ      IS          ;BR IF IT IS
S177 020720 104414 050532        DISPLY   LINB3       ;'RPBA = '
S178 020724 016046 000240        MOV      $RPBA(RO),-(SP) ;BUFFER ADDR REG CONTENTS
S179 020730 004737 022242        JSR      PC,LINOC   ;CONVERT TO OCTAL AND TYPE IT
S180 020734 104414 050542        DISPLY   LINW3       ;'RPWC = '
S181 020740 016046 000236        MOV      $RPWC(RO),-(SP) ;WORD COUNT REGISTER CONTENTS
S182 020744 004737 022242        JSR      PC,LINOC   ;CONVERT TO OCTAL AND TYPE IT
S183 020750 104414 001165        DISPLY   $CRLF
S184 020754 000207                RTS      PC
S185
S186
S187 ;PRINT LINE 3E OF ERROR MESSAGE
S188 ;'START CYL = CCC  START TRK = TT  START SEC = SS'
S189
S189 020756 104414 050426        LINE3E: DISPLY   LINS3       ;'START CYL = '
S190 020762 016046 000012        MOV      $CYL(RO),-(SP) ;MOVE CYL TO STACK
S191 020766 004737 022274        JSR      PC,LINDEC  ;TYPE IT IN DECIMAL
S192 020772 104414 052155        DISPLY   ,LINS3     ;SPACES
S193 020776 104414 050554        DISPLY   ,LINST3    ;'START TRK = '
S194 021002 005046                CLR      -(SP)       ;CLEAR STACK
S195 021004 116016 000011        MOV      $TRK(RO),(SP) ;TRACK TO STACK
S196 021010 004737 022274        JSR      PC,LINDEC  ;TYPE IT IN DECIMAL
S197 021014 104414 052155        DISPLY   ,LINS3     ;SPACES
S198 021020 104414 050571        DISPLY   ,LINS3     ;'START SEC = '
S199 021024 005046                CLR      -(SP)       ;CLEAR STACK
S200 021026 116016 000010        MOV      $SEC(RO),(SP) ;SECTOR ADDR TO STACK
S201 021032 004737 022274        JSR      PC,LINDEC  ;TYPE IT IN DECIMAL
S202 021036 104414 001165        DISPLY   $CRLF
S203 021042 000207                RTS      PC
S204
S205 ;PRINT LINE 3F OF ERROR MESSAGE
S206 ;'RPDA = XXXXXX  RPCA = XXXXXX'
S207
S208 021044 032777 000040 160066 LINE3F: BIT      #SW5,JSWR      ;SWITCH 5 SET ?
S209 021052 001420                BEQ      IS          ;BR IF NOT
S210 021054 104414 050522        DISPLY   LINDA3      ;'RPDA = '
S211 021060 016046 000242        MOV      $RPDA(RO),-(SP) ;PUT SECTOR/TRACK ADDRESS ON THE STACK
S212 021064 004737 022242        JSR      PC,LINOC   ;TYPE IT
S213 021070 104414 052155        DISPLY   ,LINS3     ;SPACES
S214 021074 104414 050511        DISPLY   LINDCA3     ;'RPCA = '
S215 021100 016046 000270        MOV      $RPCA(RO),-(SP) ;PUT DESIRED CYLINDER ADDRESS ON THE STACK
S216 021104 004737 022242        JSR      PC,LINOC   ;TYPE IT
S217 021110 104414 001165        DISPLY   $CRLF
S218 021114 000207                RTS      PC
S219
S220 ;'CCC TT SS  PREV ADR = CCC TT SS'
S221
S221 021116 016046 000272        LIN3.1: MOV      $RPCC(RO),-(SP) ;PUT CYLINDER ADDR ON STACK
S222 021122 004737 022274        JSR      PC,LINDEC  ;TYPE IT IN DECIMAL
S223 021126 104414 050356        DISPLY   T          ;PRINT 'T'
S224 021132 004737 022314        JSR      PC,READDR  ;DECREMENT TRACK AND SECTOR ADDRESSES
S225 021136 004737 022274        JSR      PC,LINDEC  ;TYPE TRACK IN DECIMAL
S226 021142 104414 050402        DISPLY   S          ;PRINT 'S'
S227 021146 004737 022274        JSR      PC,LINDEC  ;TYPE SECTOR ADDRESS
S228 021152 104414 050405        DISPLY   ,LINP3     ;PRINT 'PREV ADDR'

```


H08

COPY DOC RPO4 5 6 MLT-DR LGC
COPY DOC.P11 15-DEC-77 10:58

MACY11 30(1046) 15-DEC-77 11:03 PAGE 99
ERROR MESSAGE GENERATION ROUTINES

SEQ 0098

5286
5287
5288 021416 104414 050672
5289 021422 162760 000002
5290 021430 017046 000240
5291 021434 004737 022242
5292 021440 104414 050707
5293 021444 016046 000256
5294 021450 004737 022242
5295 021454 016046 000236
5296 021460 066016 000020
5297 021464 005046
5298 021466 016046 000022
5299 021472 004737 027054
5300 021476 012616
5301 021500 104414 050725
5302 021504 004737 022274
5303 021510 104414 001165
5304 021514 000207

: 'GOOD DATA = XXXXXX BAD DATA = XXXXXX SECT POS = XXX'

000240 LINES: DISPLY ,LINS
SUB #2,\$RPA(RO) ;PRINT 'GOOD DATA'
MOV \$RPA(RO),-(SP) ;BACK THE ADDRESS UP
JSR PC,LINOC ;'GOOD' DATA - AT THE BUFFER LOCATION
DISPLY ,LINS ;TYPE IT
MOV \$RPA(RO),-(SP) ;PRINT 'BAD DATA'
JSR PC,LINOC ;BAD DATA FROM BUFFER
MOV \$RPA(RO),-(SP) ;TYPE IT
ADD \$RPA(RO),(SP) ;WORD LENGTH ON STACK
CLR -(SP) ;MAKE INTO A POSITIVE NUMBER
MOV \$SSEC(RO),-(SP) ;UPPER DIVIDEND TO ZERO
JSR PC,LINKDV ;SECTOR SIZE ON THE STACK
MOV (SP)+(SP) ;DIVIDE WORDS XFERED BY SECTOR SIZE
DISPLY ,LINS ;MOVE REMAINDER UP THE STACK
JSR PC,LINDEC ;PRINT 'SECT POS'
DISPLY \$CRLF ;TYPE THE POSITION
RTS PC

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

5305
5306
5307
5308
5309 021516 104414 050743
5310 021522 013746 054456
5311 021526 004737 022242
5312 021532 104414 052155
5313 021536 013746 054460
5314 021542 004737 022242
5315 021546 104414 052155
5316 021552 013746 054462
5317 021556 004737 022242
5318 021562 104414 052155
5319 021566 013746 054464
5320 021572 004737 022242
5321 021576 104414 052155
5322 021602 104414 001165
5323 021606 000207

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

:PRINT LINE 5B OF ERROR MESSAGE
: 'RPEC1 = XXXXXX RPEC2 = XXXXXX'

5324
5325
5326
5327
5328 021610 104414 050777
5329 021614 016046 000300
5330 021620 004737 022242
5331 021624 104414 052155
5332 021630 104414 051010
5333 021634 016046 000302
5334 021640 004737 022242
5335 021644 104414 001165
5336 021650 000207

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

:PRINT LINE 6 OF ERROR MESSAGE
: 'SECTOR IS ECC CORRECTABLE'

5341 021652 104414 051022

LINE6: DISPLY ,LIN6 ;ECC CORRECTABLE

3342 021656 104414 001165
 3343 021662 000207
 3344
 3345
 3346
 3347
 3348 021664 104414 051055
 3349 021670 000411
 3350
 3351
 3352
 3353
 3354
 3355
 3356
 3357
 3358 021672 104414 051022
 3359 021676 000406
 3360
 3361
 3362
 3363
 3364
 3365
 3366 021700 104414 051104
 3367 021704 000414
 3368
 3369
 3370
 3371 021714 006301
 3372 021716 016137 002240 021726
 3373 021724 104414
 3374 021726 000000
 3375 021730 104414 001165
 3376 021734 000207
 3377
 3378
 3379
 3380 021736 005046
 3381 021740 113716 001253
 3382 021744 004737 022274
 3383 021750 104414 051122
 3384 021754 104414 001165
 3385 021760 000207
 3386
 3387
 3388
 3389
 3390 021762 104414 051206
 3391 021766 012746 000036
 3392 021772 060016
 3393 021774 004737 032562
 3394 022000 004737 027370
 3395 022004 104414 051265
 3396 022010 016046 000056
 3397 022014 004737 022274

```

DISPLY $CRLF
RTS PC

;PRINT LINE 6A OF THE ERROR MESSAGE
;'SECTOR READ CORRECTLY AT OFFSET N'

LINE6A: DISPLY ,LINC6 ;PRINT 'READ CORRECTLY AT OFFSET N'
BR LIN6.1 ;TYPE THE REST OF THE LINE

;PRINT LINE 6B OF THE ERROR MESSAGE
;'SECTOR IS ECC CORRECTABLE AT OFFSET N'

LINE6B: DISPLY ,LINB6 ;PRINT 'SECTOR IS ECC CORRECTABLE '
BR LIN6.1

;PRINT LINE 6C OF THE ERROR MESSAGE
;'CORRECTED ON NTH RETRY'

LINE6C: DISPLY ,LING6 ;'CORRECTED ON NTH RETRY'
BR LIN6.2 ;TYPE THE REST OF THE LINE

;PRINT LINE 6D OF THE ERROR MESSAGE
;'UNCORRECTABLE AFTER N RETRIES'

LINE6D: DISPLY ,LINU06 ;'UNCORRECTABLE AFTER N RETRIES'
BR LIN6.2 ;FINISH

;TYPE THE OFFSET VALUE IN MICRO-INCHES

LIN6.1: ASL R1 ;DOUBLE THE OFFSET TABLE INDEX
MOV OFMTBL(R1),1$ ;ADDRESS OF OFFSET POSITION MESSAGE
1$: .WORD 0 ;OFFSET VALUE
DISPLY $CRLF
RTS PC

;RETRY COUNT TYPEOUT

LIN6.2: CLR -(SP) ;CLEAR STACK
MOVB RETRY+1,(SP) ;RETRY COUNT
JSR PC,LINDEC ;TYPE IT IN DECIMAL
DISPLY ,LINR6 ;'RETRY'
DISPLY $CRLF
RTS PC

;PRINT LINE 7 OF THE ERROR MESSAGE
;'ORDEPS:XXXXX TOTAL ERRORS:XXX WRDS XFRD:XXXXXXX WRDS READ:XXXXXXX'

LINE7: DISPLY ,LIN70 ;PRINT ORDER COUNT
MOV $SOPERC,-(SP) ;TO STACK
ADD RO,(SP) ;ADD THE BASE ADDRESS
JSR PC,$0B20 ;CONVERT IT
JSR PC,$SUPRS ;PRINT IT
DISPLY ,LIN71 ;TOTAL ERRORS
MOV $TOTAL(RO),-(SP) ;TO STACK
JSR PC,LINDEC ;TYPE IT IN DECIMAL
  
```

5398 022020 104414 051277
5399 022024 012746 000046
400 022030 060016
401 022032 004737 032562
402 022036 004737 027370
403 022042 104414 051313
404 022046 012746 000052
405 022052 060016
406 022054 004737 032562
407 022060 004737 027370
408 022064 104414 001165
409 022070 104414 001165
410 022074 032777 100000
411 022102 001401
412 022104 000000
413 022106 000207

157036

```

DISPLY LIN7X ;PRINT 'WRDS XFR'
MOV #STRANS, -(SP) ;ADDRESS OF LOW WORD ON STACK
ADD RO, (SP)
JSR PC, $SUBCU ;CONVERT
JSR PC, $SUPRS ;PRINT
DISPLY LIN7R ;'BITS READ'
MOV #SREAD, -(SP) ;LOW WORD ADDRESS
ADD RO, (SP)
JSR PC, $DB20 ;CONVERT
JSR PC, $SUPRS ;PRINT
DISPLY $SCLF
DISPLY $SCLF
BIT #SW15, $SWR ;SEE IF 'HALT ON ERROR' - SWITCH 15
BEQ $S ;BR IF NOT
HALT ;SWITCH 15 HALT
IS: RTS PC

```

```

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

```

414 022110 104414 051206
415 022114 012746 000036
416 022120 060016
417 022122 004737 032562
418 022126 004737 027370
419 022132 104414 051216
420 022136 012746 000042
421 022142 060016
422 022144 004737 032562
423 022150 004737 027370
424 022154 104414 051160
425 022160 016046 000066
426 022164 004737 022274
427 022170 104414 051236
428 022174 016046 000064
429 022200 004737 022274
430 022204 104414 001165
431 022210 104414 001165
432 022214 032777 100000
433 022222 001401
434 022224 000000
435 022226 000207

156716

```

LINE7A: DISPLY LIN7O ;'ORDERS = '
MOV #SOPERC, -(SP) ;ORDER COUNT INCREMENT
ADD RO, (SP) ;ADD BASE ADDRESS
JSR PC, $DB20 ;CONVERT THE COUNT
JSR PC, $SUPRS ;PRINT IT
DISPLY LIN7P ;'TOTAL SEEKS = '
MOV #SPOSIT, -(SP) ;TOTAL SEEKS
ADD RO, (SP) ;DEVICE TABLE ADDRESS
JSR PC, $DB20 ;CONVERT THE SEEK COUNT
JSR PC, $SUPRS ;PRINT IT
DISPLY LIN7M ;' TOTAL MISPOS ERR = '
MOV #MISPO(RO), -(SP) ;TOTAL ERRORS
JSR PC, LINDEC ;TYPE IT IN DECIMAL
DISPLY LIN7S ;' TOTAL SKI,OCYL ERR = '
MOV #SKI(RO), -(SP) ;CONVERT & PRINT IT
JSR PC, LINDEC ;TYPE IT IN DECIMAL
DISPLY $SCLF
DISPLY $SCLF
BIT #SW15, $SWR ;SEE IF HALT ON ERROR - SWITCH 15 SET
BEQ $S ;BR IF NOT
HALT ;SWITCH 15 HALT
IS: RTS PC

```

```

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

```

436 022230 104414 051330
437 022234 004737 020244
438 022240 000207

```

LINEB: DISPLY LIN8M ;PRINT LINE 2 OF ERROR MESSAGE
JSR PC, LINE2
RTS PC

```

```

;OCTAL TYPEOUT ROUTINE
;CALL:
; MOV NUM, -(SP) ;PUT THE NUMBER ON THE STACK
; JSR PC, LINOCT
; RETURN

```

53

K08

CZRJDCO RPO4 5.6 MLT-DR LGC
CZRJDC.P11 15-DEC-77 10:58

MACY11 30(1046) 15-DEC-77 11:03 PAGE 102
ERROR MESSAGE GENERATION ROUTINES

SEQ 0101

5454	022242	016646	000002	LINOCT:	MOV	2(SP),-(SP)	:PUT NUMBER IN PROPER LOCATION ON STACK
5455	022246	004737	030020		JSR	PC,\$5B20	:CONVERT THE NUMBER TO OCTAL
5456	022252	012637	022266		MOV	(SP)+,1\$:GET THE ADDRESS OF THE ASCII STRING
5457	022256	062737	000005	022266	ADD	#5.,1\$:ADDRESS THE LAST 6 ASCII DIGITS
5458	022264	104414			DISPLY		:TYPE IT
5459	022266	000000		1\$:	.WORD	0	:ADDRESS
5460	022270	012616			MOV	(SP)+,(SP)	:CORRECT THE STACK
5461	022272	000207			RTS	PC	:RETURN

5462
5463
5464
5465
5466
5467
5468
5469
5470
5471
5472
5473
5474
5475
5476
5477
5478
5479
5480
5481
5482
5483
5484
5485
5486
5487
5488
5489
5490
5491
5492
5493
5494
5495
5496
5497
5498
5499
5500
5501
5502
5503
5504
5505
5506
5507
5508
5509
5510
5511
5512
5513
5514
5515
5516
5517

022274 016646 000002
022300 004737 027770
022304 004737 027370
022310 012616
022312 000207

022314 162706 000004
022320 016616 000004
022324 005066 000004
022330 005066 000002
022334 116066 000242 000004
022342 005366 000004
022346 100015
022350 012766 000025 000004
022356 116066 000243 000002
022364 005366 000002
022370 100007
022372 012766 000022 000002
022400 000403
022402 116066 000243 000002
022410 000207

022412 012737 177777 001210
022420 012737 177777 001206
022426 012737 022506 000004
022434 005037 000006
022440 005777 156530
022444 005037 001210
022450 005037 001206
022454 013701 001200
022460 012721 023546
022464 012711 000300

```
;ROUTINE TO CONVERT THE INPUT NUMBER TO DECIMAL AND TYPE IT WITH  
;LEADING ZERO SUPPRESSION  
;CALL:  
;      MOV     NUM,-(SP)      ;PUT THE NUMBER ON THE STACK  
;      JSR     PC,LINEDEC  
;      RETURN  
LINEDEC: MOV     2(SP),-(SP)   ;SET UP STACK FOR CONVERT  
;      JSR     PC,$S82D      ;CONVERT IT TO DECIMAL  
;      JSR     PC,$SUPRS     ;TYPE IT (WITH LEADING ZEROS SUPPRESSED,  
;      MOV     (SP)+,(SP)    ;RESTORE STACK POINTER  
;      RTS     PC
```

.SBTTL GENERAL SUPPORT SUBROUTINES

```
;DECREMENT THE SECTOR-TRACK ADDRESS  
;CALL:  
;      MOV     #DPB,RO      ;DPB ADDRESS  
;      JSR     PC,READDR  
;      RETURN  
;      (SP) CONTAINS THE TRACK ADDRESS  
;      2(SP) CONTAINS THE SECTOR ADDRESS
```

```
READDR: SUB     #4,SP          ;DECREMENT THE STACK POINTER  
;      MOV     4(SP),(SP)    ;MOVE THE RETURN ADDR DOWN THE STACK  
;      CLR     4(SP)         ;CLEAR STACK FOR SECTOR  
;      CLR     2(SP)         ;CLEAR STACK FOR TRACK  
;      MOV     $RPDA(RO),4(SP) ;INCREMENTED SECTOR ON STACK  
;      DEC     4(SP)         ;DECREMENT THE SECTOR ADDRESS  
;      BPL     1$           ;BR IF SECTOR GREATER THAN 0  
;      MOV     #21,4(SP)    ;JAM SECTOR ADDRESS TO 21(10)  
;      MOV     $RPDA+1(RO),2(SP) ;TRACK ADDRESS  
;      DEC     2(SP)         ;DECREMENT TRACK ADDRESS  
;      BPL     2$           ;BR IF IT DIDN'T GO NEG  
;      MOV     #18,2(SP)    ;RESET TRACK TO 18(10)  
;      BR     2$  
1$: MOV     $RPDA+1(RO),2(SP) ;TRACK ADDRESS  
2$: RTS     PC              ;RETURN
```

;ROUTINE TO CHECK FOR KW11-L OR KW11-P CLOCKS

```
CKCLK: MOV     #-1,CLKFLG    ;CLEAR CLOCK AVAILABILITY FLAG  
;      MOV     #-1,PCLOCK    ;CLEAR KW11-P CLOCK AVAILABILITY FLAG  
;      MOV     #CKCLK1,ERRVEC ;SET UP VECTOR FOR CLOCK CHECK  
;      CLR     2*ERRVEC+2    ;NEW PSW  
;      TST     2$SLKCSR      ;CHECK FOR KW11-P  
;      CLR     CLKFLG        ;SET CLOCK AVAILABILITY FLAG  
;      CLR     PCLOCK        ;SET KW11-P CLOCK FLAG  
;      MOV     $LPVEC,R1     ;KW11-P VECTOR ADDRESS  
;      MOV     #CLOCK,(R1)+  ;SET UP KW11-P VECTOR  
;      MOV     #300,(R1)     ;PSW - PRI 6
```

```

5518 022470 012777 174575 156500      MOV      #1667, @SLKCSB ;LOAD COUNTER BUFFER WITH 16.67
5519 022476 012777 000131 156470      MOV      #131, @SLKCSR ;SET CLOCK - CNT UP, 10US, CONT INT
5520 022504 000437                BR      CKCLK3
5521 022506 062706 000004                CKCLK1: ADD      #4, SP ;RESTORE THE STACK POINTER
5522 022512 012737 022554 000004      MOV      @CKCLK2, @ERRVEC ;CHANGE ERROR VECTOR TO CHECK FOR KW11-L
5523 022520 005777 156456                TST      @SLKS ;LOOK FOR KW11-L
5524 022524 005037 001210                CLR      CLKFLG ;SET CLOCK FLAG
5525 022530 013701 001204                MOV      $LLVEC, R1 ;KW11-L VECTOR ADDRESS
5526 022534 012721 023546                MOV      @CLOCK, (R1)+ ;SET UP KW11-L VECTOR
5527 022540 012711 000300                MOV      #300, (R1) ;PSW - PRI 6
5528 022544 012777 000100 156430      MOV      #100, @SLKS ;SET KW11-L INTERRUPT
5529 022552 000414                BR      CKCLK3
5530 022554 062706 000004                CKCLK2: ADD      #4, SP ;RESTORE THE STACK POINTER
5531 022560 104401 052733                TYPE    NECDCLK ;'P OR L CLOCK MUST BE ON SYSTEM'
5532 022564 005737 000042                TST      42 ;UNDER MONITOR CONTROL ?
5533 022570 001402                BEQ     1$ ;BR IF NOT
5534 022572 000137 005344                JMP     $GET42 ;ABORT PROGRAM
5535 022576 000000                1$:    HALT ;HALT
5536 022600 000137 004106                JMP     START ;TRY AGAIN
5537 022604 012737 000006 000004      CKCLK3: MOV      #6, @ERRVEC ;RESTORE THE ERROR VECTOR
5538 022612 000207                RTS     PC
5539
5540 ;ROUTINE TO DISPLAY STATISTICS FOR ALL DRIVES ASSIGNED
5541 ;CALL:
5542 ;
5543 ;      JSR      PC, STATPR
5544 ;      RETURN
5545
5546 STATPR: MOV      R0, -(SP) ;SAVE R0
5547 022616 010446 001462      MOV      R4, -(SP) ;SAVE R4
5548 022620 005737 001462      TST      @ASNLST ;ANY DRIVES ASSIGNED ^
5549 022624 001421 022724      BEQ     3$ ;BR IF NOT
5550 022632 005004                JSR     PC, SHDTYP ;TYPE THE HEADING
5551 022634 006304                CLR     R4 ;CLEAR THE DRIVE INDEX
5552 022636 016400 001740      1$:    ASL     R4 ;CHANGE TO INDEX WORDS
5553 022642 006204                MOV     BLKADR(R4), R0 ;GET THE DRIVE'S BLOCK ADDRESS
5554 022644 136437 033320 001462      ASR     R4 ;RESTORE R4
5555 022652 001402                BITB   ATABIT(R4), @ASNLST ;IS THIS DRIVE ASSIGNED ?
5556 022654 004737 022746      BEQ     2$ ;BR IF NOT
5557 022660 005204                JSR     PC, SDETAL ;TYPE THE PERFORMANCE SUMMARY
5558 022662 020427 000010      2$:    INC     R4 ;INCREMENT THE INDEX
5559 022666 001362                CMP     R4, #8. ;FINISHED ?
5560 022670 012604                BNE    1$ ;BR IF NOT
5561 022672 012600      3$:    MOV     (SP)+, R4 ;RESTORE R4
5562 022674 000207      MOV     (SP)+, R0 ;RESTORE R0
5563                RTS     PC ;RETURN
5564
5565 ;ROUTINE TO TYPE STATISTICS FOR AN INDIVIDUAL DRIVE
5566 ;CALL:
5567 ;
5568 ;      MOV     #DPB, R0 ;DPB ADDRESS
5569 ;      JSR     PC, TYPEST
5570 ;      RETURN
5571
5572 TYPEST: MOV      R0, -(SP) ;SAVE R0
5573 022700 010446 022724      MOV      R4, -(SP) ;SAVE R4
5574 022702 004737                JSR     PC, SHDTYP ;TYPE THE HEADING
5575 022706 005004                CLR     R4 ;CLEAR R4 FOR DRIVE NUMBER

```



```

5574 022710 111004          MOV#   (R0),R4          ;DRIVE NUMBER
5575 022712 004737 022746  JSR    PC,SDETAL     ;TYPE THE STATISTICS
5576 022716 012604          MOV    (SP)+,R4      ;RESTORE R4
5577 022720 012600          MOV    (SP)+,R0      ;RESTORE R0
5578 022722 000207          RTS     PC           ;RETURN
5579
5580          ;TYPE THE HEADER FOR THE DRIVE PERFORMANCE SUMMARY TYPEOUT
5581          ;CALL:
5582          ;       JSR    PC,SHDTYP
5583          ;       RETURN
5584
5585 022724 004737 023450  SHDTYP: JSR    PC,$TIME ;TYPE THE TIME OF DAY
5586 022730 004537 027430  JSR    R5,TYPRI4     ;TYPE AT PRIORITY 4
5587 022734 001165          $CRLF                ;CR-LF
5588 022736 004537 027430  JSR    R5,TYPRI4     ;TYPE THE HEADER
5589 022742 052403          STATHD              ;HEADER
5590 022744 000207          RTS     PC           ;RETURN
5591
5592          ;TYPE THE PERFORMANCE SUMMARY DATE LINE
5593          ;CALL:
5594          ;       MOV    #DRIVE,R4 ;DRIVE NUMBER
5595          ;       MOV    #DPB,R0   ;DPB ADDRESS
5596          ;       RETURN
5597
5598 022746 010246          SDETAL: MOV    R2,-(SP)   ;SAVE R2
5599 022750 010002          MOV    R0,R2        ;DPB ADDRESS
5600 022752 062702 000036  ADD    #SOPERC,R2    ;FIRST STATISTICAL FIELD
5601 022756 010446          MOV    R4,-(SP)     ;SAVE R4 FOR TYPEOUT
5602
5603 022760 104403          TYPE    TYPOS        ;TYPE DRIVE NUMBER
5604 022762          .BYTE    2         ;GO TYPE--OCTAL ASCII
5605 022763          .BYTE    0         ;TYPE 2 DIGIT(S)
5606 022764 104401 052155  TYPE    LINSF         ;SUPPRESS LEADING ZEROS
5607 022770 016046 000070  MOV    $PASSC(R0),-(SP) ;SPACES
5608 022774 004737 027770  JSR    PC,$SB2D      ;PUT THE PASS COUNT ON THE STACK
5609 023000 004537 027300  JSR    R5,REPLZ     ;CONVERT IT
5610 023004 000003          .WORD    3         ;TYPE IT
5611 023006 104401 052155  TYPE    LINSF         ;TYPE 3 DIGITS
5612 023012 010246          MOV    R2,-(SP)     ;SPACES
5613 023014 004737 032562  JSR    PC,$DB2D      ;PUT $OPERC ON THE STACK
5614 023020 004537 027300  JSR    R5,REPLZ     ;CONVERT IT
5615 023024 000006          .WORD    6         ;TYPE $OPERC
5616 023026 104401 052155  TYPE    LINSF         ;TYPE 6 DIGITS
5617 023032 062702 000004  ADD    #4,R2        ;SPACES
5618 023036 010246          MOV    R2,-(SP)     ;INCREMENT R2
5619 023040 004737 032562  JSR    PC,$DB2D      ;PUT $POSIT ON THE STACK
5620 023044 004537 027300  JSR    R5,REPLZ     ;CONVERT IT
5621 023050 000006          .WORD    6         ;TYPE $POSIT
5622 023052 104401 052155  TYPE    LINSF         ;TYPE 6 DIGITS
5623 023056 062702 000004  ADD    #4,R2        ;SPACES
5624 023062 010246          MOV    R2,-(SP)     ;INCREMENT R2
5625 023064 004737 032562  JSR    PC,$DB2D      ;PUT $STRANS ON THE STACK
5626 023070 004537 027300  JSR    R5,REPLZ     ;CONVERT $STRANS
5627 023074 000012          .WORD    10        ;TYPE IT
5628 023076 104401 052155  TYPE    LINSF         ;TYPE 10 DIGITS
5629 023102 062702 000004  ADD    #4,R2        ;SPACES
                    ;INCREMENT R2

```

```

5630 023106 010246      MOV      R2, -(SP)      ; PUT $READ ON THE STACK
5631 023110 004737 032562 JSR      PC, $S820     ; CONVERT $READ
5632 023114 004537 027300 JSR      R5, REPLZ     ; TYPE IT
5633 023120 000012      .WORD   10           ; TYPE 10 DIGITS
5634 023122 104401 052156 .TYPE   , LINSPO      ; 1 SPACE
5635 023126 062702 000004 ADD      #4, R2        ; INCREMENT R2
5636 023132 062702 000002 ADD      #2, R2        ; INCREMENT R2 AGAIN
5637 023136 012246      MOV      (R2)+, -(SP) ; PUT $SOFT ON THE STACK
5638 023140 004737 027770 JSR      PC, $S820     ; CONVERT $SOFT
5639 023144 004537 027300 JSR      R5, REPLZ     ; TYPEOUT $SOFT
5640 023150 000004      .WORD   4           ; TYPE 4 DIGITS
5641 023152 104401 052156 .TYPE   , LINSPO      ; SPACES
5642 023156 012246      MOV      (R2)+, -(SP) ; PUT $HARD ON THE STACK
5643 023160 004737 027770 JSR      PC, $S820     ; CONVERT $HARD
5644 023164 004537 027300 JSR      R5, REPLZ     ; TYPEOUT $HARD
5645 023170 000004      .WORD   4           ; TYPE 4 DIGITS
5646 023172 104401 052156 .TYPE   , LINSPO      ; SPACES
5647 023176 012246      MOV      (R2)+, -(SP) ; PUT $SKI ON THE STACK
5648 023200 004737 027770 JSR      PC, $S820     ; CONVERT $SKI
5649 023204 004537 027300 JSR      R5, REPLZ     ; TYPEOUT $SKI
5650 023210 000004      .WORD   4           ; TYPE 4 DIGITS
5651 023212 104401 052156 .TYPE   , LINSPO      ; SPACES
5652 023216 012246      MOV      (R2)+, -(SP) ; PUT $MISPO ON THE STACK
5653 023220 004737 027770 JSR      PC, $S820     ; CONVERT $MISPO
5654 023224 004537 027300 JSR      R5, REPLZ     ; TYPEOUT $MISPO
5655 023230 000004      .WORD   4           ; TYPE 4 DIGITS
5656 023232 104401 052156 .TYPE   , LINSPO      ; SPACES
5657 023236 016046 000056 MOV      $TOTAL(R0), -(SP) ; CALCULATE NUMBER OF OTHER ERRORS
5658 023242 166016 000060 SUB      $SOFT(R0), (SP) ; SUBTRACT $SOFT FROM $TOTAL
5659 023246 166016 000062 SUB      $HARD(R0), (SP) ; SUBTRACT $HARD FROM $TOTAL
5660 023252 166016 000064 SUB      $SKI(R0), (SP) ; SUBTRACT $SKI FROM $TOTAL
5661 023256 166016 000066 SUB      $MISPO(R0), (SP) ; SUBTRACT $MISPO FROM $TOTAL
5662 023262 004737 027770 JSR      PC, $S820     ; CONVERT 'OTHER' COUNT
5663 023266 004537 027300 JSR      R5, REPLZ     ; TYPE IT
5664 023272 000004      .WORD   4           ; TYPE 4 DIGITS
5665 023274 104401 001165 .TYPE   , $CRLF
5666 023300 012602      MOV      (SP)+, R2     ; RESTORE R2
5667 023302 000207      RTS      PC
5668
5669
5670
5671      ; ROUTINE TO INCREMENT $SOFT
5672      ;
5673      ; NOTE: $SOFT WILL NOT BE INCREMENTED BEYOND 9999 (10)
5674
5675 023304 005737 001264 INCSOF: TST      BADSEC      ; SEE IF BAD TRK/SEC INDICATOR SET
5676 023310 001006      BNE      IS             ; BR IF IT'S SET, DON'T INCREMENT COUNT
5677 023312 026027 000060 023417 CMP      $SOFT(R0), #9999. ; IS $SOFT ALREADY AT MAXIMUM ?
5678 023320 103002      BHS     IS             ; BR IF IT IS
5679 023322 005260 000060      INC      $SOFT(R0)      ; INCREMENT $SOFT
5680 023326 000207      IS:    RTS      PC      ; RETURN
5681
5682
5683      ; ROUTINE TO INCREMENT $HARD
5684      ;
5685      ; NOTE: $HARD WILL NOT BE INCREMENTED BEYOND 9999 (10)

```

```

5686
5687 023330 005737 001264      INCHRD: TST      BADSEC      ;SEE IF BAD TRK/SEC INDICATOR SET
5688 023334 001006                BNE      IS              ;BR IF IT'S SET, DON'T INCREMENT COUNT
5689 023336 026027 000062 023417  CMP      $SHARD(RO),#9999. ;IS $SHARD ALREADY AT MAXIMUM?
5690 023344 103002                BHIS     IS              ;BR IF IT IS
5691 023346 005260 000062      INC      $SHARD(RO)      ;INCREMENT $SHARD
5692 023352 000207                RTS      PC              ;RETURN
5693
5694
5695 ;ROUTINE TO INCREMENT $SKI
5696
5697 ;NOTE: $SKI WILL NOT BE INCREMENTED BEYOND 9999 (10)
5698
5699 023354 005737 001264      INCSKI: TST      BADSEC      ;SEE IF BAD TRK/SEC INDICATOR SET
5700 023360 001006                BNE      IS              ;BR IF IT'S SET, DON'T INCREMENT COUNT
5701 023362 026027 000064 023417  CMP      $SKI(RO),#9999.  ;IS $SKI ALREADY AT MAXIMUM?
5702 023370 103002                BHIS     IS              ;BR IF IT IS
5703 023372 005260 000064      INC      $SKI(RO)        ;INCREMENT $SKI
5704 023376 000207                RTS      PC              ;RETURN
5705
5706
5707 ;ROUTINE TO INCREMENT $MISPO
5708
5709 ;NOTE: $MISPO WILL NOT BE INCREMENTED BEYOND 9999 (10)
5710
5711 023400 005737 001264      INCMIS: TST      BADSEC      ;SEE IF BAD TRK/SEC INDICATOR SET
5712 023404 001006                BNE      IS              ;BR IF IT'S SET, DON'T INCREMENT COUNT
5713 023406 026027 000066 023417  CMP      $MISPO(RO),#9999. ;IS $MISPO ALREADY AT MAXIMUM?
5714 023414 103002                BHIS     IS              ;BR IF IT IS
5715 023416 005260 000066      INC      $MISPO(RO)     ;INCREMENT $MISPO
5716 023422 000207                RTS      PC              ;RETURN
5717
5718
5719 ;ROUTINE TO INCREMENT $TOTAL
5720
5721 ;NOTE: $TOTAL WILL NOT BE INCREMENTED BEYOND 9999 (10)
5722
5723 023424 005737 001264      INCTOT: TST      BADSEC      ;SEE IF BAD TRK/SEC INDICATOR SET
5724 023430 001006                BNE      IS              ;BR IF IT'S SET, DON'T INCREMENT COUNT
5725 023432 026027 000056 023417  CMP      $TOTAL(RO),#9999. ;IS $TOTAL ALREADY AT MAXIMUM?
5726 023440 103002                BHIS     IS              ;BR IF IT IS
5727 023442 005260 000056      INC      $TOTAL(RO)     ;INCREMENT $TOTAL
5728 023446 000207                RTS      PC              ;RETURN
5729
5730
5731 ;ROUTINE TO TYPE THE TIME
5732
5733 $TIME: TST      CLKFLG      ;CLOCK ON THE SYSTEM?
5734 023454 001033                BNE      IS              ;BR IF NOT
5735 023456 104401 001165      TYPE    $CRLF           ;CR-LF
5736 023462 013746 001266      MOV     HOUR,-(SP)      ;PUT 'HOURS' ON THE STACK
5737 023466 004737 027770      JSR    PC,$SB2D        ;CONVERT TO DECIMAL
5738 023472 004537 027300      JSR    R5,REPLZ       ;TYPE IT
5739 023476 000002                .WORD   2              ;TYPE 2 DIGITS
5740 023500 104401 053207      TYPE    ,COLON         ;:
5741 023504 013746 001270      MOV     MINUTE,-(SP)   ;PUT 'MINUTES' ON THE STACK

```

```

5742 023510 004737 027770 JSR PC,$SB2D ;CONVERT TO DECIMAL
5743 023514 004537 027300 JSR R5,REPLZ ;TYPE IT
5744 023520 000002 .WORD 2 ;TYPE 2 DIGITS
5745 023522 104401 053207 TYPE COLON ;
5746 023526 013746 001272 MOV SECOND, -(SP) ;PUT SECONDS ON THE STACK
5747 023532 004737 027770 JSR PC,$SB2D ;CONVERT TO DECIMAL
5748 023536 004537 027300 JSR R5,REPLZ ;TYPE IT
5749 023542 000002 .WORD 2 ;TYPE 2 DIGITS
5750 023544 000207 1$: RTS PC
5751 ;CLOCK HANDLER ROUTINE
5752
5753
5754 023546 005337 001274 CLOCK: DEC SIXTEE ;INCREMENT THE 1/60 SECOND COUNTER
5755 023552 001035 BNE 1$ ;BR IF A SECOND NOT COUNTED
5756 023554 013737 001212 001274 MOV HZ,SIXTEE ;RESTORE THE VALUE
5757 023562 005237 001272 INC SECOND ;COUNT THE SECOND
5758 023566 022737 000074 001272 CMP #60.,SECOND ;AT MAXIMUM ?
5759 023574 001024 BNE 1$ ;BR IF NOT
5760 023576 005037 001272 CLR SECOND ;CLEAR THE SECOND'S COUNTER
5761 023602 005237 001412 INC INTRVL+2 ;COUNT THE PERFORMANCE SUMMARY INTERVAL
5762 023606 005237 001270 INC MINUTE ;COUNT THE MINUTE
5763 023612 022737 000074 001270 CMP #60.,MINUTE ;AT MAXIMUM ?
5764 023620 001012 BNE 1$ ;BR IF NOT
5765 023622 005037 001270 CLR MINUTE ;CLEAR THE MINUTE'S COUNTER
5766 023626 005237 001266 INC HOUR ;COUNT THE HOURS
5767 023632 022737 001747 001266 CMP #999.,HOUR ;AT MAXIMUM
5768 023640 103002 BHIS 1$ ;BR IF NOT
5769 023642 005037 001266 CLR HOUR ;CLEAR THE HOURS
5770 023646 012746 000021 1$: MOV #17, -(SP) ;17 MS ON THE STACK
5771 023652 004737 037660 JSR PC,RPTMR ;DRIVER TIMER ROUTINE
5772 023656 005737 001410 TST INTRVL ;DISPLAY THE PERFORMANCE SUMMARY ?
5773 023662 001411 BEQ 2$ ;BR IF NOT
5774 023664 023737 001410 001412 CMP INTRVL,INTRVL+2 ;DISPLAY INTERVAL FINISHED ?
5775 023672 001005 BNE 2$ ;BR IF NOT
5776 023674 012737 177777 001214 MOV #-1,STATIN ;SET PERFORMANCE SUMMARY DISPLAY FLAG
5777 023702 005037 001412 CLR INTRVL+2 ;CLEAR THE PERFORMANCE INTERVAL COUNTER
5778 023706 000002 2$: RTI ;RETURN
5779
5780 ;COMMAND DECODE ROUTINE
5781 :CALL:
5782 : MOV #-1,CFLAG ;'CFLAG' IS NORMALLY SET BY THE TTY SERVICE
5783 : ;ROUTINE IN INTERRUPT MODE
5784 :
5785 : JSR PC,KSR
5786 : ;SYSTEM BUSY RETURN
5787 : RETURN1 ;RETURN AFTER KEYBOARD SERVICED
5788 : RETURN2
5789
5788 023710 005737 001440 KSR: TST ORDERQ ;ANY OPERATIONS ACTIVE ?
5789 023714 001003 BNE 1$ ;BR IF SOME ARE
5790 023716 005037 023742 CLR 3$ ;CLEAR THE LOOP COUNTER
5791 023722 000410 BR KSR1 ;PROCESS THE KEYBOARD REQUEST
5792 023724 062737 000001 023742 1$: ADD #1,3$ ;COUNT THE TIMES THROUGH THE LOOP
5793 023732 001002 BNE 2$ ;BR IF NOT ENOUGH
5794 023734 104401 053320 TYPE BUSY ;'SYSTEM BUSY'
5795 023740 000207 2$: RTS PC ;PROCESS ANY COMPLETED DRIVES
5796 023742 000002 3$: .WORD 0 ;LOOP COUNTER
5797 023744 104412 KSR1: SAVREG ;SAVE THE REGISTERS

```

```

5798 023746 012737 000200 177776 MOV #PR4,PS ;SET PRIORITY TO 4
5799 023754 005037 001262 CLR CFLAG ;CLEAR THE 'CONTROL C' FLAG
5800 023760 004737 023450 JSR PC,STIME ;TYPE THE TIME
5801 023764 005777 155156 TST @STKB ;CLEAR ANY GARBAGE IN THE TTY BUFFER
5802 023770 104401 053033 TYPE ,ENTCOM ;'ENTER COMMANDS'
5803 023774 104411 ROLIN ;READ THE KEYBOARD
5804 023776 012605 MOV (SP)+,R5 ;GET ADDRESS OF INPUT STRING
5805 024000 005737 001262 TST CFLAG ;CHECK THE CONTROL C FLAG
5806 024004 001065 BNE 7$ ;EXIT IF 'CONTROL C' ENTERED
5807 024006 005205 INC R5 ;POINT TO SECOND CHARACTER
5808 024010 122715 000101 CMPB #'A',(R5) ;EQ TO AN 'A'
5809 024014 001410 BEQ 1$ ;BR IF IT IS
5810 024016 121527 000067 CMPB (R5),#'7' ;DRIVE NUMBER GREATER THAN AN ASCII 7 ?
5811 024022 101054 BHI 6$ ;BR IF IT IS
5812 024024 121527 000060 CMPB (R5),#'0' ;DRIVE NUMBER LESS THAN AN ASCII 0 ?
5813 024030 103451 BLO 6$ ;BR IF IT IS
5814 024032 142715 177770 BICB #1C7,(R5) ;LEAVE ONLY LOWER 3 BITS IF CHAR NOT 'A'
5815 024036 122765 000124 177777 1$: CMPB #'T',-1(R5) ;EQ TO 'T'
5816 024044 001003 BNE 2$ ;BR IF NOT EQ
5817 024046 004737 024552 JSR PC,NEWASN ;ASSIGN DRIVE FOR TEST
5818 024052 000442 BR 7$ ;EXIT
5819 024054 122765 000104 177777 2$: CMPB #'D',-1(R5) ;EQ TO 'D' ?
5820 024062 001003 BNE 3$ ;BR IF NOT EQ
5821 024064 004737 024562 JSR PC,DEASGN ;DEASSIGN DRIVE
5822 024070 000433 BR 7$ ;EXIT
5823 024072 122765 000123 177777 3$: CMPB #'S',-1(R5) ;EQ TO 'S'
5824 024100 001003 BNE 4$ ;BR IF NOT EQ
5825 024102 004737 024670 JSR PC,SCMND ;TYPE STATISTICS
5826 024106 000424 BR 7$ ;EXIT
5827 024110 122765 000127 177777 4$: CMPB #'W',-1(R5) ;EQ TO 'W'
5828 024116 001007 BNE 5$ ;BR IF NOT EQ
5829 024120 032777 000001 155012 BIT #SWD,@SWR ;IS SWITCH 0 SET ?
5830 024126 001012 BNE 6$ ;BR IF SET, CAN'T DO 'W' COMMAND
5831 024130 004737 025140 JSR PC,DATAPK ;WRITE A DATA PACK
5832 024134 000411 BR 7$ ;EXIT
5833 024136 122765 000122 177777 5$: CMPB #'R',-1(R5) ;EQ TO 'R' ?
5834 024144 001003 BNE 6$ ;BR IF NOT EQ
5835 024146 004737 025152 JSR PC,REDAPK ;READ A DATA PACK
5836 024152 000402 BR 7$ ;EXIT
5837 024154 104401 053011 6$: TYPE ,INVLD ;TYPE 'INVALID COMMAND' MESSAGE
5838 024160 104413 7$: RESREG ;RESTORE R0 - R5
5839 024162 062716 000002 ADD #2,(SP) ;INCREMENT THE RETURN ADDRESS
5840 024166 005777 154754 TST @STKB ;CLEAR THE TTY BUFFER
5841 024172 052777 000100 154744 BIS #BIT06,@STKS ;SET TTY INTERRUPT ENABLE
5842 024200 005037 177776 CLR PS ;SET PRIORITY BACK TO ZERO
5843 024204 000207 RTS PC ;RETURN
5844
5845 ;ROUTINE TO PROCESS THE ASSIGN REQUEST ('T', 'R', OR 'W' COMMANDS)
5846
5847 024206 122715 000101 ASSIGN: CMPB #'A',(R5) ;ASSIGN ALL DRIVES?
5848 024212 001430 BEQ ASGN2 ;BR IF ALL DRIVES
5849 024214 111504 ASGN1: MOVB (R5),R4 ;PUT DRIVE # IN R4
5850 024216 012737 052231 026436 MOV #UNTA5N,ASNMSG ;'DRIVE ASSIGNED' MESSAGE ADDRESS
5851 024224 012703 000001 MOV #1,R3 ;RELOAD R3 FOR 1 UNIT
5852 024230 136437 033320 001462 BITB ATABIT(R4),ASNLS ;DRIVE ALREADY ASSIGNED ?
5853 024236 001013 BNE 2$ ;BR IF IT IS

```

5854	024240	005704			TST	R4	: TRYING TO ASSIGN DRIVE 0 ?
5855	024242	001007			BNE	1\$: BR IF NOT
5856	024244	012737	052315	026436	MOV	#NOTAVL,ASNMSG	: 'NOT AVAILABLE' MESSAGE ADDRESS
5857	024252	122737	000011	000041	CMPB	#11,41	: SEE IF LOADED FROM AN RPO4/5/6
5858	024260	001402			BEQ	2\$: BR IF RPO4/5/6 IS THE LOAD DEVICE
5859	024262	004737	024346		1\$: JSR	PC,ASGN3	: SEE IF DRIVE ON THE SYSTEM
5860	024266	000137	026416		2\$: JMP	ASNERR	: RETURN HERE IF DRIVE NOT AVAIL
5861	024272	000207			RTS	PC	: EXIT
5862	024274	122737	000011	000041	ASGN2: CMPB	#11,41	: LOADED FROM AN RPO4/5/6 ?
5863	024302	001005			BNE	1\$: BR IF NOT
5864	024304	012704	000001		MOV	#1,R4	: START WITH DRIVE 1
5865	024310	012703	000007		MOV	#7.,R3	: SETUP FOR ONLY 7 DRIVES
5866	024314	000403			BR	2\$: CONTINUE
5867	024316	005004			1\$: CLR	R4	: START WITH DRIVE 0
5868	024320	012703	000010		MOV	#8.,R3	: DRIVE COUNT FOR 8 DRIVES
5869	024324	004737	024346		2\$: JSR	PC,ASGN3	: ASSIGN ALL UNASSIGNED, AVAIL DRIVES
5870	024330	000137	024336		3\$: JMP	4\$: DRIVE NOT ON SYSTEM
5871	024334	000207			RTS	PC	: RETURN
5872	024336	012746	024330		4\$: MOV	#3\$,-(SP)	: PUT RETURN ADDRESS ON THE STACK
5873	024342	000137	024464		JMP	ASGN4	: LOOK FOR MORE DRIVES
5874	024346	136437	033320	001462	ASGN3: BITB	ATABIT(R4),ASNLST	: DRIVE ALREADY ASSIGNED ?
5875	024354	001043			BNE	ASGN4	: BR IF IT IS
5876	024356	005737	033316		1\$: TST	DTUW	: DATA TRANSFER UNDER WAY ?
5877	024362	100375			BPL	1\$: BR IF IT IS
5878	024364	110437	044730		MOV	R4,GENDPB	: DRIVE NUMBER
5879	024370	004737	015244		JSR	PC,RECALO	: RECALIBRATE DRIVE
5880	024374	105764	033204		TSTB	DRVSTA(R4)	: DRIVE AVAILABLE?
5881	024400	001444			BEQ	ASGN7	: BR IF DRIVE OFFLINE OR NONEXISTENT
5882	024402	100437			BMI	ASGN6	: BR IF DRIVE UNSAFE
5883	024404	006304			ASL	R4	: MAKE R4 INTO WORD INDEX
5884	024406	016464	001740	001506	MOV	BLKADR(R4),NEWUNT	: (R4) ; DPB ADDRESS
5885	024414	016400	001740		MOV	BLKADR(R4),RO	: PUT BLOCK'S ADDR INTO RO
5886	024420	004737	025164		JSR	PC,CLRDPB	: CLEAR BLOCK FOR DRIVE JUST ASSIGNED
5887	024424	004737	025360		JSR	PC,DRVPRM	: GET THE DRIVE'S ADDRESS LIMITS
5888	024430	004737	025642		JSR	PC,GETID	: GET DRIVE I.D.
5889	024434	004737	025752		JSR	PC,GETADR	: GET BAD SECTOR ADDRESSES
5890	024440	012760	000001	000070	MOV	#1,\$PASSC(RO)	: PRESET PASS COUNT TO 1
5891	024446	005737	001216		TST	PACK	: WRITE DATA PACK ?
5892	024452	001403			BEQ	2\$: BR IF NOT
5893	024454	113760	001216	000026	MOV	PACK,\$PACK(RO)	: SET READ/WRITE DATA PACK INDICATOR
5894	024462	006204			2\$: ASR	R4	: RESTORE DRIVE ADDRESS
5895	024464	005303			ASGN4: DEC	R3	: DECREMENT DRIVE COUNT
5896	024466	001402			BEQ	ASGN5	: BR IF FINISHED
5897	024470	005204			INC	R4	: INCREMENT DRIVE NUMBER
5898	024472	000725			BR	ASGN3	: CONTINUE
5899	024474	062716	000004		ASGN5: ADD	#4,(SP)	: INCREMENT RETURN
5900	024500	000207			RTS	PC	: RETURN
5901	024502	012737	052334	025436	ASGN6: MOV	#NOTSAF,ASNMSG	: 'UNSAFE' MESSAGE ADDRESS
5902	024510	000207			RTS	PC	: RETURN
5903	024512	105764	033214		ASGN7: TSTB	DRVSTP(R4)	: DRIVE PRESENT?
5904	024516	001405			BEQ	1\$: BR IF NOT
5905	024520	100010			BPL	2\$: BR IF DRIVE OFFLINE
5906	024522	012737	052257	026436	MOV	#NOTRP,ASNMSG	: ADDRESS OF 'NOT RPO4/5/6' MSG
5907	024530	000407			BR	3\$: EXIT
5908	024532	012737	052300	026436	1\$: MOV	#NOTPRS,ASNMSG	: ADDRESS OF 'NOT PRESENT' MSG
5909	024540	000403			BR	3\$: EXIT

```

5910 024542 012737 052166 026436 2$: MOV #UNTOFF,ASNMSG ;ADDRESS OF 'DRIVE OFFLINE' MESSAGE
5911 024550 000207 3$: RTS PC ;ERROR RETURN
5912
5913 ;'T' COMMAND (ROUTINE TO ASSIGN A DRIVE)
5914
5915 024552 005037 001216 NEWASN: CLR PACK ;CLEAR 'W' COMMAND INDICATOR
5916 024556 000137 024206 JMP ASSIGN ;GO TO THE ASSIGN ROUTINE
5917
5918 ;'D' COMMAND (ROUTINE TO DEASSIGN A DRIVE)
5919
5920 024562 005004 DEASGN: CLR R4
5921 024564 122715 000101 CMPB #'A,(R5) ;DEASSIGN ALL DRIVES ?
5922 024570 001434 BEQ 5$ ;BR IF YES
5923 024572 012703 000001 MOV #BIT00,R3 ;SET R3 FOR ONE UNIT
5924 024576 111504 MOVB (R5),R4 ;GET DRIVE NUMBER
5925 024600 136437 033320 001462 1$: BITB ATABIT(R4),ASNLS† ;DRIVE ASSIGNED ?
5926 024606 001414 BEQ 3$ ;BR IF NOT
5927 024610 146437 033320 001462 BICB ATABIT(R4),ASNLS† ;DELETE THE DRIVE FROM THE ASSIGNED LIST
5928 024616 006304 ASL R4 ;MAKE ADDR INTO A WORD INDEX
5929 024620 016464 001740 001464 MOV BLKADR(R4),DUNIT(R4) ;PUT ADDRESS IN DEASSIGN LIST
5930 024626 006204 ASR R4
5931 024630 005303 2$: DEC R3 ;ANY MORE DRIVES ?
5932 024632 001412 BEQ 4$ ;BR IF NOT
5933 024634 005204 INC R4
5934 024636 000760 BR 1$
5935 024640 122715 000101 3$: CMPB #'A,(R5) ;DEASSIGN ALL DRIVES ?
5936 024644 001771 BEQ 2$ ;BR IF YES
5937 024646 012737 052207 026436 MOV #UNTNOT,ASNMSG ;ADDR OF 'NOT ASSIGNED' MESSAGE
5938 024654 004737 026416 JSR PC,ASNEAR ;REPORT IT
5939 024660 000207 4$: RTS PC
5940 024662 012703 000010 5$: MOV #8.,R3 ;SET UNIT COUNT TO 8
5941 024666 000744 BR 1$
5942
5943 ;'S' COMMAND (ROUTINE TO TYPE DRIVE PERFORMANCE SUMMARY)
5944
5945 024670 005004 SCMND: CLR R4
5946 024672 122715 000101 CMPB #'A,(R5) ;ALL STATISTICS ?
5947 024676 001421 BEQ 2$ ;BR IF YES
5948 024700 111504 MOVB (R5),R4 ;GET DRIVE NUMBER
5949 024702 136437 033320 001462 BITB ATABIT(R4),ASNLS† ;SEE IF DRIVE ASSIGNED
5950 024710 001406 BEQ 1$ ;BR IF NOT
5951 024712 006304 ASL R4 ;MAKE DRIVE ADDR INTO WORD INDEX
5952 024714 016400 001740 MOV BLKADR(R4),R0 ;ADDR OF BLOCK
5953 024720 004737 022676 JSR PC,TYPEST ;TYPE DRIVE STATISTICS
5954 024724 000504 BR 9$ ;EXIT
5955 024726 012737 052207 026436 1$: MOV #UNTNOT,ASNMSG ;ADDR OF 'NOT ASSIGNED' MSG
5956 024734 004737 026416 JSR PC,ASNEAR ;TYPE ERROR MESSAGE
5957 024740 000476 BR 9$ ;EXIT
5958 024742 012703 000010 2$: MOV #8.,R3 ;DRIVE COUNT
5959 024746 136437 033320 001462 3$: BITB ATABIT(R4),ASNLS† ;SEE IF DRIVE ASSIGNED
5960 024754 001004 BNE 4$ ;BR IF YES
5961 024756 005204 INC R4 ;INCREMENT DRIVE ADDRESS
5962 024760 005303 DEC R3 ;DECREMENT COUNTER
5963 024762 001371 BNE 3$ ;MORE TO CHECK
5964 024764 000464 BR 9$ ;NONE ASSINGED, RETURN
5965 024766 004737 022676 4$: JSR PC,STATPR ;TYPE ALL STATISTICS

```

```

5966 024772 105737 001220      TSTB   DATE           ;SEE IF 'DATE' ENTERED
5967 024776 001404      BEQ    11$           ;BR IF NOT
5968 025000 104401 053211      TYPE  ,DATEIS       ;'DATE:'
5969 025004 104401 001220      TYPE  ,DATE         ;THE OPERATOR ENTERED DATE
5970 025010 105737 001232      11$:  TSTB   OPERID    ;SEE IF OPERATOR I.D. ENTERED
5971 025014 001404      BEQ    12$           ;BR IF NOT
5972 025016 104401 053222      TYPE  ,IDIS         ;'OPERATOR I.D.:'
5973 025022 104401 001232      TYPE  ,OPERID       ;THE OPERATOR I.D.
5974 025026 104401 053244      TYPE  ,HEDLIN       ;HEADER LINE
5975 025032 012737 042114      025106 MOV  #DRIVED+$ORVID,6$ ;DRIVE I.D. FIELD ADDRESS
5976 025040 005004      CLR   R4            ;DRIVE ADDRESS
5977 025042 012703 000010      MOV  #8,R3         ;COUNTER
5978 025046 136437 033320      001462 5$:  BITB   ATABIT(R4),ASNL$ ;SEE IF DRIVE ASSIGNED
5979 025054 001417      BEQ    7$           ;BR IF NOT ASSIGNED
5980 025056 010446      MOV  R4,-(SP)      ;SAVE R4 FOR TYPEOUT
5981                                ;TYPE DRIVE NUMBER
5982 025060 104403      TYPOS ;GO TYPE--OCTAL ASCII
5983 025062 002      .BYTE 2          ;TYPE 2 DIGIT(S)
5984 025063 000      .BYTE 0          ;SUPPRESS LEADING ZEROS
5985 025064 104401 052153      TYPE  ,LIN4SP      ;4 SPACES
5986 025070 105777 000012      TSTB   26$         ;SEE IF DRIVE I.D. ENTERED
5987 025074 001003      BNE   10$          ;BR IF DRIVE I.D. PRESENT
5988 025076 104401 053267      TYPE  ,NONE        ;TYPE 'NONE'
5989 025102 000404      BR    7$           ;CONTINUE
5990 025104 104401      10$:  TYPE  ,THE DRIVE I.D. ;TYPE THE DRIVE I.D.
5991 025106 000000      6$:  .WORD 0        ;ADDRESS OF DRIVE I.D. FIELD HERE
5992 025110 104401 001165      TYPE  ,$CRLF       ;CR-LF
5993 025114 005303      7$:  DEC   R3         ;DECREMENT THE COUNTER
5994 025116 001405      BEQ    8$           ;BR IF AT END
5995 025120 062737 000304      025106 ADD  #SRPEC2+2,6$ ;INCREMENT THE MESSAGE FIELD ADDRESS
5996 025126 005204      INC   R4            ;INCREMENT DRIVE ADDRESS
5997 025130 000746      BR    5$           ;CONTINUE
5998 025132 104401 001165      8$:  TYPE  , $CRLF   ;CR-LF
5999 025136 000207      9$:  RTS   PC        ;
6000                                ;
6001                                ;'W' COMMAND (ROUTINE TO WRITE A DATA PACK)
6002                                ;
6003 025140 012737 177777 001216  DATAPK: MOV  #-1,PACK ;SET THE 'W' COMMAND INDICATOR
6004 025146 000137 024206      JMP   ASSIGN        ;ASSIGN REQUESTED DRIVE
6005                                ;
6006                                ;'R' COMMAND (ROUTINE TO READ A DATA PACK)
6007                                ;
6008                                ;
6009 025152 012737 000001 001216  REDAPK: MOV  #1,PACK ;SET THE 'READ' INDICATOR
6010 025160 000137 024206      JMP   ASSIGN        ;ASSIGN THE REQUESTED DRIVE
6011                                ;
6012                                ;ROUTINE TO CLEAR THE DPB FOR THE ASSIGNED DRIVE
6013                                ;CALL:
6014                                ;
6015                                ;
6016                                ;
6017                                ;
6018                                ;
6019 025164 010346      CLDPB: MOV  R3,-(SP) ;:PUSH R3 ON STACK
6020 025166 010446      MOV  R4,-(SP) ;:PUSH R4 ON STACK
6021 025170 010546      MOV  R5,-(SP) ;:PUSH R5 ON STACK

```



```

6022 025172 010004      MOV      R0,R4      ;GET THE DPB ADDRESS
6023 025174 062704 000002  ADD      #2,R4      ;ADDRESS OF FIRST LOCN TO BE CLEARED
6024 025200 012703 000005  MOV      #5,R3      ;NUMBER OF LOCNS TO BE CLEARED
6025          1$: CLR      (R4)+      ;CLEAR THE LOCATION
6026 025206 005303      DEC      R3          ;DECREMENT THE COUNTER
6027 025210 001375      BNE     1$          ;BR IF NOT FINISHED
6028 025212 062704 000002  ADD      #2,R4      ;MOVE THE ADDRESS PAST THE 'REG' ADDR
6029 025216 012703 000070  MOV      #NEXT-$REG,R3 ;NUMBER OF LOCNS TO BE CLEARED
6030 025222 005024      CLR      (R4)+      ;CLEAR
6031 025224 162703 000002  SUB      #2,R3      ;DECREMENT THE LOCN COUNTER
6032 025230 001374      BNE     2$          ;BR IF NOT FINISHED
6033 025232 062704 000014  ADD      #12,R4     ;MOVE PAST ADDRESS LIMITS
6034 025236 012703 000162  MOV      #SPEC2-MINSEC,R3 ;NUMBER OF LOCNS TO BE CLEARED
6035 025242 005024      CLR      (R4)+      ;CLEAR A LOCATION
6036 025244 162703 000002  SUB      #2,R3      ;DECREMENT THE COUNTER
6037 025250 001374      BNE     3$          ;BR IF NOT DONE
6038 025252 113760 001434 000024  MOVB    BEGCD,$CODE(R0) ;INITIAL COMMAND CODE
6039 025260 013701 001434      MOV      BEGCD,R1    ;GET THE ACTUAL OP CODE
6040 025264 116160 001760 000002  MOVB    COMBL(R1),$COMND(R0) ;OPERATION CODE
6041 025272 113760 001432 000030  MOVB    BEGPAT,$PATTC(R0) ;PATTERN CODE
6042 025300 106360 000030      ASLB    $PATTC(R0)   ;CONVERT CODE TO A TABLE INDEX
6043 025304 013760 001436 000020  MOV      BEGSIZ,$WRDL(R0) ;BEGINNING RECORD SIZE
6044 025312 013760 001436 000004  MOV      BEGSIZ,$WRDM(R0) ;VALUE FOR DATA TRANSFER
6045 025320 005460 000004      NEG     $WRDM(R0)   ;MAKE IT INTO 2'S COMPLEMENT
6046 025324 012760 000400 000022  MOV      #256,$SSEC(R0) ;INITIAL VALUE OF SECTOR SIZE
6047 025332 132760 000001 000024  BITB    #1,$CODE(R0) ;HEADER ORDER ?
6048 025340 001403      BEQ     4$          ;BR IF NOT
6049 025342 062760 000004 000022  ADD      #4,$SSEC(R0) ;ADD HEADER SIZE TO SECTOR SIZE
6050 025350          4$:
6051 025350 012605      MOV      (SP)+,R5   ;POP STACK INTO R5
6052 025352 012604      MOV      (SP)+,R4   ;POP STACK INTO R4
6053 025354 012603      MOV      (SP)+,R3   ;POP STACK INTO R3
6054 025356 000207      RTS     PC          ;RETURN
6055
6056          ;ROUTINE TO GET ADDRESS LIMITS FROM THE OPERATOR
6057
6058 025360 010346      DRVPRM: MOV      R3,-(SP)   ;SAVE R3
6059 025362 010446      MOV      R4,-(SP)   ;SAVE R4
6060 025364 005737 000042      TST     42          ;RUNNING UNDER MONITOR CONTROL
6061 025370 001035      BNE     3$          ;BR IF YES
6062 025372 104401 053105      TYPE    'ENTLMT'    ;'ENTER ADDRESSES'
6063 025376 006204      ASR     R4          ;CONVERT INDEX TO DRIVE NUMBER
6064 025400 010446      MOV      R4,-(SP)   ;SAVE R4 FOR TYPEOUT
6065          ;TYPE DRIVE NUMBER
6066 025402 104403      TYPOS   .BYTE      ;GO TYPE--OCTAL ASCII
6067 025404          .BYTE      ;TYPE 2 DIGIT(S)
6068 025405          .BYTE      0      ;SUPPRESS LEADING ZEROS
6069 025406 104401 053576      TYPE    'SLASH'     ; '/'
6070 025412 012737 052364 025456      MOV      #RPO4B,2$   ;ADDRESS OF 'RPO4' MESSAGE
6071 025420 132764 000001 033214      BITB    #BIT00,DRV TYP(R4) ;RPO4 ?
6072 025426 001012      BNE     1$          ;BR IF YES
6073 025430 012737 052371 025456      MOV      #RPO5,2$   ;ADDRESS OF 'RPO5' MESSAGE
6074 025436 132764 000002 033214      BITB    #BIT01,DRV TYP(R4) ;RPO5 ?
6075 025444 001003      BNE     1$          ;BR IF YES
6076 025446 012737 052376 025456      MOV      #RPO6,2$   ;ADDRESS OF 'RPO6' MESSAGE
6077 025454 104401 1$: TYPE    ;TYPE THE MESSAGE WHICH FOLLOWS

```

```

6078 025456 000000 2$: WORD 0 ;MESSAGE ADDRESS
E:07 025460 104401 ;SCALF ;CR-LF
6080 025464 012737 000632 001350 3$: MOV #410,CYLIMT ;ASSUME AN MPU4/5
6081 025472 132764 000004 033214 BITB #BIT02,DRVTP(R4) ;SEE IF RPO6
6082 025500 001403 BEQ 4$ ;BR IF NOT
6083 025502 012737 001456 001350 MOV #814,CYLIMT ;CHANGE LIMIT TO 814
6084 025510 062760 177777 000122 4$: ADD #1,$FIRST(R0) ;SEE IF FIRST TIME STARTED
6085 025516 103417 BCS 5$ ;BR IF NOT
6086 025520 013760 001350 000106 MOV CYLIMT,MAXCYL(R0) ;LOAD MAXIMUM CYLINDER
6087 025526 005060 000110 CLR MINCYL(R0) ;CLEAR MINIMUM CYLINDER
6088 025532 013760 001346 000112 MOV TRKLMT,MAXTRK(R0) ;LOAD MAXIMUM TRACK
6089 025540 005060 000114 CLR MINTRK(R0) ;CLEAR MINIMUM TRACK
6090 025544 013760 001344 000116 MOV SECLMT,MAXSEC(R0) ;LOAD MAXIMUM SECTOR
6091 025552 005060 000120 CLR MINSEC(R0) ;CLEAR MINIMUM SECTOR
6092 025556 006304 5$: ASL R4 ;SETUP TO ADDRESS WORDS
6093 025560 016403 053756 MOV TABLE(R4),R3 ;PARAMETER TABLE ADDRESS
6094 025564 013763 001350 000002 MOV CYLIMT,2(R3) ;LOAD CYLINDER LIMIT FOR LAST CYLINDER
6095 025572 013763 001350 000010 MOV CYLIMT,10(R3) ;LOAD CYLINDER LIMIT FOR STARTING CYLINDER
6096 025600 005737 000042 TST 42 ;UNDER MONITOR CONTROL ?
6097 025604 001002 BNE 6$ ;BR IF YES
6098 025606 004737 026272 JSR PC,PARENT ;GET THE DRIVE'S PARAMETERS
6099 025612 116060 000120 000010 6$: MOVB MINSEC(R0),$SEC(R0) ;INITIAL SECTOR VALUE
6100 025620 116060 000114 000011 MOVB MINTRK(R0),$TRK(R0) ;INITIAL TRACK VALUE
6101 025626 016060 000110 000012 MOVB MINCYL(R0),$CYL(R0) ;INITIAL CYLINDER VALUE
6102 025634 012604 MOV (SP)+,R4 ;RESTORE R4
6103 025636 012603 MOV (SP)+,R3 ;RESTORE R3
E:04 025640 000207 RTS PC ;RETURN
6105
6106 ;ROUTINE TO GET THE DRIVE I.D. FROM THE OPERATOR
6107
6108 025642 010546 GETID: MOV R5,-(SP) ;SAVE R5
6109 025644 005737 000042 TST 42 ;UNDER MONITOR CONTROL ?
6110 025650 001036 BNE 2$ ;BR IF NOT
6111 025652 005037 001262 1$: CLR CFLAG ;CLEAR THE 'CONTROL C' FLAG
E:11 025656 104401 053060 TYPE ,ENTDRV ;'ENTER DRV I.D.:'
6112 025662 005046 CLR -(SP) ;CLEAR THE STACK
6113 025664 111016 MOVB (R0),(SP) ;PUT THE DRIVE NUMBER ON THE STACK
6114 025666 104403 TYPOS ;TYPE THE DRIVE NUMBER
6115 025670 002 .BYTE 2 ;TYPE 2 DIGITS
6116 025671 000 .BYTE 0 ;SUPPRESS LEADING ZEROS
6117 025672 104401 001165 TYPE ,SCALF ;CR-LF
6118 025676 104411 RDLIN ;READ THE ENTRY
6119 025700 012605 MOV (SP)+,R5 ;GET THE ENTRY ADDRESS
6120 025702 005737 001262 TST CFLAG ;'CONTROL C' ENTERED ?
6121 025706 001361 BNE 1$ ;BR IF IT WAS
6122 025710 121527 000056 CMPB (R5),#'. ;PERIOD ENTERED ?
E:12 025714 001414 SEG 2$ ;BR IF YES
6123 025716 112560 000224 MOVB (R5)+,$DRVID(R0) ;STORE THE I.D.
6124 025722 112560 000225 MOVB (R5)+,$DRVID+1(R0) ;STORE THE I.D.
6125 025726 112560 000226 MOVB (R5)+,$DRVID+2(R0) ;STORE THE I.D.
6126 025732 112560 000227 MOVB (R5)+,$DRVID+3(R0) ;STORE THE I.D.
6127 025736 112560 000230 MOVB (R5)+,$DRVID+4(R0) ;STORE THE I.D.
6128 025742 112560 000231 MOVB (R5)+,$DRVID+5(R0) ;STORE THE I.D.
E:13 025746 012605 2$: MOV (SP)+,R5 ;RESTORE R5
6130 025750 000207 RTS PC ;RETURN
6131
6132
6133

```

```

6134 ;ROUTINE TO GET THE ADDRESSES OF ANY BAD SECTORS (UP TO A MAX OF 16)
6135
6136 GETADR:
6137     MOV     R1,-(SP)      ;; PUSH R1 ON STACK
6138     MOV     R2,-(SP)      ;; PUSH R2 ON STACK
6139     MOV     R3,-(SP)      ;; PUSH R3 ON STACK
6140     MOV     R4,-(SP)      ;; PUSH R4 ON STACK
6141     TST     42           ;; UNDER MONITOR CONTROL ?
6142     BNE     15$          ;; BR IF YES
6143     CLR     CFLAG        ;; CLEAR 'CONTROL C' FLAG
6144     TYPE    ,ENTADR      ;; ENTER SECTOR ADDRESSES
6145     CLR     -(SP)        ;; CLEAR THE STACK
6146     MOV     MOV     (R0),(SP) ;; PUT THE DRIVE NUMBER ON THE STACK
6147     TYPOS   TYPE        ;; TYPE THE DRIVE NUMBER
6148     .BYTE   2           ;; TYPE 2 DIGITS
6149     .BYTE   0           ;; SUPPRESS LEADING ZEROS
6150     TYPE    ,SCLRF      ;; CR-LF
6151     MOV     #32,R3       ;; NUMBER OF LOCATIONS IN THE TABLE TO PRESET
6152     MOV     #SBDSEC,R4  ;; TABLE INCREMENT
6153     ADD     R0,R4        ;; BLOCK STARTING ADDRESS
6154     MOV     #-1,(R4)+   ;; SET LOCATION TO 1'S
6155     DEC     R3           ;; DECREMENT TABLE SIZE COUNT
6156     BNE     1$          ;; BR IF NOT FINISHED WITH TABLE
6157     TST     42           ;; UNDER MONITOR CONTROL ?
6158     BNE     13$        ;; BR IF YES
6159     SUB     #64,R4       ;; SET POINTER TO BEGINNING OF TABLE
6160     MOV     #16,R3       ;; NUMBER OF ADDRESSES IN TABLE
6161     RDLIN  2$:          ;; GET ADDRESS FROM OPERATOR
6162     MOV     (SP)+,R1     ;; TEXT POINTER
6163     TST     CFLAG        ;; 'CONTROL C' ENTERED ?
6164     BNE     14$        ;; BR IF IT WAS
6165     MOV     CYLIMT,R2    ;; UPPER LIMIT OF INPUT
6166     JSR    R5,CK.DIG    ;; CHECK THE DIGIT(S)
6167     12$      ;; CARRIAGE RETURN ONLY ENTERED
6168     13$      ;; PERIOD ONLY ENTERED
6169     12$      ;; ILLEGAL INPUT
6170     4$       ;; TERMINATED WITH A CARRIAGE RETURN
6171     5$       ;; TERMINATED WITH A ""
6172     3$       ;; TERMINATED WITH A "."
6173     MOV     R2,(R4)     ;; CYLINDER ADDRESS
6174     BR     13$        ;; EXIT, PERIOD ENTERED
6175     MOV     R2,(R4)     ;; CYLINDER ADDRESS
6176     BR     11$        ;; FINISHED WITH THIS ADDRESS, 'CR' ENTERED
6177     MOV     R2,(R4)     ;; CYLINDER ADDRESS
6178     MOV     TRKLMT,R2   ;; UPPER LIMIT OF INPUT
6179     JSR    R5,CK.DIG    ;; CHECK THE DIGIT(S)
6180     12$      ;; CARRIAGE RETURN ONLY ENTERED
6181     13$      ;; PERIOD ONLY ENTERED
6182     12$      ;; ILLEGAL INPUT
6183     7$       ;; TERMINATED WITH A CARRIAGE RETURN
6184     8$       ;; TERMINATED WITH A ""
6185     6$       ;; TERMINATED WITH A "."
6186     MOV     R2,3(R4)    ;; TRACK ADDRESS
6187     BR     13$        ;; EXIT, ENTRY TERMINATED BY PERIOD
6188     MOV     R2,3(R4)    ;; TRACK ADDRESS
6189     BR     11$        ;; FINISHED WITH THIS ADDRESS, 'CR' ENTERED

```

6190	026164	110264	000003	8\$:	MOVB	R2,3(R4)	: TRACK ADDRESS
6191	026170	013702	001344		MOV	SECLMT,R2	: UPPER LIMIT OF INPUT
6192	026174	004537	027632		JSR	RS,CK.DIG	: CHECK THE DIGIT(S)
6193	026200	026240			12\$: CARRIAGE RETURN ONLY ENTERED
6194	026202	026260			13\$: PERIOD ONLY ENTERED
6195	026204	026240			12\$: ILLEGAL INPUT
6196	026206	026222			10\$: TERMINATED WITH A CARRIAGE RETURN
6197	026210	026240			12\$: TERMINATED WITH A "."
6198	026212	026214			9\$: TERMINATED WITH A " "
6199	026214	110264	000002	9\$:	MOVB	R2,2(R4)	: SECTOR ADDRESS
6200	026220	000417			BR	13\$: EXIT ENTRY TERMINATED BY PERIOD
6201	026222	110264	000002	10\$:	MOVB	R2,2(R4)	: SECTOR ADDRESS
6202	026226	005303		11\$:	DEC	R3	: MORE ENTRIES ?
6203	026230	001413			BEG	13\$: BR IF NOT
6204	026232	062704	000004		ADD	#4,R4	: INCREMENT THE TABLE POINTER
6205	026236	000706			BR	2\$: CONTINUE
6206	026240	012714	177777	12\$:	MOV	#-1,(R4)	: CLEAR PRESENT TABLE ENTRY
6207	026244	012764	177777	000002	MOV	#-1,2(R4)	: CLEAR PRESENT TABLE ENTRY
6208	026252	104401	053276		TYPE	BADENT	: 'INVALID ENTRY'
6209	026256	000676			BR	2\$: TRY AGAIN
6210	026260			13\$:			
6211	026260	012604			MOV	(SP)+,R4	: POP STACK INTO R4
6212	026262	012603			MOV	(SP)+,R3	: POP STACK INTO R3
6213	026264	012602			MOV	(SP)+,R2	: POP STACK INTO R2
6214	026266	012601			MOV	(SP)+,R1	: POP STACK INTO R1
6215	026270	000207			RTS	PC	: RETURN
6216							
6217							
6218							
6219							
6220							
6221							
6222	026272	010346					
6223	026274	005037	001262				
6224	026300	012337	026310				
6225	026304	001442		1\$:	MOV	(R3)+,3\$: ADDRESS OF PARAMETER NAME
6226	026306	104401			BEG	9\$: BR IF AT END OF TABLE
6227	026310	000000		3\$:	.WORD	0	: TYPE THE PARAMETER NAME
6228	026312	012302			MOV	(R3)+,R2	: ADDRESS OF PARAMETER NAME TEXT
6229	026314	012305			MOV	(R3)+,R2	: MAXIMUM PARAMETER VALUE
6230	026316	011546			MOV	(R5),-(SP)	: ADDRESS OF PARAMETER
6231	026320	104405			TYPDS		: CURRENT VALUE OF PARAMETER
6232	026322	104401	053576		TYPE	.SLASH	: TYPE THE CURRENT VALUE OF THE PARAMETER
6233	026326	104411			RCL IN		: /
6234	026330	012601			MOV	(SP)+,R1	: READ THE KEYBOARD
6235	026332	005737	001262		TST	CFLAG	: INPUT ASCII STRING ADDRESS
6236	026336	001021			BNE	2\$: 'CONTROL C' ENTERED
6237	026340	004537	027632		JSR	RS,CK.DIG	: BR IF IT WAS
6238	026344	026300			1\$: CHECK THE DIGIT(S)
6239	026346	026412			9\$: CARRIAGE RETURN ONLY ENTERED
6240	026350	026364			6\$: PERIOD ONLY ENTERED
6241	026352	026360			5\$: ILLEGAL INPUT
6242	026354	026364			6\$: TERMINATED WITH A CARRIAGE RETURN
6243	026356	026376			7\$: TERMINATED WITH A "."
6244	026360	010215		5\$:	MOV	R2,(R5)	: TERMINATED WITH A " "
6245	026362	000746			BR	1\$: MOVE NEW VALUE TO PARAMETER LOCATION
							: GET MORE PARAMETERS

:PARAMETER ENTRY ROUTINE

:CALL
: MOV #ADR,R3
: JSR PC,PARENT

PARENT: MOV R3,-(SP)
CLR CFLAG
1\$: MOV (R3)+,3\$
BEG 9\$
TYPE
3\$: .WORD 0
MOV (R3)+,R2
MOV (R3)+,R2
MOV (R5),-(SP)

TYPDS
TYPE .SLASH
RCL IN
MOV (SP)+,R1
TST CFLAG
2\$
BNE 2\$
JSR RS,CK.DIG
1\$
9\$
6\$
5\$
6\$
7\$
5\$: MOV R2,(R5)
BR 1\$

:PARAMETER TABLE ADDRESS
:GET THE PARAMETERS
:SAVE THE PARAMETER TABLE ADDRESS
:CLEAR THE 'CONTROL C' FLAG
:ADDRESS OF PARAMETER NAME
:BR IF AT END OF TABLE
:TYPE THE PARAMETER NAME
:ADDRESS OF PARAMETER NAME TEXT
:MAXIMUM PARAMETER VALUE
:ADDRESS OF PARAMETER
:CURRENT VALUE OF PARAMETER
:TYPE THE CURRENT VALUE OF THE PARAMETER
:/
:READ THE KEYBOARD
:INPUT ASCII STRING ADDRESS
:'CONTROL C' ENTERED
:BR IF IT WAS
:CHECK THE DIGIT(S)
:CARRIAGE RETURN ONLY ENTERED
:PERIOD ONLY ENTERED
:ILLEGAL INPUT
:TERMINATED WITH A CARRIAGE RETURN
:TERMINATED WITH A "."
:TERMINATED WITH A " "
:MOVE NEW VALUE TO PARAMETER LOCATION
:GET MORE PARAMETERS

```

6246 026364 104401 053276
6247 026370 162703 000006
6248 026374 000741
6249 026376 010215
6250 026400 000404
6251 026402 005037 001262
6252 026406 011603
6253 026410 000733
6254 026412 005726
6255 026414 000207
6256
6257
6258
6259
6260
6261
6262
6263 026416 104401 053007
6264 026422 104401 052160
6265 026426 010446
6266
6267 026430 104403
6268 026432 002
6269 026433 000
6270 026434 104401
6271 026436 000000
6272 026440 104401 001165
6273 026444 000207
6274
6275
6276
6277
6278
6279
6280 026446 005004
6281 026450 111004
6282 026452 146437 033320 001462
6283 026460 006304
6284 026462 010064 001464
6285 026466 104401 001165
6286 026472 104401 001165
6287 026476 104401 052562
6288 026502 104401 052673
6289 026506 006204
6290 026510 010446
6291
6292 026512 104403
6293 026514 002
6294 026515 000
6295 026516 104401 001165
6296 026522 000207
6297
6298
6299
6300 026524 032777 000020 152406
6301 026532 001006

6S: TYPE BADENT ; 'BAD ENTRY'
SUB #6,R3 ; DECREMENT THE TABLE POINTER
BR 1S ; TRY AGAIN
7S: MOV R2,(R5) ; NEW VALUE
BR 9S ; EXIT
8S: CLR CFLAG ; CLEAR THE 'CONTROL C' FLAG
MOV (SP),R3 ; RELOAD THE PARAMETER TABLE ADDRESS
BR 1S ; TRY AGAIN
9S: TST (SP)+ ; CORRECT THE STACK POINTER
RTS PC ; RETURN

;TYPEOUT ASSIGN/DEASSIGN ERROR MESSAGE
;CALL
; MOV #MESADR,ASNMSG ;ERROR MESSAGE ADDRESS
; JSR PC,ASNERR
; RETURN

ASNERR: TYPE ,QUES ;QUESTION MARK
TYPE ,UNTMSG ;TYPE 'DRIVE'
MOV R4,-(SP) ;SAVE R4 FOR TYPEOUT
; TYPE DRIVE NUMBER
; GO TYPE--OCTAL ASCII
; TYPE 2 DIGIT(S)
; SUPPRESS LEADING ZEROS
; TYPE SPECIFIC MESSAGE
; MESSAGE ADDRESS

ASNMSG: ,WORD 0
TYPE ,SCLF
RTS PC

;DEASSIGN DRIVE IF A FATAL ERROR OCCURS
;CALL
; JSR PC,DROP
; RETURN

DROP: CLR R4 ;CLEAR R4 FOR DRIVE NUMBER
MOVB (R0),R4 ;MOVE DRIVE NUMBER TO R4
BICB ATABIT(R4),ASNLS ; REMOVE DRIVE FROM ASSIGNED LIST
ASL R4 ;MAKE DRIVE NUMBER INTO A TABLE INDEX
MOV R0,DUNIT(R4) ;PUT DRIVE IN DROP LIST
TYPE ,SCLF
TYPE ,SCLF
TYPE ,DROPNG ;TYPE 'DROPPING DRIVE'
TYPE ,DRNUM ;'DRIVE #'
ASR R4 ;DRIVE NUMBER
MOV R4,-(SP) ;SAVE R4 FOR TYPEOUT
; TYPE DRIVE NUMBER
; GO TYPE--OCTAL ASCII
; TYPE 2 DIGIT(S)
; SUPPRESS LEADING ZEROS

;ROUTINE TO DEASSIGN DRIVE IF ERRORS BECOMES EXCESSIVE
ABNRML: BIT #SW04,OSWR ;SEE IF SWITCH 4 SET
BNE 1S ;BR IF IT'S SET

```

```

6302 026534 023760 001406 000056      CMP      MAXER,$TOTAL(R0)      ;CHECK TOTAL ERROR VALUE
6303 026542 103002                BHIS     1$                    ;BR IF ERRORS DONOT EXCEED MAX
6304 026544 000137 026446                JMP      DROP                  ;DEASSING THE DRIVE
6305 026550 000207                RTS      PC                    ;RETURN
6306
6307      ;ROUTINE TO CHECK FOR END OF PASS AND END OF TEST
6308
6309      EOP:  TST      ENDET      ;END OF PASS DETERMINED BY SEEKS OR WORDS ?
6310 026552 005737 001430                BEQ     EOP1                  ;BR IF SEEKS
6311 026556 001412                CMP     $READ+2(R0),ENDCON+2 ;CHECK MSW OF WORDS READ COUNT
6312 026560 026037 000054 001374                BHI     EOP2                  ;BR IF MSW GREATER THAN LIMIT
6313 026566 101017                BLO     EOP1                  ;BR IF MSW LESS THAN LIMIT
6314 026570 103405                CMP     $READ(R0),ENDCON     ;CHECK LSW AGAINST LIMIT
6315 026572 026037 000052 001372                BHIS   EOP2                  ;BR IF EQUAL OR GREATER
6316 026600 103012                BR      EOPX                  ;EXIT
6317 026602 000510                BR      EOPX                  ;EXIT
6318 026604 026037 000044 001400      EOP1:  CMP     $POSIT+2(R0),ENDSEK+2 ;CHECK MSW OF SEEK COUNT
6319 026612 101005                BHI     EOP2                  ;BR IF MSW GREATER THAN LIMIT
6320 026614 103503                BLO     EOPX                  ;EXIT IF MSW LESS THAN LIMIT
6321 026616 026037 000042 001376                CMP     $POSIT(R0),ENDSEK    ;CHECK LSW OF SEEK COUNT
6322 026624 103477                BLO     EOPX                  ;EXIT IF LSW LESS THAN LIMIT
6323 026626 104401 001165      EOP2:  TYPE    , $CRLF              ;CR-LF
6324 026632 104401 052616                TYPE    , $ENDPAS           ;END OF PASS FOR THE DRIVE
6325 026636 016046 000070                MOV     $PASSC(R0),-(SP)    ;PUT PASS COUNT ON THE STACK
6326 026642 104405                TYPOS   ;CONVERT PASS COUNT TO DECIMAL AND TYPE IT
6327 026644 111037 001246                MOV     (R0),UNIT          ;STORE THE DRIVE NUMBER
6328 026650 032777 000020 152262                BIT     #SW04,$SWR         ;SWITCH 4 SET ?
6329 026656 001017                BNE     1$                    ;BR IF SET
6330 026660 026037 000070 001402                CMP     $PASSC(R0),PASCNT   ;SEE IF AT END OF TEST
6331 026666 103413                BLO     1$                    ;BR IF NOT
6332 026670 104401 052632                TYPE    , $ENDTST          ;TYPE 'END OF TEST'
6333 026674 104401 052673                TYPE    , $DRNUM           ;'DRIVE #'
6334 026700 013746 001246                MOV     UNIT,-(SP)         ;SAVE UNIT FOR TYPEOUT
6335                                ;TYPE DRIVE NUMBER
6336 026704 104403                TYPOS   ;GO TYPE--OCTAL ASCII
6337 026706 002                .BYTE  2                    ;TYPE 2 DIGIT(S)
6338 026707 000                .BYTE  0                    ;SUPPRESS LEADING ZEROS
6339 026710 104401 001165                TYPE    , $CRLF           ;CR-LF
6340 026714 000431                BR      3$                    ;DEASSIGN THE DRIVE
6341 026716 104401 052673      1$:  TYPE    , $DRNUM           ;'DRIVE #'
6342 026722 013746 001246                MOV     UNIT,-(SP)         ;SAVE UNIT FOR TYPEOUT
6343                                ;TYPE DRIVE NUMBER
6344 026726 104403                TYPOS   ;GO TYPE--OCTAL ASCII
6345 026730 002                .BYTE  2                    ;TYPE 2 DIGIT(S)
6346 026731 000                .BYTE  0                    ;SUPPRESS LEADING ZEROS
6347 026732 104401 001165                TYPE    , $CRLF           ;CR-LF
6348 026736 004737 022676                JSR     PC,TYPEST          ;TYPE THE DRIVE'S STATISTICS
6349 026742 010346                MOV     R3,-(SP)           ;SAVE R3
6350 026744 010446                MOV     R4,-(SP)           ;SAVE R4
6351 026746 010004                MOV     R0,R4              ;DRIVE'S BLOCK ADDRESS
6352 026750 062704 000036                ADD     # $OPERC,R4        ;ADD THE STARTING ADDR OF SECTIONS TO CLEAR
6353 026754 012703 000010                MOV     #8.,R3            ;NUMBER OF LOCNS TO BE CLEARED
6354                                ;(ERROR COUNTERS NOT CLEARED)
6355 026760 005024      2$:  CLR     (R4)+              ;CLEAR THE LOCN
6356 026762 005303                DEC     R3                  ;DECREMENT THE LOCATION COUNTER
6357 026764 001375                BNE     2$                    ;BR IF MORE TO GO
6358 026766 012604                MOV     .SP+4,R4           ;RESTORE R4

```

```

6358 026770 012603          MCV      (SP)+,R3          ;RESTORE R3
6359 026772 005260 000070  INC      $PASSC(R0)       ;INCREMENT THE PASS COUNT
6360 026776 000412          BR       EOPX             ;EXIT
6361 027000 104401 001165 3$:      TYPE      $CRLF
6362 027004 005004          CLR      R4              ;CLEAR R4 FOR DRIVE NUMBER
6363 027006 111004          MOVB    (R0),R4          ;MOVE DRIVE NUMBER
6364 027010 146437 033320 001462 BICB    ATABIT(R4),ASNLST ;DELETE DRIVE FROM ASSIGNED LIST
6365 027016 006304          ASL     R4              ;MAKE DRIVE NUMBER INTO TABLE INDEX
6366 027020 010064 001464  EOPX:    MOV      R0,DUNIT(R4) ;PUT BLOCK ADDRESS INTO DROP LIST
6367 027024 000207          RTS      PC             ;RETURN
6368
6369 ;ROUTINE TO GET THE REMAINDER OF THE RANDOM NUMBER
6370 ;CALL
6371 ;
6372 ;      MOV      NUMBER,R5          ;DIVISOR INTO R5
6373 ;      JSR     PC,GETREM          ;
6374 ;      RETURN          ;REMAINDER IS IN R5
6375 GETREM: MOV      $LONUM,-(SP)    ;STORE RANDOM NUMBER ON THE STACK FOR DIVIDE
6376         MOV      $HINUM,-(SP)    ;UPPER PART
6377         MOV      R5,-(SP)        ;PUT THE DIVISOR ONTO THE STACK
6378         JSR     PC,LINKDV        ;DIVIDE THE RANDOM NUMBERS
6379         MOV      (SP)+,R5        ;PUT THE REMAINDER INTO R5
6380         TST     (SP)+            ;ADJUST THE STACK POINTER
6381         NOP
6382         RTS      PC             ;FOR DEBUGGING HALT
6383
6384 ;LINK ROUTINE TO THE DIVISION UTILITY SUBROUTINE
6385 ; THIS ROUTINE ALLOWS THE 'SYSMAC' DIVIDE ROUTINE
6386 ; CALLING SEQUENCE TO BE USED
6387
6388 LINKDV: SAVREG          ;STORE R0 - R5
6389         MOV      26(SP),R5        ;DIVISOR
6390         CLR      R4              ;OTHER DIVISOR WORD
6391         MOV      30(SP),R2        ;UPPER DIVIDEND WORD
6392         MOV      32(SP),R3        ;LOWER DIVIDEND WORD
6393         CLR      R0              ;CLEAR OTHER DIVIDEND REGISTERS
6394         CLR      R1
6395         JSR     PC,M.DPID        ;GO TO THE DIVIDE ROUTINE
6396         MOV      R1,30(SP)        ;REMAINDER ON THE STACK
6397         MOV      R3,32(SP)        ;QUOTIENT ON THE STACK
6398         RESREG          ;RESTORE R0 - R5
6399         MOV      (SP)+,(SP)      ;MOVE RETURN UP THE STACK
6400         RTS      PC
6401
6402 ;
6403 ;      DIVISION UTILITY SUBROUTINE
6404 ;      R0-R1-R2-R3=DIVIDEND
6405 ;      R4-R5=DIVISOR
6406 ;      R0-R1=REMAINDER AFTER DIVISION
6407 ;      R2-R3=QUOTIENT AFTER DIVISION
6408 ;      ENTER WITH JSR PC,M.DPID
6409 M.DPID: MOV      #40,-(SP)        ;COUNTER FOR DIVISION CYCLES
6410         MOV      R4,-(SP)        ;HIGH ORDER
6411         MOV      R5,-(SP)        ;LOW ORDER DIVISOR TO THE STACK
6412         NEG     2(SP)            ;FORM NEGATIVE
6413         NEG     2SP             ;VERSION OF THE DIVISOR

```

```

6414 027140 005666 000002          SBC      2(SP)
6415 027144 061601          ADD      2(SP),R1
6416 027146 005500          ADC      R0          ;PERFORM THE INITIAL SUBTRACTION
6417 027150 066600 000002          ADD      2(SP),R0
6418 027154 103445          BCS     M.DP50      ;IF CARRY THEN OVERFLOW HAS OCCURRED
6419 027156 005046          CLR     -(SP)      ;THIS IS A LONGER LASTING CARRY BIT
6420 027160 006103          M.DP40: ROL     R3
6421 027162 006102          ROL     R2
6422 027164 006101          ROL     R1
6423 027166 006100          ROL     R0
6424 027170 005716          TST     QSP
6425 027172 001410          BEQ     M.DP41      ;TEST "CARRY" INDICATOR
6426 027174 005016          CLR     QSP         ;IF NO "CARRY" THEN ADD ELSE SUBTRACT
6427 027176 066601 000002          ADD      2(SP),R1   ;CLEAR UP FOR NEXT TIME
6428 027202 005500          ADC      R0          ;ADD -(DIVISOR)
6429 027204 005516          ADC     QSP          ;SET "CARRY"
6430 027206 066600 000004          ADD      4(SP),R0  ;<- I
6431 027212 000404          BR     M.DP42
6432 027214 060501          M.DP41: ADD     R5,R1
6433 027216 005500          ADC      R0          ;ADD +(DIVISOR)
6434 027220 005516          ADC     QSP          ;SET "CARRY"
6435 027222 060400          ADD     R4,R0      ;<- I
6436 027224 005516          M.DP42: ADC     QSP          ;SET "CARRY"
6437 027226 005716          TST     QSP         ;TEST THE UPDATE INDICATOR
6438 027230 001401          BEQ     .+4         ;IF ZERO FORGET IT
6439 027232 005203          INC     R3          ;NO CARRY POSSIBLE HERE
6440 027234 005366 000006          DEC     6(SP)      ;DECREMENT COUNTER
6441 027240 003347          BGT     M.DP40      ;BRANCH IF MORE TO DO
6442 027242 006003          ROR     R3
6443 027244 103404          BCS     M.DP44
6444 027246 060501          ADD     R5,R1
6445 027250 005500          ADC      R0
6446 027252 060400          ADD     R4,R0
6447 027254 000241          CLC
6448 027256 006103          M.DP44: ROL     R3
6449 027260 062706 000010          ADD     #10,SP     ;ADJUST STACK BY 4 WORDS
6450 027264 000242          CLV
6451 027266 000207          RTS     PC
6452 027270 062706 000006          M.DP50: ADD     #6,SP
6453 027274 000262          SEV
6454 027276 000207          RTS     PC
6455
6456          ;ROUTINE TO REPLACE LEADING ZEROS IN A NUMERIC STRING WITH SPACES
6457          ;CALL
6458          ;
6459          ;
6460          ;
6461          ;
6462          ;
6463 027300 010046          REPLZ: MOV     R0, -(SP) ;SAVE R0
6464 027302 012746 000012          MOV     #10, -(SP) ;MAXIMUM NUMBER OF DIGITS TO BE TYPED
6465 027306 162516          SUB     (R5)+, (SP) ;SUBTRACT DIGITS TO FORM INDEX
6466 027310 016600 000006          MOV     6(SP), R0  ;ADDRESS OF NUMBER TO R0
6467 027314 122710 000060          IS:   CMPB   #'0', (R0) ;BYTE EQUAL TO ASCII '0' ?
6468 027320 001004          BNE    2$
6469 027322 112710 000040          MOVB   #40, (R0)  ;REPLACE THE ZERO WITH A SPACE

```



```

6470 027326 005200      INC      RO      ; INCREMENT THE BYTE ADDRESS
6471 027330 000771      BR       1$      ; GO BACK AND LOOK FOR MORE LEADING ZEROS
6472 027332 105710      2$: TSTB   (RO)   ; SEE IF ZERO BYTE TERMINATOR
6473 027334 001003      BNE     3$      ; BR IF NOT
6474 027336 005300      DEC     RO      ; BACKUP STRING POINTER
6475 027340 112710 000060      MOVB   #'0,(RO) ; PUT A ZERO BACK IN
6476 027344 016637 000006 027360 3$: MOV    6(SP),4$ ; PUT ADDRESS IN LOCATION FOR TYPEOUT
6477 027352 062637 027360      ADD    (SP)+,4$ ; BEGINNING OF SIGNIFICANT DIGITS
6478 027356 104401      TYPE   0        ; TYPE THE NUMBER
6479 027360 000000      .WORD 0        ; ADDRESS OF NUMBER
6480 027362 012600      MOV    (SP)+,RO ; RESTORE RO
6481 027364 012616      MOV    (SP)+,(SP) ; MOVE RETURN ADDRESS
6482 027366 000205      RTS     R5      ; RETURN
6483
6484      ; TYPE NUMERICAL ASCIZ STRING SUPPRESS LEADING ZEROS
6485
6486      ; CALL
6487      ;
6488      ;
6489      ;
6490 027370 010046      $$SUPRS: MOV   RO,-(SP) ; SAVE RO
6491 027372 016600 000004      MOV   4(SP),RO ; PICKUP THE POINTER
6492 027376 105710      1$: TSTB (RO) ; TERMINATOR ?
6493 027400 001403      BEQ   2$      ; BR IF YES
6494 027402 122720 000060      CMPB #'0,(RO)+ ; IS THIS AN ASCII '0' ?
6495 027406 001773      BEQ   1$      ; BR IF YES
6496 027410 005300      2$: DEC   RO ; BACKUP BY '1'
6497 027412 010037 027420      MOV   RO,3$ ; SAVE FOR TYPING
6498 027416 104414      DISPLY ; GO PRINT
6499 027420 000000      3$: .WORD 0 ; ASCIZ POINTER GOES HERE
6500 027422 012600      MOV   (SP)+,RO ; RESTORE RO
6501 027424 012616      MOV   (SP)+,(SP) ; RESTORE THE STACK
6502 027426 000207      RTS   PC      ; RETURN
6503
6504      ; ROUTINE TO TYPE AT PRIORITY 4
6505
6506 027430 013746 177776      TYPRI4: MOV   @#PS,-(SP) ; SAVE THE PRESENT STATUS
6507 027434 012737 000200 177776      MOV   @200,@#PS ; CHANGE THE PRIORITY TO 4
6508 027442 012537 027452      MOV   (R5)+,1$ ; MESSAGE ADDRESS
6509 027446 004737 031034      JSR   PC,$TYPE ; TYPE THE MESSAGE
6510 027452 000000      1$: .WORD 0 ; MESSAGE ADDRESS GOES HERE
6511 027454 000205      RTS   R5      ; RETURN
6512
6513      ; ROUTINE TO TYPE ERRORS
6514      ; CALL
6515      ;
6516      ;
6517      ;
6518      ;
6519 027456 032777 020000* 151454      $DSPLY: BIT   #BIT13,@SWR ; INHIBIT ERROR TYPEOUT ^
6520 027464 001004      BNE   1$      ; BR IF YES
6521 027466 005037 177776      CLR   @#PS ; SET PRIORITY TO ZERO
6522 027472 000137 031034      JMP   $TYPE ; TYPE THE MESSAGE
6523 027476 062716 000002      1$: ADD   #2,(SP) ; INCREMENT THE RETURN
6524 027502 000002      RTI ; RETURN
6525

```

```

6526 ; THIS ROUTINE IS USED TO CHECK IF AN
6527 ; ASCII CHARACTER IS A DIGIT BETWEEN 0 AND 7.
6528 ; CALL
6529     MOV     #ADR,R1      ; ADDRESS OF ASCII CHARACTER
6530     JSR     R5,CK.OCT   ; CHECK THE CHARACTER
6531     RETURN1              ; CHARACTER IS NOT BETWEEN 0-7
6532     RETURN2              ; CHARACTER IS IN R2 AS A
6533     ;                     ; OCTAL DIGIT
6534
6535 027504 121127 000060 CK.OCT: CMPB    (R1),#'0   ; LESS THAN ZERO?
6536 027510 103407         BLO     1$              ; YES -- BRANCH
6537 027512 121127 000067     CMPB    (R1),#'7   ; GREATER THAN SEVEN?
6538 027516 101004         BHI     1$              ; YES -- BRANCH
6539 027520 111102         MOVB    (R1),R2      ; GET THE CHARACTER
6540 027522 042702 177770     BIC     #1C7,R2      ; STRIP AWAY THE ASCII
6541 027526 005725         TST     (R5)↓      ; ADJUST FOR RETURN
6542 027530 000205         1$:    RTS      R5              ; RETURN
6543
6544 ; THIS ROUTINE IS USED TO CHECK AN ASCII CHARACTER
6545 ; AND DETERMINE IF IT IS A DIGIT BETWEEN 0 AND 9.
6546 ; CALL
6547     MOV     #ADR,R1      ; ADDRESS OF ASCII CHARACTER
6548     JSR     R5,CK.DEC   ; CHECK THE CHARACTER
6549     RETURN1              ; NOT BETWEEN 0 AND 9
6550     RETURN2              ; BETWEEN 0 AND 9
6551     ;                     ; R2 = DIGIT
6552
6553 027532 121127 000060 CK.DEC: CMPB    (R1),#'0   ; LESS THAN ZERO?
6554 027536 103407         BLO     1$              ; YES -- BRANCH
6555 027540 121127 000071     CMPB    (R1),#'9   ; GREATER THAN NINE?
6556 027544 101004         BHI     1$              ; YES -- BRANCH
6557 027546 111102         MOVB    (R1),R2      ; GET THE CHARACTER
6558 027550 042702 000060     BIC     #'0,R2      ; STRIP AWAY THE ASCII
6559 027554 005725         TST     (R5)↑      ; ADJUST FOR RETURN
6560 027556 000205         1$:    RTS      R5              ; RETURN
6561
6562 ; THIS ROUTINE WILL CHECK AN ASCII CHARACTER TO
6563 ; DETERMINE WHAT IT IS.
6564 ; CALL
6565     MOV     #ADR,R1      ; ADDRESS OF ASCII CHARACTER
6566     JSR     R5,CK.CHR   ; CHECK CHARACTER
6567     RETURN ADR1          ; UNKNOWN CHARACTER
6568     RETURN ADR2          ; CARRIAGE RETURN * (R1)=ADR+1
6569     RETURN ADR3          ; COMMA * (R1)=ADR+1
6570     RETURN ADR4          ; PERIOD * (R1)=ADR+1
6571     RETURN ADR5          ; DIGIT BETWEEN 0 AND 7.
6572     RETURN ADR6          ; DIGIT BETWEEN 8 AND 9.
6573     ;                     ; R2 = DIGIT * (R1)=ADR+1
6574
6575 027560 105711         CK.CHR: TSTB   (R1)          ; "CARRIAGE RETURN"?
6576 027562 001417         BEQ     3$              ; YES -- BRANCH
6577 027564 121127 000054     CMPB    (R1),#',      ; "COMMA"?
6578 027570 001413         BEQ     2$              ; YES -- BRANCH
6579 027572 121127 000056     CMPB    (R1),#'.      ; "PERIOD"?
6580 027576 001407         BEQ     1$              ; YES -- BRANCH
6581 027600 004537 027532     JSR     R5,CK.DEC   ; "DIGIT"?

```

CZRJDCD RPO4 5/6 MLT-DR LGC
CZRJDC.P11 15-DEC-77 10.58

MACY11 30(1046) 15-DEC-77 11:03 PAGE 123
GENERAL SUPPORT SUBROUTINES

SEG 0122

6582 027604 000410
6583 027606 004537 027504
6584 027612 005725
6585 027614 005725
6586 027616 005725

BR 4\$;NO -- BRANCH
JSR R5,CK.OCT ;OCTAL ?
TST (R5)+ ;DIGIT BETWEEN 8-9
TST (R5)+ ;DIGIT BETWEEN 0-7
1\$: TST (R5)+ ;PERIOD

6587	027620	005725	2\$:	TST	(R5)+	; COMMA
6588	027622	005725	3\$:	TST	(R5)+	; CARRIAGE RETURN
6589	027624	005201		INC	R1	; MOVE POINTER TO NEXT CHARACTER
6590	027626	011505	4\$:	MOV	(R5),R5	; UNKNOWN CHARACTER
6591	027630	000205		RTS	R5	; RETURN
6592						
6593						
6594						
6595						
6596						
6597						
6598						
6599						
6600						
6601						
6602						
6603						
6604						
6605						
6606	027632	010446				
6607	027634	010346				
6608	027636	010246				
6609	027640	005002				
6610	027642	005003				
6611	027644	005004				
6612	027646	004537	027560			
6613	027652	027746				
6614	027654	027754				
6615	027656	027746				
6616	027660	027750				
6617	027662	027666				
6618	027664	027666				
6619	027666	062705	000004	1\$:	ADD	#4,R5
6620	027672	006303		2\$:	ASL	R3
6621	027674	010346			MOV	R3,-(SP)
6622	027676	006303			ASL	R3
6623	027700	006303			ASL	R3
6624	027702	062603			ADD	(SP)+,R3
6625	027704	060203			ADD	R2,R3
6626	027706	004537	027560		JSR	R5,CK.CHR
6627	027712	027752			8\$	
6628	027714	027736			5\$	
6629	027716	027734			4\$	
6630	027720	027726			3\$	
6631	027722	027672			2\$	
6632	027724	027672			2\$	
6633	027726	105711		3\$:	TSTB	(R1)
6634	027730	001010			BNE	8\$
6635	027732	005724			TST	(R4)+
6636	027734	005724		4\$:	TST	(R4)+
6637	027736	005724		5\$:	TST	(R4)+
6638	027740	020316			CMP	R3,(SP)
6639	027742	101004			BHI	9\$
6640	027744	000402			BR	8\$
6641	027746	005725		6\$:	TST	(R5)+
6642	027750	005725		7\$:	TST	(R5)+

```

; THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL
; CHARACTERS AND FORMS A DECIMAL VALUE BINARY NUMBER IN R2.
CALL
      MOV      #ADR,R1      ; ADDRESS OF ASCII 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

CK.DIG: MOV      R4,-(SP)   ; SAVE R4
      MOV      R3,-(SP)   ; SAVE R3
      MOV      R2,-(SP)   ; SAVE THE MAX. SIZE ON THE STACK
      CLR      R2         ; START WITH 0
      CLR      R3
      CLR      R4
      JSR      R5,CK.CHR  ; CHECK ONE CHARACTER
      6$           ; ILLEGAL CHARACTER
      9$           ; CARRIAGE RETURN
      7$           ; " "
      1$          ; DIGIT 0-7
      1$          ; DIGIT 8-9
      1$          ; STEP RETURN POINTER PAST "CR" & "PERIOD" RETURNS
      2$          ; INPUT NUMBER *2
      2$          ; SAVE *2
      4$          ; *4
      8$          ; *8
      10$         ; (*2)+(*8) = *10
      10$         ; UPDATE THE INPUT NUMBER
      JSR      R5,CK.CHR  ; CHECK ONE CHARACTER
      8$           ; ILLEGAL CHARACTER
      5$           ; CARRIAGE RETURN
      4$           ; " "
      2$          ; DIGIT 0-7
      2$          ; DIGIT 8-9
      3$          ; DOES A "CR" FOLLOW THE "PERIOD"
      8$          ; BR IF NOT
      4$          ; INCREMENT THE RETURN
      4$          ; INCREMENT THE RETURN
      4$          ; INCREMENT THE RETURN
      4$          ; INCREMENT THE RETURN
      9$          ; CHECK THE MAGNITUDE OF THE NUMBER
      9$          ; BR IF ENTERED NUMBER TOO LARGE
      8$          ; BYPASS INCREMENT
      4$          ; INCREMENT RETURN PAST INVALID RETURN
      4$          ; INCREMENT RETURN

```

```

6643 027752 060405      8$: ADD R4,R5 ;SETUP RETURN POINTER
6644 027754 010302      9$: MOV R3,R2 ;ENTERED VALUE
6645 027756 005726      TST (SP)+ ;CLEAN MAX. SIZE OFF OF STACK
6646 027760 012603      MOV (SP)+,R3 ;RESTORE R3
6647 027762 012604      MOV (SP)+,R4 ;RESTORE R4
6648 027764 011505      MOV (R5),R5 ;GET RETURN ADDRESS
6649 027766 000205      RTS R5 ;RETURN
6650
6651 ;THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
6652 ;UNSIGNED DECIMAL ASCIZ NUMBER.
6653 ;CALL
6654 ; MOV NUMBER, -(SP) ;PUT THE NUMBER ON THE STACK
6655 ; JSR PC, $SB2D ;CALL
6656 ; RETURN ;ADDRESS OF THE 1ST ASCIZ CHAR IS ON THE STACK
6657
6658 ;NOTE: THE PROGRAM REQUIRES THIS FORM OF '$SB2D', NOT THE VERSION ON
6659 ; THE SYSMAC LIBRARY, REV C AND LATER
6660
6661 027770 016637 000002 030014 $SB2D: MOV 2(SP), 1$ ;SAVE THE BINARY NUMBER
6662 027776 012746 030014 ; MOV #1$, -(SP) ;SET THE POINTER
6663 030002 004737 032562 ; JSR PC, $DB2D ;CALL THE DOUBLE LENGTH CONVERT
6664 030006 012666 000002 ; MOV (SP)+, 2(SP) ;PICKUP THE POINTER
6665 030012 000207 ; RTS PC ;RETURN
6666 030014 000000 000000 1$: .WORD 0,0
6667
6668 ;THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
6669 ;UNSIGNED OCTAL ASCIZ NUMBER.
6670 ;CALL
6671 ; MOV NUMBER, -(SP) ;PUT THE NUMBER ON THE STACK
6672 ; JSR PC, $SB2D ;CALL
6673 ; RETURN ;ADDRESS OF THE 1ST ASCIZ CHAR IS ON THE STACK
6674
6675 ;NOTE: THE PROGRAM REQUIRES THIS FORM OF '$SB2D', NOT THE VERSION ON
6676 ; THE SYSMAC LIBRARY, REV C AND LATER
6677
6678 030020 016637 000002 030044 $SB2D: MOV 2(SP), 1$ ;SAVE THE BINARY NUMBER
6679 030026 012746 030044 ; MOV #1$, -(SP) ;SET THE POINTER
6680 030032 004737 032756 ; JSR PC, $DB2D ;CALL THE DOUBLE LENGTH CONVERT
6681 030036 012666 000002 ; MOV (SP)+, 2(SP) ;PICKUP THE POINTER
6682 030042 000207 ; RTS PC ;RETURN
6683 030044 000000 000000 1$: .WORD 0,0
6684
6685 ;KEYBOARD INTERRUPT INITIALIZATION ROUTINE
6686 ;CALL
6687 ; JSR PC, $TKINT
6688 ; RETURN
6689
6690 030050 012737 030100 000060 $TKINT: MOV #STKSRV, TKVEC ;SETUP VECTOR
6691 030056 012737 000240 000062 ; MOV #PRS, TKVEC+2 ;PRIORITY TO 5
6692 030064 005777 151056 ; TST @STKB ;CLEAR THE BUFFER
6693 030070 012777 000100 151046 ; MOV #BIT06, @STKS ;SET INTERRUPT ENABLE
6694 030076 000207 ; RTS PC ;RETURN
6695
6696 ;KEYBOARD INTERRUPT SERVICE ROUTINE
6697 ;CALL
6698 ; ENTER VIA INTERRUPT

```

```

6699
6700 030100 104410          STKSRV: RDCHR          ; READ THE KEYBOARD
6701 030102 112637 030230  MOVB      (SP)+,5$      ; GET THE CHARACTER
6702 030106 023727 030230 000003  CMP      5$,#3        ; 'CONTROL C' ?
6703 030114 001012          BNE      1$          ; BR IF NOT
6704 030116 104401 001165  TYPE      ,SCRLF      ; CR-LF
6705 030122 104401 030522  TYPE      ,SCNTLC     ; '↑C'
6706 030126 012737 177777 001262  MOV      #-1,CFLAG    ; SET THE 'CONTROL C' FLAG
6707 030134 005077 151004  CLR      @STKS        ; CLEAR THE TTY INTERRUPT
6708 030140 000432          BR       4$          ; EXIT
6709 030142 023727 001140 000176 1$:  CMP      SWR,#SWREG   ; SOFTWARE SWITCH REGISTER IN USE ?
6710 030150 001024          BNE      3$          ; BR IF NOT
6711 030152 023727 037230 000007  CMP      5$,#7        ; 'CONTROL G' ?
6712 030160 001020          BNE      3$          ; BR IF NOT
6713 030162 104401 001165  TYPE      ,SCRLF      ; CR-LF
6714 030166 104401 032335  TYPE      ,SCNTLG     ; '↑G'
6715 030172 013746 177776  MOV      PS,-(SP)     ; PUT THE STATUS WORD ON THE STACK
6716 030176 012746 030212  MOV      #2$,-(SP)   ; RETURN ADDRESS
6717 030202 005077 150736  CLR      @STKS        ; CLEAR THE TTY INTERRUPT ENABLE
6718 030206 000137 031776  JMP      $GTSWR      ; GET THE SWITCH REGISTER ENTRY
6719 030212 012777 000100 150724 2$:  MOV      #100,@STKS  ; ENABLE TTY KEYBOARD INTERRUPT
6720 030220 000402          BR       4$          ; EXIT
6721 030222 104401 030230 3$:  TYPE      ,5$        ; ECHO THE CHARACTER
6722 030226 000002          4$:  RTI              ; RETURN
6723
6724 030230 000000          5$:  .WORD      0          ; ENTERED CHARACTER
6725
6726          ; THIS ROUTINE WILL INPUT A STRING FROM THE TTY
6727          ; CALL:
6728          ;
6729          ; RDLIN
6730          ; RETURN HERE
6731          ;
6732          ; INPUT A STRING FROM THE TTY
6733          ; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
6734          ; TERMINATOR WILL BE A BYTE OF ALL 0'S
6735
6736 $RDLIN: MOV      R3,-(SP) ; SAVE R3
6737 030232 010346          CLR      -(SP)      ; CLEAR THE RUBOUT KEY
6738 030234 005046          1$:  MOV      #STTYIN,R3 ; GET ADDRESS
6739 030236 012703 030510 2$:  CMP      #STTYIN+10.,R3 ; BUFFER FULL?
6740 030242 022703 030522 4$:  BLOS     4$          ; BR IF YES
6741 030246 101467          BLOS     4$          ; GO READ ONE CHARACTER FROM THE TTY
6742 030250 104410          RDCHR     ; GET CHARACTER
6743 030252 112613          MOVB     (SP)+,(R3) ; IS IT A RUBOUT
6744 030254 122713 000177  CMPB     #177,(R3)  ; IS THIS THE FIRST RUBOUT?
6745 030260 001022          BNE      5$          ; BR IF NO
6746 030262 005716          TST     (SP)        ; IS THIS THE FIRST RUBOUT?
6747 030264 001007          BNE      6$          ; BR IF NO
6748 030266 112737 000134 030506  MOVB     #' \,9$    ; TYPE A BACK SLASH
6749 030274 104401 030506  TYPE     #' \,9$
6750 030300 012716 177777  MOV      #-1,(SP)   ; SET THE RUBOUT KEY
6751 030304 005303          6$:  DEC      R3         ; BACKUP BY ONE
6752 030306 020327 030510  CMP      R3,#STTYIN ; STACK EMPTY?
6753 030312 103445          4$:  BLO     4$          ; BR IF YES
6754 030314 111337 030506  MOVB     (R3),9$    ; SETUP TO TYPEOUT THE DELETED CHAR.
6755 030320 104401 030506  TYPE     ,9$        ; GO TYPE
6756 030324 000746          BR       2$          ; GO READ ANOTHER CHAR.
6757 030326 005716          5$:  TST     (SP)        ; RUBOUT KEY SET?
6758 030330 001406          7$:  BEQ     7$          ; BR IF NO
6759 030332 112737 000134 030506  MOVB     #' \,9$    ; TYPE A BACK SLASH

```

```

6755 030340 104401 030506          TYPE          9$
6756 030344 005016          CLR          (SP)
6757 030346 122713          CMPB         #25,(R3)
6758 030352 001003          BNE          10$
6759 030354 104401 032330          TYPE          $CNTLU
6760 030360 000726          BR           1$
6761 030362 122713 000003          CMPB         #3,(R3)
6762 030366 001006          BNE          8$
6763 030370 012737 177777 001262          MOV          #-1,CFLAG
6764 030376 104401 030522          TYPE          $CNTLC
6765 030402 000427          BR           11$
6766 030404 122713 000012          CMPB         #12,(R3)
6767 030410 001011          BNE          3$
6768 030412 105013          CLRB        (R3)
6769 030414 104401 001165          TYPE          $SCRLF
6770 030420 104401 030510          TYPE          $TTYIN
6771 030424 000706          BR           2$
6772 030426 104401 001164          TYPE          $QUES
6773 030432 000701          BR           1$
6774 030434 111337 030506          MOVB        (R3),9$
6775 030440 104401 030506          TYPE          9$
6776 030444 122723 000015          CMPB         #15,(R3)+
6777 030450 001274          BNE          2$
6778 030452 105063 177777          CLRB        -1(R3)
6779 030456 104401 001166          TYPE          $LF
6780 030462 005726          TST         (SP)+
6781 030464 012603          MOV          (SP)+,R3
6782 030466 011646          MOV          (SP)-,(SP)
6783 030470 016666 000004 000002          MOV          4(SP),2(SP)
6784 030476 012766 030510 000004          MOV          #TTYIN,4(SP)
6785 030504 000002          RTI
6786 030506          .BYTE      0
6787 030507          .BYTE      0
6788 030510 000012          $TTYIN:    .BLKB 10
6789 030522 041536 005015 000          $CNTLC:    .ASCIZ /<C/<CR/<LF>
6790
6791          030530          .EVEN
6792
6793          ;;*****
6794
6795          .SBTTL  MACRO ROUTINES
6796
6797          .SBTTL  ERROR HANDLER ROUTINE
6798
6799          ;;*****
6800          ;THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT.
6801          ;SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
6802          ;AND GO TO $ERRTYP ON ERROR
6803          ;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
6804          ;$SW15=1      HALT ON ERROR
6805          ;$SW13=1      INHIBIT ERROR TYPEOUTS
6806          ;$SW10=1     BELL ON ERROR
6807          ;CALL
6808          ;*          ERROR      N          ;;ERROR=EMT AND N=ERROR ITEM NUMBER
6809
6810          $ERROR:

```

```

6811 030530 104407          CKSWR          ;; TEST FOR CHANGE IN SOFT-SWP
6812 030532 010337 001244    MOV          R3,ATTN      ;; SAVE THE ATTENTION REGISTER CONTENTS
6813 030536 010137 001242    MOV          R1,DRIVE     ;; DRIVE NUMBER
6814 030542 032777 020000 150370  BIT          #SW13,@SWR   ;; INHIBIT PRINTOUTS ?
6815 030550 001002          BNE          +6          ;; BR IF YES
6816 030552 004737 023450    JSR          PC,$TIME     ;; TYPE THE TIME
6817 030556 105237 001103 7$:  INCB        $ERFLG      ;; SET THE ERROR FLAG
6818 030562 001775          BEQ          7$          ;; DON'T LET THE FLAG GO TO ZERO
6819 030564 013777 001102 150350  MOV          $TSTNM,@DISPLAY ;; DISPLAY TEST NUMBER AND ERROR FLAG
6820 030572 032777 002000 150340  BIT          #BIT10,@SWR  ;; BELL ON ERROR?
6821 030600 001402          BEQ          1$          ;; NO - SKIP
6822 030602 104401 001160    TYPE        $BELL        ;; RING BELL
6823 030606 005237 001112 1$:  INC          $ERTTL      ;; COUNT THE NUMBER OF ERRORS
6824 030612 011637 001116    MOV          (SP),$ERRPC  ;; GET ADDRESS OF ERROR INSTRUCTION
6825 030616 162737 000002 001116  SUB          #2,$ERRPC
6826 030624 117737 150266 001114  MOVB        @ERRPC,$ITEMB ;; STRIP AND SAVE THE ERROR ITEM CODE
6827 030632 032777 020000 150300  BIT          #BIT13,@SWR  ;; SKIP TYPEOUT IF SET
6828 030640 001004          BNE          20$        ;; SKIP TYPEOUTS
6829 030642 004737 030700    JSR          PC,$ERRTYP  ;; GO TO USER ERROR ROUTINE
6830 030646 104401 001165    TYPE        , $CRLF
6831 030652          20$:
6832 030652 005777 150262 2$:  TST          @SWR        ;; HALT ON ERROR
6833 030656 100002          BPL          3$          ;; SKIP IF CONTINUE
6834 030660 000000          HALT
6835 030662 104407          CKSWR          ;; HALT ON ERROR!
6836 030664          3$:  TEST FOR CHANGE IN SOFT-SWP
6837 030664 023737 000042 000046  CMP          @#42,@#46   ;; ARE WE IN ACT-11 AUTO MODE?
6838 030672 001001          BNE          +4          ;; BRANCH IF NOT
6839 030674 000000          HALT
6840 030676 001002          RTI          ;; RETURN
6841
6842          .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
6843
6844          ;*****
6845          ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
6846          ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
6847          ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
6848
6849          $ERRTYP:
6850 030700 104401 001165    TYPE        $CRLF        ;; "CARRIAGE RETURN" & "LINE FEED"
6851 030704 010046    MOV          RO,-(SP)    ;; SAVE RO
6852 030706 005000    CLR          RO          ;; PICKUP THE ITEM INDEX
6853 030710 153700 001114  B1SB        @#$ITEMB,RO
6854 030714 001004          BNE          1$          ;; IF ITEM NUMBER IS ZERO, JUST
6855          ;; TYPE THE PC OF THE ERROR
6856 030716 013746 001116  MOV          $ERRPC,-(SP) ;; SAVE $ERRPC FOR TYPEOUT
6857          ;; ERROR ADDRESS
6858 030722 104402          TYP0C
6859 030724 000426 1$:  BR          6$          ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
6860 030726 005300          DEC          RO        ;; GET OUT
6861 030730 006300          ASL          RO        ;; ADJUST THE INDEX SO THAT IT WILL
6862 030732 006300          ASL          RO        ;; WORK FOR THE ERROR TABLE
6863 030734 006300          ASL          RO
6864 030736 062700 004026  ADD          #$ERRTB,RO  ;; FORM TABLE POINTER
6865 030742 012037 030752  MOV          (RO)+,2$    ;; PICKUP "ERROR MESSAGE" POINTER
6866 030746 001404          BEQ          3$          ;; SKIP TYPEOUT IF NO POINTER

```



```

6867 030750 104401
6868 030752 000000
6869 030754 104401 001165
6870 030760 012037 030770
6871 030764 001404
6872 030766 104401
6873 030770 000000
6874 030772 104401 001165
6875 030776 011000
6876 031000 001004
6877 031002 01260C
6878 031004 104401 001165
6879 031010 000207
6880 031012
6881 031012 013046
6882 031014 104402
6883 031016 005710
6884 031020 001770
6885 031022 104401 031030
6886 031026 000771
6887 031030 020040 000
6888 031034
6889
6890
6891
6892
6893
6894
6895
6896
6897
6898
6899
6900
6901
6902
6903
6904
6905
6906
6907 031034 105737 001157
6908 031040 100002
6909 031042 000000
6910 031044 000407
6911 031046 010046
6912 031050 017600 000002
6913 031054 112046
6914 031056 001005
6915 031060 005726
6916 031062 012600
6917 031064 062716 000002
6918 031070 000002
6919 031072 12271E 000011
6920 031076 001430
6921 031100 12271E 000200
6922 031104 001006

```

```

2$: TYPE WORD 0
TYPE %CRLF
3$: MOV (RO)+,4$
BEQ 5$
TYPE
4$: WORD 0
TYPE %CRLF
5$: MOV (RO),RO
BNE 7$
6$: MOV (SP)+,RO
TYPE %CRLF
7$: RTS PC
MOV @ (RO)+,-(SP)
TYPOC
TST (RO)
BEQ 6$
TYPE %$
BR 7$
8$: .ASCIZ / /
.EVEN

```

```

:: TYPE THE "ERROR MESSAGE"
:: "ERROR MESSAGE" POINTER GOES HERE
:: "CARRIAGE RETURN" & "LINE FEED"
:: PICKUP "DATA HEADER" POINTER
:: SKIP TYPEOUT IF 0
:: TYPE THE "DATA HEADER"
:: "DATA HEADER" POINTER GOES HERE
:: "CARRIAGE RETURN" & "LINE FEED"
:: PICKUP "DATA TABLE" POINTER
:: GO TYPE THE DATA
:: RESTORE RO
:: "CARRIAGE RETURN" & "LINE FEED"
:: RETURN
:: SAVE @ (RO)+ FOR TYPEOUT
:: GO TYPE--OCTAL ASCII (ALL DIGITS)
:: IS THERE ANOTHER NUMBER?
:: BR IF NO
:: TYPE TWO(2) SPACES
:: LOOP
:: TWO(2) SPACES

```

.SBTTL TYPE ROUTINE

```

*****
*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
*NOTE1: %NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
*NOTE2: %FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
*NOTE3: %FILLC CONTAINS THE CHARACTER TO FILL AFTER.
*

```

```

*CALL:
*1) USING A TRAP INSTRUCTION
* TYPE ,MESADR ;; MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
*OR
* TYPE
* MESADR
*

```

```

$TYPE: TSTB $TPFLG
BPL 1$
HALT
BR 3$
1$: MOV RO,-(SP)
MOV @2(SP),RO
2$: MOVB (RO)+,-(SP)
BNE 4$
TST (SP)+
60$: MOV (SP)+,RO
3$: ADD #2,(SP)
RTI
4$: CMPB #HT,(SP)
BEQ 8$
CMPB #CRLF,(SP)
BNE 5$

```

```

:: IS THERE A TERMINAL?
:: BR IF YES
:: HALT HERE IF NO TERMINAL
:: LEAVE
:: SAVE RO
:: GET ADDRESS OF ASCIZ STRING
:: PUSH CHARACTER TO BE TYPED ONTO STACK
:: BR IF IT ISN'T THE TERMINATOR
:: IF TERMINATOR POP IT OFF THE STACK
:: RESTORE RO
:: ADJUST RETURN PC
:: RETURN
:: BRANCH IF (HT)
:: BRANCH IF NOT CRLF

```

```

6923 031106 005726          TST      (SP)+          ;; POP (CR)<LF> EQUIV
6924 031110 104401          TYPE                    ;; TYPE A CR AND LF
6925 031112 001165          $CALF                    ;
6926 031114 105037 031250  CLRFB   $CHARCNT       ;; CLEAR CHARACTER COUNT
6927 031120 000755          BR      2$              ;; GET NEXT CHARACTER
6928 031122 004737 031204  5$: JSR     PC,$TYPEC      ;; GO TYPE THIS CHARACTER
6929 031126 123726 001156  6$: CMPB  $FILLC,(SP)+   ;; IS IT TIME FOR F.LLER CHARS.?
6930 031132 001350          BNE     2$              ;; IF NO GO GET NEXT CHAR.
6931 031134 013746 001154  MCV     $NULL,-(SP)     ;; GET # OF FILLER CHARS. NEEDED
6932                                     AND THE NULL CHAR.
6933 031140 105356 000001  7$: DECB  1(SP)         ;; DOES A NULL NEED TO BE TYPED?
6934 031144 002770          BLT     6$              ;; BR IF NO--GO POP THE NULL OFF OF STACK
6935 031146 004737 031204  JSR     PC,$TYPEC      ;; GO TYPE A NULL
6936 031152 105337 031250  DECB  $CHARCNT        ;; DO NOT COUNT AS A COUNT
6937 031156 000770          BR      7$              ;; LOOP

```

;HORIZONTAL TAB PROCESSOR

```

6941 031160 112716 000040  8$: MOVB  #' (SP)         ;; REPLACE TAB WITH SPACE
6942 031164 004737 031204  9$: JSR     PC,$TYPEC      ;; TYPE A SPACE
6943 031170 132737 000007 031250 BITB   #7,$CHARCNT     ;; BRANCH IF NOT AT
6944 031176 001372          BNE     9$              ;; TAB STOP
6945 031200 005726          TST      (SP)+          ;; POP SPACE OFF STACK
6946 031202 000724          BR      2$              ;; GET NEXT CHARACTER
6947 031204 105777 147740  $TYPEC: TSTB  2$TPS        ;; WAIT UNTIL PRINTER IS READY
6948 031210 100375          BPL     $TYPEC
6949 031212 116677 000002 147732 MOVB  2(SP),2$TPB      ;; LOAD CHAR TO BE TYPED INTO DATA REG.
6950 031220 122766 000015 000002 CMPB  #CR,2(SP)        ;; IS CHARACTER A CARRIAGE RETURN?
6951 031226 001003          BNE     1$              ;; BRANCH IF NO
6952 031230 105037 031250  CLRFB  $CHARCNT       ;; YES--CLEAR CHARACTER COUNT
6953 031234 000406          BR      $TYPEX
6954 031236 122766 000012 000002 1$: CMPB  #LF,2(SP)      ;; IS CHARACTER A LINE FEED?
6955 031244 001402          BEQ     $TYPEX        ;; BRANCH IF YES
6956 031246 105227          INCB  (PC)+          ;; COUNT THE CHARACTER
6957 031250 000000          $CHARCNT: .WORD 0    ;; CHARACTER COUNT STORAGE
6958 031252 000207          $TYPEX: RTS      PC

```

.SPTTL BINARY TO OCTAL (ASCII) AND TYPE

```

6961                                     ;*****
6962                                     ;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
6963                                     ;OCTAL (ASCII) NUMBER AND TYPE IT.
6964                                     ;$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
6965                                     ;CALL:
6966                                     ;
6967                                     ;      MOV     NUM,-(SP)          ;; NUMBER TO BE TYPED
6968                                     ;      TYPOS          ;; CALL FOR TYPEOUT
6969                                     ;      .BYTE   N              ;; N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
6970                                     ;      .BYTE   M              ;; M=1 OR 0
6971                                     ;
6972                                     ;      ;:1=TYPE LEADING ZEROS
6973                                     ;      ;:0=SUPPRESS LEADING ZEROS
6974                                     ;
6975                                     ;$STYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
6976                                     ;$TYPOS OR $TYFOC
6977                                     ;CALL:
6978                                     ;      MOV     NUM,-(SP)          ;; NUMBER TO BE TYPED

```

```

6979          ;*      TYPON                      ;;CALL FOR TYPEOUT
6980          ;*
6981          ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
6982          ;*CALL:
6983          ;*      MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
6984          ;*      TYPOC                      ;;CALL FOR TYPEOUT
6985
6986 031254 017646 000000          $TYPOS: MOV      2(SP),-(SP)          ;; PICKUP THE MODE
6987 031260 116637 000001 031477 MOVVB   1(SP),%SOFILL          ;; LOAD ZERO FILL SWITCH
6988 031266 112637 031501          MOVVB   (SP)+,%SOMODE+1          ;; NUMBER OF DIGITS TO TYPE
6989 031272 062716 000002          ADD     #2,(SP)                ;; ADJUST RETURN ADDRESS
6990 031276 000406
6991 031300 112737 000001 031477 $TYPOC: MOVVB   #1,%SOFILL          ;; SET THE ZERO FILL SWITCH
6992 031306 112737 000006 031501          MOVVB   #6,%SOMODE+1          ;; SET FOR SIX(6) DIGITS
6993 031314 112737 000005 031476 $TYPON: MOVVB   #5,%SOCNT          ;; SET THE ITERATION COUNT
6994 031322 010346
6995 031324 010446
6996 031326 010546
6997 031330 113704 031501          MOV     R3,-(SP)              ;; SAVE R3
6998 031334 005404          MOV     R4,-(SP)              ;; SAVE R4
6999 031336 062704 000006          MOV     R5,-(SP)              ;; SAVE R5
7000 031342 110437 031500          MOVVB   %SOMODE+1,R4          ;; GET THE NUMBER OF DIGITS TO TYPE
7001 031346 113704 031477          ADD     #6,R4
7002 031352 016605 000012          MOVVB   R4,%SOFILL,R4          ;; SUBTRACT IT FOR MAX. ALLOWED
7003 031356 005003          MOVVB   %SOFILL,R4          ;; SAVE IT FOR USE
7004 031360 006105          MOV     R3,%SOFILL,R4          ;; GET THE ZERO FILL SWITCH
7005 031362 000404          MOV     R3,%SOFILL,R4          ;; PICKUP THE INPUT NUMBER
7006 031364 006105          CLR     R3                    ;; CLEAR THE OUTPUT WORD
7007 031366 006105          ROL     R5                    ;; ROTATE MSB INTO "C"
7008 031370 006105          BR     3$                    ;; GO DO MSB
7009 031372 010503          ROL     R5                    ;; FORM THIS DIGIT
7010 031374 006103          ROL     R5,R3
7011 031376 105337 031500          MOV     R5,R3
7012 031402 100016          ROL     R3,R3
7013 031404 042703 177770          DEC     %SOCNT                ;; GET LSB OF THIS DIGIT
7014 031410 001002          BPL     #177770,R3            ;; TYPE THIS DIGIT
7015 031412 005704          BIC     #177770,R3            ;; BR IF NO
7016 031414 001403          BNE     #4$                    ;; GET RID OF JUNK
7017 031416 005204          TST     R4                    ;; TEST FOR 0
7018 031420 052703 000060          BEQ     #5$                    ;; SUPPRESS THIS 0?
7019 031424 052703 000040          INC     R4                    ;; BR IF YES
7020 031430 110337 031474          BIS     #'0,R3                ;; DON'T SUPPRESS ANYMORE 0'S
7021 031434 104401 031474          MOVVB   #'0,R3                ;; MAKE THIS DIGIT ASCII
7022 031440 105337 031476          MOVVB   R3,%SOFILL,R4          ;; MAKE ASCII IF NOT ALREADY
7023 031444 003347          TYPE    #8$                    ;; SAVE FOR TYPING
7024 031446 002402          DECB   %SOCNT                ;; GO TYPE THIS DIGIT
7025 031450 005204          BGT     #2$                    ;; COUNT BY 1
7026 031452 000744          BLT     #6$                    ;; BR IF MORE TO DO
7027 031454 012605          INC     R4                    ;; BR IF DONE
7028 031456 012604          BR     #7$                    ;; INSURE LAST DIGIT ISN'T A BLANK
7029 031460 012603          MOV     R5,(SP)+,R5          ;; GO DO THE LAST DIGIT
7030 031462 016666 000002 000004 6$: MOV     (SP)+,R5          ;; RESTORE R5
7031 031470 012616          MOV     (SP)+,R4          ;; RESTORE R4
7032 031472 000002          MOV     (SP)+,R3          ;; RESTORE R3
7033 031474 000          MOV     2(SP),4(SP)          ;; SET THE STACK FOR RETURNING
7034 031476 000          R11
7035 031478 000          ;*      .BYTE      C          ;; RETURN
7036 031480 000          ;*      .BYTE      0          ;; STORAGE FOR ASCII DIGIT
7037 031482 000          ;*      .BYTE      0          ;; TERMINATOR FOR TYPE ROUTINE

```

7035 031476 000
7036 031477 000
7037 031500 000000

\$COUNT: .BYTE 0 ;: OCTAL DIGIT COUNTER
\$OFILL: .BYTE 0 ;: ZERO FILL SWITCH
\$OMOD: .WORD 0 ;: NUMBER OF DIGITS TO TYPE

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

7038
7039
7040
7041
7042
7043
7044
7045
7046
7047
7048
7049
7050

; 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

7051 031502
7052 031502 010046
7053 031504 010146
7054 031506 010246
7055 031510 010346
7056 031512 010546
7057 031514 012746 020200
7058 031520 016605 000020
7059 031524 100004
7060 031526 005405
7061 031530 112756 000055 000001
7062 031536 005000 15:
7063 031540 012703 031716
7064 031544 112723 000040
7065 031550 005002 25:
7066 031552 016001 031706
7067 031556 160105 35:
7068 031560 002402
7069 031562 005202
7070 031564 000774
7071 031566 060105 45:
7072 031570 005702
7073 031572 001002
7074 031574 105716
7075 031576 100407
7076 031600 106316 55:
7077 031602 103003
7078 031604 116663 000001 177777
7079 031612 052702 000060
7080 031616 052702 000040
7081 031622 110223
7082 031624 005720
7083 031626 020027 000010
7084 031632 002746
7085 031634 003002
7086 031636 010502
7087 031640 000764
7088 031642 105726 85:
7089 031644 100003
7090 031646 116663 177777 177776

\$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 R0(SP), R5 ;: GET THE INPUT NUMBER
BPL 1\$;: BR IF INPUT IS POS.
NEG R5 ;: MAKE THE BINARY NUMBER POS.
MOV #-1, 1(SP) ;: MAKE THE ASCII NUMBER NEG.
CLR R0 ;: ZERO THE CONSTANTS INDEX
MOV #0, R3 ;: SETUP THE OUTPUT POINTER
MOV R0, (R3)+ ;: SET THE FIRST CHARACTER TO A BLANK
CLR R2 ;: CLEAR THE BCD NUMBER
MOV \$DTBL(R0), R1 ;: GET THE CONSTANT
SUB R1, R5 ;: FORM THIS BCD DIGIT
BLT 4\$;: BR IF DONE
INC R2 ;: INCREASE THE BCD DIGIT BY 1
BR 3\$
4\$:
ADD R1, R5 ;: ADD BACK THE CONSTANT
TST R2 ;: CHECK IF BCD DIGIT=0
BNE 5\$;: FALL THROUGH IF 0
TSTB (SP) ;: STILL DOING LEADING 0'S?
BMI 7\$;: BR IF YES
ASLB (SP) ;: MSD?
BCC 6\$;: BR IF NO
MOVB 1(SP), -1(R3) ;: YES--SET THE SIGN
BIS #'0, R2 ;: MAKE THE BCD DIGIT ASCII
BIS #' , R2 ;: MAKE IT A SPACE IF NOT ALREADY A DIGIT
MOV R2, (R3)+ ;: PUT THIS CHARACTER IN THE OUTPUT BUFFER
TST (R0)+ ;: JUST INCREMENTING
CMP R0, #10 ;: CHECK THE TABLE INDEX
BLT 2\$;: GO DO THE NEXT DIGIT
BGT 8\$;: GO TO EXIT
MOV R5, R2 ;: GET THE LSD
BR 6\$;: GO CHANGE TO ASCII
9\$:
TSTB (SP)+ ;: WAS THE LSD THE FIRST NON-ZERO?
BPL 9\$;: BR IF NO
MOVB -1(SP), -2(R3) ;: YES--SET THE SIGN FOR TYPING

```

7091 031654 105013          9$:  CLRB   (R3)           ;; SET THE TERMINATOR
7092 031656 012605          MOV    (SP)+,R5         ;; POP STACK INTO R5
7093 031660 012603          MOV    (SP)+,R3         ;; POP STACK INTO R3
7094 031662 012602          MOV    (SP)+,R2         ;; POP STACK INTO R2
7095 031664 012601          MOV    (SP)+,R1         ;; POP STACK INTO R1
7096 031666 012600          MOV    (SP)+,R0         ;; POP STACK INTO R0
7097 031670 104401 031716   TYPE    $DBLK          ;; NOW TYPE THE NUMBER
7098 031674 016666 000002 000004  MOV    2(SP),4(SP)      ;; ADJUST THE STACK
7099 031702 012616          MOV    (SP)+,(SP)
7100 031704 000002          RTI                    ;; RETURN TO USER
7101 031706 023420          $DTBL: 10000.
7102 031710 001750          1000.
7103 031712 000144          100.
7104 031714 000012          10.
7105 031716 000004          $DBLK: .BLKW 4
7106
7107          .SBTTL  TTY INPUT ROUTINE
7108
7109          ;*****
7110          .ENABL  LSB
7111
7112          ;*****
7113          ;SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
7114          ;ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
7115          ;SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
7116          ;WHEN OPERATING IN TTY FLAG MODE.
7117 031726 022737 000176 001140 $CKSWR: CMP    #SWREG,SWR      ;; IS THE SOFT-SWR SELECTED?
7118 031734 001074          BNE    15$             ;; BRANCH IF NO
7119 031736 105777 147202          TSTB   2$TKS          ;; CHAR THERE?
7120 031742 100071          BPL    15$             ;; IF NO, DON'T WAIT AROUND
7121 031744 117746 147176          MOVVB  2$TKB,(SP)     ;; SAVE THE CHAR
7122 031750 042716 177600          BIC    #1C177,(SP)   ;; STRIP-OFF THE ASCII
7123 031754 022726 000007          CMP    #7,(SP)+      ;; IS IT A CONTROL G?
7124 031760 001062          BNE    15$             ;; NO, RETURN TO USER
7125 031762 123727 001134 000001  CMPB   $AUTOB,#1     ;; ARE WE RUNNING IN AUTO-MODE?
7126 031770 001456          BEQ    15$             ;; BRANCH IF YES
7127
7128 031772 104401 032335          TYPE   , $CNTLG      ;; ECHO THE CONTROL-G (1G)
7129 031776 104401 032342          TYPE   , $MSWR      ;; TYPE CURRENT CONTENTS
7130 032002 013746 000176          MOV    $WREG,-(SP)   ;; SAVE SWREG FOR TYPEOUT
7131 032006 104402          TYPOC          ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
7132 032010 104401 032353          TYPE   , $MNEW      ;; PROMPT FOR NEW SWR
7133 032014 005046          19$:  CLR    -(SP)       ;; CLEAR COUNTER
7134 032016 005046          CLR    -(SP)         ;; THE NEW SWR
7135 032020 105777 147120          7$:  TSTB   2$TKS          ;; CHAR THERE?
7136 032024 100375          BPL    7$             ;; IF NOT TRY AGAIN
7137
7138 032026 117746 147114          MOVVB  2$TKB,-(SP)   ;; PICK UP CHAR
7139 032032 042716 177600          BIC    #1C177,(SP)   ;; MAKE IT 7-BIT ASCII
7140
7141
7142
7143 032036 021627 000025          9$:  CMP    (SP),#25    ;; IS IT A CONTROL-U?
7144 032042 001005          BNE    10$           ;; BRANCH IF NOT
7145 032044 104401 032330          TYPE   , $CNTLU      ;; YES, ECHO CONTROL-U (1U)
7146 032050 062706 000006          20$:  ADD    #6,SP      ;; IGNORE PREVIOUS INPUT

```

```

7147 032054 000757 BR 19$ ;;LET'S TRY IT AGAIN
7148
7149
7150 032056 021627 000015 10$: CMP (SP),#15 ;; IS IT A <CR>?
7151 032062 001022 BNE 16$ ;; BRANCH IF NO
7152 032064 005766 000004 TST 4(SP) ;; YES, IS IT THE FIRST CHAR?
7153 032070 001403 BEQ 11$ ;; BRANCH IF YES
7154 032072 016677 000002 147040 MOV 2(SP),2SWR ;; SAVE NEW SWR
7155 032100 062706 000006 11$: ADD #6,SP ;; CLEAR UP STACK
7156 032104 104401 001165 14$: TYPE $,SRLF ;; ECHO <CR> AND <LF>
7157 032110 123727 001135 000001 CMPB $,IN1HG,#1 ;; RE-ENABLE TTY KBD INTERRUPTS?
7158 032116 001003 BNE 15$ ;; BRANCH IF NOT
7159 032120 012777 000100 147016 MOV #100,2STKS ;; RE-ENABLE TTY KBD INTERRUPTS
7160 032126 000002 15$: RTI ;; RETURN
7161 032130 004737 031204 16$: JSR PC,STYPEC ;; ECHO CHAR
7162 032134 021627 000060 CMP (SP),#60 ;; CHAR < 0?
7163 032140 002420 BLT 18$ ;; BRANCH IF YES
7164 032142 021627 000067 CMP (SP),#67 ;; CHAR > 7?
7165 032146 003015 BGT 18$ ;; BRANCH IF YES
7166 032150 042726 000060 BIC #60,(SP)+ ;; STRIP-OFF ASCII
7167 032154 005766 000002 TST 2(SP) ;; IS THIS THE FIRST CHAR
7168 032160 001403 BEQ 17$ ;; BRANCH IF YES
7169 032162 006316 ASL (SP) ;; NO, SHIFT PRESENT
7170 032164 006316 ASL (SP) ;; CHAR OVER TO MAKE
7171 032166 006316 ASL (SP) ;; ROOM FOR NEW ONE.
7172 032170 005266 000002 17$: INC 2(SP) ;; KEEP COUNT OF CHAR
7173 032174 056616 177776 BIS -2(SP),(SP) ;; SET IN NEW CHAR
7174 032200 000707 BR 7$ ;; GET THE NEXT ONE
7175 032202 104401 001164 18$: TYPE $,SQUES ;; TYPE ?<CR><LF>
7176 032206 000720 BR 20$ ;; SIMULATE CONTROL-U
7177
7178
7179
7180
7181 *****
7182 *THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
7183 *CALL:
7184 * RDCHR ;; INPUT A SINGLE CHARACTER FROM THE TTY
7185 * RETURN HERE ;; CHARACTER IS ON THE STACK
7186 * ;; WITH PARITY BIT STRIPPED OFF
7187
7188 $RDCHR: MOV (SP),-(SP) ;; PUSH DOWN THE PC
7189 032212 016666 000004 000002 MOV 4(SP),2(SP) ;; SAVE THE PS
7190 032220 105777 146720 1$: TSTB 2STKS ;; WAIT FOR
7191 032224 100375 BPL 1$ ;; A CHARACTER
7192 032226 117766 146714 000004 MOVB 2STKB,4(SP) ;; READ THE TTY
7193 032234 042766 177600 000004 BIC #177,4(SP) ;; GET RID OF JUNK IF ANY
7194 032242 026627 000004 000023 CMP 4(SP),#23 ;; IS IT A CONTROL-S?
7195 032250 001013 BNE 3$ ;; BRANCH IF NO
7196 032252 105777 146666 2$: TSTB 2STKS ;; WAIT FOR A CHARACTER
7197 032256 100375 BPL 2$ ;; LOOP UNTIL ITS THERE
7198 032260 117746 146662 MOVB 2STKB,-(SP) ;; GET CHARACTER
7199 032264 042716 177600 BIC #177,(SP) ;; MAKE IT 7-BIT ASCII
7200 032270 022627 000021 CMP (SP)+,#21 ;; IS IT A CONTROL-Q?
7201 032274 001366 BNE 2$ ;; IF NOT DISCARD IT
7202 032276 000750 BR 1$ ;; YES, RESUME

```

```

7203 032300 026627 000004 000140 3$: CMP 4(SP),#140 ;; IS IT UPPER CASE?
7204 032306 002407 000000 000140 BLT 4$ ;; BRANCH IF YES
7205 032310 026627 000004 000175 CMP 4(SP),#175 ;; IS IT A SPECIAL CHAR?
7206 032316 003003 000000 000175 BGT 4$ ;; BRANCH IF YES
7207 032320 042766 000040 000004 BIC #40,4(SP) ;; MAKE IT UPPER CASE
7208 032326 000002 000000 000004 4$: RTI ;; GO BACK TO USER
7209 032330 052536 005015 000 SCNTLU: .ASCIZ /↑U/<15><12> ;; CONTROL "U"
7210 032335 136 006507 000012 SCNTLG: .ASCIZ /↑G/<15><12> ;; CONTROL "G"
7211 032342 005015 053523 020122 SMSWR: .ASCIZ <15><12>/SWR = /
7212 032350 020075 000 SMNEW: .ASCIZ / NEW = /
7213 032353 040 047040 053505
7214 032360 036440 000040

```

.SBTTL RANDOM NUMBER GENERATOR ROUTINE

```

;*****
;THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR
;WITH A RANGE OF 0 TO 2(+33)-1.
;CALL:
;* JSR PC,$RAND ;; CALL THE ROUTINE
;* RETURN ;; RETURN HERE THE RANDOM
;* ;; NUMBER WILL BE IN
;* ;; $HINUM,$LONUM

```

```

7228 032364 010046 032464 $RAND: MOV R0,-(SP) ;; PUSH R0 ON STACK
7229 032366 010146 032464 MOV R1,-(SP) ;; PUSH R1 ON STACK
7230 032370 010246 032464 MOV R2,-(SP) ;; PUSH R2 ON STACK
7231 032372 013700 032464 MOV $LONUM,R0 ;; SET R0 WITH LOW
7232 032376 013701 032464 MOV $HINUM,R1 ;; SET R1 WITH HIGH
7233 032402 012702 177771 MOV #-7,R2 ;; SET SHIFT COUNT
7234 032406 006300 1$: ASL R0 ;; SHIFT R0 LEFT AND
7235 032410 006101 ROL R1 ;; ROTATE CARRY INTO R1 AND
7236 032412 005202 INC R2 ;; CHECK FOR DONE
7237 032414 001374 BNE 1$ ;; CONTINUE SHIFT LOOP
7238 032416 063700 032464 ADD $LONUM,R0 ;; ADD NUMBER TO MAKE X 129
7239 032422 005501 ADC R1 ;; PROPOGATE CARRY
7240 032424 063701 032464 ADD $HINUM,R1 ;; ADD NUMBER TO MAKE X 129
7241 032430 062700 001057 ADD #1057,R0 ;; ADD LOW CONSTANT
7242 032434 005501 ADC R1 ;; PROPOGATE CARRY
7243 032436 062701 047401 ADD #47401,R1 ;; ADD HIGH CONSTANT
7244 032442 010037 032464 MOV R0,$LONUM ;; SAVE R0
7245 032446 010137 032464 MOV R1,$HINUM ;; SAVE R1
7246 032452 012602 MOV (SP)+,R2 ;; POP STACK INTO R2
7247 032454 012601 MOV (SP)+,R1 ;; POP STACK INTO R1
7248 032456 012600 MOV (SP)+,R0 ;; POP STACK INTO R0
7249 032460 000207 RTS PC ;; RETURN
7250 032462 176543 $HINUM: .WORD 176543
7251 032464 123456 $LONUM: .WORD 123456

```

.SBTTL SAVE AND RESTORE R0-R5 ROUTINES

```

;*****
;SAVE R0-R5
;CALL:
;* SAVREG

```

```

7259
7260
7261
7262
7263
7264
7265
7266
7267
7268
7269
7270
7271 032466 010046
7272 032466 010146
7273 032472 010246
7274 032474 010346
7275 032476 010446
7276 032500 010546
7277 032502 016646 000022
7278 032506 016646 000022
7279 032512 016646 000022
7280 032516 016646 000022
7281 032522 000002
7282
7283
7284
7285
7286 032524 012666 000022
7287 032524 012666 000022
7288 032530 012666 000022
7289 032534 012666 000022
7290 032540 012666 000022
7291 032544 012605
7292 032546 012604
7293 032550 012603
7294 032552 012602
7295 032554 012601
7296 032556 012600
7297 032560 000002
7298
7299
7300
7301
7302
7303
7304
7305
7306
7307
7308
7309
7310
7311
7312 032562 104412
7313 032564 016602 000002
7314 032570 012700 032742

```

;;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

```

\$\$SAVREG:

```

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 R4, -(SP) ;; PUSH R4 ON STACK
MOV R5, -(SP) ;; PUSH R5 ON STACK
MOV 22(SP), -(SP) ;; SAVE PS OF MAIN FLOW
MOV 22(SP), -(SP) ;; SAVE PC OF MAIN FLOW
MOV 22(SP), -(SP) ;; SAVE PS OF CALL
MOV 22(SP), -(SP) ;; SAVE PC OF CALL
RTI

```

;;RESTORE RO-R5

;;CALL:

;;RESREG:

```

RESREG:
MOV (SP)+, 22(SP) ;; RESTORE PC OF CALL
MOV (SP)+, 22(SP) ;; RESTORE PS OF CALL
MOV (SP)+, 22(SP) ;; RESTORE PC OF MAIN FLOW
MOV (SP)+, 22(SP) ;; RESTORE PS OF MAIN FLOW
MOV (SP)+, R5 ;; POP STACK INTO R5
MOV (SP)+, R4 ;; POP STACK INTO R4
MOV (SP)+, R3 ;; POP STACK INTO R3
MOV (SP)+, R2 ;; POP STACK INTO R2
MOV (SP)+, R1 ;; POP STACK INTO R1
MOV (SP)+, R0 ;; POP STACK INTO R0
RTI

```

.SBTTL DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

```

*****
*THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
*DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
*POSITIVE.

```

```

*CALL
* MOV #PNTR, -(SP) ;; POINTER TO LOW WORD OF BINARY NUMBER
* JSR PC, @#$DB2D
* RETURN ;; THE FIRST ADDRESS OF ASCII
;; IS ON THE STACK

```

```

$DB2D: SAVREG ;; SAVE REGISTERS
MOV 2(SP), R2 ;; PICKUP THE DATA POINTER
MOV #SDECVL, R0 ;; GET ADDRESS OF "SDECVL" STRING

```



```

7315 032574 010066 000032          MOV     R0,2(SP)          ;;PUT ADDRESS OF ASCIZ STRING ON STACK
7316 032600 012201          MOV     (R2)+,R1        ;;PICKUP THE BINARY NUMBER
7317 032602 012202          MOV     (R2)+,R2
7318 032604 012737 000012 032660  MOV     #10,4$          ;;SET UP TO DO 10 CONVERSIONS
7319 032612 012704 032672  MOV     #STNPNR,R4       ;;ADDRESS OF TEN POWER
7320 032616 012705 032674  MOV     #STNPNR+2,R5
7321 032622 005003          CLR     R3              ;;CLEAR PARTIAL
7322 032624 161401          SUB     (R4),R1         ;;SUBTRACT TEN POWER
7323 032626 005602          SBC     R2
7324 032630 161502          SUB     (R5),R2
7325 032632 002402          BLT     3$              ;;BR IF TEN POWER TOO LARGE
7326 032634 005203          INC     R3              ;;ADD 1 TO PARTIAL
7327 032636 000772          BR     2$              ;;LOOP
7328 032640 062401          ADD     (R4)+,R1       ;;RESTORE SUBTRACTED VALUE
7329 032642 005502          ADC     R2
7330 032644 062402          ADD     (R4)+,R2
7331 032646 022525          CMP     (R5)+,(R5)+    ;;MOVE TO NEXT TEN POWER
7332 032650 052703 000060  BIS     #0,R3           ;;CHANGE PARTIAL TO ASCII
7333 032654 110320          MOVB   R3,(R0)+        ;;SAVE IT
7334 032656 005327          DEC     (PC)+          ;;DONE?
7335 032660 000000          .WORD  0
7336 032662 001357          BNE     1$              ;;BR IF NO
7337 032664 105020          CLRB   (R0)+          ;;TERMINATOR
7338 032666 104413          RESREG          ;;RESTORE REGISTERS
7339 032670 000207          RTS     PC             ;;RETURN
7340 032672 145000          STNPNR: 145000        ;;1.0E09
7341 032674 035632          35632
7342 032676 160400          160400                ;;1.0E08
7343 032700 002765          2765
7344 032702 113200          113200                ;;1.0E07
7345 032704 000230          230
7346 032706 041100          041100                ;;1.0E06
7347 032710 000017          17
7348 032712 103240          103240                ;;1.0E05
7349 032714 000001          1
7350 032716 023420          23420                ;;1.0E04
7351 032720 000000          0
7352 032722 001750          1750                  ;;1.0E03
7353 032724 000000          0
7354 032726 000144          144                   ;;1.0E02
7355 032730 000000          0
7356 032732 000012          12                    ;;1.0E01
7357 032734 000000          0
7358 032736 000001          1                      ;;1.0E00
7359 032740 000000          0
7360 032742 000014          $DECVL: .BLKB 12.    ;;RESERVE STORAGE FOR ASCIZ STRING
7361
7362          .SBTTL DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
7363
7364          ;;*****
7365          ;;THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
7366          ;;UNSIGNED OCTAL ASCIZ NUMBER.
7367          ;;CALL
7368          ;;
7369          ;;      MOV     #PNTR, -(SP)          ;; POINTER TO LOW WORD OF BINARY NUMBER
7370          ;;      JSR     PC, @#$DB20         ;; CALL THE ROUTINE
7371          ;;      RETURN                    ;; THE ADDRESS OF THE FIRST ASCIZ CHAR. IS ON THE STACK

```

```

032756 104412
032760 016601 000002
032764 012705 033075
032770 012704 000014
032774 012703 177770
033000 012100
033002 012101
033004 005002
033006 110245
033010 010002
033012 005304
033014 003007
033016 001405
033020 005205
033022 010566 000002
033026 104413
033030 000207
033032 006203
033034 006001
033036 006000
033040 006001
033042 006000
033044 006001
033046 006000
033050 040302
033052 062702 000060
033056 000753
033060 000016
033076 010046
033100 016600 000002
033104 005740
033106 111000
033110 006300
033112 016000 033132
033116 000200
033120 011646
033122 016666 000004 000002
033130 000002

```

```

$DB20: SAVREG          ;; SAVE ALL REGISTERS
MOV      2(SP),R1      ;; PICKUP THE POINTER TO LOW WORD
MOV      #SOCTVL+13.,R5 ;; POINTER TO DATA TABLE
MOV      #12.,R4       ;; DO ELEVEN CHARACTERS
MOV      #C7,R3        ;; MASK
MOV      (R1)+,R0      ;; LOWER WORD
MOV      (R1)+,R1      ;; HIGH WORD
CLR      R2            ;; TERMINATOR
1$: MOVB  R2,-(R5)     ;; PUT CHARACTER IN DATA TABLE
MOV      R0,R2        ;; GET THIS DIGIT
DEC      R4           ;; COUNT THIS CHARACTER
BGT     3$           ;; BR IF NOT THE LAST DIGIT
BEQ     2$           ;; BR IF IT IS THE LAST DIGIT
INC      R5           ;; ALL DIGITS DONE-ADJUST POINTER FOR FIRST
MOV      R5,2(SP)     ;; ASCII CHAR. & PUT IT ON THE STACK
RESREG          ;; RESTORE ALL REGISTERS
RTS      PC          ;; RETURN TO USER
2$: ASR   R3          ;; POSITION THE MASK FOR THE LAST DIGIT
3$: ROR  R1           ;; POSITION THE BINARY NUMBER FOR
ROR     R0           ;; THE NEXT OCTAL DIGIT
ROR     R1
ROR     R0
ROR     R1
ROR     R0
ROR     R1
ROR     R0
BIC     R3,R2        ;; MASK OUT ALL JUNK
ADD     #0,R2        ;; MAKE THIS CHAR. ASCII
BR      1$          ;; GO PUT IT IN THE DATA TABLE
$SOCTVL: .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.

$TRAP: MOV  R0,-(SP)  ;; SAVE R0
MOV      2(SP),R0   ;; GET TRAP ADDRESS
TST     -(R0)       ;; BACKUP BY 2
MOVB   (R0),R0     ;; GET RIGHT BYTE OF TRAP
ASL    R0           ;; POSITION FOR INDEXING
MOV    $TRAPAD(R0),R0 ;; INDEX TO TABLE
RTS    R0           ;; GO TO ROUTINE

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

$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.

```

: 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
          $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
          $SAVREG ;;CALL=SAVREG   TRAP+12(104412) SAVE R0-R5 ROUTINE
          $RESREG ;;CALL=RESREG   TRAP+13(104413) RESTORE R0-R5 ROUTINE
          $DSPLY  ;;CALL=DISPLY   TRAP+14(104414) ROUTINE TO TYPE ERROR MESSAGES

$TERM=-$TRPAD

```

: *****

.SBTTL SINGLE/DUAL PORT RH11/RPD4/5 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

```

7468 033164 000000 000000 000000 RPERRS: .WORD 0,0,0,0
7469 033172 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

```

DRVACT: .BYTE 0      ;DRIVE 0
         .BYTE 00    ;DRIVE 1
         .BYTE 00    ;DRIVE 2
         .BYTE 00    ;DRIVE 3
         .BYTE 00    ;DRIVE 4
         .BYTE 00    ;DRIVE 5
         .BYTE 0     ;DRIVE 6

```

```

7421
7422
7423
7424
7425
7426
7427
7428
7429
7430
7431
7432 033132 033120
7433 033134 031034
7434 033136 031300
7435 033140 031254
7436 033142 031314
7437 033144 031502
7438
7439 033146 031776
7440
7441 033150 031726
7442 033152 032210
7443 033154 030232
7444 033156 032456
7445 033160 032524
7446 033162 027456
7447 000032
7448
7449
7450
7451
7452
7453
7454
7455
7456
7457
7458
7459
7460
7461
7462

```

```

75183 033203 000
75184
75185
75186
75187
75188
75189
75190
75191
75192
75193
75194
75195
75196
75197
75198
75199
75200
75201
75202
75203
75204
75205
75206
75207
75208
75209
75210
75211
75212
75213
75214
75215
75216
75217
75218
75219
75220
75221
75222
75223
75224
75225
75226
75227
75228
75229
75230
75231
75232
75233
75234
75235
75236
75237
75238
75239
75240
75241
75242

```

```

.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

DRVSTA: .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 DRIVE TYPES (DRVTYP=8 BYTES)
;DRVTYP=0 IF DRIVE IS NONEXISTENT (DRVSTA=). ALSO
;DRVTYP=1 IF DRIVE IS RPO4
;DRVTYP=2 IF DRIVE IS RPO5
;DRVTYP=4 IF DRIVE IS RPO6
;DRVTYP=-1 IF NOT RPO4/5/6

DRVTYP: .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

```

K11

CZRJDCO RPO4 5 6 MLT-DR LGC
CZRJDC.P11 15-DEC-77 10:58

MACY11 30(1046) 15-DEC-77 11:03 PAGE 141
SINGLE/DUAL PORT RH11/RPO4/5/6 DRIVER (REV 1.0)

SEG 0140

```

7539 033243 000 .BYTE 0 ;DRIVE 7
7540 ;TRANSFER WAIT FLAG (TRNSWT=1 WORD)
7541 ;THIS IS A ONE WORD QUEUE. IT WILL CONTAIN THE ADDRESS OF
7542 ;"DPB" OF THE I/O OPERATION.
7543
7544 033244 000000 TRNSWT: .WORD 0
7545 ;SEARCH WAIT KEYS (SRCHWT=1 WORD)
7546 ;THIS IS A ONE WORD QUEUE THAT WILL CONTAIN A KEY FOR EACH OF
7547 ;THE DRIVES THAT ARE PERFORMING A SEARCH COMMAND FOR THE I/O
7548 ;REQUEST THAT IS AT THE TOP OF THEIR REQUEST QUEUE.
7549 ;EACH DRIVE IS ASSIGNED ONE BIT, STARTING AT BIT 00 FOR DRIVE 0.
7550
7551 033246 000000 SRCHWT: .WORD 0
7552 ;RPO4/5/6 DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)
7553 ;ACTDRV=0 IF DRIVER IS INACTIVE
7554 ;ACTDRV>0 IF DRIVER IS ACTIVE
7555
7556 033250 000 ACTDRV: .BYTE 0
7557 ;SOFTWARE TIMER ROUTINE ACTIVE FLAG (ACTSTR=1 BYTE)
7558 ;ACTSTR=0 IF SOFTWARE TIMER ROUTINE IS INACTIVE
7559 ;ACTSTR>0 IF SOFTWARE TIMER ROUTINE IS ACTIVE
7560
7561 033251 000 ACTSTR: .BYTE 0
7562 ;UNLOAD FLAG (ULDFLG=8 BYTES)
7563 ;ULDFLG=0 IF NO UNLOAD COMMAND
7564 ;ULDFLG>0 IF UNLOAD COMMAND IN PROGRESS
7565 ;ULDFLG<0 IF UNLOAD COMMAND IN WAIT QUEUE
7566
7567 033252 000 ULDFLG: .BYTE 0 ;DRIVE 0
7568 033253 000 .BYTE 0 ;DRIVE 1
7569 033254 000 .BYTE 0 ;DRIVE 2
7570 033255 000 .BYTE 0 ;DRIVE 3
7571 033256 000 .BYTE 0 ;DRIVE 4
7572 033257 000 .BYTE 0 ;DRIVE 5
7573 033260 000 .BYTE 0 ;DRIVE 6
7574 033261 000 .BYTE 0 ;DRIVE 7
7575
7576 ;LOOK AHEAD COUNT (LACNT=8 BYTES)
7577 ;LACNT WILL INDICATE THE NUMBER OF LOOK AHEADS PERFORMED
7578
7579 033262 000 LACNT: .BYTE 0 ;DRIVE 0
7580 033263 000 .BYTE 0 ;DRIVE 1
7581 033264 000 .BYTE 0 ;DRIVE 2
7582 033265 000 .BYTE 0 ;DRIVE 3
7583 033266 000 .BYTE 0 ;DRIVE 4
7584 033267 000 .BYTE 0 ;DRIVE 5
7585 033270 000 .BYTE 0 ;DRIVE 6
7586 033271 000 .BYTE 0 ;DRIVE 7
7587
7588 ;SAVE REGISTERS FLAG (SAVEFG =1 WORD)
7589 ;SAVEFG <0 IF SAVE THE RH11/RPO4.5.6 REGISTERS WHEN THE
7590
7591
7592
7593
7594

```

```

7595
7596
7597
7598
7599 033272 000000
7600
7601
7602
7603
7604
7605
7606
7607 033274 000000
7608
7609
7610
7611
7612 033276 177777
7613 033300 177777
7614 033302 177777
7615 033304 177777
7616 033306 177777
7617 033310 177777
7618 033312 177777
7619 033314 177777
7620
7621
7622
7623
7624
7625 033316 177777
7626
7627
7628
7629
7630
7631 033320 001
7632 033321 002
7633 033322 004
7634 033323 010
7635 033324 020
7636 033325 040
7637 033326 100
7638 033327 200
7639
7640
7641
7642
7643 033330 000003
7644
7645
7646
7647
7648 033332 176700
7649 033334 000254 000240
7650

```

```

; OPERATION IS COMPLETED AS PER (DPB+14).
; SAVEFG=0 IF SAVE THE RH11/RPO4/S/6 REGISTERS, AS PER
; (DPB+14), AFTER AN ERROR.
SAVEFG: .WORD 0

; SEEK FLAG (SEEKFG=1 WORD)
; SEEKFG=0 IF WHEN THE DISK ADDRESS ISN'T IN THE WINDOW
; FOR A DATA TRANSFER START A SEARCH COMMAND
; SEEKFG=0 IF DATA TRANSFER WILL DO IMPLIED SEEKS.
; DISREGARD THE WINDOW
SEEKFG: .WORD 0

; TIMEOUT TABLE (TIMER=8 WORDS)
; THIS TABLE CONTAINS THE TIME ALLOWED FOR AN OPERATION
TIMER: .WORD -1 ;DRIVE 0
       .WORD -1 ;DRIVE 1
       .WORD -1 ;DRIVE 2
       .WORD -1 ;DRIVE 3
       .WORD -1 ;DRIVE 4
       .WORD -1 ;DRIVE 5
       .WORD -1 ;DRIVE 6
       .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
DTUW: .WORD -1

; ATTENTION BITS TABLE (ATABIT=8 BYTES)
; THIS TABLE CONTAINS THE CORRESPONDING BIT TO EACH DRIVES
; ATTENTION BIT
ATABIT: .BYTE 1 ;DRIVE 0
        .BYTE 2 ;DRIVE 1
        .BYTE 4 ;DRIVE 2
        .BYTE 10 ;DRIVE 3
        .BYTE 20 ;DRIVE 4
        .BYTE 40 ;DRIVE 5
        .BYTE 100 ;DRIVE 6
        .BYTE 200 ;DRIVE 7

; RPO4/S/6 TO RH11 "MASSBUS CONTROL BUS PARITY ERRORS" (MCPE) ALLOWED BEFORE
; CALLING IT FATAL (MCPEM=1 WORD)
MCPEM: .WORD 3

; STORAGE FOR RPADR (THE FIRST ADDRESS (776700) OF THE RH11/RPO4/S/6).
; RPVEC (THE VECTOR ADDRESS (254)), AND RPVEC+2 (THE BR LEVEL (5)).
RPADR: .WORD 176700
RPVEC: .WORD 254,5*32.

```

M11

CZRJDC0, RPO4 5 6 MLT-DP LGC
CZRJDC.P11 15-DEC-77 10:59

MACY11 30(1046) 15-DEC-77 11:03 PAGE 143
SINGLE/DUAL PORT RH11/RPO4/5/6 DRIVER (REV 1.0)

SEQ 0142

```

7651 ;MAXIMUM NUMBER OF LOOK AHEADS ALLOWED IS 4 (MXLACT=1 WORD)
7652
7653 033340 000004 MXLACT: .WORD 4
7654 ;MAXIMUM DELTA DELAY IS 8 SECTORS (MXDLTA=1 WORD)
7655
7656 033342 001000 MXDLTA: .WORD 8.*64.
7657 ;MINIMUM DELTA DELAY IS 2 SECTORS (MNDLTA=1 WORD)
7658
7659 033344 000200 MNDLTA: .WORD 2.*64.
7660 ;MAXIMUM SEARCH FOR I/O WINDOW IS 5 SECTORS (MXWNDW=1 WORD)
7661
7662 033346 000005 MXWNDW: .WORD 5
7663
7664 ;DEFINITIONS OF THE RH11/RPO4/5/6 ADDRESS INDEXES
7665
7666 000000 RPCS1=0 ;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 00)
7667 000002 RPWC=2 ;WORD COUNT REGISTER (NOT A DRIVE REG)
7668 000004 RPBA=4 ;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)
7669 000006 RPOA=6 ;DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 05)
7670 000010 RPCS2=10 ;CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
7671 000012 RPOS1=12 ;DRIVE STATUS REGISTER (DRIVE REG 01)
7672 000014 RPER1=14 ;ERROR REGISTER #1 (DRIVE REG. 02)
7673 000016 RPAS=16 ;ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 04)
7674 000020 RPLA=20 ;LOOK AHEAD REGISTER (DRIVE REG. 07)
7675 000022 RPOB=22 ;DATA BUFFER REGISTER (NOT A DRIVE REG.)
7676 000024 RPMR=24 ;MAINTAINABILITY REGISTER (DRIVE REG. 03)
7677 000026 RPOT=26 ;DRIVE TYPE REGISTER (DRIVE REG. 06)
7678 000030 RPSN=30 ;SERIAL NUMBER REGISTER (DRIVE REG. 10)
7679 000032 RPOF=32 ;OFFSET REGISTER (DRIVE REG. 11)
7680 000034 RPCA=34 ;DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
7681 000036 RPCC=36 ;CURRENT CYLINDER ADDRESS REGISTER (DRIVE REG. 13)
7682 000040 RPER2=40 ;ERROR REGISTER #2 (DRIVE REG. 14)
7683 000042 RPER3=42 ;ERROR REGISTER #3 (DRIVE REG. 15)
7684 000044 RPEC1=44 ;ECC POSITION REGISTER (DRIVE REG. 16)
7685 000046 RPEC2=46 ;ECC PATTERN REGISTER (DRIVE REG. 17)
7686
7687 ;RH11/RPO4/5/6 DRIVER INITIALIZATION CODE
7688 ;THIS ROUTINE WILL DETERMINE WHICH RPO4/5/6 DRIVES ARE
7689 ;AVAILABLE FOR TESTING AND SET THE DRVSTA INDICATOR
7690 ;TO THE PROPER STATE FOR EACH DRIVE.
7691 ;NOTE: THIS ROUTINE CALLS DRVINT
7692
7693 ;CALL
7694
7695 ; JSR PC,RPINIT
7696 ; RETURN
7697
7698 ;NOTE: THE 'P' OR 'L' CLOCK MUST BE STARTED
7699
7700 033350 104412 RPINIT: SAVREG ;SAVE RC - R5
7701 033352 013746 177776 MOV #PS, -(SP) ;SAVE THE PRESENT PROCESSOR STATUS
7702 033356 012737 000240 177776 MOV #(<5*32.>), #PS ;CHANGE THE PRIORITY TO 5
7703 033364 004737 041432 JSR PC, CLRQUE ;CLEAR ALL REQUEST QUEUES
7704 033370 012701 033164 MOV #RPERRS, R1 ;FIRST ADDRESS TO BE CLEARED
7705 033374 012702 033274 MOV #SEEKFG, R2 ;LAST ADDRESS TO BE CLEARED
7706 033400 005021 IS: CLR (R1)+ ;CLEAR

```

N11

```

7707 033402 020102      CMP      R1,R2      ;ARE WE DONE?
7708 033404 101775      BLOS     1$         ;BRANCH IF NO
7709 033406 012702 033316     MOV      #DTUW,R2   ;LAST ADDRESS
7710 033412 012721 177777     2$:      MOV      #-1,(R1)+ ;INITIALIZE
7711 033416 020102      CMP      R1,R2      ;DONE?
7712 033420 101774      BLOS     2$         ;LOOP IF NO
7713 033422 005037 033204     CLR      DRVSTA     ;SET ALL DRIVES TO OFFLINE
7714 033426 005037 033206     CLR      DRVSTA+2
7715 033432 005037 033210     CLR      DRVSTA+4
7716 033436 005037 033212     CLR      DRVSTA+6
7717 033442 013703 033334     MOV      RPVEC,R3   ;SETUP THE RH11/RPO4/5/6 VECTOR
7718 033446 012723 036214     MOV      #ISR,(R3)+
7719 033452 013713 033336     MOV      RPVEC+2,(R3)
7720 033456 013704 033332     MOV      RPADR,R4   ;FIRST ADDRESS OF RH11/RPO4
7721 033462 012764 000040 000010     MOV      #BIT05,RPCS2(R4) ;MASSBUS INIT
7722 033470 005001      CLR      R1         ;START WITH DRIVE 0
7723 033472 004037 033562     3$:      JSR      RO,DRVINT  ;INIT THE DRIVE
7724 033476 000401      BR      4$         ;'DVA' NOT SET OR PARITY ERROR
7725 033500 000402      BR      5$         ;NORMAL RETURN
7726 033502 105061 033204     4$:      CLRB    DRVSTA(R1) ;SET DRIVE STATUS TO OFFLINE
7727 033506 005201     5$:      INC     R1         ;GO TO NEXT DRIVE
7728 033510 042701 177770     BIC     #1C7,R1     ;MASK OUT UNUSED BITS
7729 033514 001366      BNE     3$         ;BR IF MORE DRIVES TO GO
7730 033516 012701 000007     MOV      #7,R1     ;START WITH DRIVE 7
7731 033522 005037 177776     CLR      #PS       ;CLEAR THE PROCESSOR STATUS
7732 033526 105761 033224     6$:      TSTB   DPINT(R1)  ;WAITING FOR DRIVE TO SWITCH PORTS ?
7733 033532 001405      BEQ     8$         ;BR NOT WAITING
7734 033534 004737 041066     JSR      PC,SET.IE ;SET INTERRUPT
7735 033540 105761 033224     7$:      TSTB   DPINT(R1)  ;DRIVE SWITCHED PORTS ?
7736 033544 001375      BNE     8$         ;BR IF NOT
7737 033546 005301     8$:      DEC     R1         ;GO TO THE NEXT DRIVE
7738 033550 100366      BPL     6$         ;CHECK NEXT DRIVE
7739 033552 012637 177776     MOV      (SP)+,#PS  ;RESTORE THE PROCESSOR STATUS
7740 033556 104413      RESREG ;RESTORE RO - RS
7741 033560 000207      RTS     PC         ;BYE-BYE
7742
7743 ;DRIVE INITIALIZATION ROUTINE
7744 ;THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
7745 ;AN RPO4/5/6. IF IT IS, A "READ-IN PRESET" IS ISSUED AND FMT22
7746 ;IS SET TO A "1". THEN MOL, DPR, DRY, AND VV ARE CHECKED TO
7747 ;INSURE THEY ARE ALL ON A "1". AND DEPENDING ON THEIR STATE,
7748 ;DRVSTA IS SET TO THE PROPER CONDITION.
7749
7750 ;CALL
7751 :      MOV      #DRVNUM,R1 ;DRIVE NUMBER TO R1
7752 :      MOV      RPADR,R4   ;UNIBUS ADDRESS OF RH11/RPO4/5/6 (RPCS1)
7753 :      JSR      RO,DRVINT  ;CALLED BY A JSR
7754 :      RETURN1 ;ERROR OCCURRED (PARITY)
7755 :      RETURN2 ;NORMAL RETURN
7756 :
7757 DRVINT: MOV      R5,-(SP)     ;SAVE R5
7758 033564 105061 033204     CLRB    DRVSTA(R1)  ;START DRIVE STATUS AS OFFLINE
7759 033570 105061 033214     CLRB    DRVSTYP(R1) ;CLEAR THE DRIVE TYPE INDICATOR
7760 033574 105061 033252     CLRB    ULDFLG(R1)  ;CLEAR THE UNLOAD FLAG
7761 033600 010164 000010     MOV      R1,RPCS2(R4) ;SELECT A DRIVE
7762 033604 112764 000111 000000     MOVB   #111,RPCS1(R4) ;DO A DRIVE CLEAR COMMAND (3 SEIZE DRIVE

```


7763	033612	032764	010000	000010	BIT	#BIT12,RPCS2(R4)	:NONEXISTENT DRIVE?
7764	033620	001403			BEQ	15	:NO---BRANCH
7765	033622	004737	041066		JSR	PC,SET.IE	:GO SET "IE" WITHOUT A "TRE"
7766	033626	000520			BR	65	:LEAVE THIS ROUTINE
7767	033630	105061	033204	15:	CLRB	DRVSTA(R1)	:SET DRIVE STATUS TO OFFLINE
7768	033634	032764	004000	000000	BIT	#BIT11,RPCS1(R4)	:SEE IF DRIVE AVAILABLE
7769	033642	001514			BEQ	75	:BR IF DRIVE NOT AVAILABLE
7770	033644	004037	040406		JSR	RO,RD.RP	:READ THE DRIVE TYPE REG.
7771	033650	000026			RPOF		
7772	033652	034114			85		:ERROR RETURN ADDRESS
7773	033654	012605			MOV	(SP)+,R5	:PUT DRIVE TYPE IN R5
7774	033656	112761	000001	033214	MOVB	#1,DRVTYP(R1)	:SET RPO4 INDICATOR
7775	033664	022705	020020		CMP	#20020,R5	:IS IT A SINGLE PORT RPO4?
7776	033670	001431			BEQ	25	:BRANCH IF YES
7777	033672	022705	024020		CMP	#24020,R5	:IS IT A DUAL PORT RPO4?
7778	033676	001426			BEQ	25	:BR IF YES
7779	033700	112761	000002	033214	MOVB	#2,DRVTYP(R1)	:SET RPO5 INDICATOR
7780	033706	022705	020021		CMP	#20021,R5	:SINGLE PORT RPO5 ?
7781	033712	001420			BEQ	25	:BR IF YES
7782	033714	022705	024021		CMP	#24021,R5	:DUAL PORT RPO5 ?
7783	033720	001415			BEQ	25	:BR IF YES
7784	033722	112761	000004	033214	MOVB	#4,DRVTYP(R1)	:SET RPO6 INDICATOR
7785	033730	022705	020022		CMP	#20022,R5	:SINGLE PORT RPO6 ?
7786	033734	001407			BEQ	25	:BR IF YES
7787	033736	022705	024022		CMP	#24022,R5	:DUAL PORT RPO6 ?
7788	033742	001404			BEQ	25	:BR IF YES
7789	033744	112761	177777	033214	MOVB	#-1,DRVTYP(R1)	:SET INDICATOR TO 'OTHER'
7790	033752	000446			BR	65	:EXIT
7791	033754	012746	000121	25:	MOV	#121,-(SP)	:DO A "READ-IN PRESET"
7792	033760	004037	040562		JSR	RO,WRT.RP	
7793	033764	000000			RPCS1		
7794	033766	034114			85		
7795	033770	012746	010000		MOV	#BIT12,-(SP)	:SET FMT22=1
7796	033774	004037	040562		JSR	RO,WRT.RP	
7797	034000	000032			RPOF		
7798	034002	034114			85		
7799	034004	004037	040406		JSR	RO,RD.RP	:READ RPO51
7800	034010	000012			RPDS1		
7801	034012	034114			85		
7802	034014	012605			MOV	(SP)+,R5	:AND SAVE IT IN R5
7803	034016	100015			BPL	45	:BRANCH IF ATA=0
7804	034020	116164	033320	000016	MOVB	ATABIT(R1),RPAS(R4)	:CLEAR ATTENTION BIT
7905	034026	004037	040406		JSR	RO,RD.RP	:FIND OUT WHY ATA=1
7806	034032	000014			RPER1		
7807	034034	034114			85		
7808	034036	006126			ROL	(SP)+	:IS IT UNSAFE?
7809	034040	100004			BPL	45	:BR IF NOT
7810	034042	112761	177777	033204	MOVB	#-1,DRVSTA(R1)	:SET UNSAFE INDICATOR
7811	034050	000407			BR	65	:EXIT
7812	034052	005105	45:		COM	R5	:CHECK MOL, DPR, DRY, AND VV
7813	034054	042705	167077		BIC	#1<BIT12!BIT08!BIT07!BIT06>,R5	
7814	034060	001003			BNE	65	:BRANCH IF MOL, DPR, DRY, OR VV IS CLEAR
7815	034062	112761	000001	033204	MOVB	#1,DRVSTA(R1)	:SET DRIVE STATUS TO ONLINE
7816	034070	005720	65:		(R0)+		:STEP OVER THE ERROR RETURN
7817	034072	000410			BR	85	:EXIT
7818	034074	006301	75:		ASL	R1	:CHANGE INDEX TO ADDRESS WORDS

```

7819 034076 012761 003720 033276      MOV      #2000.,TIMER(R1)      ;START 2 SEC TIMER
7820 034104 006201                ASR      R1                    ;RESTORE R1
7821 034106 105161 033224      COMB    DPINT(R1)             ;SET PORT INITIALIZE INDICATOR
7822 034112 005720                TST     (R0)+                 ;
7823 034114 012605      8$:     MOV      (SP)+,R5      ;RESTORE R5
7824 034116 000200                RTS      R0                    ;EXIT
7825
7826                ;REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
7827                ;CALL
7828                ;
7829                ;
7830                ;
7831                ;
7832                ;
7833                ;
7834                ;
7835                ;
7836 034120 013746 177776      RPO4:   MOV      3#PS, -(SP)      ;SAVE THE CALLING STATUS
7837 034124 013737 033336 177776      MOV      RVEC+2, 2#PS      ;DON'T ALLOW ANY RPO4/5/6 INTERRUPTS
7838 034132 112737 000001 033250      MOVVB   #1,ACTDRV         ;SET "ACTIVE DRIVER" FLAG
7839 034140 104412                SAVREG                    ;SAVE R0 - R5
7840 034142 011002                MOV      (R0),R2           ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
7841 034144 005062 000016      CLR     16(R2)            ;CLEAR THE STATUS/ERROR INDICATOR
7842 034150 111201                MOVVB   (R2),R1           ;PICKUP THE DRIVE NUMBER
7843 034152 013704 033332      MOV      RPAR,R4          ;UNIBUS ADDRESS OF RPO4
7844 034156 105761 033204      TSTB   DRVSTA(R1)        ;CHECK DRIVES STATUS
7845 034162 003014                BGT     1$                ;BRANCH IF ONLINE
7846 034164 105761 033252      TSTB   ULDFLG(R1)        ;UNLOAD COMMAND IN QUEUE?
7847 034170 001036                BNE     3$                ;BRANCH IF YES
7848 034172 105761 033224      TSTB   DPINT(R1)         ;TRYING TO INIT THE DRIVE
7849 034176 001042                BNE     5$                ;BR IF YES
7850 034200 004037 033562      JSR    R0,DRVINT         ;GO INIT. THE DRIVE
7851 034204 000434                BR     4$                ;ERROR RETURN
7852 034206 105761 033204      TSTB   DRVSTA(R1)        ;IS DRIVE STATUS ONLINE?
7853 034212 003445                BLE     6$                ;BR IF NOT
7854 034214 105761 033234      1$:    TSTB   DPRQS(R1)      ;OUTSTANDING PORT REQUEST FOR THE DRIVE
7855 034220 001031                BNE     5$                ;BR IF YES
7856 034222 010164 000010      MOV      R1,RPCS2(R4)     ;SELECT THE DRIVE
7857 034226 004037 041530      JSR    R0,DRVGUF         ;PUT THIS REQUEST IN QUEUE
7858 034232 000460                BR     9$                ;QUEUE IS FULL
7859 034234 122762 000103 000002      CMPB   #103,2(R2)        ;IS THIS REQ. FOR AN UNLOAD?
7860 034242 001003                BNE     2$                ;BR IF NO
7861 034244 112761 177777 033252      MOVVB   #-1,ULDFLG(R1)   ;SET THE "UNLOAD IN QUEUE" FLAG
7862 034252 105761 033174      2$:    TSTB   DRVACT(R1)     ;IS THIS DRIVE ACTIVE?
7863 034256 001043                BNE     8$                ;BR IF YES
7864 034260 004737 034412      JSR    PC,OPT            ;CALL THE OPTIMIZER
7865 034264 000440                BR     8$                ;
7866 034266 012762 120000 000016 3$:    MOV      #BIT15!BIT13,16(R2) ;SET THE "UNLOAD IN QUEUE" ERROR FLAG
7867 034274 000434                BR     8$                ;EXIT
7868 034276 004737 035522      4$:    JSR    PC,C17          ;GO HANDLE THE PARITY ERROR
7869 034302 000431                BR     8$                ;
7870 034304 004037 041530      5$:    JSR    R0,DRVQUE        ;PUT REQUEST IN QUEUE
7871 034310 000431                BR     9$                ;QUEUE IS FULL
7872 034312 032714 000100      BIT    #BIT06,(R4)       ;IS 'IE' SET ALREADY?
7873 034316 001023                BNE     8$                ;BR IF IT IS
7874 034320 004737 041066      JSR    PC,SET.IE        ;SET INTERRUPT

```

```

7875 034324 000420
7876 034326 105761 033204
7877 034332 002412
7878 034334 012762 140000 000016
7879 034342 105761 033214
7880 034346 001007
7881 034350 012762 100002 000016
7882 034356 000403
7883 034360 012762 110000 000016
7884 034366 104413
7885 034370 005720
7886 034372 000401
7887 034374 104413
7888 034376 005720
7889 034400 105037 033250
7890 034404 012637 177776
7891 034410 000200
7892
7893
7894
7895
7896
7897
7898
7899 034412 104412
7900 034414 013746 177776
7901 034420 146137 033320 033246
7902 034426 004737 041604
7903 034432 005702
7904 034434 001505
7905 034436 032764 004000 000000
7906 034444 001407
7907 034446 032764 000100 000012
7908 034454 001003
7909 034456 004037 033562
7910 034462 000470
7911 034464 105761 033204
7912 034470 003014
7913 034472 004737 041626
7914 034476 012762 140000 000016
7915 034504 105761 033204
7916 034510 100064
7917 034512 012762 110000 000016
7918 034520 000460
7919 034522 012746 000111
7920 034526 004037 040562
7921 034532 000000
7922 034534 034644
7923 034536 032714 004000
7924 034542 001427
7925 034544 122762 000150 000002
7926 034552 002403
7927 034554 004737 035106
7928 034560 000440
7929 034562 005737 033316
7930 034566 002012

BR 8$ ;RETURN, REQUEST IN QUEUE
TSTB 6$: DRVSTA(R1) ;SEE IF DRIVE OFFLINE OR UNSAFE
BLT 7$ ;BR IF UNSAFE
MOV #BIT15!BIT14,16(R2) ;SET OFFLINE ERROR INDICATOR
TSTB DRVSTYP(R1) ;SEE IF OFFLINE OR NONEXISTENT
BNE 8$ ;BR IF OFFLINE
MOV #BIT15!BIT01,16(R2) ;REPORT DRIVE NONEXISTENT
BR 8$ ;GO TO EXIT
MOV 7$: #BIT15!BIT12,16(R2) ;DRIVE IS UNSAFE
8$: RESREG ;RESTORE R0 - R5
TST (R0)+ ;SETUP FOR NORMAL RETURN
BR 10$ ;FINISH UP THEN EXIT
RESREG 9$: ;RESTORE R0 - R5
TST (R0)+ 10$: ;CORRECT THE RETURN ADDRESS
CLRE ACTDRV ;CLEAR "ACTIVE DRIVER" FLAG
MOV (SP)+,2*PS ;RETURN "PS" TO USER LEVEL
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
BICB 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 #BIT11,RPCS1(R4) ;IS DVA SET?
BEQ 10$ ;BRANCH IF NOT
BIT #BIT6,RPDS1(R4) ;IS VV SET?
BNE 10$ ;BR IF IT IS
JSR R0,DRVINT 9$: ;SEE IF DRIVE STILL ONLINE?
BR 6$ ;PARITY OR 'DVA' NOT SET
TSTB DRVSTA(R1) 10$: ;IS DRIVE ONLINE?
BGT 1$ ;YES--BRANCH
JSR PC,POPQUE ;NO--REMOVE REQUEST FROM QUEUE
MOV #BIT15!BIT14,16(R2) ;SET OFFLINE STATUS/ERROR INDICATOR
TSTB 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
MOV #111 -(SP) 1$: ;LOAD COMMAND ONTO THE STACK
JSR R0,WRP ;LOAD THE REGISTER
RPCS1 ;REGISTER INCREMENT
6$ ;ERROR RETURN ADDRESS
BIT #BIT11,(R4) ;DRIVE AVAILABLE?
BEQ 5$ ;BR IF NOT
CMPB #150,2(R2) ;IS THE REQUEST FOR I/O?
BLT 2$ ;YES--BRANCH
JSP PC,C14 ;CALL THE COMMAND INITIATOR
BR 8$ ;BRANCH TO EXIT
TST DTUW 2$: ;DATA TRANSFER UNDERWAY?
BGE 4$ ;YES--GO START A SEARCH

```

```

7931 034570 005737 033274      TST      SEEKFG      ;DO IMPLIED SEEKS?
7932 034574 100404      BMI      3$          ;YES---BRANCH
7933 034576 004037 036056      JSR      RO,LA       ;NO--DO LOOK AHEAD
7934 034602 000427      BR       8$          ;RETURN HERE ON A PARITY ERROR
7935 034604 000403      BR       4$          ;GO START A SEARCH
7936 034606 004737 034672      3$:     JSR      PC,C11 ;START A DATA TRANSFER
7937 034612 000423      BR       8$
7938 034614 004737 035000      4$:     JSR      PC,C13 ;START A SEARCH
7939 034620 000420      BR       8$          ;GO TO THE EXIT
7940 034622 112761 177777 033234 5$:     MOVB     #-1,DPRQS(R1) ;SET PORT REQUEST INDICATOR
7941 034630 010103      MOV     R1,R3        ;SET UP TO ADDRESS WORDS
7942 034632 006303      ASL     R3           ;CONVERT TO WORD INDEX
7943 034634 012763 023420 033276 7$:     MOV     #10000.,TIMER(R3) ;START 10 SEC TIMER
7944 034642 000402      BR       7$          ;EXIT
7945 034644 004737 035522      6$:     JSR      PC,C17   ;PROCESS THE PARITY ERROR
7946 034650 032714 000100      7$:     BIT     #BIT06,(R4) ;SEE IF 'IE' ALREADY SET
7947 034654 001002      BNE     8$          ;BR IF SET
7948 034656 004737 041066      JSR      PC,SET.IE  ;SET "IE" WITHOUT A "TRE"
7949 034662 012637 177776      8$:     MOV     (SP)+,2#PS ;RESTORE PROC. STATUS
7950 034666 104413      RESREG
7951 034670 000207      RTS     PC          ;RESTORE RO - RS
7952
7953      ;COMMAND INITIATOR
7954      ;CALL
7955      ;
7956      ;
7957      ;
7958      ;
7959      ;
7960      ;
7961      ;
7962      ;
7963      ;
7964 034672 004737 041626      C11:    JSR      PC,POPQUE  ;REMOVE REQUEST FROM "DRIVES WAIT" QUEUE
7965 034676 010237 033244      MOV     R2,TRANSW   ;PUT REQ. IN TRANSFER WAIT QUEUE
7966 034702 010203      MOV     R2,R3        ;DPB ADDRESS TO R3
7967 034704 013704 033332      MOV     RPADR,R4     ;RPCS1 ADDRESS
7968 034710 010164 000010      MOV     R1,RPCS2(R4) ;SELECT DRIVE
7969 034714 062703 000004      ADD     #4,R3        ;DESIRED WORD COUNT
7970 034720 062704 000002      ADD     #2,R4        ;RPWC ADDRESS
7971 034724 012324      MOV     (R3)+,(R4)+  ;LOAD WORD COUNT
7972 034726 012324      MOV     (R3)+,(R4)+  ;LOAD BUFFER ADDRESS
7973 034730 012346      MOV     (R3)+,-(SP)  ;LOAD SECTOR AND TRACK
7974 034732 004037 040562      JSR      RO,WRT.RP   ;CALL THE LOAD(WRITE) ROUTINE
7975 034736 000006      RPDA
7976 034740 035522      C17
7977 034742 012346      MOV     (R3)+,-(SP)  ;INDEX OF REGISTER TO LOAD
7978 034744 004037 040562      JSR      RO,WRT.RP   ;ERROR RETURN ADDRESS
7979 034750 000034      RPCA
7980 034752 035522      C17
7981 034754 016246 000002      MOV     2(R2)-,(SP)  ;LOAD "COMMAND+GO", "A17&A16". AND "PSEL"
7982 034760 004037 040562      JSR      RO,WRT.RP
7983 034764 000000      RPCS1
7984 034766 035522      C17
7985 034770 010137 033316      MOV     R1,DTUW
7986 034774 000137 035464      JMP     C15          ;SET "DATA TRANSFER UNDERWAY"

```

7987	035000	013704	033332		C13:	MOV	RPADR,R4	;RPCS1 ADDRESS
7988	035004	010164	000010			MOV	R1,RPCS2(R4)	;SELECT DRIVE
7989	035010	016246	000012			MOV	12(R2),-(SP)	;DESIRED CYLINDER ADDRESS
7990	035014	004037	040562			JSR	RO,WRT.RP	
7991	035020	000034				RPCA		
7992	035022	035522				CI7		
7993	035024	116203	000010			MOV	10(R2),R3	;PICKUP SECTOR ADDRESS
7994	035030	163703	033346			SUB	MXWNOV,R3	;BACKUP BY MAX. SEARCH FOR I/O WINDOW
7995	035034	002002				BGE	1\$	
7996	035036	062703	000026			ADD	#22,R3	
7997	035042	010346			1\$:	MOV	R3,-(SP)	;COMBINE THE ADJUSTED SECTOR WITH
7998	035044	116266	000011	000001		MOV	11(R2),1(SP)	;THE DESIRED TRACK
7999	035052	004037	040562			JSR	RO,WRT.RP	;LOAD DESIRED TRACK & SECTOR
8000	035056	000006				RPCA		
8001	035060	035522				CI7		
8002	035062	012746	000131			MOV	#131,-(SP)	;START A SEARCH
8003	035066	004037	040562			JSR	RO,WRT.RP	
8004	035072	000000				RPCS1		
8005	035074	035522				CI7		
8006	035076	156137	033320	033246		BISB	ATABIT(R1),SRCHWT	;SET "SEARCH WAIT" KEY
8007	035104	000567				BR	C15	
8008	035106	013704	033332		C14:	MOV	RPADR,R4	;RPCS1 ADDRESS
8009	035112	010164	000010			MOV	R1,RPCS2(R4)	;SELECT DRIVE
8010	035116	116203	000002			MOV	2(R2),R3	;PICKUP THE REQUESTED COMMAND
8011	035122	122703	000131			CMPB	#131,R3	;IS IT A SEARCH COMMAND?
8012	035126	001007				BNE	1\$;BRANCH IF NO
8013	035130	016246	000010			MOV	10(R2),-(SP)	;LOAD DESIRED TRACK & SECTOR
8014	035134	004037	040562			JSR	RO,WRT.RP	
8015	035140	000006				RPCA		
8016	035142	035522				CI7		
8017	035144	000403				BR	2\$;GO LOAD CYLINDER
8018	035146	122703	000105		1\$:	CMPB	#105,R3	;IS IT A SEEK COMMAND
8019	035152	001007				BNE	3\$;BRANCH IF NO
8020	035154	016246	000012		2\$:	MOV	12(R2),-(SP)	;LOAD DESIRED CYLINDER
8021	035160	004037	040562			JSR	RO,WRT.RP	
8022	035164	000034				RPCA		
8023	035166	035522				CI7		
8024	035170	000546				BR	C16	
8025	035172	122703	000115		3\$:	CMPB	#115,R3	;IS IT AN "OFFSET" COMMAND?
8026	035176	001013				BNE	4\$;BR IF NO
8027	035200	004037	040406			JSR	RO,RD.RP	;MERGE THE OFFSET VALUE INTO RPOF
8028	035204	000032				RPCF		;BUT DON'T CHANGE THE UPPER
8029	035206	035522				CI7		
8030	035210	116216	000001			MOV	1(R2),-(SP)	;BYTE WHEN LOADING THE
8031	035214	004037	040562			JSR	RO,WRT.RP	;REGISTER (RPOF)
8032	035220	000032				RPOF		
8033	035222	035522				CI7		
8034	035224	000530				BR	C16	;GO START THE COMMAND
8035	035226	122703	000107		4\$:	CMPB	#107,R3	;IS IT A "RECALIBRATE" COMMAND?
8036	035232	001525				BEQ	C16	;BRANCH IF YES
8037	035234	122703	000117			CMPB	#117,R3	;IS IT A RETURN TO CENTER?
8038	035240	001522				BEQ	C16	;BRANCH IF YES
8039	035242	122703	000103			CMPB	#103,R3	;IS IT AN "UNLOAD" COMMAND?
8040	035246	001016				BNE	5\$;BRANCH IF NO
8041	035250	112761	000001	033174		MOV	#1,DRVACT(R1)	;SET THE DRIVE ACTIVE INDICATOR
8042	035256	105061	033204			CLRB	DRVSTAT,R1	;PUT DRIVE STATUS TO OFFLINE

8043	035262	112761	000001	033252	MOVB	#1,ULDFLG(R1)	;SET "UNLOAD IN PROGRESS" FLAG
8044	035270	010346			MOV	R3, -(SP)	;START THE "UNLOAD" COMMAND
8045	035272	004037	040562		JSR	RD,WRT.RP	
8046	035276	000000			RPCS1		
8047	035300	035522			CI7		
8048	035302	000207			RTS	PC	;RETURN TO USER
8049	035304	122703	000143	5\$:	CMPB	#143,R3	;IS IT A "SET FORMAT" COMMAND?
8050	035310	001014			BNE	6\$;BRANCH IF NO
8051	035312	004037	040406		JSR	RD, RD.RP	;READ THE OFFSET REGISTER
8052	035316	000032			RPOF		
8053	035320	035522			CI7		
8054	035322	116266	000001	000001	MOVB	1(R2), 1(SP)	;COMBINE "FMT22" "ECI" AND "HCI"
8055	035330	004037	040562		JSR	RD,WRT.RP	;LOAD "FMT22", "ECI", AND/OR "HCI".
8056	035334	000032			RPOF		
8057	035336	035522			CI7		
8058	035340	000436			BR	12\$	
8059	035342	122703	000141	6\$:	CMPB	#141,R3	;IS IT A "GET REGISTER" COMMAND?
8060	035346	001023			BNE	10\$;BRANCH IF NO
8061	035350	016203	000006	7\$:	MOV	6(R2), R3	;POINTS TO 1ST ADDRESS OF WHERE ;TO PUT THE REGISTER(S)
8062							;INIT. THE INDEX FOR THE FIRST REG.
8063	035354	116237	000010	035372	MOVB	10(R2), 9\$;INDEX OF LAST REG. TO MOVE
8064	035362	116205	000011		MOVB	11(R2), R5	;READ RPO4/5/6 REGISTER
8065	035366	004037	040406	8\$:	JSR	RD, RD.RP	;INDEX OF REG. TO READ
8066	035372	000000		9\$:	RPCS1		
8067	035374	035522			CI7		
8068	035376	012623			MOV	(SP)+, (R3)+	;GET THE CONTENTS OF RH11/RPO4/5 6 REG.
8069	035400	023705	035372		CMP	9\$, R5	;LAST REG. BEEN READ?
8070	035404	001414			BEQ	12\$;GET OUT IF YES
8071	035406	062737	000002	035372	ADD	#2, 9\$;INCREASE THE INDEX BY 2
8072	035414	000764			BR	8\$;LOOP--MORE TO READ
8073	035416	122703	000145	10\$:	CMPB	#145,R3	;IS IT A "SELECT DRIVE" COMMAND?
8074	035422	001405			BEQ	12\$;BRANCH IF YES
8075	035424	010346		11\$:	MOV	R3, -(SP)	;LOAD THE COMMAND
8076	035426	004037	040562		JSR	RD,WRT.RP	
8077	035432	000000			RPCS1		
8078	035434	035522			CI7		
8079	035436	004737	041626	12\$:	JSR	PC, POPQUE	;REMOVE REQ. FROM QUEUE
8080	035442	052762	000200	000016	BIS	#BIT07, 16(R2)	;SET THE "DONE" BIT
8081	035450	005737	033272		TST	SAVEFG	;SAVE THE RH11/RPO4/5/6 REGISTERS
8082	035454	100002			BPL	13\$;BRANCH IF NO
8083	035456	004737	040750		JSR	PC, SVRH11	;YES--GO SAVE THE REGISTERS
8084	035462	000207		13\$:	RTS	PC	;RETURN TO USER
8085	035464	006301		C15:	ASL	R1	
8086	035466	012761	001750	033276	MOV	#1000, TIMER(R1)	;SET A ONE SECOND TIMER
8087	035474	006201			ASR	R1	
8088	035476	112761	000001	033174	MOVB	#1, DRVACT(R1)	;SET THE DRIVE ACTIVE
8089	035504	000207			RTS	PC	;RETURN TO THE USER
8090	035506	010346		C16:	MOV	R3, -(SP)	;LOAD THE COMMAND
8091	035510	004037	040562		JSR	RD,WRT.RP	
8092	035514	000000			RPCS1		
8093	035516	035522			CI7		
8094	035520	000761			BR	C15	
8095	035522	032764	010000	000010	CI7:	#BIT12, RPCS2(R4)	;DRIVE NON-EXISTENT ?
8096	035530	001034			BYE	C18	;BR IF YES
8097	035532	005702		1\$:	TST	R2	;ANYTHING IN QUEUE ?
8098	035534	001405			BEQ	CI7B	;BR IF NOT

H12

CRJDCO. PPO4 5 6 MLT-DR LGC
CRJDCO.P11 15-DEC-77 10:58

MACY11 30(1046) 15-DEC-77 11:03 PAGE 151
SINGLE/DUAL PORT RH11/RPO4/5/6 DRIVER (REV 1.0)

SC0 0150

```
0099 035536 012762 104000 000016      MOV      #BIT15:BIT11,16(R2)      ;SET "PARITY" ERROR INDICATOR
0100 035544 004737 040750      JSR      PC,SVRH11              ;GO SAVE THE RH11/RPO4/5/6 REGISTERS
0101 035550 012746 000111      CI7B:   MOV      #111-(SP)          ;DO A "DRIVE CLEAR"
0102 035554 004037 040562      JSR      RO,WAT.RP
0103 035556 000000      RPCS1
0104 035562 035622      CIB
0105 035564 004737 041510      JSR      PC,EMPTYQ             ;EMPTY THE QUEUE
0106 035570 105061 033252      CLR     ULDFLG(R1)             ;CLEAR THE UNLOAD IN QUEUE FLAG
0107 035574 105061 033174      CLR     DRVACT(R1)            ;DRIVE IS IDLE
0108 035600 020137 033316      CMP     R1,DTUW               ;IF THIS DRIVE HAD AN I/O REQUEST
0109 035604 001005      BNE     1$                    ;IN PROGRESS CLEAR ALL OF THE FLAGS
0110 035606 005037 033244      CLR     TRNSWT
0111 035612 012737 177777      MOV     #-1,DTUW
0112 035620 000207      RTS
0113 035622 104412      1$:
0114 035624 032764 010000      CI8:   SAVREG                      ;SAVE R0 - R5
0115 035632 001002      BIT     #BIT12,RPCS2(R4)      ;IS 'NED' SET ?
0116 035634 005001      BNE     1$                    ;BR IF YES
0117 035636 005003      CLR     R1
0118 035640 105761 033174      1$:   CLR     R3
0119 035644 001443      TSTB   DRVACT(R1)            ;DRIVE ACTIVE?
0120 035646 013702 033244      BEQ     5$                    ;BRANCH IF NO
0121 035652 020137 033316      MOV     TRNSWT,R2             ;GET THE "TRANSFER WAIT" QUEUE
0122 035656 001402      CMP     R1,DTUW               ;DID THIS DRIVE HAVE AN I/O IN PROGRESS?
0123 035660 004737 041604      BEQ     2$                    ;BRANCH IF YES
0124 035664 005702      JSR      PC,GETREQ            ;GET THE OPB POINTER
0125 035666 001415      2$:   TST     R2                    ;QUEUE ENTRY FOR DRIVE ?
0126 035670 032764 010000      BFO     4$                    ;BR IF NOT
0127 035676 001404      BIT     #BIT12,RPCS2(R4)      ;'NED' SET ?
0128 035700 012762 100002      BEQ     3$                    ;BR IF NOT
0129 035706 000405      MOV     #BIT15:BIT10,16(R2)    ;SET 'DRIVE NON-EXISTENT' INDICATOR
0130 035710 012762 102000      BR      4$                    ;CONTINUE
0131 035716 004737 040750      MOV     #BIT15:BIT10,16(R2)    ;SET "NON-CLEARABLE PARITY" ERROR INDICATOR
0132 035722 012763 177777      JSR      PC,SVRH11            ;SAVE RH11/RPO4/5/6 REGISTERS
0133 035730 105061 033174      MOV     #-1,TIMER(R3)         ;STOP THE TIMER
0134 035734 020137 033316      CLR     DRVACT(R1)            ;SET "DRIVE ACTIVE" TO IDLE
0135 035740 001005      CMP     R1,DTUW               ;IS THIS DRIVE SETUP FOR A TRANSFER
0136 035742 012737 177777      BNE     5$                    ;BR IF NOT
0137 035750 005037 033244      MOV     #-1,DTUW              ;RESET THE INDICATOR
0138 035754 105061 033252      CLR     TRNSWT                ;CLEAR THE TRANSFER QUEUE
0139 035760 032764 010000      CLR     ULDFLG(R1)            ;CLEAR UNLOAD FLAG
0140 035766 001021      BIT     #BIT12,RPCS2(R4)      ;'NED' SET ?
0141 035770 005201      BNE     6$                    ;BR IF YES
0142 035772 062703 000002      INC     R1                    ;MOVE TO THE NEXT DRIVE
0143 035776 042701 177770      ADD     #2,R3
0144 036002 001316      BIC     #1<7,R1
0145 036004 012737 177777      BNE     1$                    ;BRANCH IF MORE DRIVES
0146 036012 005037 033244      MOV     #-1,DTUW              ;NO DATA TRANSFERS UNDERWAY
0147 036016 004737 041432      CLR     TRNSWT                ;CLEAR THE "TRANSFER WAIT" QUEUE
0148 036022 012764 000040      JSR      PC,CLRQIF            ;CLEAR ALL OF THE REQUEST QUEUES
0149 036030 000406      MOV     #BIT05,RPCS2(R4)      ;DO A MASSBUS INIT.
0150 036032 004737 041510      BR      7$                    ;CONTINUE
0151 036036 105061 033204      JSR      PC,EMPTYQ            ;CLEAR THE DRIVE'S QUEUE
0152 036042 105061 033214      CLR     DRVSTA(R1)            ;SET DRIVE TO OFFLINE
0153 036046 004737 041066      CLR     DRVTP(R1)            ;CLEAR THE DRIVE TYPE INDICATOR
0154 036052 104413      JSR      PC,SET.IE            ;SET "IE" WITHOUT "TRE"
                                PRESREG                          ;RESTORE R0 - R5
```



```

8211 036252 105037 033250          CLRB   ACTDRV   ;CLEAR "ACTIVE DRIVER" FLAG
8212 036256 000002                RTI       ;RETURN
8213
8214          ;TRANSFER DONE ROUTINE
8215
8216 036260 105061 033174          TD:      CLRB   DRVACT(R1) ;SET DRIVE ACTIVE INDICATOR TO IDLE
8217 036264 012737 177777 033316      MOV     #-1,DTUW ;NO DATA TRANSFERS UNDERWAY
8218 036272 006301                ASL     R1
8219 036274 012761 177777 033276      MOV     #-1,TIMER(R1) ;CANCEL TIMEOUT
8220 036302 006201                ASR     R1
8221 036304 013702 033244          MOV     TRNSWT,R2 ;GET "DPB" ADDRESS FROM THE
8222 036310 005037 033244          CLR     TRNSWT ;TRANSFER WAIT QUEUE--CLEAR QUEUE
8223 036314 052762 000200 000016      BIS     #BIT07,16(R2) ;SET DONE
8224 036322 010164 000010          MOV     R1,RPCS2(R4) ;SELECT THE DRIVE
8225 036326 004037 040406          JSR     RD,RD.RP ;TRANSFER ERROR(TRE=1)?
8226 036332 000000                RPCS1
8227 036334 035522                CI7
8228 036336 006126                ROL
8229 036340 100421                BMI     3$ ;BR IF YES
8230 036342 005737 033272          TST     SAVEFG ;SAVE THE RH11/RPO4/5/6 REGISTERS?
8231 036346 100002                BPL     1$ ;BRANCH IF NO
8232 036350 004737 040750          JSR     PC,SVRH11 ;YES--SAVE THE REGISTERS
8233 036354 004737 036434          1$:    JSR     PC,WC ;SEE IF WRITE CHECK TO BE PUT IN QUEUE
8234 036360 004737 041604          JSR     PC,GETREQ ;GET DPB POINTER
8235 036364 005702                TST     R2 ;ENTRY FOR DRIVE ?
8236 036366 001403                BEQ     2$ ;BR IF NOT
8237 036370 004737 034412          JSR     PC,OPT ;CALL OPTIMIZER
8238 036374 000462                BR     SC ;CHECK OTHER DRIVES
8239 036376 012714 000113          2$:    MOV     #113,(R4) ;RELEASE THE DRIVE
8240 036402 000457                BR     SC ;CHECK FOR OTHER DRIVES
8241 036404 052762 100100 000016 3$:    BIS     #BIT15:BIT06,16(R2) ;SET DATA ERROR FLAG
8242 036412 004737 041510          JSR     PC,EMPTYQ ;EMPTY THE "DRIVE'S WAIT" QUEUE
8243 036416 004737 040750          JSR     PC,SVRH11 ;SAVE THE RH11/RPO4/5/6 REGISTERS
8244 036422 012714 040111          MOV     #40111,(R4) ;ISSUE A "DRIVE CLEAR"
8245 036426 012714 000113          MOV     #113,(R4) ;ISSUE A RELEASE TO THE DRIVE
8246 036432 000443                BR     SC ;CHECK FOR OTHER DRIVES
8247
8248          ;FORCED WRITE CHECK ROUTINE
8249
8250 036434 005737 001424          WC:    TST     AUTOCK ;AUTOMATIC WRITE CHECKS ?
8251 036440 001437                BEQ     2$ ;BR IF NOT
8252 036442 122762 000002 000024      CMPB   #2,$CODE(R2) ;LAST OPERATION WRITE DATA ?
8253 036450 001404                BEQ     1$ ;BR IF IT WAS
8254 036452 122762 000003 000024      CMPB   #3,$CODE(R2) ;LAST OPERATION WRITE HEADER 3 DATA ?
8255 036460 001027                BNE     2$ ;BR IF NOT
8256 036462 004037 041530          1$:    JSR     RD,DRVQUE ;PUT THE OPERATION IN THE QUEUE
8257 036466 000424                BR     2$ ;QUEUE IS FULL
8258 036470 005062 000016          CLR     16(R2) ;CLEAR "DONE" BIT IN DPB
8259 036474 116262 000234 000027      MOVB   $RPCS1(R2),$PREV0(R2) ;SAVE WRITE OPERATION CODE
8260 036502 016262 000012 000034      MOV     $CYL(R2),$PREVA+2(R2) ;SAVE CYLINDER
8261 036510 016262 000010 000032      MOV     $SEC(R2),$PREVA(R2) ;SAVE SECTOR AND TRACK ADDRESSES
8262 036516 142762 000002 000024      BICB   #2,$CODE(R2) ;CHANGE WRITE TO CHECK
8263 036524 142762 000020 000002      BICB   #20,$COMND(R2) ;CHANGE DRIVER CODE TO WRITE CHECK
8264 036532 152762 000010 000002      BICB   #10,$COMND(R2) ;FINISH CHANGING CODE TO WRITE CHECK
8265 036540 000207                2$:    RTS     PC ;EXIT
8266

```

:SPECIAL CONDITION ROUTINE

8267							
8268							
8269	036542	116403	000016	SC:	MOVB	RPAS(R4),R3	:READ "RPAS"
8270	036546	001012			BNE	2\$:BRANCH IF ANY 'ATA' BITS SET
8271	036550	004037	040406		JSR	RO,RO.RP	:READ CONTROL AND STATUS REGISTER
8272	036554	000000			RPCS1		
8273	036556	035622			CIB		
8274	036560	106126			ROLB	(SP)+	:IS "IE"=1?
8275	036562	100403			BMI	1\$:YES, NO DRIVES TO CHECK
8276	036564	104001			ERROR	1	:REPORT AN ILLEGAL INTERRUPT
8277	036566	004737	041066		JSR	PC,SET.IE	:SET INTERRUPT ENABLE
8278	036572	000207		1\$:	RTS	PC	:RETURN
8279	036574	005046		2\$:	CLR	-(SP)	:PROCESS ALL DRIVES THAT HAVE
9280	036576	110316			MOVB	R3,(SP)	:AN "ATA"=1
8281	036600	012703	000001		MOV	#1,R3	
8282	036604	005001			CLR	R1	
8283	036606	030316		SC3:	BIT	R3 (SP)	:ATA=1?
8284	036610	001005			BNE	SC5	:YES--BRANCH
8285	036612	005201		SC4:	INC	R1	:MOVE TO THE NEXT DRIVE
8286	036614	106303			ASLB	R3	
8287	036616	001373			BNE	SC3	:BRANCH IF MORE TO CHECK?
8288	036620	005726			TST	(SP)+	:CLEAN OFF THE STACK
8289	036622	000207			RTS	PC	:RETURN TO USER
8290	036624	105761	033224	SC5:	TSTB	DPINT(R1)	:INITIALIZING THE DRIVE ?
8291	036630	001402			BEQ	1\$:BR IF NOT
8292	036632	000137	037520		JMP	SC13	:PROCESS THE DRIVE
8293	036636	105761	033234	1\$:	TSTB	DPROS(R1)	:PORT REQUEST OUTSTANDING ?
8294	036642	001402			BEQ	2\$:BR IF NOT
8295	036644	000137	037520		JMP	SC13	:START THE OUTSTANDING COMMAND
8296	036650	105761	033204	2\$:	TSTB	DRVSTA(R1)	:CHECK THE DRIVE STATUS
8297	036654	003025			BGT	5\$:BRANCH IF ONLINE
8298	036656	105761	033252		TSTB	ULDFLG(R1)	:UNLOAD IN PROGRESS?
8299	036662	003422			BLE	5\$:BRANCH IF NOT
8300	036664	004737	041604		JSR	PC,CETREQ	:GET DPB POINTER
8301	036670	004737	040750		JSR	PC,SVRH11	:SAVE THE RH11/RPO4/5/6 REGISTERS
8302	036674	004737	037450		JSR	PC,SC12	:SAVE RPS1, RPER1, RPER2, AND RPER3
8303							:ALSO DO A DRIVE INIT (DRVINT)
8304	036700	105761	033204		TSTB	DRVSTA(R1)	:DID DRIVE COME ONLINE?
8305	036704	003416			BLE	6\$:NO---BRANCH
8306	036706	032737	040000	033164	BIT	#BIT14,RPERRS	:WAS THERE AN ERROR?
8307	036714	001002			BNE	3\$:BR IF ERROR
8308	036716	000137	037360		JMP	SC11	:NO ERROR
8309	036722	013705	033166	3\$:	MOV	RPERRS+2,R5	:YES -- PICKUP RPER1 AND
8310	036726	000476			BR	SC6A	:GO PROCESS THE ERROR
8311	036730	105761	033174	5\$:	TSTB	DRVACT(R1)	:DRIVE ACTIVE WITH COMMAND OR ERROR RECOVERY ?
8312	036734	001027			BNE	SC6	:BR IF EITHER
8313	036736	004737	037450		JSR	PC,SC12	:SAVE RPS1, RPER1, RPER2, AND RPER3
8314							:ALSO DO A DRVINT
8315	036742	105761	033224	6\$:	TSTB	DPINT(R1)	:TRYING TO INIT THE DRIVE ?
8316	036746	001321			BNE	SC4	:BR IF YES, CHECK ON MORE DRIVES
8317	036750	105761	033204		TSTB	DRVSTA(R1)	:CHECK ON DRIVE'S STATUS
8318	036754	100412			BMI	7\$:BR IF UNSAFE
8319	036756	032737	020000	033172	BIT	#BIT13,RPERRS+6	:ADDRESS PLUG CHANGED ?
8320	036764	001011			BNE	8\$:BR IF YES
8321	036766	012746	000113		MOV	#113, -(SP)	:RELEASE COMMAND
8322	036772	004037	040562		JSR	RO,WAT.RP	:WRITE THE COMMAND INTO RPCS1

```

8323 046776 000000          RPCS1          ;REGISTER INDEX
8324 037000 037330          SCB           ;PARITY EXIT ADDRESS
8325 057002 011605          7S:  MOV      (SP),R5      ;PICKUP (RPAS) BEFORE THE ERROR CALL
8326 037004 104002          ERROR        2          ;REPORT THE UNEXPECTED ATTENTION
8327 037006 000701          BR          SC4          ;GO CHECK FOR MORE ATA'S
8328 037010
8329 037010 104005          ERROR        5          ;REPORT THE ADDRESS PLUG CHANGE
8330 037012 000677          BR          SC4          ;CHECK FOR MORE DRIVES
8331 037014 006301          SC6:  ASL      R1          ;SETUP TO ADDRESS WORDS
8332 037016 012761 177777 033276  MOV      #-1,TIMER(R1) ;STOP THE TIMER
8333 037024 006201          ASR      R1          ;RESTORE THE DRIVE ADDRESS
8334 037026 004737 041604  JSR      PC,GETREQ      ;GET THE DPB POINTER FROM THE QUEUE
8335 037032 010164 000010  MOV      R1,RPCS2(R4)  ;SELECT DRIVE
8336 037036 004037 040406  JSR      R0,RD.RP      ;READ THE RPO4'S STATUS REG.
8337 037042 000012          RPDS1
8338 037044 037330          SCB
8339 037046 011605          MOV      (SP),R5      ;AND PUT IT IN R5
8340 037050 006126          ROL      (SP)+        ;WAS THERE AN ERROR?
8341 037052 100407          BMI     1$          ;BR IF ERROR
8342 037054 105761 033174  TSTB   DRVACT(R1)    ;CHECK DRIVE'S STATE
8343 037060 003137          BGT     SC11        ;BR IF DRIVE ACTIVE WITH ORDER
8344 037062 052762 100210 000016  BIS    #BIT15!BIT07!BIT03,16(R2) ;INFORM USER OF ERROR RECOVER COMPLETION
8345 037070 000470          BR          SC7
8346 037072 004037 040406  1$:  JSR      R0,RD.RP      ;READ ERROR REGISTER #1
8347 037076 000014          RPER1
8348 037100 037330          SCB
8349 037102 012605          MOV      (SP)+,R5      ;AND SAVE IT IN R5
8350 037104 004737 040750  JSR      PC,SVRH11     ;SAVE RH11/RPO4/5/6 REGISTERS
8351 037110 012746 000111  MOV      #111,-(SP)    ;ISSUE A DRIVE CLEAR
8352 037114 004037 040562  JSR      R0,WRT.RP
8353 037120 000000          RPCS1
8354 037122 037330          SCB
8355 037124 006105          SC6A:  ROL      R5          ;WAS "UNSAFE" CONDITION =1?
8356 037126 100406          BMI     1$          ;BRANCH IF YES
8357 037130 005702          TST     R2          ;ANYTHING IN QUEUE ?
8358 037132 001447          BEQ     SC7          ;BR IF NOT
8359 037134 052762 100240 000016  BIS    #BIT15!BIT07!BIT05,16(R2) ;INFORM USER OF ERROR
8360 037142 000443          BR          SC7
8361 037144 004037 040406  1$:  JSR      R0,RD.RP      ;READ DRIVE STATUS REG. #1
8362 037150 000012          RPDS1
8363 037152 037330          SCB
8364 037154 011605          MOV      (SP),R5      ;SAVE RPDS1 IN R5
8365 037156 006126          ROL      (SP)+        ;"ERR"=1?
8366 037160 100011          BPL     2$          ;BR IF NO--UNSAFE CLEARED
8367 037162 112761 177777 033204  MOVB   #-1,DRVSTA(R1) ;DRIVE IS UNSAFE
8368 037170 004737 040750  JSR      PC,SVRH11     ;SAVE RH11/RPO4/5/6 REGISTERS
8369 037174 052762 110000 000016  BIS    #BIT15!BIT12,16(R2) ;INFORM USER OF UNSAFE ERROR
8370 037202 000423          BR          SC7
8371 037204 032705 010000  2$:  BIT     #BIT12,R5      ;"MOL" = 1 ?
8372 037210 001015          BNE     3$          ;BR IF YES
8373 037212 112761 177777 033174  MOVB   #-1,DRVACT(R1) ;ACTIVE ERROR RECOVER
8374 037220 112761 000001 033204  MOVB   #1,DRVSTA(R1)  ;ONLINE
8375 037226 006301          ASL      R1
8376 037230 012761 072460 033276  MOV      #30000.,TIMER(R1) ;START 30 SECOND TIMER
8377 037236 006201          ASR      R1
8378 037240 000137 036612  JMP     SC4

```

M12

CZRJDC, RPO4 5 6 MLT-DR LGC
CZRJDC.P11 15-DEC-77 10:58

MACY11 30(1046) 15-DEC-77 11:03 PAGE 156
SINGLE DUAL PORT RH11/RPO4/5/6 DRIVER (REV 1.0)

SEG 0155

```

8379 037244 052762 100220 000016 3$: BIS #BIT15!BIT07!BIT04,16(R2) ;INFORM USER OF ERROR
8380 037252 105061 033174 SC7: CLRB DRVACT(R1) ;DRIVE IS IDLE
8381 037256 004737 041510 JSR PC,EMPTY0 ;DUMP THE QUEUE
8382 037262 105761 033252 TSTB ULDFLG(R1) ;UNLOAD IN PROGRESS OR QUEUE?
8383 037266 003002 BGT 1$ ;BR IF NOT
8384 037270 105061 033252 CLRB ULDFLG(R1) ;CLEAR UNLOAD FLAG
8385 037274 116164 033320 000016 1$: MOVB ATABIT(R1),RPAS(R4) ;CLEAR ATTENTION BIT
8386 037302 105761 033204 TSTB DRVSTA(R1) ;IS THE DRIVE UNSAFE?
8387 037306 100406 BMI 2$ ;BR IF IT IS
8388 037310 012746 000113 MOV #113,-(SP) ;RELEASE COMMAND
8389 037314 004037 040562 JSR RO,WAT.RP ;WRITE THE COMMAND INTO RPCS1
8390 037320 000000 RPCS1 ;REGISTER INDEX
8391 037322 037330 SCB ;PARITY EXIT ADDRESS
8392 037324 000137 036612 2$: JMP SC4 ;CHECK FOR MORE DRIVES
8393 037330 105761 033174 SC8: TSTB DRVACT(R1) ;IS DRIVE IDLE?
8394 037334 001405 BEQ 1$ ;YES--BRANCH
8395 037336 004737 041604 JSR PC,GETREG ;GET DPB POINTER
8396 037342 004737 035522 JSR PC,C17 ;PROCESS THE PARITY ERROR
8397 037346 000402 BR 2$ ;CONTINUE
8398 037350 004737 035550 1$: JSR PC,C17B ;PROCESS THE UNCORRECTABLE PARITY ERROR
8399 037354 000137 036612 2$: JMP SC4 ;CHECK MORE DRIVES
8400 037360 105761 033252 SC11: TSTB ULDFLG(R1) ;"UNLOAD IN PROGRESS"?
8401 037364 003402 BLE 1$ ;BRANCH IF NO
8402 037366 105061 033252 CLRB ULDFLG(R1) ;CLEAR UNLOAD FLAG
8403 037372 105061 033174 1$: CLRB DRVACT(R1) ;SET DRIVE IDLE
8404 037376 136137 033320 033246 BITB ATABIT(R1),SRCHWT ;DOING A SEARCH OPERATION FOR
8405 ; AN I/O COMMAND?
8406 037404 001012 BNE 2$ ;BRANCH IF YES
8407 037406 004737 041626 JSR PC,POPQUE ;REMOVE REQUEST FROM QUEUE
8408 037412 052762 000200 000016 BIS #BIT07,16(R2) ;SET "DONE" BIT
8409 037420 005737 033272 TST SAVEFG ;SAVE THE REGISTERS?
8410 037424 100002 BPL 2$ ;BRANCH IF NO
8411 037426 004737 040750 JSR PC,SVRH11 ;YES--SAVE ALL OF THE RH11/RPO4/5/6 REG'S
8412 037432 116164 033320 000016 2$: MOVB ATABIT(R1),RPAS(R4) ;CLEAR ATTENTION BIT
8413 037440 004737 034412 JSR PC,OPT ;START A REQUEST
8414 037444 000137 036612 JMP SC4 ;CHECK FOR MORE DRIVES
8415 037450 010164 000010 SC12: MOV R1,RPCS2(R4) ;SELECT DRIVE
8416 037454 016437 000012 033164 MOV RPOS1(R4),RPERRS ;SAVE THE FOUR REGISTERS THAT
8417 037462 016437 000014 033166 MOV RPER1(R4),RPERRS+2 ;WILL TELL US SOMETHING
8418 037470 016437 000040 033170 MOV RPER2(R4),RPERRS+4
8419 037476 016437 000042 033172 MOV RPER3(R4),RPERRS+6
8420 037504 004037 033562 JSR RO,DRVINT ;INIT. THE STATE OF THE DRIVE
8421 037510 000401 BR 1$ ;TAKE ERROR EXIT
8422 037512 000207 RTS PC ;RETURN
8423 037514 005726 1$: TST (SP)+ ;POP PC OFF OF THE STACK
8424 037516 000704 BR SC8 ;PROCESS THE PARITY ERROR
8425 037520 006301 SC13: ASL R1 ;SETUP TO ADDRESS WORDS
8426 037522 012761 177777 033276 MOV #-1,TIMER(R1) ;STOP THE TIMER
8427 037530 006201 ASL R1
8428 037532 010164 000010 MOV R1,RPCS2(R4) ;SELECT THE DRIVE
8429 037536 116164 033320 000016 MOVB ATABIT(R1),RPAS(R4) ;CLEAR THE ATTENTION BIT
8430 037544 032714 004000 BIT #BIT11,(R4) ;DRIVE AVAILABLE
8431 037550 001006 BNE 1$ ;BR IF AVAILABLE
8432 037552 006301 ASL R1
8433
8434 037554 012761 023420 033276 MOV #10000.,TIMER(R1) ;START 10 SEC TIMER AGAIN

```

```

0435
0436 037562 006201          ASR      R1
0437 037564 000433          BR      3$
0438 037566 105761 033224    1$:    TSTB   DPINT(R1)      ;EXIT
0439 037572 001424          BEQ     2$           ;INITIALIZING THE DRIVE ?
0440 037574 105061 033224    CLRB   DPINT(R1)      ;BR IF NOT
0441 037600 004037 033562    JSR    RD,DRVINT     ;CLEAR THE INIT INDICATOR
0442 037604 000240          NOP
0443 037606 105761 033204    TSTB   DRVSTA(R1)    ;GO INIT THE DRIVE
0444 037612 003014          BGT     2$           ;DUMMY PARITY ERROR RETURN
0445 037614 005702          TST     R2           ;DRIVE ONLINE ?
0446 037616 001416          BEQ     3$           ;BR IF YES -- START ORDER
0447 037620 004737 041604    JSR    PC,GETREQ     ;QUEUE ENTRY FOR THE DRIVE
0448 037624 052762 140000 000016  BIS    #BIT15:BIT14,16(R2) ;GET DPB ADDRESS
0449 037632 004737 040750    JSR    PC,SVRH11     ;INFORM USER THAT DRIVE OFFLINE
0450 037636 004737 041510    JSR    PC,EMPTYQ     ;SAVE THE REGISTERS
0451 037642 000404          BR      3$           ;EMPTY THE REQUEST QUEUE
0452 037644 105061 033234    2$:    CLRB   DPRQS(R1)  ;CLEAR THE PORT REQUEST INDICATOR
0453 037650 004737 034412    JSR    PC,OPT        ;START THE PENDING REQUEST
0454 037654 000137 036612    JMP     SC4          ;PROCESS OTHER DRIVES
0455
0456 ;RPO4/5/6 TIMER ROUTINE
0457 ;CALL
0458 ;
0459 ;   MOV     #TIME -(SP) ;ELAPSED TIME IN MILLISECONDS ON THE STACK
0460 ;   JSR    PC,RPTMR    ;CALL RPO4/5/6 TIME ROUTINE
0461 037660 005737 033250    RPTMR: TST     ACTDRV   ;CHECK "ACTDRV & ACTSTR"
0462 037664 001030          BNE     4$           ;IF NON ZERO EXIT
0463 037666 112737 000001 033251  MOVB   #1,ACTSTR     ;SET "ACTSTR"
0464 037674 104412          SAVREG
0465 037676 005001          CLR     R1           ;SAVE R0 - R5
0466 037700 005003          CLR     R3           ;START WITH DRIVE 0
0467 037702 005763 033276    1$:    TST     TIMER(R3)  ;IS THE TIMER RUNNING?
0468 037706 002407          BLT     2$           ;BRANCH IF NO
0469 037710 166663 000002 033276  SUB    2(SP),TIMER(R3) ;COUNT THE INTERVAL
0470 037716 003003          BGT     2$           ;BR IF NO SOFTWARE TIMEOUT
0471 037720 004737 037752    JSR    PC,STO        ;CALL SOFTWARE TIMEOUT ROUTINE
0472 037724 000405          BR      3$           ;GO TO THE EXIT
0473 037726 005201          INC     R1           ;MOVE TO NEXT DRIVE
0474 037730 005723          TST    (R3)+
0475 037732 022701 000010  CMP    #8.,R1        ;OUT OF DRIVES?
0476 037736 003361          BGT     1$           ;BRANCH IF NO
0477 037740 104413          RESREG   ;RESTORE R0 - R5
0478 037742 105037 033251  CLRB   ACTSTR        ;ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
0479 037746 012616          MOV    (SP)+,(SP)   ;ADJUST THE STACK
0480 037750 000207          RTS     PC           ;RETURN
0481
0482 ;SOFTWARE TIMEOUT ROUTINE
0483 ;
0484 ;NOTE: THIS ROUTINE MUST BE ENTERED AT PRIORITY 6
0485 ;OR GREATER
0486 ;
0487 ;CALL:  STO     #DRVNUM,R1 ;DRIVE NUMBER
0488 ;        MOV    #DRVNUM,R1 ;DRIVE NUMBER
0489 ;        JSR    PC,STO     ;CALL
0490 ;
0491 ;        RETURN

```

8491											
8492	037752	010146		STO:	MOV	R1,-(SP)		:SAVE R1			
8493	037754	010346			MOV	R3,-(SP)		:SAVE R3			
8494	037756	013704	033332		MOV	RPADR,R4		:GET ADDRESS OF "RPCS1"			
8495	037762	010164	000010		MOV	R1,RPCS2(R4)		:SELECT THE DRIVE			
8496	037766	004037	040406		JSR	RD,RD.RP		:READ "DRIVE STATUS REG"			
8497	037772	000012			RPDS1						
8498	037774	040274			STO5						
8499	037776	105726			TSTB	(SP)+		:IS "DRY"=1?			
8500	040000	100477			BMI	STO2		:BR IF YES			
8501	040002	105761	033224	STO1:	TSTB	DPINT(R1)		:TRYING TO INITIALIZE THE DRIVE ?			
8502	040006	001074			BNE	STO2		:BR IF YES			
8503	040010	105761	033234		TSTB	DPRQS(R1)		:OUTSTANDING PORT REQUEST FOR THE DRIVE ?			
8504	040014	001071			BNE	STO2		:P? IF YES			
8505	040016	013702	033244		MOV	TRNSWT,R2		:PICKUP TRANSFER WAIT QUEUE			
8506	040022	020137	033316		CMP	R1,DTUW		:TRANSFER UNDERWAY ON THIS DRIVE?			
8507	040026	001402			BEQ	15		:BRANCH IF YES			
8508	040030	004737	041604		JSR	PC.GETREG		:GET DPB ADDRESS			
8509	040034	052762	101000	000016	15:	BIS	#BIT15:BIT09,16(R2)	:SET THE ERROR FLAGS			
8510	040042	004737	040750	000010	JSR	PC.SVRH11		:SAVE RH11/RPO4/5/6 REGISTERS			
8511	040046	012764	000040		MOV	#BIT05,RPCS2(R4)		: "INIT" THE MASS BUS			
8512	040054	105061	033174		CLRB	DRVACT(R1)		:DRIVE IS IDLE			
8513	040060	105061	033252		CLRB	ULDFLG(R1)		:CLEAR THE UNLOAD FLAG			
8514	040064	005001			CLR	R1		:START WITH DRIVE 0			
8515	040066	005003			CLR	R3					
8516	040070	004037	033562	25:	JSR	RD.DRVINT		:INIT. THIS DRIVE			
8517	040074	000477			STO5			:PARITY ERROR RETURN			
8518	040076	105761	033174		TSTB	DRVACT(R1)		:DRIVE IDLE BEFORE THE INIT.?			
8519	040102	001414			BEQ	45		:YES--BRANCH			
8520	040104	013702	033244		MOV	TRNSWT,R2		:GET TRANSFER WAIT QUEUE			
8521	040110	023701	033316		CMP	DTUW,R1		:WAS THERE I/O ON THIS DRIVE?			
8522	040114	001402			BEQ	35		:YES--BRANCH			
8523	040116	004737	041604		JSR	PC.GETREG		:GET THE DPB POINTER FROM QUEUE			
8524	040122	052762	100400	000016	35:	BIS	#BIT15:BIT08,16(R2)	:INFORM USER OF INIT.			
8525	040130	105061	033174		CLRB	DRVACT(R1)		:SET DRIVE ACTIVE TO IDLE			
8526	040134	105061	033252		CLRB	ULDFLG(R1)		:NO UNLOAD			
8527	040140	012763	177777	033276	MOV	#-1,TIMER(R3)		:STOP THE TIMER			
8528	040146	005723			TST	(R3)+		:UPDATE THE INDEX			
8529	040150	005201			INC	R1		:INCREMENT THE DRIVE NUMBER			
8530	040152	022701	000010		CMP	#R1		:LAST DRIVE BEEN CHECKED?			
8531	040156	003344			BGT	25		:NO--LOOP			
8532	040160	012737	177777	033316	MOV	#-1,DTUW		:NO DATA TRANSFERS UNDERWAY			
8533	040166	005037	033244		CLR	TRNSWT		:CLEAR TRANSFER WAIT QUEUE			
8534	040172	004737	041432		JSR	PC.CLRQUE		:CLEAR ALL REQUEST QUEUES			
8535	040176	000500			BR	STO9		:EXIT			
8536	040200	116405	000016	STO2:	MOVB	RPAS(R4),R5		:READ ATTENTION REG			
8537	040204	136105	033320		BITB	ATABIT(R1),R5		:IS ATTENTION FOR THIS DRIVE UP ?			
8538	040210	001017			BNE	STO3		:YES--BRANCH			
8539	040212	105761	033224		TSTB	DPINT(R1)		:TRYING TO INITIALIZE THE DRIVE ?			
8540	040216	001031			BNE	STO6		:BR IF YES - DRIVE NOT ONLINE			
8541	040220	105761	033234		TSTB	DPRQS(R1)		:OUTSTANDING PORT REQUEST FOR THE DRIVE ?			
8542	040224	001045			BNE	STO7		:BR IF YES - NO RESPONSE TO REQUEST			
8543	040226	020137	033316		CMP	R1,DTUW		:DATA TRANSFER UNDERWAY FOR THIS DRIVE			
8544	040232	001263			BNE	STO1		:BR IF NO			
8545	040234	004037	040406		JSR	RD,RD.RP		:YES--CHECK "RDY"			
8546	040240	000000			RPCS1						

```

0547 040242 040274 STOS
0548 040244 105726 TSTB (SP)+
0549 040246 100255 BPL ST01 ;BR IF "RDY"=0
0550 040250 105761 033224 ST03: TSTB DPINT(R1) ;INITIALIZING THE DRIVE ?
0551 040254 001003 BNE IS ;BR IF INIT PENDING
0552 040256 105761 033234 TSTB DPRQS(R1) ;PORT REQUEST PENDING ?
0553 040262 001446 BEQ ST09 ;BR IF NOT
0554 040264 012763 177777 033276 IS: MOV #-1,TIMER(R3) ;STOP THE TIMER
0555 040272 000442 BR ST09 ;EXIT
0556 040274 004737 035622 ST05: JSR PC,CIB ;GO HANDLE THE PARITY ERROR
0557 040300 000437 BR ST09
0558 040302 105061 033224 ST06: CLRB DPINT(R1) ;CLEAR THE INITIALIZE INDICATOR
0559 040306 105061 033204 CLRB DRVSTA(R1) ;SET UNIT OFFLINE
0560 040312 012763 177777 033276 MOV #-1,TIMER(R3) ;STOP THE TIMER
0561 040320 004737 041604 JSR PC,GETREQ ;GET THE DFB ADDRESS
0562 040324 005702 TST R2 ;REQUEST IN QUEUE ?
0563 040326 001424 BEQ ST09 ;BR IF NOT
0564 040330 052762 140000 000016 BIS #BIT15:BIT14,16(R2) ;INFORM THE USER DRIVE NOT AVAILABLE
0565 040336 000414 BR ST08 ;FINISH
0566 040340 012763 177777 033276 ST07: MOV #-1,TIMER(R3) ;STOP THE TIMER
0567 040346 105061 033234 CLRB DPRQS(R1) ;CLEAR PORT REQUEST INDICATOR
0568 040352 004737 041604 JSR PC,GETREQ ;GET DFB ADDRESS
0569 040356 005702 TST R2 ;QUEUE ENTRY FOR DRIVE ?
0570 040360 001407 BEQ ST09 ;BR IF NONE
0571 040362 012762 100004 000016 MOV #BIT15:BIT2,16(R2) ;INFORM USER OF PORT REQUEST ERROR
0572 040370 004737 041510 ST08: JSR PC,EMPTYQ ;CLEAR THE QUEUE FOR THE DRIVE
0573 040374 004737 040750 JSR PC,SVRH11 ;SAVE THE REGISTERS
0574 040400 012603 ST09: MOV (SP)+,R3 ;RESTORE R3
0575 040402 012601 MOV (SP)+,R1 ;RESTORE R1
0576 040404 00C207 RTS PC ;RETURN

;ROUTINE TO READ A RH11/RP04/5/6 REGISTER
;CALL
; JSR RD, RD.RP ;GO READ A REGISTER
; INDEX ;REG. INDEX FROM BASE
; ERRADR ;ERROR ADDRESS--PROCESS ERROR STARTING
; ;AT THIS ADDRESS
; ;CONTENTS OF REG. IS ON THE STACK
; RETURN

RD.RP: MOV MCPMX, RD.RP2 ;MAX. RETRYs ALLOWED
MOV (SP)-,(SP) ;SAVE RD FOR RETURN
MOV RPADR, RD.ADR ;FORM THE DESIRED ADDRESS
ADD (RD)+, RD.ADR ;USING THE BASE AND THE INDEX
RD.RP1: MOV @ (PC)+, (PC)+ ;READ THE DESIRED REGISTER OF THE RP04
RD.ADR: .WORD 0 ;ADDRESS IS FORMED HERE
RD.WRD: .WORD 0 ;REG. CONTENTS PUT HERE
MOV RD.WRD, 2(SP) ;RETURN IT TO THE USER
MOV RPADR, -(SP) ;PUT THE ADDRESS ON THE STACK
ADD #RPCS2, (SP) ;FORM THE ADDRESS OF RPCS2
BIT #BIT12, @ (SP)+ ;CHECK THE 'NED' BIT
BNE RD.RP3 ;BR IF DRIVE NON-EXISTENT
MOV @RPADR, -(SP) ;READ RPCS1
BIT #BIT13, (SP) ;DID MCPE SET?
BNE IS ;BRANCH IF YES
CMP (SP)+, (RD)+ ;ADJUST FOR RETURN

```

```

8603 040476 000430          BR      RD.RP4          ;EXIT
8604 040500          1$:      ERROR      3          ;REPORT "MCPE" ERROR
8605 040500 104003          TST      DTUW          ;DATA TRANSFER UNDERWAY?
8606 040502 005737 033316  BMI      2$          ;NO--BRANCH
8607 040506 100405          BIT      #BIT14,(SP) ;NO--"TRE"=1?
8608 040510 032716 040000  BEQ      2$          ;NO--BRANCH
8609 040514 001402          TST      (SP)+        ;YES--CLEAN OFF THE STACK AND
8610 040516 005726          BR      RD.RP3        ;TAKE THE FATAL ERROR EXIT
8611 040520 000415          BIS      #BIT14,(SP) ;CLEAR "MCPE" BY SENDING A "1" TO "TRE"
8612 040522 052716 040000  SWAB     (SP)         ;POSITION BEFORE WRITING
8613 040526 000316          MOV      RPADR,3$     ;FORM ADDRESS OF HIGH BYTE
8614 040530 013737 033332 040544  INC      3$          ;WRITE THE HIGH BYTE OF RPCS1
8615 040536 005237 040544  MOV      (SP)+,2(PC)+ ;ADDRESS STORAGE
8616 040542 112637          .WORD   0           ;EXCEEDED MAX. RETRYS
8617 040544 000000          DEC     (PC)+        ;BRANCH IF NO
8618 040546 005327          RD.RP2: .WORD   3     ;FATAL ERROR EXIT
8619 040550 000003          BGE     RD.RP1
8620 040552 002326          RD.RP3: MOV     (RD),PO
8621 040554 011000          MOV     (SP)+,(SP)
8622 040556 012615          RD.RP4: RTS      RD
8623 040560 000200          ;ROUTINE TO WRITE A REGISTER
8624
8625          ;CALL
8626          ;
8627          MOV     DATA,-(SP) ;DATA TO BE LOADED ON THE STACK
8628          JSR     RD,WRT.RP    ;CALL THE ROUTINE TO LOAD(WRITE) THE REG.
8629          INDEX ;INDEX OF THE REGISTER TO BE LOADED
8630          ERRADR ;ADDRESS TO RETURN TO ON AN ERROR
8631          RETURN ;ERROR FREE RETURN
8632
8633
8634 040562 013737 033330 040732 WRT.RP: MOV     MCPEMX,WRT.R2 ;MAX RETRYS ALLOWED
8635 040570 016637 000002 040650  MOV     2(SP),WRT.WD ;SAVE THE WORD TO WRITE
8636 040576 012616          MOV     (SP)+,(SP)   ;ADJUST THE STACK
8637 040600 012037 040652          MOV     (RD)+,WRT.AD ;GET INDEX OF REGISTER TO BE WRITTEN
8638 040604 001015          BNE     1$           ;BRANCH IF NOT RPCS1
8639 040606 122737 000150 040650  CMPB   #150,WRT.WD   ;IS THE COMMAND FOR DATA TRANSFERS?
8640 040614 002411          BLT     1$           ;YES--DON'T GET THE OLD A16 & A17, & PSEL
8641 040616 004037 040406          JSR     RD,RD.RP    ;NO---COMBINE A16&A17, & PSEL WITH
8642 040622 000000          RPCS1 ;THE COMMAND BEFORE SENDING IT TO
8643 040624 040740          WRT.R3 ;THE RH11/RP04
8644 040626 000316          SWAB   (SP)
8645 040630 042716 177770  BIC     #1C7,(SP)
8646 040634 112637 040651          MOV     (SP)+,WRT.WD+1
8647 040640 063737 033332 040652 1$:      ADD     RPADR,WRT.AD ;FORM THE ADDRESS OF THE DISK REG.
8648 040646 012737          WRT.R1: MOV     (PC)+,2(PC)+ ;LOAD THE DESIRED REG.
8649 040650 000000          WRT.WD: .WORD   0   ;WORD TO WRITE GOES HERE
8650 040652 000000          WRT.AD: .WORD   0   ;ADDRESS IS FORMED HERE
8651 040654 013746 033332          MOV     RPADR,-(SP) ;PUT THE ADDRESS ON THE STACK
8652 040660 062716 000010          ADD     #RPCS2,(SP) ;FORM THE ADDRESS OF RPCS2
8653 040664 032736 010000          BIT     #BIT12,2(SP)+ ;CHECK THE 'NED' BIT
8654 040670 001023          BNE     WRT.R3      ;BR IF DRIVE NON-EXISTENT
8655 040672 004037 040406          JSP     RD,RD.RP   ;CHECK FOR PARITY ERROR ON WRITE
8656 040676 000014          RPER1
8657 040700 040740          WRT.R3
8658 040702 032726 000010          BIT     #BIT03,(SP)+

```



```

8659 040706 001416
8660 040710 016037 177776 040722
8661 040716 004037 040406
8662 040722 000000 1$: .WORD 0
8663 040724 040740 WRT.R3: .WORD 0
8664 040726 104004 ERROR 4
8665 040730 005327 DEC (PC)+
8666 040732 000003 WRT.R2: .WORD 3
8667 040734 002344 BGE WRT.R1
8668 040736 005726 TST (SP)+
8669 040740 011000 WRT.R3: MOV (R0),R0
8670 040742 000401 BR WRT.R5
8671 040744 005720 WRT.R4: TST (R0)+
8672 040746 000200 WRT.R5: RTS R0
8673
8674 ;ROUTINE TO SAVE THE RH11/RPO4/5/6 REGISTERS AS PER DPB+14
8675
8676 ;CALL
8677
8678 ;
8679
8680 040750 104412 SVRH11: SAVREG
8681 040752 005702 TST R2
8682 040754 001430 BEQ 4$
8683 040756 013704 033332 MOV RPADR,R4
8684 040762 111264 000010 MOVB (R2),RPCS2(R4)
8685 040766 016203 000014 MOV 14(R2),R3
8686 040772 001433 BEQ 6$
8687 040774 005037 041030 CLR 3$
8688 041000 023727 041030 000022 1$: CMP 3$,#RPDB
8689 041006 001006 BNE 2$
8690 041010 032764 000200 000010 BIT #BIT07,RPCS2(R4)
8691 041016 001002 BNE 2$
8692 041020 005023 CLR (R3)+
8693 041022 000405 BR 4$
8694 041024 004037 040406 2$: JSR R0,RD,RP
8695 041030 000000 3$: .WORD 0
8696 041032 041056 S$
8697 041034 012623 MOV (SP)+,(R3)+
8698 041036 023727 041030 000046 4$: CMP 3$,#RPEC2
8699 041044 001406 BEQ 6$
8700 041046 062737 000002 041030 ADD #2,3$
8701 041054 000751 BR 1$
8702 041056 004737 035522 5$: JSR PC,C17
8703 041062 104413 6$: RESREG
8704 041064 000207 RTS PC
8705
8706 ;ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A "TRE"
8707 ;CALL
8708
8709 ;
8710 ;
8711 ;
8712 041066 010446 SET.IE: MOV R4,-(SP)
8713 041070 013704 033332 MOV RPADR,R4
8714 041074 010164 000010 MOV R1,RPCS2(R4)

```

```

;BRANCH IF "PAR=0"
;PICKUP THE INDEX
;READ THE REG.
;REG. INDEX
;RETURN TO THIS ADDRESS ON ERROR
;REPORT THE PARITY ON WRITE ERROR
;DECREMENT THE ERROR COUNT
;RETRY COUNTER
;TRY AGAIN IF NOT FINISHED
;CLEAN OFF THE STACK
;TAKE THE "PARITY ON WRITE" ERROR EXIT
;EXIT
;ADJUST FOR ERROR FREE EXIT

```

```

;DPB POINTER TO R2
;SAVE THE DRIVES REG'S
;SAVE R0 - R5
;QUEUE ENTRY FOR THE DRIVE ?
;BR IF NONE
;SELECT DRIVE
;GET THE ERROR TABLE POINTER
;EXIT IF NO ADDRESS
;COUNTER & POINTER
;REACHED THE BUFFER REGISTER ?
;BR IF NOT
; 'OR' SET ?
;BR IF SET
;STORE RPDB AS ZEROES
;CONTINUE
;READ THE SELECTED REGISTER
;REGISTER INDEX
;ERROR RETURN ADDRESS
;STORE THE REGISTER CONTENTS
;REACHED THE END ?
;BR IF YES
;INCREMENT THE REGISTER INDEX
;CONTINUE READING THE REGISTERS
;PROCESS THE UNCORRECTABLE PARITY ERROR
;RESTORE R0 - R5
;RETURN

```

```

;DRIVE NUMBER TO R1
;SET "IE"
;SAVE R4
;PICKUP ADDRESS OF RPCS1
;SELECT DRIVE

```

```

0715 041100 011446      MOV      (R4),-(SP)      ;READ RPCS1
0716 041102 052715 040000  BIS      #BIT14,(SP)    ;SET THE "TRE" BIT OF THE WORD READ
0717 041106 000316      SWAB     (SP)          ;ADJUST FOR DATO
0718 041110 112714 000100  MOVVB   #BIT06,(R4)    ;SET "IE"
0719 041114 032764 010000 000010  BIT      #BIT12,RPCS2(R4) ;IS "NED"=1?
0720 041122 001002      BNE     1$            ;YES--CLEAR "TRE"
0721 041124 005726      TST     (SP)+         ;CLEAN OFF THE STACK
0722 041126 000402      BR      2$
0723 041130 112664 000001  1$:     MOVVB  (SP)+,1(R4) ;CLEAR "TRE"
0724 041134 012604      MOV     (SP)+,R4      ;RESTORE R4
0725 041136 000207      RTS      PC          ;RETURN TO CALLER

;QUEUE COUNT
QCNT:  .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

;QUEUE INPUT POINTERS
QINPT: .WORD QDRV0      ;DRIVE 0
       .WORD QDRV1      ;DRIVE 1
       .WORD QDRV2      ;DRIVE 2
       .WORD QDRV3      ;DRIVE 3
       .WORD QDRV4      ;DRIVE 4
       .WORD QDRV5      ;DRIVE 5
       .WORD QDRV6      ;DRIVE 6
       .WORD QDRV7      ;DRIVE 7

;QUEUE OUTPUT POINTERS
QOUTPT: .WORD QDRV0     ;DRIVE 0
        .WORD QDRV1     ;DRIVE 1
        .WORD QDRV2     ;DRIVE 2
        .WORD QDRV3     ;DRIVE 3
        .WORD QDRV4     ;DRIVE 4
        .WORD QDRV5     ;DRIVE 5
        .WORD QDRV6     ;DRIVE 6
        .WORD QDRV7     ;DRIVE 7

CSTART: .WORD QDRV0     ;DRIVE 0 START ADDRESS
CSTOP:  .WORD QDRV1     ;DRIVE 0 STOP ADDRESS & DRIVE 1 START ADDRESS
        .WORD QDRV2     ;STOP DRIVE 1--START DRIVE 2
        .WORD QDRV3     ;STOP DRIVE 2--START DRIVE 3
        .WORD QDRV4     ;STOP DRIVE 3--START DRIVE 4
        .WORD QDRV5     ;STOP DRIVE 4--START DRIVE 5
        .WORD QDRV6     ;STOP DRIVE 5--START DRIVE 6
        .WORD QDRV7     ;STOP DRIVE 6--START DRIVE 7
        .WORD QTERM

;DRIVE REQUEST QUEUES

```

0411232 000010
 0411252 000010
 0411272 000010
 0411292 000010
 0411312 000010
 0411332 000010
 0411352 000010
 0411372 000010
 0411392 000010
 0411412 041432

QDRV0: .BLKW 10
 QDRV1: .BLKW 10
 QDRV2: .BLKW 10
 QDRV3: .BLKW 10
 QDRV4: .BLKW 10
 QDRV5: .BLKW 10
 QDRV6: .BLKW 10
 QDRV7: .BLKW 10
 QTERM=.

:ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES

:CALL

JSR PC,CLRQUE

CLRQUE: SAVREG

MOV #QCNT,R2

CLR (R2)+

CLR (R2)+

CLR (R2)+

CLR (R2)+

MOV #B,R3

MOV #QSTART,R1

1\$: MOV (R1)+,(R2)+

DEC R3

BNE 1\$

MOV #B,R3

2\$: MOV (R1)+,(R2)+

DEC R3

BNE 2\$

RESREG

RTS PC

:SAVE R0 - R5
 :ZERO THE QUEUE COUNTS
 :DRIVES 0 & 1
 :DRIVES 2 & 3
 :DRIVES 4 & 5
 :DRIVES 6 & 7
 :MOVE THE STARTING
 :ADDRESS OF THE QUEUE INTO
 :THE QUEUE INPUT POINTER

:MOVE THE STARTING ADDRESS
 :OF THE QUEUE INTO THE
 :QUEUE OUTPUT POINTER

:RESTORE R0 - R5

:EMPTY THE QUEUE SPECIFIED BY R1

:CALL

MOV DRVNUM,R1

JSR PC,EMPTYQ

:DRIVE NUMBER TO R1

EMPTYQ: CLRB 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 R0,DRVQUE

:GO PUT REQUEST IN QUEUE

RETURN1

:RETURN HERE IF QUEUE IS FULL

RETURN2

:RETURN HERE IF REQUEST IS IN QUEUE

041432 104412
 041434 012702 041140
 041440 005022
 041442 005022
 041444 005022
 041446 005022
 041450 012703 000010
 041452 012701 041210
 041454 012122
 041456 005303
 041458 001375
 041460 012703 000010
 041462 012701 041210
 041464 012122
 041466 005303
 041468 001375
 041470 104412
 041472 000207
 041474 000207
 041476 000207
 041478 000207
 041480 000207
 041482 000207
 041484 000207
 041486 000207
 041488 000207
 041490 000207
 041492 000207
 041494 000207
 041496 000207
 041498 000207
 041500 000207
 041502 000207
 041504 000207
 041506 000207
 041508 000207
 041510 000207
 041512 000207
 041514 000207
 041516 000207
 041518 000207
 041520 000207
 041522 000207
 041524 000207
 041526 000207

04.530 :22761 000010 041140

DRVQUE: CMPB #10,QCNT(R1) :IS QUEUE FULL?

H13

CPY000 RPO4 5 6 MLT-DR LGC
CPY000.F11 15-DEC 77 10:58

MAC711 30(1046) 15-DEC-77 11.03 PAGE 164
SINGLE DUAL PORT RH11/RPO4/5 6 DRIVER (REV 1.0)

SEQ 0163

```

0827 041536 001421          BEQ      2$          ;BR IF YES-TAKE RETURN1
0828 041540 105261 041140    INCB     QCNT(R1)    ;INCREMENT QUEUE COUNT
0829 041544 006301          ASL      R1
0830 041546 010271 041150    MOV      R2,QINPT(R1) ;PUT THIS REQUEST IN QUEUE
0831 041552 062761 000002 041150    ADD      #2,QINPT(R1) ;UPDATE THE QUEUE POINTER
0832 041556 026161 041150 041212    CMP      QINPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER
0833 041566 001003          BNE      1$          ;BRANCH IF NO
0834 041570 016161 041210 041150    MOV      QSTART(R1),QINPT(R1) ;YES--RESET POINTER
0835 041576 006201          1$:      ASR      R1
0836 041600 005720          TST     (R0)+       ;TAKE RETURN 2
0837 041602 000200          2$:      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

0847 041604 005002          GETREQ:  CLR      R2
0848 041606 105761 041140    TSTB    QCNT(R1)    ;IS THERE ANY REQUEST IN QUEUE?
0849 041612 001404          BEQ      2$          ;NO---BRANCH
0850 041614 006301          1$:      ASL      R1
0851 041616 017102 041170    MOV      QOUTPT(R1),R2 ;PICKUP "DPB" POINTER FOR THIS DRIVE
0852 041622 006201          ASR      R1
0853 041624 000207          2$:      RTS      PC          ;RETURN TO USER

;ROUTINE TO "POP" THE REQUEST FROM QUEUE
;CALL
;      MOV      #DRVNUM,R1          ;DRIVE NUMBER TO R1
;      JSR      PC,POPQUE          ;CALL TO REMOVE REQUEST
;      RETURN   ;R2=ADDRESS OF DPB REMOVED

0862 041626 105361 041140    POPQUE:  DECB     QCNT(R1)    ;DECREMENT QUEUE COUNT
0863 041632 006301          ASL      R1
0864 041634 017102 041170    MOV      QOUTPT(R1),R2 ;GET THE "DPB" POINTER
0865 041640 062761 000002 041170    ADD      #2,QOUTPT(R1) ;UPDATE THE QUEUE POINTER
0866 041646 026161 041170 041212    CMP      QOUTPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER?
0867 041654 001003          BNE      1$          ;NO--BRANCH TO EXIT
0868 041656 016161 041210 041170    MOV      QSTART(R1),QOUTPT(R1) ;YES--RESET THE POINTER
0869 041664 006201          1$:      ASR      R1
0870 041666 000207          RTS      PC          ;RETURN TO USER

;*****
;SBTTL DATA, CONTROL, & STATUS BLOCKS
;*****
;BLOCK LOCATION EQUATE STATEMENTS
0880          000001          $FMT    =          1          ;FMT HCI,ECI OR OFFSET CODE
0881          000002          $COMNC  =          $FMT+1      ;OPERATION CODE

```

```

000003      COC003      $PSEL  =      $FMT+2      ;PORT SELECT & BITS A16, A17
000004      000004      $WRDM  =      $FMT+3      ;WORD COUNT (2'S COMP)
000005      000006      $BUF   =      $FMT+5      ;BUFFER ADDR OR REGISTER TABLE POINTER
000010      000010      $SEC   =      $FMT+7      ;SECTOR ADDRESS OR 1ST REG ADDR
000011      000011      $TRK  =      $FMT+10     ;TRACK ADDRESS OF LAST REG ADDR
000012      000012      $CYL  =      $FMT+11     ;CYLINDER ADDR
000014      000014      $REG   =      $FMT+13     ;REGISTER STORAGE (IF ERROR)
000016      000016      $STAT  =      $FMT+15     ;STATUS WORD (SET BY DRIVER)

;DRIVE'S HISTORY AND CURRENT INDICATOR STORAGE EQUATES

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

;INDEX EQUATES TO THE NEXT OPERATION PARAMETERS

000074      000074      $NCODE =      $WRDL+54    ;NEXT OPERATION CODE
000075      000075      $NPATC =      $NCODE+1    ;NEXT PATTERN
000076      000076      $NSEC  =      $NCODE+2    ;NEXT SECTOR
000077      000077      $NTRK  =      $NCODE+3    ;NEXT TRACK
000100      000100      $NCYL  =      $NCODE+4    ;NEXT CYLINDER
000102      000102      $NWRDL =      $NCODE+6    ;NEXT BUFFER SIZE
000104      000104      $NEXT  =      $NCODE+10   ;PARAMETER SELECTION INDICATOR

;INDEX EQUATES FOR MAXIMUM/MINIMUM ADDRESSES

000106      000106      $MAXCYL =      $NCODE+12  ;MAXIMUM CYLINDER ADDRESS
000110      000110      $MINCYL =      $MAXCYL+2  ;MINIMUM CYLINDER ADDRESS
000112      000112      $MAXTRK =      $MAXCYL+4  ;MAXIMUM TRACK ADDRESS
000114      000114      $MINTRK =      $MAXCYL+6  ;MINIMUM TRACK ADDRESS
000116      000116      $MAXSEC =      $MAXCYL+10 ;MAXIMUM SECTOR ADDRESS
000120      000120      $MINSEC =      $MAXCYL+12  ;MINIMUM SECTOR ADDRESS
000122      000122      $FIRST =      $MAXCYL+14  ;FIRST OPERATION INDICATOR

;BAD SECTOR/TRACK ADDRESS STORAGE AREA INDEX EQUATE

000124      000124      $BDSEC =      $MAXCYL+16  ;BAD SECTOR STORAGE TABLE

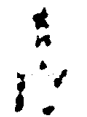
;DRIVE ID AREA INDEX EQUATE

```

```

8939          000224          $ORVID =          $B0SEC+100          ;DRIVE ID
8940
8941          :RH11/RPO4/5/6 REGISTER EQUATES
8942
8943          000234          $RPCS1 =          $ORVID+10          ;RPO4 REGISTER STORAGE
8944          000236          $RPWC =          $RPCS1+2
8945          000240          $RPBA =          $RPCS1+4
8946          000242          $RPOA =          $RPCS1+6
8947          000244          $RPCS2 =          $RPCS1+10
8948          000246          $RPDS1 =          $RPCS1+12
8949          000250          $RPER1 =          $RPCS1+14
8950          000252          $RPAS =          $RPCS1+16
8951          000254          $RPLA =          $RPCS1+20
8952          000256          $RPOB =          $RPCS1+22
8953          000260          $RPMR =          $RPCS1+24
8954          000262          $RPDT =          $RPCS1+26
8955          000264          $RPSN =          $RPCS1+30
8956          000266          $RPOF =          $RPCS1+32
8957          000270          $RPCA =          $RPCS1+34
8958          000272          $RPCC =          $RPCS1+36
8959          000274          $RPER2 =          $RPCS1+40
8960          000276          $RPER3 =          $RPCS1+42
8961          000300          $RPEC1 =          $RPCS1+44
8962          000302          $RPEC2 =          $RPCS1+46
8963
8964          ;BLOCK FOR DRIVE 0
8965
8966          041670          000          000          DRIVED: .BYTE 0,0          ;DRIVE NUMBER
8967          041672          000005          .BLKW 5
8968          041704          042124          .WORD .+$RPCS1-$REG
8969          041706          000266          .BLKB $RPEC2-$REG
8970
8971          ;BLOCK FOR DRIVE 1
8972
8973
8974          042174          001          000          DRIVE1: .BYTE 1,0          ;DRIVE NUMBER
8975          042176          000005          .BLKW 5
8976          042210          042430          .WORD .+$RPCS1-$REG
8977          042212          000266          .BLKB $RPEC2-$REG
8978
8979          ;BLOCK FOR DRIVE 2
8980
8981
8982          042500          002          000          DRIVE2: .BYTE 2,0          ;DRIVE NUMBER
8983          042502          000005          .BLKW 5
8984          042514          042734          .WORD .+$RPCS1-$REG
8985          042516          000266          .BLKB $RPEC2-$REG
8986
8987          ;BLOCK FOR DRIVE 3
8988
8989
8990          043004          003          000          DRIVE3: .BYTE 3,0          ;DRIVE NUMBER
8991          043006          000005          .BLKW 5
8992          043020          043240          .WORD .+$RPCS1-$REG
8993          043022          000266          .BLKB $RPEC2-$REG

```



```

8995
8996
8997          ;BLOCK FOR DRIVE 4
8998
8999 043310      004      000      DRIVE4: .BYTE 4,0          ;DRIVE NUMBER
9000 043312      000005          .BLKW 5
9001 043324      013544          .WORD  +$RPCS1-$REG
9002 043326      000266          .BLKB  $RPEC2-$REG
9003
9004
9005          ;BLOCK FOR DRIVE 5
9006
9007 043614      005      000      DRIVES: .BYTE 5,0          ;DRIVE NUMBER
9008 043616      000005          .BLKW 5
9009 043630      044050          .WORD  +$RPCS1-$REG
9010 043632      000266          .BLKB  $RPEC2-$REG
9011
9012
9013          ;BLOCK FOR DRIVE 6
9014
9015 044120      006      000      DRIVE6: .BYTE 6,0          ;DRIVE NUMBER
9016 044122      000005          .BLKW 5
9017 044134      044354          .WORD  +$RPCS1-$REG
9018 044136      000266          .BLKB  $RPEC2-$REG
9019
9020
9021          ;BLOCK FOR DRIVE 7
9022
9023 044424      007      000      DRIVE7: .BYTE 7,0          ;DRIVE NUMBER
9024 044426      000005          .BLKW 5
9025 044440      044660          .WORD  +$RPCS1-$REG
9026 044442      000266          .BLKB  $RPEC2-$REG
9027
9028
9029          ;GENERAL PURPOSE DPB - USED BY 'READHD', 'RECALT', 'OFFSET', & 'RTNCTR'
9030
9031 044730      000000      000000      177774      GENDPB: .WORD 0,0,-4,CYLDER
9032 044736      054456
9033 044740      000000      000000      044750          .WORD 0,0,GENREG,0
9034 044746      000000
9035 044750      000024          GENREG: .BLKW 24          ;REGISTER STORAGE IF ERROR
9036
9037          ;*****
9038          .SBTTL  ERROR MESSAGES
9039          ;*****
9040
9041
9042
9043 045020      044122      030461      044440      EM1:  .ASCIZ  /RH11 INTERRUPT OCCURRED (RPAS = 0)/
9044 045026      052116      051105      052522
9045 045034      052120      047440      041503
9046 045042      051125      042522      020104
9047 045050      051050      040520      020123
9048 045056      020075      024460          000
9049
9050 045062          125      042516      050130      EM2:  .ASCIZ  /UNEXPECTED ATTENTION OCCURRED/

```

9051	045070	041505	042524	020104		
9052	045076	052101	042524	052116		
9053	045104	047511	020116	041517		
9054	045112	052503	051122	042105		
9055	045120	000				
9056						
9057	045121	115	051501	041123	EM3:	.ASCIZ /MASSBUS PARITY ERROR (MCPE=1)/
9058	045126	051525	050040	051101		
9059	045134	052111	020131	051105		
9060	045142	047522	020122	046450		
9061	045150	050103	036505	024461		
9062	045156	000				
9063						
9064	045157	115	051501	041123	EM4:	.ASCIZ /MASSBUS PARITY ERROR (PAR=1)/
9065	045164	051525	050040	051101		
9066	045172	052111	020131	051105		
9067	045200	047522	020122	050050		
9068	045206	051101	030475	000051		
9069						
9070	045214	042101	051104	051505	EM5:	.ASCIZ /ADDRESS PLUG CHANGE BIT SET/
9071	045222	020123	046120	043525		
9072	045230	041440	040510	043516		
9073	045236	020105	044502	020124		
9074	045244	042523	000124			
9075						
9076	045250	044122	030461	042040	EM6:	.ASCIZ /RH11 DIDN'T RESPOND TO ADDRESSING/
9077	045256	042111	023516	020124		
9078	045264	042522	050123	047117		
9079	045272	020104	047524	040440		
9080	045300	042104	042522	051523		
9081	045306	047111	000107			
9082						
9083	045312	047125	047503	051122	EM10:	.ASCIZ /UNCORRECTABLE MASSBUS PARITY ERROR/
9084	045320	041505	040524	046102		
9085	045326	020105	040515	051523		
9086	045334	052502	020123	040520		
9087	045342	044522	054524	042440		
9088	045350	051122	051117	000		
9089						
9090	045355	106	052101	046101	EM11:	.ASCIZ /FATAL MASSBUS PARITY ERROR/
9091	045362	046440	051501	041123		
9092	045370	051525	050040	051101		
9093	045376	052111	020131	051105		
9094	045404	047522	000122			
9095						
9096	045410	042520	051522	051511	EM12:	.ASCIZ /PERSISTENT DEVICE UNSAFE/
9097	045416	042524	052116	042040		
9098	045424	053105	041511	020105		
9099	045432	047125	040523	042506		
9100	045440	000				
9101						
9102	045441	117	042520	040522	EM13:	.ASCIZ /OPERATION NOT COMPLETED WITHIN TIME LIMIT/
9103	045446	044524	047117	047040		
9104	045454	052117	041440	046517		
9105	045462	046120	052105	042105		
9106	045470	053440	052111	044510		

9107	045476	020116	044524	042515	
9108	045504	046040	046511	052111	
9109	045512	000			
9110					
9111	045513	104	044522	042526	EM14: .ASCIZ /DRIVE WENT OFFLINE/
9112	045520	053440	047105	020124	
9113	045528	043117	046106	047111	
9114	045534	000105			
9115					
9116	045536	047516	051040	051505	EM15: .ASCIZ /NO RESPONSE TO PORT REQUEST/
9117	045544	047520	051516	020105	
9118	045552	047524	050040	051117	
9119	045560	020124	042522	052521	
9120	045566	051505	000124		
9121					
9122	045572	042510	042101	051105	EM20: .ASCIZ /HEADER CRC ERROR/
9123	045600	041440	041522	042440	
9124	045606	051122	051117	000	
9125					
9126	045613	104	052101	020101	EM21: .ASCIZ /DATA CHECK ('DCK') ERROR/
9127	045620	044103	041505	020113	
9128	045626	023450	041504	023513	
9129	045634	020051	051105	047522	
9130	045642	000122			
9131					
9132	045644	051127	052111	020105	EM22: .ASCIZ /WRITE CHECK ERROR - DATA CHECK ('DCK') SET/
9133	045652	044103	041505	020113	
9134	045660	051105	047522	020122	
9135	045666	020055	040504	040524	
9136	045674	041440	042510	045503	
9137	045702	024040	042047	045503	
9138	045710	024447	051440	052105	
9139	045716	000			
9140					
9141	045717	127	044522	042524	EM23: .ASCIZ /WRITE CHECK ERROR - DATA CHECK ('DCK') NOT SET/
9142	045724	041440	042510	045503	
9143	045732	042440	051122	051117	
9144	045740	026440	042040	052101	
9145	045746	020101	044103	041505	
9146	045754	020113	023450	041504	
9147	045762	023513	020051	047516	
9148	045770	020124	042523	000124	
9149					
9150	045776	042510	042101	051105	EM24: .ASCIZ /HEADER READ ERROR - 'FMT' BIT DROPPED/
9151	046004	051040	040505	020104	
9152	046012	051105	047522	020122	
9153	046020	020055	043047	052115	
9154	046026	020047	044502	020124	
9155	046034	051104	050117	042520	
9156	046042	000104			
9157					
9158	046044	042510	042101	051105	EM25: .ASCIZ /HEADER READ ERROR - HEADER COMPARE ('HCE') ERROR
9159	046052	051040	040505	020104	
9160	046060	051105	047522	020122	
9161	046066	020055	042510	042101	
9162	046074	051105	041440	046517	

9163	046102	040520	042522	024040		
9164	046110	044047	042503	024447		
9165	046116	042440	051122	051117		
9166	046124	000				
9167						
9168	046125	106	051117	040515	EM26:	.ASCIZ /FORMAT ERROR ('FER')/
9169	046132	020124	051105	047522		
9170	046140	020122	023450	042506		
9171	046146	023522	000051			
9172						
9173	046152	042510	042101	051105	EM27:	.ASCIZ /HEADER COMPARE ('HCE') ERROR/
9174	046160	041440	046517	040520		
9175	046166	042522	024040	044047		
9176	046174	042503	024447	042440		
9177	046202	051122	051117	000		
9178						
9179	046207	115	051511	042503	EM30:	.ASCIZ /MISCELLANEOUS DRIVE ERROR/
9180	046214	046114	047101	047505		
9181	046222	051525	042040	044522		
9182	046230	042526	042440	051122		
9183	046236	051117	000			
9184						
9185	046241	117	042520	040522	EM31:	.ASCIZ /OPERATION INCOMPLETE ('OPI') ERROR/
9186	046246	044524	047117	044440		
9187	046254	041516	046517	046120		
9188	046262	052105	020105	023450		
9189	046270	050117	023511	020051		
9190	046276	051105	047522	000122		
9191						
9192	046304	051104	053111	020105	EM32:	.ASCIZ /DRIVE TIMING ('DTE') ERROR/
9193	046312	044524	044515	043516		
9194	046320	024040	042047	042524		
9195	046326	024447	042440	051122		
9196	046334	051117	000			
9197						
9198	046337	120	051101	052111	E'133:	.ASCIZ /PARITY ('PAR') ERROR AFTER OPERATION STARTED/
9199	046344	020131	023450	040520		
9200	046352	023522	020051	051105		
9201	046360	047522	020122	043101		
9202	046366	042524	020122	050117		
9203	046374	051105	052101	047511		
9204	046402	020116	052123	051101		
9205	046410	042524	000104			
9206						
9207	046414	051127	052111	020105	EM34:	.ASCIZ /WRITE CLOCK FAILURE ('WCF') ERROR/
9208	046422	046103	041517	020113		
9209	046430	040506	046111	051125		
9210	046436	020105	023450	041527		
9211	046444	023506	020051	051105		
9212	046452	047522	000122			
9213						
9214	046456	047111	040526	044514	EM35:	.ASCIZ /INVALID ADDRESS ('IAE') ERROR/
9215	046464	020104	042101	051104		
9216	046472	051505	020123	023450		
9217	046500	040511	023505	020051		
9218	046506	051105	047522	000122		

9219					
9220	046514	051127	052111	020105	EM36: .ASCIZ /WRITE LOCK ('WLE') ERROR/
9221	046522	047514	045503	024040	
9222	046530	053447	042514	024447	
9223	046536	042440	051122	051117	
9224	046544	000			
9225					
9226	046545	104	052101	020101	EM37: .ASCIZ /DATA CHECK ('DCK') SET DURING WRITE CHECK COMMAND/
9227	046552	044103	041505	020113	
9228	046560	023450	041504	023513	
9229	046566	020051	042523	020124	
9230	046574	052504	044522	043516	
9231	046602	053440	044522	042524	
9232	046610	041440	042510	045503	
9233	046616	041440	046517	040515	
9234	046624	042116	000		
9235					
9236	046627	122	030510	020061	EM40: .ASCIZ /RH11 OR UNIBUS TRANSFER ERROR/
9237	046634	051117	052440	044516	
9238	046642	052502	020123	051124	
9239	046650	047101	043123	051105	
9240	046656	042440	051122	051117	
9241	046664	000			
9242					
9243	046665	102	051525	040440	EM41: .ASCIZ /BUS ADDRESS OR WORD COUNT INCORRECT/
9244	046672	042104	042522	051523	
9245	046700	047440	020122	047527	
9246	046706	042122	041440	052517	
9247	046714	052116	044440	041516	
9248	046722	051117	042522	052103	
9249	046730	000			
9250					
9251	046731	104	052101	020101	EM42: .ASCIZ /DATA COMPARE ERRORS - NO OTHER ERROR(S) DETECTED/
9252	046736	047503	050115	051101	
9253	046744	020105	051105	047522	
9254	046752	051522	026440	047040	
9255	046760	020117	052117	042510	
9256	046766	020122	051105	047522	
9257	046774	024122	024523	042040	
9258	047002	052105	041505	042524	
9259	047010	000104			
9260					
9261	047012	040503	023516	020124	EM43: .ASCIZ CAN'T MATCH DATA READ WITH A PATTERN
9262	047020	040515	041524	020110	
9263	047026	040504	040524	051040	
9264	047034	040505	020104	044527	
9265	047042	044124	040440	050040	
9266	047050	052101	042524	047122	
9267	047056	000			
9268					
9269	047057	105	051122	051117	EM44: .ASCIZ ERROR BIT 51 SET, BUT NO ERROR SIGNALLED BY THE RH11/
9270	047064	041040	052111	051450	
9271	047072	020051	042523	026124	
9272	047100	041040	052125	047040	
9273	04710E	020117	051105	047522	
9274	047114	020122	044523	047107	

9275	047122	046101	042105	041040	
9276	047130	020131	044124	020105	
9277	047136	044122	030461	000	
9278	047143	0435105	041503	046040	EM45: .ASCIZ /ECC LOGIC FAILURE - POSITION REGISTER VALUE TOO LARGE/
9279	047150	044501	041511	043040	
9280	047156	026440	052514	042522	
9281	047164	026440	050040	051517	
9282	047172	052111	047511	020116	
9283	047200	042522	044507	052123	
9284	047206	051105	053040	046101	
9285	047214	042525	052040	047517	
9286	047222	046040	051101	042507	
9288	047230	000			
9289					
9290	047231	102	051525	040440	EM46: .ASCIZ /BUS ADDRESS AND WORD COUNT NOT CONSISTENT/
9291	047236	042104	042522	051523	
9292	047244	040440	042116	053440	
9293	047252	051117	020104	047503	
9294	047260	047125	020124	047516	
9295	047266	020124	047503	051516	
9296	047274	051511	042524	052116	
9297	047302	000			
9298					
9299	047303	123	042505	020113	EM50: .ASCIZ /SEEK INCOMPLETE ('SKI') OR OFF CYLINDER ('CYL') ERROR
9300	047310	047111	047503	050115	
9301	047316	042514	042524	024040	
9302	047324	051447	044513	024447	
9303	047332	047440	020122	043117	
9304	047340	020106	054503	044514	
9305	047346	042116	051105	024040	
9306	047354	047447	054503	023514	
9307	047362	020051	051105	047522	
9308	047370	000122			
9309					
9310	047372	051120	043517	040522	EM51: .ASCIZ /PROGRAM DETECTED POSITIONING ERROR/
9311	047400	020115	042504	042524	
9312	047406	052103	042105	050040	
9313	047414	051517	052111	047511	
9314	047422	044516	043516	042440	
9315	047430	051122	051117	000	
9316					
9317	047435	104	044522	042526	EM60: .ASCIZ /DRIVE UNSAFE ERROR/
9318	047442	052440	051516	043101	
9319	047450	020105	051105	047522	
9320	047456	000122			
9321					
9322	047460	050122	051501	000	DM1: .ASCIZ /PPAS/
9323					
9324	047465	104	044522	042526	DM2: .ASCIZ /DRIVE RPOS1 RPER1 RPER2 RPER3 RPAS
9325	047472	020040	051040	042120	
9326	047500	030523	020040	051040	
9327	047506	042520	030522	020040	
9328	047514	051040	042520	031122	
9329	047522	020040	051040	042520	
9330	047530	031522	020040	051040	

9331	047536	040520	000123						
9332									
9333	047542	051104	053111	020105	DH3:	.ASCIZ	/DRIVE	REG	ADR DATA/
9334	047550	020040	042522	020107					
9335	047556	042101	020122	042040					
9336	047564	052101	000101						
9337									
9338	047570	051104	053111	020105	DH4:	.ASCIZ	/DRIVE	REG	ADR GOOD BAD/
9339	047576	020040	042522	020107					
9340	047604	042101	020122	020040					
9341	047612	043440	047517	020104					
9342	047620	020040	041040	042101					
9343	047626	000							
9344									
9345	047627	044	050122	042101	DH6:	.ASCIZ	/SRPADR/		
9346	047634	000122							
9347									
9348	047636	051104	020126	050122	DH14:	.ASCII	/DRV	RPCS1	RPCS2 RPDS1 RPER1 /
9349	047644	051503	020061	020040					
9350	047652	050122	051503	020062					
9351	047660	020040	050122	051504					
9352	047666	020061	020040	050122					
9353	047674	051105	020061	020040					
9354	047702	050122	051105	020062		.ASCIZ	/RPER2	RPER3	RPEC1 RPEC2/<CR><LF>
9355	047710	020040	050122	051105					
9356	047716	020063	020040	050122					
9357	047724	041505	020061	020040					
9358	047732	050122	041505	006462					
9359	047740	000012							
9360									
9361	047742	050122	041527	020040	DH15:	.ASCII	/RPWC	RPBA	RPDA RPAS RPLA /
9362	047750	020040	050122	040502					
9363	047756	020040	020040	050122					
9364	047764	040504	020040	020040					
9365	047772	050122	051501	020040					
9366	050000	020040	050122	040514					
9367	050006	020040	020040						
9368	050012	050122	041104	020040		.ASCIZ	/RPDB	RPMR	RPDT/<CR><LF>
9369	050020	020040	050122	051115					
9370	050026	020040	020040	050122					
9371	050034	052104	005015	000					
9372									
9373	050041	122	051520	020116	DH16:	.ASCIZ	/RPSN	RPOF	RPCA RPCC STATUS/<CR><LF>
9374	050046	020040	051040	047520					
9375	050054	020106	020040	051040					
9376	050062	041520	020101	020040					
9377	050070	051040	041520	020103					
9378	050076	020040	051440	040524					
9379	050104	052524	006523	000012					
9380									
9381						.EVEN			
9382									
9383	050112	001244	000000		DT1:	.WORD	ATTN,0		
9384									
9385	050116	001242	033164	033166	DT2:	.WORD	DRIVE,RPERRS,RPERRS+2,RPERRS+4,RPERRS+6,ATTN,0		
9386	050124	033170	033172	001244					

9387	050132	000000							
9388									
9389	050134	001242	040432	040434	DT3:	.WORD	DRIVE, RD. ADR, RD. WRD, 0		
9390	050142	000000							
9391									
9392	050144	001242	040652	040650	DT4:	.WORD	DRIVE, WRT. AD, WPT. WD, RD. WRD, 0		
9393	050152	040434	000000						
9394									
9395	050156	001170	000000		DT6:	.WORD	SRPADR, 0		
9396									
9397	050162	000234	000244	000246	DT14:	.WORD	SRPCS1, SRPCS2, SRPDS1, SRPEF1, SRPER2, SRPER3, SRPEC1, SRPEC2, 0		
9398	050170	000250	000274	000276					
9399	050176	000300	000302	000000					
9400									
9401	050204	000236	000240	000242	DT15:	.WORD	SRPWC, SRPBA, SRPDA, SRPAS, SRPLA, SRPOB, SRPMA, SRPCT, 0		
9402	050212	000252	000254	000256					
9403	050220	000260	000262	000000					
9404									
9405	050226	000264	000266	000270	DT16:	.WORD	SRPSN, SRPOF, SRPCA, SRPCC, STATUS, 0		
9406	050234	000272	000016	000000					
9407									
9408	050242	051120	051505	047105	LIN2C:	.ASCIZ	/PRESENT ORDER = /		
9409	050250	020124	051117	042504					
9410	050256	020122	020075	000					
9411	050263	040	050040	042522	LIN2P:	.ASCIZ	/ PREVIOUS ORDER = /		
9412	050270	044526	052517	020123					
9413	050276	051117	042504	020122					
9414	050304	020075	000						
9415	050307	052	042440	051122	LIN2S:	.ASCIZ	2* ERROR AT BAD TRACK/SECTOR2		
9416	050314	051117	040440	020124					
9417	050322	040502	020104	051124					
9418	050330	041501	027513	042523					
9419	050336	052103	051117	000					
9420	050343	105	051122	051117	LINM3:	.ASCIZ	/ERROR AT C/		
9421	050350	040440	020124	000103					
9422	050356	052040	000		T:	.ASCIZ	/ T /		
9423	050361	120	042522	042523	LINN3:	.ASCIZ	/PRESENT ADDR = C/		
9424	050366	052116	040440	042104					
9425	050374	020122	020075	000103					
9426	050402	051440	000		S:	.ASCIZ	/ S /		
9427	050405	040	020040	051120	LINP3:	.ASCIZ	/ PREV ADDR = C/		
9428	050412	053105	040440	042104					
9429	050420	020122	020075	000103					
9430	050426	052123	051101	020124	LINS3:	.ASCIZ	/START CYL = /		
9431	050434	054503	020114	020075					
9432	050442	000							
9433	050443	040	042440	042116	LINEN3:	.ASCIZ	/ END CYL = /		
9434	050450	041440	046131	036440					
9435	050456	000040							
9436	050460	020040	041501	052524	LINA3:	.ASCIZ	/ ACTUAL CYL = /		
9437	050466	046101	041440	046131					
9438	050474	036440	000040						
9439	050500	020040	051124	020113	LINT3:	.ASCIZ	/ TRK = /		
9440	050506	020075	000						
9441	050511	040	050122	040503	LINCA3:	.ASCIZ	/ RPCA = /		
9442	050516	036440	000040						

9143	050522	050122	040504	036440	LINDA3: .ASCIZ /RPOA = /
9144	050530	000040			
9145	050532	050122	040502	036440	LINB3: .ASCIZ /RPBA = /
9146	050540	000040			
9147	050542	020040	050122	041527	LINW3: .ASCIZ / RPWC = /
9148	050550	036440	000040		
9149	050554	052123	051101	020124	LINST3: .ASCIZ /START TRK = /
9150	050562	051124	020113	020075	
9151	050570	000			
9152	050571	123	040524	052122	LINSS3: .ASCIZ /START SEC = /
9153	050576	051440	041505	036440	
9154	050604	000040			
9155	050606	052502	043106	051105	LINM4: .ASCIZ /BUFFER ADDR = /
9156	050614	040440	042104	020122	
9157	050622	020075	000		
9158	050625	040	051440	055111	LINS4: .ASCIZ / SIZE = /
9159	050632	020105	020075	000	
9160	050637	040	040440	052103	LINX4: .ASCIZ / ACTUAL NMBR WRDS XFRO = /
9161	050644	040525	020114	046516	
9162	050652	051102	053440	042122	
9163	050660	020123	043130	042122	
9164	050666	036440	000040		
9165	050672	047507	042117	042040	LIND5: .ASCIZ /GOOD DATA = /
9166	050700	052101	020101	020075	
9167	050706	000			
9168	050707	040	041040	042101	LINB5: .ASCIZ / BAD DATA = /
9169	050714	042040	052101	020101	
9170	050722	020075	000		
9171	050725	040	051440	041505	LINP5: .ASCIZ / SECT POS = /
9172	050732	020124	047520	020123	
9173	050740	020075	000		
9174	050743	110	040505	042504	LINS5: .ASCIZ /HEADER FROM ERROR SECTOR = /
9175	050750	020122	051106	046517	
9176	050756	042440	051122	051117	
9177	050764	051440	041505	047524	
9178	050772	020122	020075	000	
9179	050777	122	042520	030503	LINEP5: .ASCIZ /RPEC1 = /
9180	051004	036440	000040		
9181	051010	051040	042520	031103	LINE05: .ASCIZ / RPEC2 = /
9182	051016	036440	000040		
9183	051022	042523	052103	051117	LINB6: .ASCIZ /SECTOR IS ECC CORRECTABLE /
9184	051030	044440	020123	041505	
9185	051036	020103	047503	051122	
9186	051044	041505	040524	046102	
9187	051052	020105	000		
9188	051055	123	041505	047524	LINC6: .ASCIZ /SECTOR READ CORRECTLY
9189	051062	020122	042522	042101	
9190	051070	041440	051117	042522	
9191	051076	052103	054514	000040	
9192	051104	047503	051122	041505	LING6: .ASCIZ /CORRECTED ON /
9193	051112	042524	020104	047117	
9194	051120	000040			
9195	051122	051040	052105	044522	LINR6: .ASCIZ / RETRIES/
9196	051130	051505	000		
9197	051133	125	041516	051117	LINU06: .ASCIZ /UNCORRECTABLE AFTER
9198	051140	042522	052103	041101	

9499	051146	042514	040440	052106					
9500	051154	051105	000040						
9501	051160	020040	047524	040524	LIN7M:	.ASCIZ	/	TOTAL MISPOS ERR = /	
9502	051166	020114	044515	050123					
9503	051174	051517	042440	051122					
9504	051202	036440	000040						
9505	051206	051117	042504	051522	LIN7O:	.ASCIZ	/	ORDERS:/	
9506	051214	000072							
9507	051216	052040	052117	046101	LIN7P:	.ASCIZ	/	TOTAL SEEKS = /	
9508	051224	051440	042505	051513					
9509	051232	036440	000040						
9510	051236	052040	052117	046101	LIN7S:	.ASCIZ	/	TOTAL SKI,OCYL ERR = /	
9511	051244	051440	044513	047454					
9512	051252	054503	020114	051105					
9513	051260	020122	020075	000					
9514	051265	040	042440	051122	LIN7T:	.ASCIZ	/	ERRORS:/	
9515	051272	051117	035123	000					
9516	051277	040	053440	042122	LIN7X:	.ASCIZ	/	WRDS XFR:/	
9517	051304	020123	043130	035122					
9518	051312	000							
9519	051313	040	053440	042122	LIN7R:	.ASCIZ	/	WRDS READ:/	
9520	051320	020123	042522	042101					
9521	051326	000072							
9522	051330	044504	043106	051105	LIN8M:	.ASCIZ	/	DIFFERENT ERROR DURING RETRY/	
9523	051336	047105	020124	051105					
9524	051344	047522	020122	052504					
9525	051352	044522	043516	051040					
9526	051360	052105	054522	000					
9527	051365	104	052101	020101	LIN9B:	.ASCIZ	/	DATA COMPARISON ERRORS/	
9528	051372	047503	050115	051101					
9529	051400	051511	047117	042440					
9530	051406	051122	051117	000123					
9531	051414	020040	020040	020040	LIN9H:	.ASCII	/	GOOD BAD<<CR><LF>	
9532	051422	020040	043440	047517					
9533	051430	020104	020040	041040					
9534	051436	042101	005015						
9535	051442	047514	020103	020040		.ASCIZ	/	LOC DATA DATA<<CR><LF>	
9536	051450	020040	042040	052101					
9537	051456	020101	020040	042040					
9538	051464	052101	006501	000012					
9539	051472	047514	020103	020040	LIN9I:	.ASCIZ	/	LOC DATA<<CR><LF>	
9540	051500	020040	042040	052101					
9541	051506	006501	000012						
9542	051512	047524	040524	020114	LIN9E:	.ASCIZ	/	TOTAL COMPARE ERRORS = /	
9543	051520	047503	050115	051101					
9544	051526	020105	051105	047522					
9545	051534	051522	036440	000040					
9546	051542	044124	020105	040504	LIN9G:	.ASCIZ	/	THE DATA COMPARED OK<<CR><LF>	
9547	051550	040524	041440	046517					
9548	051556	040520	042522	020104					
9549	051564	045517	005015	000					
9550	051571	105	051122	051117	LIN10A:	.ASCIZ	/	ERROR BURST BEGINS AT WORD /	
9551	051576	041040	051125	052123					
9552	051604	041040	043505	047111					
9553	051612	020123	052101	053440					
9554	051620	051117	020104	000					


```

9555 051625 040 047111 042040 LIN108: .ASCIZ / IN DATA FIELD OF ERROR SECTOR/ <CR> <LF>
9556 051632 052101 020101 044506
9557 051640 046105 020104 043117
9558 051646 042440 051122 051117
9559 051654 051440 041506 047524
9560 051662 006522 000012
9561 051666 051105 047522 020122 LIN109: .ASCII /ERROR WAS NOT IN THE DATA READ - / <CR> <LF>
9562 051674 040527 020123 047516
9563 051702 020124 047111 052040
9564 051710 042510 042040 052101
9565 051716 020101 042522 042101
9566 051724 026440 006440 012
9567 051731 105 041503 041440 .ASCIZ /ECC CORRECTION CAN'T BE PERFORMED/
9568 051736 051117 042522 052103
9569 051744 047511 020116 040503
9570 051752 023516 020124 042502
9571 051760 050040 051105 047506
9572 051766 046522 042105 000
9573 051773 105 041503 041440 LIN10H: .ASCII /ECC CORRECTION RESULTS/ <CR> <LF>
9574 052000 051117 042522 052103
9575 052006 047511 020116 042522
9576 052014 052523 052114 006523
9577 052022 012
9578 052023 101 042104 020122 .ASCIZ /ADDR BAD CORRECTED / <CR> <LF>
9579 052030 020040 041040 042101
9580 052036 020040 020040 041440
9581 052044 051117 042522 052103
9582 052052 042105 006440 000012
9583 052060 047503 052116 047105 LIN11H: .ASCIZ /CONTENTS OF ERPOR SECTOR (REPORTED ABOVE)/ <CR> <LF>
9584 052066 051524 047440 020106
9585 052074 051105 047522 020122
9586 052102 042523 052103 051117
9587 052110 024040 042522 047520
9588 052116 052122 042105 040440
9589 052124 047502 042526 006451
9590 052132 000012
9591 052134 042101 051104 020040 .ASCIZ /ADDR DATA/ <CR> <LF>
9592 052142 020040 040504 040524
9593 052150 005015 000
9594 052153 040 040 LIN4SP: .ASCII //
9595 052155 040 LIN5SP: .ASCII //
9596 052156 000040 LIN6SP: .ASCIZ //
9597
9598 .SBTTL TELETYPE MESSAGES
9599
9600 052160 051104 053111 000105 UNTMSG: .ASCIZ /DRIVE/
9601 052166 047440 043106 044514 UNTOFF: .ASCIZ / OFFLINE/
9602 052174 042516 000
9603 052177 040 047117 044514 UNTON: .ASCIZ / ONLINE/
9604 052204 042516 000
9605 052207 040 047516 020124 UNTNOT: .ASCIZ / NOT BEING TESTED/
9606 052214 042502 047111 020107
9607 052222 042524 052123 042105
9608 052230 000
9609 052231 040 046101 042522 UNTASH: .ASCIZ / ALREADY BEING TESTED/
9610 052236 042101 020131 042502

```

9611	052244	047111	020107	042524	
9612	052252	052123	042105	000	
9613	052257	040	047516	020124	NOTRP: .ASCIZ @ NOT AN RPO4 5/6@
9614	052264	047101	051040	030120	
9615	052272	027464	027465	000066	
9616	052300	047040	052117	050040	NOTPRS: .ASCIZ / NOT PRESENT/
9617	052306	042522	042523	052116	
9618	052314	000			
9619	052315	040	047516	020124	NOTAVL: .ASCIZ / NOT AVAILABLE/
9620	052322	053101	044501	040514	
9621	052330	046102	000105		
9622	052334	052440	051516	043101	NOTSAF: .ASCIZ / UNSAFE/
9623	052342	000105			
9624	052344	047125	052111	051440	SYSTAT: .ASCIZ /UNIT STATUS:\ <cr\>\<lf\>\<lf\></lf\></lf\></cr\>
9625	052352	040524	052524	035123	
9626	052360	005015	000012		
9627	052364	050122	032060	000	RPO4B: .ASCIZ /RPO4/
9628	052371	122	030120	000065	RPO5: .ASCIZ /RPO5/
9629	052376	050122	033060	000	RPO6: .ASCIZ /RPO6/
9630	052403	104	044522	042526	STATHC: .ASCII /DRIVE PERFORMANCE SUMMARY\ <cr\>\<lf\></lf\></cr\>
9631	052410	050040	051105	047506	
9632	052416	046522	047101	042503	
9633	052424	051440	046525	040515	
9634	052432	054522	005015		
9635	052436	051104	020126	040520	.ASCII /DRV PASS ORDERS SEEKS WRDS XFER WRDS REAC /
9636	052444	051523	047440	042122	
9637	052452	051105	020123	020040	
9638	052460	042523	045505	020123	
9639	052466	020040	051127	051504	
9640	052474	054040	042506	020122	
9641	052502	020040	051127	051504	
9642	052510	051040	040505	020104	
9643	052516	047523	052106	044040	.ASCIZ /SOFT HARD SKI MISP OTHER\ <cr\>\<lf\></lf\></cr\>
9644	052524	051101	020104	051440	
9645	052532	044513	046440	051511	
9646	052540	020120	052117	042510	
9647	052546	006522	000012		
9648	052552	047504	042516	005015	PDONE: .ASCIZ /DONE\ <cr\>\<lf\>\<lf\></lf\></lf\></cr\>
9649	052560	000012			
9650	052562	037407	040506	040524	DROPNG: .ASCIZ /07\>?FATAL OR EXCESSIVE ERRORS.
9651	052570	020114	051117	042440	
9652	052576	041530	051505	044523	
9653	052604	042526	042440	051122	
9654	052612	051117	000123		
9655	052616	047105	020104	043117	ENDPAS: .ASCIZ /END OF PASS
9656	052624	050040	051501	000123	
9657	052632	005015	047105	020104	ENDTST: .ASCIZ /\ <cr\>\<lf\>/END OF TEST/</lf\></cr\>
9658	052640	043117	052040	051505	
9659	052646	000124			
9660	052650	005015	051104	053111	DEASSG: .ASCIZ /\ <cr\>\<lf\>/DRIVE DEASSIGNED/</lf\></cr\>
9661	052656	020105	042504	051501	
9662	052664	044523	047107	042105	
9663	052672	000			
9664	052673	015	025012	025052	DRNUM: .ASCIZ /\ <cr\>\<lf\>/***** DRIVE #</lf\></cr\>
9665	052700	025052	025052	025052	
9666	052706	020052	051104	053111	

9667	052714	020105	U00043		
9668	052720	051440	040524	052122	ASGND: .ASCIZ / STARTED<<CR><LF>
9669	052726	042105	005015	000	
9670	052733	015	037412	023440	MEDCLK: .ASCIZ <CR><LF>/? 'L' OR 'P' CLOCK REQUIRED ON SYSTEM<<CR><LF>
9671	052740	023514	047440	020122	
9672	052746	050047	020047	046103	
9673	052754	041517	020113	042522	
9674	052762	052521	051111	042105	
9675	052770	047440	020116	054523	
9676	052776	052123	046505	005015	
9677	053004	000			
9678	053005	056	000		PERIOD: .ASCIZ /:/
9679	053007	077	000		QUES: .ASCIZ /:/
9680	053011	111	053116	046101	INVL0: .ASCIZ /INVALID COMMAND<<CR><LF>
9681	053016	042111	041440	046517	
9682	053024	040515	042116	005015	
9683	053032	000			
9684	053033	015	042412	052116	ENTCOM: .ASCIZ <CR><LF>/ENTER COMMANDS: <<CR><LF>
9685	053040	051105	041440	046517	
9686	053046	040515	042116	035123	
9687	053054	006440	000012		
9688	053060	047105	042524	020122	ENTDRV: .ASCIZ /ENTER I.D. FOR DRV #/
9689	053066	027111	027104	043040	
9690	053074	051117	042040	053122	
9691	053102	021440	000		
9692	053105	015	042412	052116	ENTLMT: .ASCIZ <CR><LF>/ENTER ADDRESS LIMITS FOR DRV #/
9693	053112	051105	040440	042104	
9694	053120	042522	051523	046040	
9695	053126	046511	052111	020123	
9696	053134	047506	020122	051104	
9697	053142	020126	000043		
9698	053146	047105	042524	020122	ENTADR: .ASCIZ /ENTER BAD TRK/SEC ADRS FOR DRV #/
9699	053154	040502	020104	051124	
9700	053162	027513	042523	020103	
9701	053170	042101	051522	043040	
9702	053176	051117	042040	053122	
9703	053204	021440	000		
9704	053207	072	000		COLON: .ASCIZ /:/
9705	053211	015	042012	052101	DATEIS: .ASCIZ <CR><LF>/DATE: /
9706	053216	035105	000040		
9707	053222	005015	050117	051105	IDIS: .ASCIZ <CR><LF>/OPERATOR I.D.: /
9708	053230	052101	051117	044440	
9709	053236	042056	035056	000040	
9710	053244	005015	042012	053122	MEDLIN: .ASCIZ <CR><LF><LF>/DRV DRV I.D.<<CR><LF>
9711	053252	020040	051104	020126	
9712	053260	027111	027104	005015	
9713	053266	000			
9714	053267	116	047117	006505	NONE: .ASCIZ /NONE<<CR><LF>
9715	053274	000012			
9716	053276	020077	047111	040526	BADENT: .ASCIZ /? INVALID ENTRY<<CR><LF>
9717	053304	044514	020104	047105	
9718	053312	051124	006531	000012	
9719	053320	054523	052123	046505	BUSY: .ASCIZ /SYSTEM BUSY...<<CR><LF>
9720	053326	041040	051525	027131	
9721	053334	027056	005015	000	
9722	053341	015	050012	047522	INTD0N: .ASCII <CR><LF>/PROGRAM INITIALIZATION COMPLETE

```

9723 053346 051107 046501 044440
9724 053354 044516 044524 046101
9725 053362 055111 052101 047511
9726 053370 020116 047503 050115
9727 053376 042514 042524
9728 053402 005015 054524 042520
9729 053410 040440 023440 047503
9730 053416 052116 047522 020114
9731 053424 023503 052040 020117
9732 053432 047105 042524 020122
9733 053440 047503 046515 047101
9734 053446 051504 005015 000012
9735
9736
9737
9738
9739
9740 053454 053602 000000 001404
9741 053462 053610 177777 001410
9742 053470 053740 177777 001402
9743 053476 053617 177777 001414
9744 053504 053707 000001 001420
9745 053512 053715 000007 001422
9746 053520 053732 000001 001430
9747 053526 053700 000001 001416
9748 053534 053723 000001 001424
9749 053542 053747 000001 001426
9750 053550 000000
9751
9752 053552 047105 042524 020122
9753 053560 040520 040522 042515
9754 053566 042524 051522 020072
9755 053574 000040
9756 053576 027440 000040
9757
9758 053602 040515 042130 000114
9759 053610 047111 051124 046126
9760 053616 000
9761 053617 103 050115 046514
9762 053624 000124
9763 053626 040515 041530 046131
9764 053634 000
9765 053635 115 047111 054503
9766 053642 000114
9767 053644 040515 052130 045522
9768 053652 000
9769 053653 115 047111 051124
9770 053660 000113
9771 053662 040515 051530 041505
9772 053670 000
9773 053671 115 047111 042523
9774 053676 000103
9775 053700 047506 046522 052101
9776 053706 000
9777 053707 127 051503 046105
9778 053714 000

```

.ASCIZ <CR>'LF' TYPE A 'CONTROL C' TO ENTER COMMANDS//CR>'LF'<LF>

.EVEN

:PARAMETER ENTRY TABLE

```

PARLST: .WORD PAR1,0,MAXDL
         .WORD PAR2,-1,INTRVL
         .WORD PAR19,-1,PASCNT
         .WORD PAR3,-1,CPLMT
         .WORD PAR11,1,WCSEL
         .WORD PAR14,7,RATIO
         .WORD PAR16,1,ENDET
         .WORD PAR10,1,FORMAT
         .WORD PAR15,1,AUTOCK
         .WORD PAR20,1,NOTPRT
         .WORD 0 ;TABLE TERMINATOR

```

ASKPAR: .ASCIZ /ENTER PARAMETERS: /

SLASH: .ASCIZ @ / @

PAR1: .ASCIZ /MAXDL/

PAR2: .ASCIZ /INTRVL/

PAR3: .ASCIZ /CPLMT/

PAR4: .ASCIZ /MAXCYL/

PAR5: .ASCIZ /MINCYL/

PAR6: .ASCIZ /MAXTRK/

PAR7: .ASCIZ /MINTRK/

PAR8: .ASCIZ /MAXSEC/

PAR9: .ASCIZ /MINSEC/

PAR10: .ASCIZ /FORMAT/

PAR11: .ASCIZ /WCSEL/

9779 053715 122 052101 047511 PAR14: .ASCIZ /RATIO/
 9780 053722 000
 9781 053723 101 052125 041517 PAR15: .ASCIZ /AUTOCK/
 9782 053730 000113
 9783 053732 047105 042504 000124 PAR16: .ASCIZ /ENDET/
 9784 053740 040520 041523 052116 PAR19: .ASCIZ /PASCNT/
 9785 053746 000
 9786 053747 116 052117 051120 PAR20: .ASCIZ /NOTPRY/
 9787 053754 000124

.EVEN

:PARAMETER TABLE POINTERS FOR ADDRESS LIMITS

9793	053756	053776			TABLE:	.WORD	TABLE0		:PARAMETER TABLE FOR DRIVE 0
9794	053760	054044				.WORD	TABLE1		:PARAMETER TABLE FOR DRIVE 1
9795	053762	054112				.WORD	TABLE2		:PARAMETER TABLE FOR DRIVE 2
9796	053764	054160				.WORD	TABLE3		:PARAMETER TABLE FOR DRIVE 3
9797	053766	054226				.WORD	TABLE4		:PARAMETER TABLE FOR DRIVE 4
9798	053770	054274				.WORD	TABLE5		:PARAMETER TABLE FOR DRIVE 5
9799	053772	054342				.WORD	TABLE6		:PARAMETER TABLE FOR DRIVE 6
9800	053774	054410				.WORD	TABLE7		:PARAMETER TABLE FOR DRIVE 7

:PARAMETER TABLE FOR ADDRESS LIMITS

9804	053776	053635	000000	042000	TABLE0:	.WORD	PAR5,0,MINCYL+DRIVE0
9805	054004	053626	000000	041776		.WORD	PAR4,0,MAXCYL+DRIVE0
9806	054012	053653	000022	042004		.WORD	PAR7,18,MINTRK+DRIVE0
9807	054020	053644	000022	042002		.WORD	PAR6,18,MAXTRK+DRIVE0
9808	054026	053671	000025	042010		.WORD	PAR9,21,MINSEC+DRIVE0
9809	054034	053662	000025	042006		.WORD	PAR8,21,MAXSEC+DRIVE0

9812	054044	053635	000000	042304	TABLE1:	.WORD	PAR5,0,MINCYL+DRIVE1
9813	054052	053626	000000	042302		.WORD	PAR4,0,MAXCYL+DRIVE1
9814	054060	053653	000022	042310		.WORD	PAR7,18,MINTRK+DRIVE1
9815	054066	053644	000022	042308		.WORD	PAR6,18,MAXTRK+DRIVE1
9816	054074	053671	000025	042314		.WORD	PAR9,21,MINSEC+DRIVE1
9817	054102	053662	000025	042312		.WORD	PAR8,21,MAXSEC+DRIVE1

9820	054112	053635	000000	042610	TABLE2:	.WORD	PAR5,0,MINCYL+DRIVE2
9821	054120	053626	000000	042606		.WORD	PAR4,0,MAXCYL+DRIVE2
9822	054126	053653	000022	042614		.WORD	PAR7,18,MINTRK+DRIVE2
9823	054134	053644	000022	042612		.WORD	PAR6,18,MAXTRK+DRIVE2
9824	054142	053671	000025	042620		.WORD	PAR9,21,MINSEC+DRIVE2
9825	054150	053662	000025	042616		.WORD	PAR8,21,MAXSEC+DRIVE2

9828	054160	053635	000000	043114	TABLE3:	.WORD	PAR5,0,MINCYL+DRIVE3
9829	054166	053626	000000	043112		.WORD	PAR4,0,MAXCYL+DRIVE3
9830	054174	053653	000022	043120		.WORD	PAR7,18,MINTRK+DRIVE3
9831	054202	053644	000022	043116		.WORD	PAR6,18,MAXTRK+DRIVE3
9832	054210	053671	000025	043124		.WORD	PAR9,21,MINSEC+DRIVE3
9833	054216	053662	000025	043122		.WORD	PAR8,21,MAXSEC+DRIVE3

9834 054224 000000

```

9835
9836 054226 053635 000000 043420 TABLE4: .WORD PAR5,0,MINCYL+DRIVE4
9837 054234 053626 000000 043416 .WORD PAR4,0,MAXCYL+DRIVE4
9838 054242 053653 000022 043424 .WORD PAR7,18.,MINTRK+DRIVE4
9839 054250 053644 000022 043422 .WORD PAR6,18.,MAXTRK+DRIVE4
9840 054256 053671 000025 043430 .WORD PAR9,21.,MINSEC+DRIVE4
9841 054264 053662 000025 043426 .WORD PAR8,21.,MAXSEC+DRIVE4,0
9842 054272 000000
9843
9844 054274 053635 000000 043724 TABLE5: .WORD PAR5,0,MINCYL+DRIVES
9845 054302 053626 000000 043722 .WORD PAR4,0,MAXCYL+DRIVES
9846 054310 053653 000022 043730 .WORD PAR7,18.,MINTRK+DRIVES
9847 054316 053644 000022 043726 .WORD PAR6,18.,MAXTRK+DRIVES
9848 054324 053671 000025 043734 .WORD PAR9,21.,MINSEC+DRIVES
9849 054332 053662 000025 043732 .WORD PAR8,21.,MAXSEC+DRIVES,0
9850 054340 000000
9851
9852 054342 053635 000000 044230 TABLE6: .WORD PAR5,0,MINCYL+DRIVE6
9853 054350 053626 000000 044225 .WORD PAR4,0,MAXCYL+DRIVE6
9854 054356 053653 000022 044234 .WORD PAR7,18.,MINTRK+DRIVE6
9855 054364 053644 000022 044232 .WORD PAR6,18.,MAXTRK+DRIVE6
9856 054372 053671 000025 044240 .WORD PAR9,21.,MINSEC+DRIVE6
9857 054400 053662 000025 044236 .WORD PAR8,21.,MAXSEC+DRIVE6,0
9858 054406 000000
9859
9860 054410 053635 000000 044534 TABLE7: .WORD PAR5,0,MINCYL+DRIVE7
9861 054416 053626 000000 044532 .WORD PAR4,0,MAXCYL+DRIVE7
9862 054424 053653 000022 044540 .WORD PAR7,18.,MINTRK+DRIVE7
9863 054432 053644 000022 044536 .WORD PAR6,18.,MAXTRK+DRIVE7
9864 054440 053671 000025 044544 .WORD PAR9,21.,MINSEC+DRIVE7
9865 054446 053662 000025 044542 .WORD PAR8,21.,MAXSEC+DRIVE7,0
9866 054454 000000
9867
9868
9869 054456 000000 000000 000000 CILDER: .WORD 0,0,0,0 ;HEADER BUFFER FOR 'READHD' ROUTINE
9870 054464 000000
9871
9872 054466
9873
9874
9875
9876
9877
9878
9879
9880
9881
9882
9883 054466 104401 054600
9884 054472 104411
9885 054474 012605
9886 054476 112537 001220
9887 054502 112537 001221
9888 054506 112537 001222
9889 054512 112537 001223
9890 054516 112537 001224

```

```

ENDPGM = . ;LAST LOCATION OF PROG + 2
;*****
;ROUTINE TO GET THE DATE AND THE OPERATOR FROM THE OPERATOR
;CALL:
; JSP PC,OPRDAT
; RETURN
;NOTE: THIS ROUTINE IS ENTERED ONLY AT INITIAL START
OPRDAT: TYPE .ENTDAT ;'ENTER DATE'
ROLIN ;READ THE ENTRY
MOV (SP)+,R5 ;PUT THE ENTRY ADDRESS INTO R5
MOV8 (R5)+,DATE ;STORE THE DATE
MOV8 (R5)+,DATE+1 ;STORE THE DATE
MOV8 (R5)+,DATE+2 ;STORE THE DATE
MOV8 (R5)+,DATE+3 ;STORE THE DATE
MOV8 (R5)+,DATE+4 ;STORE THE DATE

```

9891 054522 112537 001225
 9892 054526 112537 001226
 9893 054532 112537 001227
 9894 054536 104401 054617
 9895 054542 104411
 9896 054544 012605
 9897 054546 112537 001232
 9898 054552 112537 001233
 9899 054556 112537 001234
 9900 054562 112537 001235
 9901 054566 112537 001236
 9902 054572 112537 001237
 9903 054576 000207
 9904
 9905 054600 005015 047105 042524
 9906 054606 020122 040504 042524
 9907 054614 020072 000
 9908 054617 105 052116 051105
 9909 054624 047440 042520 040522
 9910 054632 047524 020122 027111
 9911 054640 027104 020072 000
 9912
 9913
 9914
 9915
 9916
 9917
 9918
 9919
 9920
 9921
 9922 054646 010046
 9923 054650 010146
 9924 054652 013746 000004
 9925 054656 013746 000006
 9926 054662 010600
 9927
 9928 054664 104400
 9929 054666 012637 000006
 9930 054672 012737 054712 000004
 9931 054700 012701 020000
 9932 054704 005711
 9933 054706 005721
 9934 054710 000775
 9935 054712 162701 000002
 9936 054716 010006
 9937 054720 012637 000006
 9938 054724 012637 000004
 9939 054730 010137 054742
 9940 054734 012601
 9941 054736 012600
 9942 054740 000207
 9943 054742 000000
 9944
 9945
 9946

```

MOVW (R5)+,DATE+5 ;STORE THE DATE
MOVW (R5)+,DATE+6 ;STORE THE DATE
MOVW (R5)+,DATE+7 ;STORE THE DATE
TYPE ,ENTID ;'ENTER OPERATOR I.D.'
RDLIN ;READ THE ENTRY
MOV (SP)+,R5 ;ENTRY ADDRESS
MOVW (R5)+,OPERID ;STORE THE I.D.
MOVW (R5)+,OPERID+1 ;STORE THE I.D.
MOVW (R5)+,OPERID+2 ;STORE THE I.D.
MOVW (R5)+,OPERID+3 ;STORE THE I.D.
MOVW (R5)+,OPERID+4 ;STORE THE I.D.
MOVW (R5)+,OPERID+5 ;STORE THE I.D.
RTS PC ;RETURN

ENTDAT: .ASCIZ <CR><LF>/ENTER DATE: /
042524
042524

ENTID: .ASCIZ /ENTER OPERATOR I.D.: /
051105
040522
027111
000

.EVEN

.SBTL ROUTINE TO SIZE MEMORY

;*****
;CALL:
;* JSR PC,$SIZE
;* RETURN
;*$LSTAD WILL CONTAIN THE LAST AVAILABLE MEMORY LOCATION

$SIZE: MOV RO,-(SP) ;SAVE RO ON THE STACK
MOV R1,-(SP) ;SAVE R1 ON THE STACK
MOV @#ERRVEC,-(SP) ;SAVE PRESENT ERROR VECTOR PS & PC
MOV @#ERRVEC+2,-(SP)
MOV SP,RO ;SAVE THE STACK POINTER
;;SET THE ERRVEC PS TO THE PRESENT PS
TRAP ;PUSH OLD PSW AND PC ON STACK
MOV (SP)+,@#ERRVEC+2 ;SAVE THE PSW IN @#ERRVEC+2
MOV #25,@#ERRVEC ;SET FOR TIMEOUT
MOV #2000,R1 ;FIRST ADDRESS
1$: TST (R1) ;TEST THIS ADDRESS
TST (R1)+ ;STEP TO NEXT ADDRESS
BR 1$ ;TRY ANOTHER
2$: SUB #2,R1 ;DROP BACK
MOV RO,SP ;RESTORE THE STACK
MOV (SP)+,@#ERRVEC ;RESTORE ERROR VECTOR
MOV R1,$LSTAD ;LAST ADDRESS
MOV (SP)+,R1 ;RESTORE R1
MOV (SP)+,RO ;RESTORE RO
RTS PC
$LSTAD: .WORD 0 ;CONTAINS THE LAST ADDRESS

.SBTL BUSADR - GET BUS ADDRESS AND VECTOR ADDRESS FOR RH11
;THIS ROUTINE IS USED TO INSURE THE BUS ADDRESS

```

```

9947
9948
9949
9950
9951
9952
9953
9954
9955
9956 054744 177700
9957 054746 000776
9958 054750 000000
9959
9960 054752 005737 001260
9961 054756 001456
9962 054760 005037 001260
9963 054764 012700 001170
9964 054770 104401 055176
9965 054774 012046
9966 054776 104402
9967 055000 104401 052155
9968 055004 104411
9969 055006 012601
9970 055010 013737 054744 054750
9971 055016 004537 055220
9972 055022 055042
9973 055024 055114
9974 055026 054764
9975 055030 055036
9976 055032 054764
9977 055034 055110
9978 055036 010260 177776
9979 055042 104401 055207
9980 055046 012046
9981 055050 104402
9982 055052 104401 052155
9983 055056 104411
9984 055060 012601
9985 055062 013737 054746 054750
9986 055070 004537 055220
9987 055074 055114
9988 055076 055114
9989 055100 055042
9990 055102 055110
9991 055104 055042
9992 055106 055110
9993 055110 010260 177776
9994 055114 013701 000004
9995 055120 012737 055154 000004
9996 055126 005777 124036
9997 055132 010137 000004
9998 055136 012700 001170
9999 055142 012701 033332
00000 055146 012021
00001 055150 012021
00002 055152 000207

```

```

: AS THE RH11 IS SETUP FOR THE PROPER ADDRESS.
: IT WILL ALSO READ THE ADDRESS FROM THE TTY IF
: REQUIRED.
: NOTE: THIS ROUTINE DESTROYS R0-R4
: CALL
:
:
:

```

```

: JSR PC,BUSADR
: RETURN

```

```

HIAD: .WORD 177700
HIVEC: .WORD 776
BOUND: .WORD 0

```

```

BUSADR: TST CHGADR
: BEQ 7$
: CLR CHGADR
1$: MOV #SAPADR,R0
: TYPE MRPCSI
: MOV (R0)+, -(SP)
: TPOC
: TYPE ,LINS
: RDLIN
: MOV (SP)+, R1
: MOV HIAD, BOUND
: JSR R5, CK.NUM
: 3$
: 7$
: 1$
: 2$
: 1$
: 4$
2$: MOV R2, -2(R0)
3$: TYPE MRHVEC
: MOV (R0)+, -(SP)
: TPOC
: TYPE ,LINS
: RDLIN
: MOV (SP)+, R1
: MOV HIVEC, BOUND
: JSR R5, CK.NUM
: 7$
: 7$
: 3$
: 4$
4$: MOV R2, -2(R0)
7$: MOV ERVVEC, R1
: MOV #BS, ERVVEC
: TST JSRPADR
: MOV R1, ERVVEC
: MOV #SAPADR, R0
: MOV #RPADR, R1
: MOV (R0)+, (R1)+
: MOV (R0)+, (R1)+
: RTS PC

```

```

: IMPLT FROM TTY REQUESTED?
: NO--BRANCH
: YES--CLEAR THE REQUEST FLAG
: FIRST ADDRESS
: "RPCSI="
: PRESENT RPCSI ADDRESS
: TYPE IT
: 2 SPACES
: GET THE ENTRY
: ADDRESS OF ASCII TEXT
: SET THE ADDRESS MAX
: CHECK THE NUMBER
: CARRIAGE RETURN ONLY ENTERED
: PERIOD ONLY ENTERED
: ILLEGAL INPUT
: TERMINATED WITH A CARRIAGE RETURN
: TERMINATED WITH A "."
: TERMINATED WITH A ":"
: SAVE NEW RPCSI
: "RHVEC="
: PRESENT RH11 VECTOR ADDRESS ON THE STACK
: TYPE IT
: 2 SPACES
: READ THE ENTRY
: ASCII TEXT ADDRESS
: SET THE VECTOR-MAX
: CHECK THE NUMBER
: CARRIAGE RETURN ONLY ENTERED
: PERIOD ONLY ENTERED
: ILLEGAL INPUT
: TERMINATED WITH A CARRIAGE RETURN
: TERMINATED WITH A "."
: TERMINATED WITH A ":"
: SAVE INPUT
: SAVE THE ERROR VECTOR
: SETUP FOR TRAP
: CHECK FOR RH11
: RESTORE ERROR VECTOR
: FIRST ADDRESS OF NEW PARAMETERS
: FIRST ADDRESS OF WHERE TO PUT THEM
: BUS ADDRESS
: VECTOR ADDRESS
: RETURN

```



```

10003 055154 010137 000004
10004 055160 022626
10005 055162 104006
10006 055164 005737 000042
10007 055170 001675
10008 055172 000137 005344
10009
10010 055176 050122 051503 020061 MRFC51: .ASCIZ BRFC51 = 0
10011 055204 020075 000
10012 055207 122 053110 041505 MRHVEC: .ASCIZ BRHVEC = 0
10013 055214 036440 000040
10014
10015
10016
10017
10018
10019
10020
10021
10022
10023
10024
10025
10026
10027
10028
10029 055220 010446
10030 055222 010346
10031 055224 010246
10032 055226 005004
10033 055230 005003
10034 055232 005002
10035 055234 004537 027560
10036 055240 055340
10037 055242 055344
10038 055244 055340
10039 055246 055342
10040 055250 055254
10041 055252 055340
10042 055254 062705 000004
10043 055260 006303
10044 055262 103426
10045 055264 006303
10046 055266 103424
10047 055270 006303
10048 055272 103422
10049 055274 060203
10050 055276 004537 027560
10051 055302 055344
10052 055304 055326
10053 055306 055324
10054 055310 055316
10055 055312 055260
10056 055314 055344
10057 055316 105711
10058 055320 001011

```

```

83: MOV R1,ERRVEC ;RESTORE ERROR VECTOR
    CMP (SP)+,(SP)+ ;CLEAN OFF THE STACK
    ERROR 6 ;REPORT THE ERROR
    TST 2842 ;IS THERE A MONITOR?
    BEQ 18 ;NO--GO ASK FOR ADDRESS
    JMP $GET42 ;GO TO END OF PROGRAM

020061 MRFC51: .ASCIZ BRFC51 = 0
041505 MRHVEC: .ASCIZ BRHVEC = 0

.SBTL CK.NUM - CHECK NUMBER (OCTAL)
;THIS ROUTINE CHECKS AN ASCIZ STRING FOR LEGAL CHARACTERS
;AND FORMS AN OCTAL NUMBER IN R2
CALL:
    MOV #ADR,R1 ;ADDRESS OF ASCIZ STRING
    MOV #NUM,R2 ;MAX SIZE OF INPUT NUMBER
    JSR R5,CK.NUM ;GO FORM THE NUMBER
    RETURN ADR1 ;"CR" ONLY ENTERED -- R2 = C
    RETURN ADR2 ;"PERIOD" ONLY ENTERED -- R2 = 0
    RETURN ADR3 ;ILLEGAL CHARACTER IN THE INPUT STRING
    RETURN ADR4 ;"CR" ENTERED -- R2 = NUMBER
    RETURN ADR5 ;"COMMA" -- R2 = NUMBER
    RETURN ADR6 ;"PERIOD" -- R2 = NUMBER

CK.NUM: MOV R4,-(SP) ;SAVE R4
        MOV R3,-(SP) ;SAVE R3
        MOV R2,-(SP) ;SAVE R2
        CLR R4 ;RETURN POINTER
        CLR R3 ;START NUMBER AT ZERO
        CLR R2 ;STORE RESULT
        JSR R5,CK.CHR ;CHECK ONE CHARACTER
        B3 ;ILLEGAL CHARACTER
        B3 ;CARRIAGE RETURN
        ;
        ;
        B3 ;DIGIT 0-7
        B3 ;DIGIT 8-9
        B3 ;INCREMENT RETURN PAST "CR" AND "PERIOD" ONLY RETURNS
        B3 ;FOR THE OCTAL NUMBER IN R3
        B3 ;DON'T LET IT GET TO BIG

        JSR R5,CK.CHR ;CHECK ONE CHARACTER
        B3 ;ILLEGAL CHARACTER
        B3 ;CARRIAGE RETURN
        ;
        ;
        B3 ;DIGIT 0-7
        B3 ;DIGIT 8-9
        B3 ;DOES A "CR" FOLLOW THE "PERIOD"
        B3 ;OR IF NOT

```

```

10059 055322 005724
10060 055324 005724
10061 055326 005724
10062 055330 023703 054750
10063 055334 101003
10064 055336 000401
10065 055340 005725
10066 055342 005725
10067 055344 060405
10068 055346 010302
10069 055350 005726
10070 055352 012603
10071 055354 012604
10072 055356 011505
10073 055360 000205
10074
10075
10076 055362 005015 020040 020040
10077 055370 020040 020040 020040
10078 055376 055132 041455 051132
10079 055404 042112 041455 005015
10080 055412 050122 032060 032457
10081 055420 033057 046440 046125
10082 055426 044524 042055 044522
10083 055434 042526 042440 042530
10084 055442 041522 051511 051105
10085 055450 050040 047522 051107
10086 055456 046501 005015 000012
10087 055464 005015 047524 052040
10088 055472 051505 020124 051104
10089 055500 053111 020105 026060
10090 055506 051040 050105 040514
10091 055514 042503 052040 042510
10092 055522 023440 054130 050104
10093 055530 020047 040520 045503
10094 055536 047440 020116 051104
10095 055544 053111 020105 006460
10096 055552 012
10097 055553 127 052111 020110
10098 055560 047101 052117 042510
10099 055566 020122 040520 045503
10100 055574 020054 046103 040505
10101 055602 020122 042515 047515
10102 055610 054522 046040 041517
10103 055616 052101 047511 020116
10104 055624 030064 006454 012
10105 055631 101 042116 051040
10106 055636 051505 040524 052122
10107 055644 052040 042510 050040
10108 055652 047522 051107 046501
10109 055660 005015 000
10110 055663 015 051412 051531
10111 055670 042524 020115 040510
10112 055676 020123 035061 020113
10113 055704 042515 047516 054522
10114 055712 020054 054047 042120

```

```

4$: TST (R4)+ ; INCREMENT THE RETURN INDEX
5$: TST (R4)+ ; INCREMENT THE RETURN INDEX
TST (R4)+ ; INCREMENT THE RETURN INDEX
CMP BOUND,R3 ; INPUT VALUE TOO LARGE?
BHI B$ ; BR IF IT IS
BR 7$ ; BR IF NOT
6$: TST (R5)+ ; INCREMENT THE RETURN ADDRESS
7$: TST (R5)+ ; INCREMENT THE RETURN ADDRESS
8$: ADD R4,R5 ; SETUP FOR PROPER RETURN
MOV R3,R2 ; LOAD ENTERED VALUE
TST (SP)+ ; CLEAN OFF THE STACK
MOV (SP)+,R3 ; RESTORE R3
MOV (SP)+,R4 ; RESTORE R4
MOV (R5),R5 ; GET RETURN ADDRESS
RTS R5 ; RETURN

```

```

TITLE: .ASCII <CR><LF>/ ZZ-CZRJD-C<CR><LF>
.ASCIIZ DRPO4/5/6 MULTI-DRIVE EXERCISER PROGRAM<CR><LF><LF>
LOADRV: .ASCII <CR><LF>/TO TEST DRIVE D, REPLACE THE 'XXDP' PACK ON DRIVE D<CR><LF>
.ASCII /WITH ANOTHER PACK, CLEAR MEMORY LOCATION 40,<CR><LF>
.ASCIIZ /AND RESTART THE PROGRAM.<CR><LF>
NOLOAD: .ASCIIZ <CR><LF>/SYSTEM HAS 16K MEMORY, 'XXDP' LOADER WILL BE OVERWRITTEN<CR><L

```

E15

CZRJDCO RPO4.5 6 MLT-DR LGC
CZRJDC.P11 15-DEC-77 10:58

MACY11 30(1046) 15-DEC-77 11:03 PAGE 187
CK.NUM - CHECK NUMBER (OCTAL)

SEG 0106

10115	055720	023520	046040	040517
10116	055726	042504	020122	044527
10117	055734	046114	041040	020105
10118	055742	053117	051105	051127
10119	055750	052111	042524	006516
10120	055756	000012		
10121				
10122		000001		

.END

ABNRM 026524
 ABS = 000200
 ACK = 000123
 ACL = 000040
 ACTDRV 033250
 ACTSTR 033251
 ACU = 100000
 ACE = 001000
 ASGNO 052720
 ASGN1 024214
 ASGN2 024274
 ASGN3 024346
 ASGN4 024464
 ASGN5 024474
 ASGN6 024502
 ASGN7 024512
 ASKPAR 053552
 ASNEAR 026416
 ASNLST 001462
 ASMSG 026436
 ASSIGN 024206
 ATP = 100000
 ATBIT 033320
 ATTN 001244
 AT0 = 000001
 AT1 = 000002
 AT2 = 000004
 AT3 = 000010
 AT4 = 000020
 AT5 = 000040
 AT6 = 000100
 AT7 = 000200
 ALTCK 001424
 AVAIL 001530
 A16 = 000400
 A17 = 001000
 BADENT 053276
 BADSEC 001264
 BPI = 000010
 BEGCOO 001434
 BEGPAT 001432
 BEGSIZ 001436
 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 001740
 BOUND 054750
 BPTVEC = 000014
 BUFTBL 001616
 BUSADR 054752
 BUSY 053320
 CFLAG 001262
 CHGADR 001260
 C11 034672
 C13 035000
 C14 035106
 C15 035464
 C16 035506
 C17 035522
 C17B 035550
 C18 035562
 CKBUS 013070
 CKCLK 022412
 CKCLK1 022506
 CKCLK2 022554
 CKCLK3 022604
 CKERR 012770
 CKFMT 011072
 CKHCE 011266
 CKSWR = 104407
 CKCHR 027560
 CKDEC 027532
 CKDIG 027632
 CKNUM 055220
 CKOC 027504
 CLKFLG 001210
 CLOCK 023546
 CLR = 000040
 CLROPB 025164
 CLRGUE 041432
 CMCNT 001312
 CMCYL 001314
 CMDAT 013436

CMHED 013342
 CMPAR 013154
 CMPAR0 013172
 CMPLMT 001414
 CMPRES 017530
 CMPRT 013702
 CMPRX 013674
 CMSEC 001316
 CMSTR 013264
 CMTRK 001317
 COLON 053207
 COMTBL 001760
 CR = 000015
 CRLF = 000200
 CSF = 000002
 CSU = 000010
 CYLDER 054456
 CYLMT 001350
 DATAPK 025140
 DATA 003026
 DATAI 003066
 DATE 001220
 DATEIS 053211
 DCK = 100000
 DCKER = 007570
 DCKER1 007722
 DCL = 000100
 DCU = 000001
 DDISP = 177570
 DEASGN 024562
 DEASSG 052650
 DE1 = 000040
 DFF20 = 000002
 DH1 047460
 DH14 047636
 DH15 047742
 DH16 050041
 DH2 047465
 DH3 047542
 DH4 047570
 DH6 047627
 DIGB = 000004
 DISPLA 001142
 DISPLY = 104414
 DISPRE = 000174
 DLT = 100000
 DL64 = 000020
 DMO = 000001
 DONE 007264
 DPINT 033224
 DPR = 000400
 DPRGS 033234
 DRIVE 001242

DRIVE0 041670
 DRIVE1 042174
 DRIVE2 042500
 DRIVE3 043004
 DRIVE4 043310
 DRIVE5 043614
 DRIVE6 044120
 DRIVE7 044424
 DRNUM 052673
 DROP 026446
 DROPNG 052562
 DRQ = 004000
 DRVACT 033174
 DRVCLA = 000111
 DRIVER 011040
 DRVINT 033562
 DRVPRM 025360
 DRVQUE 041530
 DRVS'A 033204
 DRV'TP 033214
 DRY = 000200
 DSWR = 177570
 DTE = 010000
 DTEER 011676
 DTSY = 000200
 DTUW 033316
 DT00 = 000001
 DT01 = 000002
 DT02 = 000004
 DT03 = 000010
 DT04 = 000020
 DT05 = 000040
 DT06 = 000100
 DT07 = 000200
 DT08 = 000400
 DT1 050112
 DT14 050162
 DT15 050204
 DT16 050226
 DT2 050116
 DT3 050134
 DT4 050144
 DT6 050156
 DUNIT 001464
 DVA = 004000
 ECBAD0 001334
 ECBAD1 001342
 ECBIT 001320
 ECC 014302
 ECCX 015034
 ECC1 014700
 ECC2 015030
 ECC3 001332

ECG01 001340
 ECH = 000100
 ECI = 004000
 ECMASK0 = 001324
 ECMASK1 001326
 ECSEC 001322
 ECWR0 001330
 ECWR01 001336
 EMPTYQ 041510
 EMTVEC = 000030
 EM1 045020
 EM10 045311
 EM11 045355
 EM12 045410
 EM13 045441
 EM14 045513
 EM15 045536
 EM2 045506
 EM20 045513
 EM21 045513
 EM22 045614
 EM23 045717
 EM24 045776
 EM25 046044
 EM26 046125
 EM27 046152
 EM3 045121
 EM30 046207
 EM31 046241
 EM32 046304
 EM33 046337
 EM34 046414
 EM35 046456
 EM36 046514
 EM37 046545
 EM4 045157
 EM40 046621
 EM41 046666
 EM42 046731
 EM43 047012
 EM44 047057
 EM45 047143
 EM46 047231
 EM5 045214
 EM50 047303
 EM51 047372
 EM6 045250
 EM60 047435
 ENCCMP 014156
 ENDCN 001352
 ENDCON 001372
 ENDET 001430
 ENDPAS 052615

ENDPGM= 054466
ENDSEK 001376
ENDSK 001356
ENDST 052632
ENTADP 053146
ENTCOM 053033
ENTDAT 054600
ENTDRV 053060
ENTIO 054617
ENTMT 053105
ENTPR 004772
ENTPRR 026552
ENTPRX 027024
ENTPR1 026604
ENTPR2 026626
ENTPRR 001306
ENTPRC1 006644
ENTPROC 006630
ENTRR = 040000
ENTRVEC = 000004
ENTXT1 000001
ENTXT10 000010
ENTXT2 000002
ENTXT20 000020
ENTXT4 000004
ENTXT40 000040
ENTACTOR 001567
ENTAIRNS 001254
ENTALPAR 007030
ENTALPRI 007040
ENTEN = 000200
ENTER = 000020
ENTFILBUF 012624
ENTITER 012072
ENTT22 = 010000
ENTFORMAT 001416
ENTRSTER 001300
ENTT1 = 000002
ENTT2 = 000004
ENTT3 = 000010
ENTT4 = 000020
ENTT5 = 000040
ENTNOPB 044730
ENTNREG 044750
ENTADP 025752
ENTBUF 015672
ENTIO 025642
ENTPAR 017322
ENTPAT 017274
ENTREG = 000141
ENTREM 027026
ENTREG = 041604
ENT = 000001

GODRIV 016370
GRV = 000010
GTSWR = 104406
HCE = 000200
HCEER 012150
HCI = 002000
HCRC = 000400
HCRCER 010710
HEDLIN 053244
HIAD 054744
HIVEC 054746
HOUR = 001266
HT = 000011
HZ = 001212
IAE = 002000
IAFER 012012
IDIS 053222
IDLE 006202
IOLE1 006300
IE = 000100
ILF = 000001
ILR = 000002
INCHRO 023330
INCMIS 023400
INCSKI 023354
INCSOF 023304
INCTOT 023424
INTDOM 053341
INTRVL 001410
INVLD 053011
IOTVEC = 000020
IR = 000100
ISR 036214
IXE = 004000
KSP 023710
KSRI 023744
LA 036056
LACNT 033262
LF = 000012
LIMIT 001310
LINA3 050460
LINB3 050532
LINB5 050707
LINB6 051022
LINCAB 050511
LINC6 051055
LINCAB 050522
LINOEC 022274
LINOS 050672
LINEN3 050443
LINE05 051010
LINEP5 050777
LINE1 020200

LINE2 020244
LINE2A 020414
LINE2B 020432
LINE3 020652
LINE3A 020660
LINE3B 020666
LINE3C 020700
LINE3D 020710
LINE3E 020756
LINE3F 021044
LINE4 021326
LINE5 021416
LINE5A 021516
LINE5B 021610
LINE6 021652
LINE6A 021664
LINE6B 021672
LINE6C 021700
LINE6D 021706
LINE7 021762
LINE7A 022110
LINE8 022230
LINE66 051104
LINKOV 027054
LINM3 050343
LINM4 050606
LINM3 050361
LINOCT 022242
LINP3 050405
LINP5 050725
LINR6 051122
LINSP 052155
LINSPO 052156
LINS53 050571
LINS3 050554
LINS3 050426
LINS4 050625
LINS5 050743
LINT3 050500
LINU06 051133
LINW3 050542
LINX4 050637
LIN10A 051571
LIN10B 051625
LIN10C 051666
LIN10H 051773
LIN11H 052060
LIN2C 050242
LIN2P 050263
LIN2S 050307
LIN3.1 021116
LIN3.3 021230
LIN3.4 021262

LIN45P 052153
LIN6.1 021714
LIN6.2 021736
LIN7M 051160
LIN7O 051206
LIN7P 051216
LIN7R 051313
LIN7S 051236
LIN7T 051265
LIN7X 051277
LIN8M 051330
LIN9B 051365
LIN9E 051512
LIN9G 051542
LIN9H 051414
LIN9I 051472
LKPAR 004746
LOADRV 055464
LST = 002000
LSTAD 001256
MAIN 005374
MAIN1 005522
MAIN2 005642
MAIN3 006076
MASK 001250
MATCH 014224
MAXCYL = 000106
MAXDL 001404
MAXER 001406
MAXSEC = 000116
MAXTRK = 000112
MCLK = 000002
MCPE = 020000
MCPEMX 033330
MHS = 001000
MINCYL = 000110
MINSEC = 000120
MINTRK = 000114
MINUTE 001270
MINX = 000004
MNDLTA 033344
MNTBL 002010
MOH = 020000
MOL = 010000
MONTR 005266
MPE = 000400
MRD = 000020
MRHVEC 055207
MRPCS1 055176
MSE = 000020
MSTCK = 000010
MWR = 000040
MXDLTA 033342

MXF = 001000
MXLACT 033340
MXWNOH 033346
M.DPIO 027122
M.DP40 027160
M.DP41 027214
M.DP42 027224
M.DP44 027256
M.DP50 027270
NBA = 100000
NED 010000
NEDCLK 052733
NEM = 004000
NEWASN 024552
NEWUNT 001506
NHS = 002000
NOLORD 055663
NOMTCH 012546
NONE 053267
NOTAVL 052315
NOTPRS 052300
NOTPRT 001426
NOTRP 052257
NOTSAF 052334
OCYL = 100000
OFFCOD 002220
OFFSET = 000115
OFFST 015272
OFLIN 007136
OFMSG0 002724
OFMSG0 002274
OFMSG1 002327
OFMSG2 002363
OFMSG3 002417
OFMSG4 002453
OFMSG5 002507
OFMSG6 002543
OFMSG7 002577
OFMSG8 002633
OFMSG9 002667
OFMTBL 002240
OFRE = 000200
OF100 = 000004
OF200 = 000010
OF25 = 000001
OF400 = 000020
OF50 = 000002
OF800 = 000040
OPE = 020000
OPEPIO 001332
OPT = 020000
OPTER = 011566
OPTER = 011632

JPRDAT 054466
JPT 034412
OPTBL 001766
OR = 000200
ORDERO 001440
PACK 001216
PAR = 000010
PARENT 026270
PARENT 011720
PARLIST 053475
PARO 001574
PAR1 053360
PAR10 053370
PAR11 053370
PAR15 053371
PAR16 053372
PAR19 053374
PAR2 053361
PAR20 053374
PAR3 053361
PAR4 053362
PAR5 053363
PAR6 053364
PAR7 053365
PAR8 053366
PAR9 053367
PARSCNT 001400
PARSCNT = 000020
PARLOCK 001200
PARLOCK = 052555
PARM100 053300
PARM100 = 002000
PARM100 = 001000
PARM100 = 020000
PARM100 = 177777
PARM100 = 000200
PARM100 = 041620
PARSER 011511
PROCES 006544
PROCES = 015044
PROCES = 007220
PROCES = 000000
PROCES = 000040
PROCES = 000100
PROCES = 000140
PROCES = 000200
PROCES = 000300
PROCES = 000340
PROCES = 177777
PROCES = 002000

PSU = 000001
PSW = 177776
PUNSAF 006776
PWAVEC = 000024
QCNT 041140
QDRVO 041232
QDRV1 041252
QDRV2 041272
QDRV3 041312
QDRV4 041332
QDRV5 041352
QDRV6 041372
QDRV7 041412
QINPT 041150
QOUTPT 041170
QSTART 041210
QSTOP 041212
QTERM = 041432
QUES 053007
QVCON 001362
QVSEK 001366
RANCYL 016702
RANPAT 017150
RANSEC 016526
RANSTZ 017014
RANTRK 016602
RANWRT 017170
RANXIT 017160
RATIO 001422
RAW = 000020
ROCHR 104410
RODAT 000171
ROHD 000173
ROLIN 104411
ROBY = 000200
RO.AOR 040432
RO.RP 040406
RO.RP1 040430
RO.RP2 040550
RO.RP3 040554
RO.RP4 040560
RO.WRD 040434
READDR 022314
READHD 015316
READIN = 000121
RECAL = 000107
RECALT 015240
RECALO 015244
REDAFK 025152
REFMT 006374
REFMTX 006536
RELBUF 016026
RELSE = 000113

REPLZ 027300
RESREG = 104412
RESVEFC = 000010
RETRY 001252
RMR = 000004
RNOF = 000101
RPAOR 033332
RPAS 000016
RPBA 000004
RPCA = 000034
RPCC = 000036
RPCS1 000000
RPCS2 000010
RPCA 000006
RPOB 000026
RPOS1 000012
RPO2 000026
RPEC1 000044
RPEC2 000046
RPERAS 033164
RPER1 000014
RPER2 000040
RPER3 000042
RPINIT 033350
RPLA 000020
RPMR 000024
RPOF = 000032
RPSN 000030
RPMR 037660
RPVEC 033334
RPWC = 000002
RPO4 034120
RPO4B 052364
RPO5 052371
RPO6 052376
RTC = 000117
RTNCTR 015214
R6 = 000006
R7 = 000007
S 050402
SAVEFG 033272
SAVER1 001302
SAVERS 001304
SAVREG = 104412
SC 036542
SCMND 024670
SC1 = 000100
SC10 = 001000
SC11 037360
SC12 037450
SC13 037520
SC2 = 000200
SC3 = 002000

SC3 036606
SC4 036612
SC5 036624
SC6 037014
SC6A 037124
SC7 037252
SC8 037330
SCETAL 022746
SEARCH = 000131
SECLMT 001344
SEFCO 001272
SEEK = 000105
SEEKFG = 033274
SELOPV = 000145
SELPAR 016446
SETFMT = 000143
SETUP 004154
SETVEC 005020
SET. IFC 041066
SHOTYP 022724
SIXTEE 001274
SIZMEM 004576
SKI = 040000
SKIER 012332
SLASH 053576
SPOTCK 020042
SRCHMT 033246
STACK = 001100
START 004106
START1 004116
START2 004122
STATHO 052403
STATIN 001214
STATIS 015526
STATPR 022614
STKLMT = 177774
STNDAT 002762
STO 037752
STO1 040002
STO2 040200
STO3 040250
STO5 040274
STO6 040302
STO7 040340
STO8 040370
STO9 040400
SVRH11 040750
SWR 001140
SWREG 000176
SWT.M 007100
SWO = 000001
SWO0 = 000001
SWC: = 000002

SWO2 = 000004
SWO3 = 000010
SWO4 = 000020
SWO5 = 000040
SWO6 = 000100
SWO7 = 000200
SWO8 = 000400
SWO9 = 001000
SW1 = 000002
SW10 = 002000
SW11 = 004000
SW12 = 010000
SW13 = 020000
SW14 = 040000
SW15 = 100000
SW2 = 000004
SW3 = 000010
SW4 = 000020
SW5 = 000040
SW6 = 000100
SW7 = 000200
SW8 = 000400
SW9 = 001000
S. STAT 052344
T 050356
TABLE 053756
TABLE0 053776
TABLE1 054044
TABLE2 054112
TABLE3 054160
TABLE4 054226
TABLE5 054274
TABLE6 054342
TABLE7 054410
TAP = 040000
TBITVE = 000014
TD 036260
TDF = 000040
TIMER 033276
TITLE 055362
TKVEC = 000060
TPVEC = 000064
TRAPVE = 000034
TRF = 040000
TRFER 012232
TRK.LMT 001346
TRK1 = 004000
TRK10 = 040000
TRK2 = 010000
TRK3 = 100000
TRK4 = 020000
TRNSWT 033244
TR.VEC = 000014

WJF = 000100	WATPK1 017576	\$ERRPC 001116	\$SPACK = 000126	\$SIZE = 054646
WYPOS = 104405	WATPK2 017756	\$ERRTB 004026	\$SPASS = 001100	\$SKI = 000064
WYPE = 104401	WATPK3 020002	\$ERRTY 030700	\$SPASSC = 000070	\$SOFT = 000060
WYPEST = 022676	WATPK4 020034	\$ERTTL 001112	\$PATTC = 000030	\$SSEC = 000022
WYPOC = 104402	WAT.A0 040652	\$FAIR = 000072	\$POSIT = 000042	\$STUP = 177777
WYPON = 104404	WAT.RP 040562	\$FILLC = 001156	\$PREVA = 000032	\$SUPRS = 027370
WYPOS = 104403	WAT.R1 040646	\$FILLS = 001156	\$PREVC = 000027	\$SVPC = 000210
WYPRI4 027430	WAT.R2 040732	\$FIRST = 000122	\$PSEL = 000003	\$SWR = 122000
WCPAP 007016	WAT.R3 040740	\$FMT = 000001	\$QUES = 001164	\$STATUS = 000016
WJDFLG 033252	WAT.R4 040744	\$GDADR 001120	\$RAND = 032364	\$TERM = 000032
WJIT = 001246	WAT.R5 040746	\$GDOAT 001124	\$RCHR = 032210	\$TIME = 023450
WJLORD = 000103	WAT.WD 040650	\$GET42 005344	\$RDIN = 030232	\$TYB = 001146
WJNS = 040000	WRS = 000400	\$GTSWR 031776	\$RSZ = 000001	\$TKINT = 030050
WJNSAF 012466	WRSU = 000004	\$HARD = 000062	\$READ = 000052	\$TKS = 001144
WJYASN 052231	ZROIND 001276	\$MC = 000000	\$REG = 000014	\$TKSRV = 030100
WJYMSG 052160	\$AUTOB 001134	\$HINUM 032452	\$RESRE 032524	\$TY = 000001
WJYNOT 052207	\$BODDR 001122	\$ICNT 001104	\$RETRY 015400	\$TNPWR = 032672
WJYOFF 052156	\$BDDAT 001126	\$INTAG 001135	\$RPADR 001170	\$TOTAL = 000056
WJYON 052177	\$BDOSEC = 000124	\$ITEMB 001114	\$RPDS = 000252	\$TPB = 001152
WJY = 020000	\$BELL 001160	\$LF = 001166	\$RPBA = 000240	\$TPFLG = 001157
WJY1 = 000001	\$BUF = 000006	\$LKCSB 001176	\$RPCA = 000270	\$TPS = 001150
WJY2 = 000002	\$CHARC 031250	\$LKCSR 001174	\$RPCC = 000272	\$TRANS = 000046
WJY3 = 000004	\$CKSWR 031726	\$LKS = 001202	\$RPCS1 = 000234	\$TRAP = 033076
WJY4 = 000010	\$CMTAG 001100	\$LLVEC 001204	\$RPCS2 = 000244	\$TRAP2 = 033120
WJY5 = 000002	\$CM3 = 000000	\$LONUM 032464	\$RPDA = 000242	\$TRK = 000011
WJY6 = 010000	\$CNTLC 030522	\$LPAOR 001106	\$RPDB = 000256	\$TRP = 000015
WJY7 = 000100	\$CNTLG 032335	\$LPERR 001110	\$RPDS1 = 000246	\$TRPAD = 033132
WJY8 = 001552	\$CNTLU 032330	\$LPVEC 001200	\$RPDT = 000262	\$TSTNM = 001102
WJY9 = 000002	\$CODE = 000024	\$LSTAD 054742	\$RPEC1 = 000300	\$TYIN = 030510
WJY10 = 036434	\$COMND = 000002	\$MISPO = 000066	\$RPEC2 = 000302	\$TYPOS = 031502
WJY11 = 040000	\$CRLF = 001165	\$MNEW 032353	\$RPER1 = 000250	\$TYPE = 031034
WJY12 = 000040	\$CYL = 000012	\$MSWR 032342	\$RPER2 = 000274	\$TYPEC = 031204
WJY13 = 012370	\$DBLK 031716	\$NCODE = 000074	\$RPER3 = 000276	\$TYPEX = 031252
WJY14 = 000151	\$DB2D 032562	\$NCYL = 000100	\$RPLA = 000254	\$TYPOC = 031300
WJY15 = 010362	\$DB2D 032756	\$NEXT = 000104	\$RPMR = 000260	\$TYPON = 031314
WJY16 = 000153	\$DECVL 032742	\$NPATC = 000075	\$RPOF = 000266	\$TYPOS = 031254
WJY17 = 001420	\$DORGX 005370	\$NSEC = 000076	\$RPSN = 000264	\$WRDL = 000020
WJY18 = 000001	\$DRVID = 000224	\$NTRK = 000077	\$RPVEC = 001172	\$WRDM = 000004
WJY19 = 004000	\$DSPLY 027456	\$NULL 001154	\$RPWC = 000236	\$OFILL = 031477
WJY20 = 012044	\$DTBL 031706	\$NWRDL = 000102	\$SAVRE 032466	= 055760
WJY21 = 004000	\$ENDAD 005360	\$OCNT 031476	\$SB2D 027770	
WJY22 = 000161	\$ERFLG 001103	\$OCTVL 033060	\$SB2D 030020	
WJY23 = 000163	\$ERMAX 001115	\$OMCDE 031500	\$SEC = 000010	
WJY24 = 017546	\$ERROR 030530	\$OPERC = 000036	\$SETUP = 000146	

. ABS. 055760 CJC

ERRORS DETECTED: 0

RM03: CZRJDC, CZRJDC, SEQ/SOL/NL: MC: MD: CND=RPO456.011, RM03: CZRJDC.P11
RUN-TIME: 30 26 1 SECONDS
RUN-TIME PATIC: 577/57=9.9
CORE USED: 407 179 PAGES.

J15