

RP07

PERF EXER
CZRJOAO

AH-F965A-MC
FICHE 1 OF 2

MAY 1983
COPYRIGHT © 1983
MADE IN USA



The main body of the document is a large grid of approximately 15 columns and 25 rows of data. Each cell in the grid contains a small, dense table or set of data points, likely representing performance metrics or exercise results. The text is very small and difficult to read, but the overall structure is a regular grid of data blocks.

RP07

PERF EXER
CZRJOAO

AH-F965A-MC
FICHE 2 OF 2

MAY 1983
COPYRIGHT © 1983
MADE IN USA



.REM @

IDENTIFICATION

PRODUCT CODE: AC-F964A-MC
PRODUCT NAME: CZRJOAO RP07 PERF EXER
PRODUCT DATE: JANUARY 1, 1983
MAINTAINER: CX DIAGNOSTIC ENGINEERING
AUTHOR: MIKE LEAVITT

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

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

COPYRIGHT (C) 1983 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL
DEC

PDP
DECUS

UNIBUS
DECTAPE

MASSBUS

@

.REM @

CONTENTS

1. ABSTRACT
 - 1.1 GENERAL DOCUMENT NOTES
2. REQUIREMENTS
 - 2.1 EQUIPMENT
 - 2.2 MEDIA
 - 2.3 PRELIMINARY PROGRAMS
3. OPERATING PROCEDURE
 - 3.1 LOADING THE PROGRAM
 - 3.2 STARTING ADDRESSES
 - 3.3 PROGRAM CONTROL
 - 3.4 SWITCH OPTIONS
 - 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 DUAL PORT OPERATION
 - 3.8 XXDP, ACT11, APT11
 - 3.9 APT ENVIRONMENTAL TABLE DEFINITIONS
4. CONTROLLING THE PROGRAM
 - 4.1 PARAMETERS
 - 4.1.1 PROGRAM CONTROL PARAMETERS
 - 4.1.2 CHANGE DEVICE ADDRESS
 - 4.2 KEYBOARD COMMANDS
 - 4.2.1 'T' COMMAND
 - 4.2.2 'D' COMMAND
 - 4.2.3 'S' COMMAND
 - 4.2.4 'W' COMMAND
 - 4.2.5 'R' COMMAND
 - 4.2.6 'WT' COMMAND
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 BAD ADDRESS 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 SELECTION OF OPERATION VARIABLES
- 8.4 DATA PATTERNS

9. RP SOFTWARE DRIVER DOCUMENT

1. ABSTRACT

THE RP07 PERFORMANCE EXERCISER PROGRAM IS DESIGNED TO PERFORM AN INTERACTIVE TEST ON RP DISK DRIVES CONNECTED TO A MASSBUS SUBSYSTEM. THE DRIVES MAY BE CONTROLLED BY AN RH70 CONTROLLER. 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 (SEE ERROR RATE SPECIFICATIONS).

THE PERFORMANCE 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 SINGLE PORT DRIVES.

TO OBTAIN INTERACTIVE TESTING, OPERATIONS ON THE MULTI-DRIVE CONFIGURATIONS ARE OVERLAPPED (OTHER DRIVES ARE PERFORMING SEEK/SEARCH OPERATIONS WHILE ONE DRIVE IS PERFORMING A DATA TRANSFER). 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 EXCEPT WRITE HEADER & DATA AND WRITE CHECK HEADER & DATA ARE USED. RECALIBRATE AND READ-IN PRESET COMMANDS ARE USED AT STARTUP AND DRIVE INITIALIZATION. RECALIBRATE, OFFSET, AND RETURN-TO-CENTERLINE COMMANDS ARE USED DURING ERROR RECOVERY.

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.

DEPENDING UPON WHETHER THE PROGRAM HAS BEEN LOADED VIA APT AUTOMATIC MODE OR APT DUMP MODE WILL DETERMINE WHETHER; PROGRAM/OPERATOR COMMUNICATIONS ARE THROUGH THE KEYBOARD, DYNAMIC PROGRAM OPTIONS ARE SELECTED VIA SWITCH REGISTER SETTINGS AND ERRORS ARE REPORTED ON THE CONSOLE TERMINAL.

1.1 GENERAL DOCUMENT NOTES

A. IN REFERENCE TO ALL NUMBERS IN THIS DOCUMENTATION, TO INDICATE THE BASE OF A NUMBER LARGER THAN SEVEN, A PERIOD(.) WILL FOLLOW THE NUMBER TO INDICATE DECIMAL OR NO PERIOD WILL FOLLOW THE NUMBER TO INDICATE OCTAL. IF THE NUMBER OCCURS AT THE END OF A SENTENCE, A DOUBLE PERIOD(. .) INDICATES DECIMAL AND A SINGLE PERIOD(.) INDICATES OCTAL. ALSO, ANY REFERENCES TO TIME ARE ALWAYS IN DECIMAL.

2. REQUIREMENTS -----

2.1 EQUIPMENT

PDP-11 PROCESSOR
20K MEMORY
KW11-L OR KW11-P CLOCK
PROGRAM LOADING DEVICE
TERMINAL
RH11 OR RH70 CONTROLLER
1 TO 8 DISK DRIVES (RP07'S)

2.2 MEDIA

THE PERFORMANCE EXERCISER PROGRAM REQUIRES FORMATTED DISK
PACKS GENERATED BY THE RP07 FORMATTER PROGRAM (ISSFMT).
THE PACKS MUST BE FORMATTED IN 32. SECTOR (16 BIT) MODE; THE
ALTERNATE (30. SECTOR - 18 BIT) MODE IS NOT SUPPORTED.

2.3 PRELIMINARY PROGRAMS

RP07 FRONT-END TEST
RP07 FUNCTIONAL TEST

3. OPERATING PROCEDURE -----

3.1 LOADING THE PROGRAM

THE PROGRAM MAY BE LOADED BY EITHER OF THE FOLLOWING MEDIA:

.PAPER TAPE, USING THE STANDARD PAPER TAPE PROCEDURE
.XXDP MEDIA, USING ANY XXDP DEVICE

3.2 STARTING ADDRESSES

200 - START ADDRESS, ALL SWITCHES CLEAR (SEE SECTION 3.4)

WHEN THE PROGRAM IS STARTED, A DATA PATTERN WILL BE WRITTEN TO
ALL ON-LINE DRIVES IN A SEQUENTIAL SEEK MODE. UPON COMPLETION OF
THE WRITE, THE PROGRAM GOES INTO A TESTING MODE.

204 - RESTART ADDRESS, THE RESTART ADDRESS PROVIDES THE OPERATOR WITH
THE ABILITY TO CHANGE THE DEFAULT RP/RH ADDRESSES (SEE SECTION
4.1.2), ANY PROGRAM PARAMETERS (SEE SECTION 4.1) OR CHANGE
DRIVE LIMIT PARAMETERS (SEE SECTION 4.2).

3.3 PROGRAM CONTROL

PROVIDED THE PROGRAM HAS BEEN LOADED AND STARTED VIA THE APT DUMP MODE OR THE DIAGNOSTIC IS RUNNING IN STAND ALONE PROCESSOR/DRIVE OPERATIONS ARE INITIATED AND CONTROLLED BY KEYBOARD COMMANDS AND SWITCH REGISTER SWITCH SETTINGS.

HOWEVER, IF THE PROGRAM IS LOADED VIA APT SCRIPT MODE ALL SETUP AND SWITCH REGISTER SETTINGS WILL BE PROVIDED THROUGH THE APT E TABLE. TYPEOUTS FROM THE USER DIAGNOSTIC MAY OR MAYNOT BE INHIBITED DEPENDING UPON WHETHER OR NOT THE APPROPRIATE BIT IN THE E TABLE HAS BEEN SET.

3.4 SWITCH OPTIONS

IF THE PROGRAM IS BEING RUN ON A SWITCHLESS PROCESSOR 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. 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 DRIVE 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 SETTINGS 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 1'S IN THE SWITCHES. ALL SWITCH REGISTER REFERENCES WILL BE TO THE 'SOFTWARE' REGISTER AND THE PROCEDURES DESCRIBED ABOVE MUST BE FOLLOWED.

SW<15>=1	HALT ON ERROR
SW<14>	NOT USED
SW<13>=1	1:19BIT ERROR TYPEOUT
SW<12>	NOT USED
SW<11>	NOT USED
SW<10>=1	BELL ON ERROR
SW<09>=1	CHANGE END OF PASS TO 1/4 OF NORMAL
SW<09>	NOT USED
SW<08>=1	INHIBIT END OF PASS MESSAGES
SW<07>=1	DISPLAY ALL DATA COMPARE ERRORS
SW<06>=1	DO NOT ALTER THE CURRENT OPERATION PARAMETERS
SW<05>=1	PARTIAL REGISTER DISPLAY IF ERROR; DO NOT DISPLAY ECC CORRECTION RESULTS
SW<04>=1	INHIBIT MAXIMUM ERROR COUNT CHECK; DO NOT DEASSIGN DRIVES WHEN END OF TEST IS REACHED
SW<03>=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<07> SET, DISPLAY REST OF BUFFER
SW<02>=1	INHIBIT SUBSYSTEM STATUS TYPEOUT DURING STARTUP

SW<01>=1 INHIBIT PERFORMANCE REPORT AFTER SPECIFIED TIME
SW<00>=1 PROMPT 'WRITE ANYWHERE' QUESTION DURING AUTO TEST MODE
INHIBIT DATA COMPARE AFTER READ COMMAND, W/O ERROR
READ ONLY MODE

3.5 PASS/TEST TERMINATION

A PASS IN RANDOM 'T' COMMAND MODE OR SEQUENTIAL 'T' COMMAND MODE 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 SOFT ERROR SPECIFICATION FOR THE RP DRIVE IS NO MORE THAN 1 SOFT ERROR (NON-DISK RELATED) IN 1×10^{10} BITS READ. (SEE SECTION 3.5.1 FOR THE 90% CONFIDENCE LEVEL)

THE SEEK ERROR SPECIFICATION FOR THE RP DRIVE IS NO MORE THAN 1 SEEK ERROR IN 1×10^6 SEEKS. (SEE SECTION 3.5.1 FOR THE 90% CONFIDENCE LEVEL)

A PASS IN 'W' OR 'R' COMMAND MODE IS DETERMINED BY THE MAXIMUM DISK ADDRESS LIMITS SETUP BY THE OPERATOR.

3.5.1 PASS TERMINATION

END OF PASS FOR A SINGLE DRIVE IN THE RANDOM 'T' COMMAND MODE OR SEQUENTIAL 'T' COMMAND MODE, IS DETERMINED BY ONE OF THE FOLLOWING CONDITIONS.

- A. IF PARAMETER 'ENDING' IS 1, END OF PASS OCCURS WHEN THE DRIVE HAS READ 4.128×10^9 BITS (2.58×10^8 WORDS). IF SW09=1, THE END OF PASS OCCURS WHEN THE DRIVE HAS READ 1.032×10^9 BITS ($.645 \times 10^8$ WORDS).
- B. IF PARAMETER 'ENDING' IS 0, END OF PASS OCCURS WHEN THE DRIVE HAS PERFORMED 1×10^6 SEEKS.

END OF PASS FOR A SINGLE DRIVE IN 'W' OR 'R' COMMAND MODE, IS DETERMINED AS FOLLOWS.

- A. WHEN A SEQUENTIAL SEEK IS MADE BEYOND THE MAXIMUM DISK ADDRESS LIMITS SET BY THE OPERATOR, THE PASS IS CONSIDERED ENDED.

3.5.2 TEST TERMINATION

IF SW04 IS CLEAR(0), THE TEST FOR A DRIVE IS TERMINATED WHEN:

- A. THE DRIVE HAS COMPLETED THE NUMBER OF PASSES SPECIFIED IN PARAMETER 'PASSES'.
- B. THE TOTAL ERRORS ACCUMULATED EXCEED 25.
- C. A FATAL ERROR OCCURS: EM12 OR EM14.
- D. OPERATOR DEASSIGNS THE DRIVE
- E. THE NUMBER OF PASSES SPECIFIED BY THE MONITOR HAVE BEEN REACHED, WHEN RUNNING IN 'XXDP' CHAIN MODE, 'ACT11' CHAIN MODE OR 'APT' SCRIPT MODE (ANY AUTO MODE).

3.6 RUN TIME

THE EXERCISER PROGRAM MAY BE RUN IN TWO MODES. (SEE SECTION 3.5.1)
THE PROGRAM RUN TIME VARIES GREATLY DEPENDING ON THE OPERATION
MODE SELECTED, THE READ/WRITE RATIO PARAMETER ('RATIO'), AND BY
SWR SWITCHES 0, 1, AND 2.

3.6.1 DATA TRANSFER MODE (DEFAULT)

ONE DRIVE - APPROX. 45 MIN. (TO REACH 4.128×10^9 BITS (2.58×10^8 WORDS))

3.6.2 SEEK VERIFICATION MODE

PARAMETER 'MAX WRD CNT' = 256. (1 SECTOR)
PARAMETER 'MAX TRK' = 'MIN TRK' (SAME VALUES)
PARAMETER 'MAX SEC' = 'MIN SEC' (SAME VALUES)
SW<01> = 1 (NO DATA COMPARE)
SW<00> = 1 (READ ONLY MODE)

ONE DRIVE - APPROX. 4.0 HRS (TO REACH 1×10^6 SEEKS)

3.7 DUAL PORT OPERATION

- A. LOAD THE PERFORMANCE 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 AND CYCLE THE DRIVES UP.
- C. START THE PROGRAM IN EACH PROCESSOR. RUN THE PROGRAM AS THOUGH EACH PROCESSOR WERE RUNNING INDEPENDENTLY OF THE OTHER.

3.8 XXDP, ACT11, APT11 COMPATIBILITY

THIS PROGRAM IS COMPATIBLE WITH ACT11 AND APT11 IN BOTH DUMP AND AUTOMATIC MODES.

THIS PROGRAM IS ALSO, COMPATIBLE WITH XXDP IN DUMP AND CHAIN MODES, AND PROVIDES MEDIA PROTECTION IN THE CASE WHERE THE RP07 IS THE XXDP LOADING DEVICE.

AUTOMATIC MODE OR CHAIN MODE (MONITOR)

1. THE BUS ADDRESS AND CONTROLLER INTERRUPT VECTOR ARE DEFAULTED TO 176700 AND 254 RESPECTIVELY.

DUMP MODE (NO MONITOR)

1. INPUT DIALOGUE PROMPTED AFTER PROGRAM STARTS

3.9 APT ETABLE DEFINITIONS

THE FOLLOWING DEFINITIONS ARE VALID FOR SPECIFYING APT ENVIRONMENTAL TABLE (ETABLE) ENTRIES, VIA RUNNING THE APT UTILITY PROGRAM 'TSP':

1. SOFTWARE ENVIRONMENT:
= 1 IF APT SCRIPT MODE
= 0 IF STANDLONE MODE
2. ENVIRONMENT MODE:
BIT 7 = 1 ETABLE DOES SIZING
= 0 PROGRAM DOES SIZING

BIT 6 = 1 SPOOL MESSAGES TO APT IF SCRIPT MODE
= 0 DON'T SPOOL TO APT

BIT 5 = 1 SUPPRESS TTY CONSOLE OUTPUT
= 0 ALLOW TTY CONSOLE OUTPUT

BIT 4 TO BIT 0 ARE NOT USED
3. SWITCH 1 (SOFTWARE SWITCH REGISTER)
IF ENVIRONMENT MODE BIT 7 (SIZING BIT) IS SET TO 1,
THE SOFTWARE SWITCH REGISTER WILL BE USED, INSTEAD
OF THE HARDWARE TTY CONSOLE SWITCH REGISTER.
4. SWITCH 2 (USER SWITCH REGISTER)
NOT USED
5. CPU OPTIONS
NOT USED
6. MEMORY TYPES 1-4 AND MAX MEMORY ADDRESSES
NOT USED
7. INTERRUPT VECTOR 1:
USED WHEN ENVIRONMENT MODE BIT 7 = 1;DEFAULT = 254
8. BUS PRIORITY 1:
NOT USED.
9. INTERRUPT VECTOR 2:
NOT USED
10. BUS PRIORITY 2:
NOT USED
11. BASE ADDRESS:
USED WHEN ENVIRONMENT MODE BIT 7 = 1;DEFAULT = 176700
12. DEVICE MAP:
NOT USED
13. CONTROLLER DESCRIPTOR WORDS:
NOT USED
14. CONTROLLER DESCRIPTOR WORDS:
NOT USED

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'. TYPING A RUBOUT 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 DRIVE TESTING MODE, THE PROGRAM WILL ENTER THE COMMAND MODE. IF A 'CONTROL C' IS TYPED DURING 'ENTER COMMAND' SEQUENCE, WITH NO DRIVES ASSIGNED, THE PROGRAM WILL BE RESTARTED AT LOCATION 204. OTHERWISE, THE PROGRAM WILL RETURN TO 'ENTER COMMAND' PROMPT AND WAIT FOR A CORRECT SEQUENCE OF CHARACTERS. IF 'CONTROL C' IS TYPED DURING ANY OTHER ENTRY SEQUENCE, THE PROGRAM WILL RETURN TO THE BEGINNING OF THE GROUP SEQUENCE BEING ENTERED.

4.1 PARAMETERS

THE FOLLOWING QUESTIONS ARE ASKED TO DETERMINE HOW THE RP07 WILL BE TESTED. THE DEFAULT ANSWERS TO THESE QUESTIONS ARE ALWAYS AS DOCUMENTED HERE.

'THE HELP MESSAGE CAN ONLY BE TYPED ONCE,
DO YOU WANT IT TYPED (L) ?

A 'Y' RESPONSE WILL TYPE A SUMMARY OF THE SWITCH REGISTER DEFINITIONS, ALONG WITH ALL THE COMMANDS AVAILABLE TO THE USER. A 'N' RESPONSE WILL NOT TYPE ANY HELP MESSAGE AND THE NEXT PROMPT WILL BE TYPED.

IF RUNNING THE MANUFACTURES VERSION OF THIS PROGRAM, THE FOLLOWING QUESTION WILL ASKED. OTHERWISE, CONTINUE TO THE NEXT PROMPT.

'DO YOU WANT 132. CHARACTER ERROR MODE (L) Y ?

A 'Y' RESPONSE WILL ALLOW THE 132. CHARACTER ERROR MESSAGES TO BE TYPED TO THE OUTPUT DEVICE WHEN AN ERROR OCCURS. A 'N' RESPONSE WILL OUTPUT THE NORMAL LONG ERROR TYPE MESSAGES TO THE OUTPUT DEVICE.

THE 132. CHARACTER ERROR MODE, IS A 132. CHARACTER STRING, WHICH IS SEPERATED INTO COLUMNS OF DATA FOR THE ERROR REPORT. THIS FORM OF ERROR REPORTING HELPS MANUFACTURING FIND DIFFERENT TYPES OF ERRORS MORE

EASILY. THE USER CAN SCAN DOWN A COLUMN, INSTEAD OF LOOKING THRU THE TYPICAL LONG ERROR REPORT. THE ONE DISADVANTAGE OF THE 132 CHARACTER REPORT, IS THE AMOUNT OF ERROR DATA WHICH CAN BE REPORTED TO THE USER.

THE FOLLOWING QUESTION AND WARNING MESSAGE ONLY APPEAR IN THE FIELD VERSION OF THIS DIAGNOSTIC.

'DO YOU WANT TO WRITE ANYWHERE ON MEDIA (L) N ?'

A 'N' ANSWER WILL PROCEED WITH TESTING ONLY THE FE CYLINDER AND SKIP THE FOLLOWING QUESTION. A 'Y' ANSWER WILL PROCEED TO NEXT WARNING MESSAGE AND QUESTION. IF THE PROGRAM IS IN 'READ ONLY' MODE (SW0=1), THE WARNING MESSAGE WILL BE OMITTED BUT THE QUESTION WILL BE ASKED.

! CUSTOMER DATA WILL BE OVERWRITTEN !

CONTINUE (L) ?'

A 'Y' ANSWER WILL PROCEED WITH TESTING THE ENTIRE DISK. A 'N' ANSWER WILL PROCEED WITH TESTING ONLY THE FE CYLINDER.

IF ONLY THE FE CYLINDER IS TO BE TESTED, THE FOLLOWING MESSAGE WILL BE PRINTED.

* TESTING WILL OCCUR ON FE CYLINDER ONLY *

AT THIS POINT, IF THE PROGRAM IS LOCKED IN 'READ ONLY' MODE, THE FOLLOWING MESSAGE WILL BE TYPED. IF THE PROGRAM IS NOT LOCKED IN 'READ ONLY' MODE, THE FOLLOWING MESSAGE WILL BE OMITTED.

* PROGRAM IS LOCKED IN 'READ ONLY' MODE *

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

'CHANGE PARAMETERS (L) N ?'

IF THE ENTRY IS A 'N' FOLLOWED BY A CARRIAGE RETURN OR JUST A CARRIAGE RETURN (DEFAULT), THE PROGRAM WILL NOT ALLOW ANY PARAMETERS TO CHANGED AND WILL CONTINUE.
IF THE ENTRY IS A 'Y' FOLLOWED BY CARRIAGE RETURN, THE OPERATOR WILL BE ALLOWED TO CHANGE THE PROGRAM PARAMETERS.

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. (SEE SECTION 4.1.1)

IF THIS IS THE PROGRAM'S FIRST START, THE STATUS OF THE DRIVES ON THE SELECTED MASSBUS SUBSYSTEM WILL BE PRINTED. ON ALL SUBSEQUENT STARTS, THIS TYPEOUT MAY BE INHIBITED BY SETTING SW<02> =1.

THE FOLLOWING IS AN EXAMPLE DRIVE STATUS PRINTOUT:

```
'DRIVE STATUS:
0  ONLINE RP07
1  LOAD DEVICE
2  OFFLINE RP07
3  ONLINE RP07      NON-INTERLEAVED
4  NOT PRESENT
5  NOT AN RP07
6  NOT PRESENT
7  NOT PRESENT'
```

THE ABOVE DRIVE STATUS SHOWS THAT DRIVE 0 WILL BE TESTED, WHILE DRIVES 1 - 7 WILL NOT BE TESTED.

4.1.1 KEYBOARD ENTRY PARAMETERS

QUESTION #	BASE	DEFAULT VALUE	VALUE RANGE	FUNCTION
1	10.	8192. (SEE NOTE)	6 - 8192.	CONTROLS THE MAXIMUM WORD COUNT USED FOR DATA TRANSFERS NOTE: THE PROGRAM WILL SELECT A MAXIMUM WORD COUNT, WHICH IS DETERMINED BY THE MEMORY AVAILABLE. THE MAX. WORD COUNT ASSIGNED BY THE PROGRAM IS 8192.(1 TRK) WORDS. THE OPERATOR MAY SPECIFY ANY OTHER MAX. WORD COUNT AS LONG AS THE VALUE SPECIFIED IS AT LEAST 6 WORDS BUT NO LARGER THAN 8192. WORDS OR MEMORY AVAILABLE. (WHICH EVER VALUE IS SMALLER)
2	10.	0	0 - 32767.	DETERMINES THE INTERVAL (IN MINUTES) BETWEEN AUTOMATIC PERFORMANCE REPORT TYPEOUTS; NO TYPEOUT IF THIS PARAMETER IS 0 OR IF SW<02> =1
3	10.	15.	0 - 32767.	DETERMINES THE INTERVAL (IN MINUTES) BETWEEN DATA COMPARES TO MEMORY AFTER A READ DATA COMMAND. ALWAYS DO COMPARE IF THIS PARAMETER IS 0. IF SW01 =1, THEN NO DATA COMPARES WILL BE GRANTED, UNLESS A 'DCK' ERROR OCCURS.
4	10.	1	1 - 32767.	NUMBER OF PASSES TO END OF TEST. (THIS PARAMETER IS NOT USED WHEN THE PROGRAM IS

				OPERATING IN AUTO RUN MODE)
5	10.	0	0 - 15.	IF PARAMETER=0, DATA PATTERN IS RANDOMLY SELECTED. IF PARAMETER>0, SPECIFIES ONE OF THE 15. PATTERNS. THE SELECTED DATA PATTERN IS POINTED BY THE PARAMETER 'PATTERN'. (SEE SECTION 8.4)
6	8	000000	0 OR 1	IF PARAMETER = 0, THE WORD COUNT IS RANDOMLY SELECTED BETWEEN 6 AND THE VALUE 'WRDCNT' (MAX WRD CNT). IF PARAMETER = 1, THE WORD COUNT WILL BE THE VALUE 'WRDCNT' (MAX WRD CNT).
7	8	000002	0 - 7	CONTROLS THE APPROXIMATE RATIO OF READ TO WRITE COMMANDS.
				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
8	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 SEEKS.
9	8	000001	0 OR 1	IF EQ 1, DO AN APPROPRIATE WRITE CHECK AFTER EACH WRITE COMMAND. IF EQ 0, SELECT WRITE CHECK COMMAND RANDOMLY.
10	8	000000	0 OR 1	IF PARAMETER=0, RANDOM DATA BLOCK ADDRESS IS USED IN 'T' COMMAND. IF PARAMETER=1, SEQUENTIAL DATA BLOCK IS USED IN 'T' COMMAND.

4.1.2 CHANGE DEVICE ADDRESS

THE RP/RH ADDRESS AND VECTOR MAY BE CHANGED WHEN THE PROGRAM IS STARTED AT ADDRESS 204 OR IF THE PROGRAM DOES NOT RECEIVE A RESPONSE WHEN IT ACCESSES THE DEFAULT RP/RH ADDRESS.

(DEFAULT ADDRESS = 176700, VECTOR = 254)

ADDRESS SELECTION EXAMPLES

EXAMPLE 1

RPCS1=176700 <CR> ;NO CHANGE IN ADDRESS
RPVEC=0C0254 <CR> ;NO CHANGE IN ADDRESS

EXAMPLE 2

RPCS1=176700 172400<CR> ;CHANGE BASE ADDRESS TO 172400
RPVEC=000254 224<CR> ;CHANGE VECTOR ADDRESS TO 224

4.2 KEYBOARD COMMANDS

THROUGH THE KEYBOARD COMMANDS, THE OPERATOR MAY ASSIGN DRIVES FOR TEST ('T' COMMAND), WRITE SEQUENTIAL DATA ('W' COMMAND), PERFORM A SEQUENTIAL READ ('R' COMMAND), PERFORM WRITE DATA AND FOLLOWED BY TEST ('WT' COMMAND), REQUEST A DRIVE PERFORMANCE SUMMARY ('S' COMMAND), OR DEASSIGN A DRIVE ('D' COMMAND).

THE 'T', 'W', 'R' AND 'WT' COMMANDS ARE EXCLUSIVE TO ONE ANOTHER ON THE SAME DRIVE UNDER TEST. THE 'D' COMMAND MUST BE ENTERED IN ORDER TO ISSUE A DIFFERENT COMMAND TO THE SAME DRIVE UNDER TEST. EXCEPT FOR THE 'S' COMMAND, WHICH CAN BE ENTERED AT ANY TIME DURING THE TEST.

IF THE PROGRAM WAS STARTED AT ADDRESS 204 OR IF NO DRIVES ARE ASSIGNED FOR TESTING, THE FOLLOWING MESSAGE WILL BE TYPE BEFORE ENTERING THE COMMAND MODE. HOWEVER, IF A 'CONTROL C' IS TYPED WHILE TESTING IS IN PROGRESS, THE FOLLOWING MESSAGE WILL BE OMITTED AND THE PROGRAM WILL ENTER COMMAND MODE.

'NO DRIVES ASSIGNED'

WHEN THE PROGRAM ENTERS THE COMMAND MODE, THE FOLLOWING PROMPT WILL BE TYPED:

'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 AND TRY TO ASSIGN THE DRIVE(S) THAT WERE REQUESTED. IF THE DRIVE(S) CANNOT BE ASSIGNED, ONE OF THE FOLLOWING ERROR MESSAGES WILL BE REPORTED AND THE PROCESS CONTINUES FOR EACH DRIVE.

RESPONSE -----	COMMAND(S) -----
?DRIVE N LOAD DEVICE	T, W, R, WT
?DRIVE N OFFLINE	T, W, R, WT
?DRIVE N NOT ASSIGNED	D, S

?DRIVE N ALREADY ASSIGNED	T, W, R, WT
?DRIVE N NOT PRESENT	T, W, R, WT
?DRIVE N UNSAFE	T, W, R, WT
?DRIVE N NOT AN RP07	T, W, R, WT

NEXT, THE PROGRAM WILL PROCESS ALL THE ASSIGNED DRIVES AS FOLLOWS:

WHEN THE PROGRAM IS ASSIGNING THE DRIVES, THE OPERATOR WILL BE ASKED TO CHANGE THE DRIVE PARAMETERS WITH THE FOLLOWING PROMPT:

'CHANGE DRIVE PARAMETERS (L) N ?'

IF THE ENTRY IS A 'N' FOLLOWED BY A CARRIAGE RETURN CR JUST A CARRIAGE RETURN (DEFAULT), THE PROGRAM WILL NOT ALLOW ANY DRIVE PARAMETERS TO BE CHANGED AND WILL PROCEED TO TEST THE DRIVES AS COMMANDED. IF THE ENTRY IS A 'Y' FOLLOWED BY CARRIAGE RETURN, THE OPERATOR WILL BE ALLOWED TO CHANGE THE DRIVE PARAMETERS AS FOLLOWS.

THE PROGRAM WILL FIRST TELL THE OPERATOR WHICH DRIVE IS BEING REFERENCED FOR CHANGES AND THE HARD WIRED DRV SERIAL NUMBER FORMAT:

'DRIVE # N, PGXXXX. ADDRESS LIMITS:'

WHERE 'XXXX' IS THE HARD WIRED DECIMAL SERIAL NUMBER CONTAINED IN THE RPSN REGISTER OF THE MBA. IF THE DRV SERIAL NUMBER IS NOT JUMPED IN THE RPSN REGISTER, 'XXXX' WILL APPEAR AS '????'.

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

NAME	DEFAULT VALUE	VALUE RANGE	FUNCTION
MIN CYL	*	* - 630.	THE MINIMUM CYLINDER ADDRESS
MAX CYL	630.	* - 630.	THE MAXIMUM CYLINDER ADDRESS
MIN TRK	0	0 - 31.	THE MINIMUM TRACK ADDRESS
MAX TRK	31.	0 - 31.	THE MAXIMUM TRACK ADDRESS
MIN SEC	0	0 - 49.	THE MINIMUM SECTOR ADDRESS
MAX SEC	49.	0 - 49.	THE MAXIMUM SECTOR ADDRESS

* IF RUNNING THE FIELD VERSION OF THIS PROGRAM AND TESTING OCCURS ONLY ON THE FE CLYINDER, THIS VALUE WILL BE 630. IF RUNNING THE MANUFACTURES VERSION OR THE FIELD VERSION OF THIS PROGRAM AND TESTING IS ANYWHERE ON THE MEDIA, THIS VALUE WILL BE 0.

4.2.1 'T' COMMAND

USED TO ASSIGN A DRIVE(S) FOR A TEST. THIS COMMAND IS REQUIRED TO PERFORM THE TEST OF THE DRIVE(S).

FORMAT: TN<CR>

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

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

4.2.2 '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
TERMINATED BY A CARRIAGE RETURN <CR>.

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

4.2.3 'S' COMMAND

USED TO REQUEST A PERFORMANCE SUMMARY TYPEOUT FOR THE REFERENCED
DRIVE(S). AFTER THE 'S' COMMAND HAS BEEN PERFORMED, THE PROGRAM
WILL AUTOMATICALLY RESUME TESTING THE DRIVE(S) WHICH WERE UNDER TEST.

FORMAT: SN<CR>

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

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

4.2.4 'W' COMMAND

USED TO PERFORM A SEQUENTIAL WRITE OF THE DISK, WITH DATA ACCEPTABLE
TO THE PERFORMANCE 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: WO<CR> - WRITE A DATA PATTERN ON DRIVE 0.
WA<CR> - WRITE A DATA PATTERN ON ALL AVAILABLE DRIVES.

4.2.5 'R' COMMAND

USED TO PERFORM A SEQUENTIAL READ OF THE DISK.

FORMAT: RN<CR>

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

EXAMPLE: RO<CR> - READ THE DATA ON DRIVE 0.
RA<CR> - READ THE DATA ON ALL AVAILABLE DRIVES.

4.2.6 'WT' COMMAND

USED TO PERFORM A SEQUENTIAL WRITE DATA, FOLLOWED BY A 'T' COMMAND.

FORMAT: WTN<CR>

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

EXAMPLE: WTO<CR> - WRITE A DATA PATTERN AND TEST DRIVE 0
WTA<CR> - WRITE A DATA PATTERN AND TEST ALL DRIVES

5. PERFORMANCE SUMMARY TYPEOUT

5.1 THE PROGRAM WILL DISPLAY A PERFORMANCE REPORT FOR THE DRIVES
BEING EXERCISED. THIS REPORT WILL BE DISPLAYED AUTOMATICALLY
IF THE PARAMETER 'INTRVL' IS NOT ZERO AND SW<02>=0, OR IF THE
DRIVE HAS REACHED THE DEFINED NUMBER OF PASSES AND SW<08>=0,
OR IF THE OPERATOR REQUESTS TO DO SO BY USE OF THE 'S' COMMAND.

THE REPORT TYPEOUT CONTAINS THE FOLLOWING FIELDS:

'TIME'	ELAPSED TIME OF PROGRAM
'DRIVE'	DRIVE NUMBER - DRIVE TYPE
'DRV S/N'	HARD WIRED MASSBUS ADAPTER SERIAL NUMBER(RMSN)
'PASS'	PRESENT PASS COUNT FOR THE DRIVE
'WRDS WRITN /'	
'PASS'	NUMBER OF WORDS WRITTEN EACH PASS BY THE DRIVE
'TOTAL'	TOTAL NUMBER OF WORDS (X10^6) WRITTEN BY THE DRIVE
'WRDS READ /'	
'PASS'	NUMBER OF WORDS READ EACH PASS BY THE DRIVE
'TOTAL'	TOTAL NUMBER OF WORDS (X10^6) READ BY THE DRIVE
'SEEKS'	
'PASS'	NUMBER OF SEEK OPERATIONS EACH PASS BY THE DRIVE
'TOTAL'	TOTAL NUMBER OF SEEK OPERATIONS BY THE DRIVE
'SOFT'	NUMBER OF SOFT DATA ERRORS
'HARD'	NUMBER OF HARD DATA ERRORS
'SKI'	NUMBER OF 'SKI' ERRORS
'MISP'	NUMBER OF PROGRAM DETECTED POSITIONING ERRORS
'OTHER'	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 CORRECTABLE OR DOES NOT BECOME 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

BOTH AT POSITIVE AND NEGATIVE OFFSETS.

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 COMMANDS
- D. 'DTE' ERRORS WHICH ARE CORRECTED OR WHICH BECOME ECC CORRECTABLE 'DCK' ERROR DURING THE RETRY SEQUENCE.

6. DATA CHECKING & ERROR RECOVERY

6.1 DATA COMPARISON

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

- A. THE COMMAND TERMINATED WITH NO ERRORS AND SW<01>=0
- 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

DATA VERIFICATION IS DONE EITHER THROUGH READING THE DATA BACK AND MATCHING THE DATA WITH ONE OF THE 15. PATTERNS OR THROUGH ISSUING A WRITE CHECK COMMAND AFTER DOING A WRITE DATA COMMAND.

6.3 BAD ADDRESS FLAGGING

ACCOMPLISHED BY THE RP07 HARDWARE SKIP DEFECT.

7. ERROR MESSAGES

ERRORS ARE REPORTED ON THE TTY CONSOLE. 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

(THE MESSAGE TAGS ARE GIVEN FOR REFERENCE.)

MESSAGE

TAG ---	TEXT ----
EM1	RH CONTROLLER INTERRUPT OCCURRED (RMAS=0) THE RH CONTROLLER INTERRUPTED AND THE ATTENTION SUMMARY REGISTER (RMAS) 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 RH DETECTED A CONTROL BUS PARITY ERROR WHEN READING THE INDICATED REGISTER FROM THE INDICATED DRIVE.
EM4	MASSBUS PARITY ERROR (PAR=1) THE INDICATED RP DETECTED A CONTROL BUS PARITY ERROR WHEN THE RH LOADED THE SPECIFIED REGISTER.
EM5	ADDRESS PLUG CHANGE BIT SET THE 'OPE' BIT WAS SET WHEN THE INDICATED DRIVE INTERRUPTED.
EM6	RH DIDN'T RESPOND TO ADDRESSING WHEN THE PROGRAM ADDRESSED THE RH, 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 RH 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 BY MANUAL INTERVENTION.
EM13	OPERATION NOT COMPLETED WITHIN TIME LIMIT THE DRIVE DID NOT COMPLETE THE OPERATION WITHIN 10. SECONDS 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 15. 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:
'AOE', 'KMR', '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 LOCK FAILURE ('WCF')

A WRITE LOCK 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 RH CONTROLLER OR UNIBUS TRANSFER ERROR

'TRE' IS SET IN THE RH 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 RH CONTROLLER

THE OPERATION COMPLETED NORMALLY; HOWEVER, THE PROGRAM FOUND EITHER ERROR BITS IN THE RP SET OR ERROR BITS IN THE RH CONTROLLER SET.

EM45 ECC LOGIC FAILURE

DURING 'DCK' ERROR PROCESSING, THE CONTENTS OF THE ECC POSITION REGISTER (RMEC1) OR THE CONTENTS OF ECC PATTERN REGISTER (RMEC2) WERE NOT VALID. THE POSITION REGISTER WAS EITHER 0 OR GREATER THAN 010040, OR THE PATTERN REGISTER CONTAINED 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 ERROR

THE DRIVE SIGNALLED EITHER 'SKI' ERROR.

EM51 NOT USED

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

HH:MM:SS

'HH:MM:SS' IS THE TIME SINCE THE PROGRAM WAS STARTED.
(HOURS, MINUTES, SECONDS)

LINE 2

'PRSNT COMMAND= XXXX PREV COMMAND= YYYY'

MNEMONICS USED FOR THE COMMANDS ARE DEFINED BELOW:

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)
*GETREG - GET REGISTERS (OCTAL 41)
*SETFMT - SET FORMAT (ECI OR HCI) (OCTAL 43)
*SELDRV - SELECT DRIVE (OCTAL 5)
WCKD - WRITE CHECK DATA (OCTAL 51)
WCKHD - WRITE CHECK HEADER & DATA (OCTAL 53)
WRTDAT - WRITE DATA (OCTAL 61)
FMTRK - FORMAT TRACK (WRITE HEADER & DATA) (OCTAL 63)
RDDAT - READ DATA (OCTAL 71)
RDHD - READ HEADER & DATA (OCTAL 73)

* SPECIAL RP DRIVER COMMAND (NOT A CONTROLLER COMMAND)

(DISPLAY OF THE RH/RP REGISTERS IN TWO GROUPS:
RPCS1, RPCS2, RPDS, RPER1, RPER2, RPEC1 AND 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 OCCURS AT AN ADDRESS
ON THE DISK WHICH THE OPERATOR HAS IDENTIFIED AS BEING BAD. PARAMETER
'NOTPRT' MUST BE 0 FOR THE ERROR TO BE REPORTED.

A WORD CALLED 'STATUS' IS DISPLAYED WITH THE RP REGISTERS. THE
CONTENTS OF THIS WORD IDENTIFY HOW THE ERROR WAS PROCESSED BY THE
RP DRIVE HANDLER ROUTINE. (SEE SECTION 9.7)

LINE 3

ERROR AT CXXX TYY SZZ PREV ADDR= CUUU TVV SWW

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

LINE 4

PRSNT 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 RH CONTROLLER BUFFER ADDRESS
REGISTER AND THE RH CONTROLLER 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 RP 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 WRD CNT= YYYY ACTUAL NUMBR WRDS XFRD= ZZZZ

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

LINE 11

EXPCD DATA= XXXX RECEVD DATA= YYYY WORD POS= ZZZ

THIS LINE GIVES THE EXPECTED DATA, THE RECEIVED DATA FROM THE DISK,
AND THE LOCATION OF THE WORD IN THE SECTOR. THE WORD POSITION IS IN
DECIMAL.

LINE 12

HEADER CONTENTS OF ERROR SECTOR= XXXX XXXX XXXX XXXX

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

LINE 13

RPEC1= XXXX RPEC2= YYYY

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

LINE 14

ECC CORRECTABLE WITHOUT OFFSET

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

LINE 15

READ CORRECTLY AT (NEG OR POS) OFFSET

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

LINE 16

ECC CORRECTABLE AT (NEG OR POS) OFFSET

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 COULD NOT BE PERFORMED CORRECTLY AFTER THE
INDICATED NUMBER OF RETRY ATTEMPTS.

LINE 19

DIFFERENT ERROR DURING RETRY

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

LINE 20

DATA COMPARISON ERRORS

A PRINTOUT OF THE DATA COMPARISON ERRORS FOLLOW THIS LINE.

LINE 21

TOTAL COMPARE ERRORS= XXXX

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

LINE 22

THE DATA COMPARED OK

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

LINE 23

ECC CORRECTION RESULTS

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

LINE 24

ERROR BURST BEGINS AT WORD XXY 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 'RMEC1' 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

TOTALS; ERRORS:X WRDS WRITN: YYYY WRDS READ: ZZZZ

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

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

'WRDS WRITN' IS THE TOTAL NUMBER OF WORDS WRITTEN THE DRIVE.

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

LINE 28

TOTALS; SEEKS: XXX TOTAL POS ERR= YYY TOTAL SKI ERR= Z

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

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

'TOTAL POS ERR' IS THE TOTAL NUMBER OF PROGRAM DETECTED POSITIONING
ERROR BY THE DRIVE.

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

8. PROGRAM DESCRIPTION -----

8.1 PROGRAM OPERATION

WHEN THE PROGRAM IS STARTED, PROVIDING APT TTY ENABLE BIT IS SET OR DIAGNOSTIC LOADED BY OTHER THAN APT SCRIPT MODE, ALL TABLES AND PARAMETERS ARE CLEARED OR INITIALIZED. THE PARAMETERS WHICH ARE UNDER OPERATOR TTY ENTRY CONTROL ARE CHECKED FOR VALIDITY AND CONSISTENCY. RH CONTROLLER INTERRUPT ENABLE ('IE') IS SET, TTY KEYBOARD INTERRUPT ENABLE IS SET, AND THE KW11-L OR KW11-P CLOCK IS STARTED. 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 REPORT 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 RPO7, AND IS ONLINE. THE ASSIGNMENT ROUTINE THEN ISSUES A 'READIN PRESET' INSTRUCTION, SETS 'FMT16', 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 COMMAND, THE ASSIGNED BUFFER WILL BE FILLED WITH THE SELECTED PATTERN. (WRITE CHECK COMMANDS ARE ISSUED AFTER EACH WRITE COMMAND. THE WRITE CHECK COMMAND USES THE PARAMETERS SELECTED FOR THE PRECEDING WRITE COMMAND.) CONTROL IS THEN PASSED TO THE COMMAND INITIATION ROUTINE.

THE COMMAND INITIATION ROUTINE FIRST LOOKS AT THE CYLINDER ADDRESS OF THE REQUESTED OPERATION. IF THE DRIVE MUST SEEK TO ANOTHER CYLINDER TO PERFORM THE OPERATION, THE PROGRAM ISSUES A SEARCH INSTRUCTION TO THE DRIVE WITH A 'TARGET' SECTOR WHICH IS 1 SECTOR 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 THEN ISSUES THE REQUESTED COMMAND TO THE DRIVE THAT INTERRUPTED.

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 RH CONTROLLER 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 COMMAND WAS A READ COMMAND, 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.

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 ERROR - EM50
WRITE CHECK WITHOUT 'DCK' SET - EM23
RH CONTROLLER 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 RH CONTROLLER - 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 COMMAND TERMINATION.

WHEN THE DUAL PORT HANDLER ROUTINE IN THE EXERCISER PROGRAM RECEIVES A REQUEST FOR A DRIVE, THE PROGRAM VERIFIES THAT THE DRIVE IS ONLINE. THE DRIVE IS SELECTED AND THE RPCS1 REGISTER IS READ TO TEST THE 'DVA' BIT. IF THE DRIVE IS IN NEUTRAL, THIS WILL SEIZE THE DRIVE. IF THE DRIVE IS SEIZED BY THE OTHER PORT, A DRIVE CLEAR COMMAND IS ISSUED TO THE DRIVE TO SET 'PORT REQUEST'. THE PROGRAM THEN 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 15. SECOND TIMER FOR THE DRIVE. IF THE DRIVE HAS NOT SWITCHED TO THE REQUESTING SYSTEM WITHIN THE 15. 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 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 COMMAND 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 SWAP 'MAX' AND 'MIN' ADDRESSES AND CONTINUE.
- B. THE WORD COUNT IS RANDOMLY SELECTED BETWEEN 6 AND THE VALUE 'WRDCNT' (MAX WRD CNT). 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 AND NEEDS 2 MORE LOCATIONS IF A READ HEADER & DATA COMMAND IS ISSUED.
- C. THE DATA WRITTEN IS RANDOMLY SELECTED AMONG THE 15. STANDARD PATTERNS. THE PARAMETER 'PATTERN' ENABLES THE RANDOM PATTERN SELECTION. IF THIS PARAMETER IS 0.
- D. THE COMMANDS ARE SELECTED RANDOMLY. WRITE CHECK DATA COMMAND IS PERFORMED ONLY IF THE PREVIOUS COMMAND WAS THE APPROPRIATE WRITE DATA COMMAND.

8.4 DATA PATTERNS

THE PROGRAM SELECTS ONE OF THE FOLLOWING DATA PATTERNS TO WRITE WHEN A WRITE COMMAND 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. IF THE PARAMETER 'PATTERN' IS 0 THE PROGRAM WILL ATTEMPT TO MATCH THE FIRST 4 DATA WORDS OF EACH SECTOR, TO ONE OF THE FOLLOWING PATTERNS. HOWEVER, IF THE PARAMETER 'PATTERN' IS NOT 0, THE PROGRAM WILL ASSUME THAT THE DESIRED DATA PATTERN IN LOCATION 'PATTERN' IS THE DATA TO LOOK FOR AND WILL NOT TRY TO MATCH ANY PATTERNS. THIS ALLOWS THE OPERATOR TO SCAN THE DISK FOR ANY SPECIFIC PATTERN.

PAT 1 PAT 2 PAT 3 PAT 4 PAT 5 PAT 6 PAT 7 *PAT 8

000001	177776	000000	133331	052525	147556	155555	030221
000003	177774	000000	133331	052525	147556	133333	030221
000007	177770	000000	133331	052525	147556	155555	030221
000017	177760	177777	133331	125252	147556	133333	030221
000037	177740	177777	133331	125252	147556	155555	030221
000077	177700	177777	133331	125252	147556	133333	030221
000177	177600	000000	133331	052525	147556	155555	030221
000377	177400	000000	133331	052525	147556	133333	030221
000777	177000	177777	133331	125252	147556	155555	030221
001777	176000	177777	133331	125252	147556	133333	030221
003777	174000	000000	133331	052525	147556	155555	030221
007777	170000	177777	133331	125252	147556	133333	030221
017777	160000	000000	133331	052525	147556	155555	030221
037777	140000	177777	133331	125252	147556	133333	030221
077777	100000	000000	133331	052525	147556	155555	030221
177777	000000	177777	133331	125252	147556	133333	030221

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	177377	153333	177777	000000
004000	173777	155555	177577	066667	177777	000000
010000	167777	172666	177677	153333	177777	000000
020000	157777	155555	177773	066667	177777	000000
040000	137777	172666	177775	153333	177777	000000
100000	077777	155555	177776	066667	177777	000000

* WORST CASE PATTERN

9.1 RH/RP DRIVER

THIS DOCUMENT IS THE USER'S GUIDE FOR THE RH/RP DRIVER.

9.2 TO INITIALIZE THE DRIVER:

```
JSR   PC,RPINIT
RETURN
```

UPON RETURN YOU MUST EXAMINE THE 'DRVSTA' TABLE TO DETERMINE THE DRIVES THAT ARE ONLINE FOR TESTING. THE 'DRVSTA' TABLE IS EIGHT BYTES; ONE BYTE PER DRIVE. THE STATE OF EACH DRIVE WILL BE INDICATED AS FOLLOWS:

DRVSTA	DRIVE STATE
--------	-------------


```

>0      ONLINE
=0      OFFLINE, DRIVE
        IS NOT AN RP07, OR
        NONEXISTENT DRIVE
<0      UNSAFE
  
```

THE DRIVE TYPE IS DEFINED IN AN 8 BYTE LONG TABLE TAGGED 'DRVTYP'.
 THE TABLE CONTAINS ONE BYTE FOR EACH DRIVE AND IS INDEXED BY THE
 DRIVE NUMBER. ENTRIES ARE ENCODED AS FOLLOWS:

DRVTYP	CONDITION
0	NONEXISTENT DRIVE
4	RP07
5	RP07
-1	NOT AN RP07

THE 'RMINIT' ROUTINE WILL DO A READIN PRESET AND WILL SET FMT16.

9.3 AFTER THE DRIVER HAS BEEN INITIALIZED, IT IS CALLED USING THE
 FOLLOWING SEQUENCE.

```

CALL:
      JSR      R0,RP07      ;MAKE THE CALL
      PNTDPB      ;ADDRESS OF DPB*
      RETURN1     ;RETURN IF QUEUE IS FULL
      RETURN2     ;RETURN IF REQUEST IS IN
                  ;QUEUE OR THERE IS AN
                  ;ERROR CONDITION
  
```

*DPB (DATA PARAMETER BLOCK)

```

PNTDPB: .BYTE 0      ;(0) DRIVE NUMBER
        .BYTE 0      ;(1) OFFSET VALUE OR FMT16, ECT, AND HCI
        .BYTE 0      ;(2) COMMAND
        .BYTE 0      ;(3) PSEL AND A17 AND A16
        .WORD 0      ;(4) WORD COUNT (MUST BE NEG.)
        .WORD 0      ;(6) BUFFER ADDRESS OR
                    ;REGISTER TABLE POINTER
        .BYTE J      ;(10) SECTOR ADDRESS OR
                    ;FIRST REG. INDEX
        .BYTE 0      ;(11) TRACK ADDRESS OR
                    ;LAST REG. INDEX
        .WORD 0      ;(12) CYLINDER ADDRESS
        .WORD 0      ;(14) ERROR TABLE POINTER
                    ;POINTS TO THE FIRST OF TWENTY
                    ;LOCATIONS OF WHERE THE DRIVER
                    ;IS TO STORE THE RH/RP
                    ;REGISTERS ON AN ERROR. IF LEFT
                    ;ZERO REGISTERS ARE NOT SAVED.
        .WORD 0      ;(16) STATUS/ERROR INDICATOR
                    ;BIT15=1=ERROR OCCURRED
                    ;BIT07=1=DONE
                    ;BIT14-BIT09 AND BIT06-BIT03
                    ;INDICATE TYPE OF ERROR
  
```

9.4 THE DRIVER PROVIDES A SOFTWARE TIMEOUT CAPABILITY. TO UTILIZE THIS CAPABILITY YOU MUST SUPPLY THE "RP TIMER" ROUTINE WITH THE ELAPSED TIME IN THE FOLLOWING MANNER:

```

MOV    #16.,-(SP)    ;16. MILLISECONDS BETWEEN
                    ;CLOCK TICKS
JSR    PC,RPTMR     ;CALL THE TIMER ROUTINE
  
```

IT SHOULD BE NOTED THAT YOU MUST PROVIDE THE CODE TO DRIVE THE CLOCK AND THE ELAPSED TIME MUST BE IN MILLISECONDS.

9.4.1 EXAMPLE - WRITE 1000. WORDS

```

1$:    JSR    R0,RP07    ;CALL THE DRIVER
        WFTDPB        ;DPB ADDRESS
        F'    1$        ;WAIT FOR QUEUE IF FULL
2$:    ST     WRTDPB+16  ;WAIT FOR COMMAND TO COMPLETE
        BEQ    2$
        BMI    ERROR1   ;ERROR OCCURRED
        .
        .
        .
  
```

```

WRTDPB: .BYTE    5        ;DRIVE #5
        .BYTE    0
        .BYTE    161     ;WRITE COMMAND
        .BYTE    0
        .WORD    -1000.  ;WORD COUNT
        .WORD    WRTBUF  ;BUFFER ADDRESS
        .BYTE    3
        .BYTE    5
        .WORD    400     ;CYLINDER
        .WORD    ERRTB5  ;ERROR TABLE
        .WORD    0      ;STATUS/ERROR INDICATOR
  
```

ALTERNATE DPB SETUP

```

WRTDPB: .WORD    5        ;THIS SETUP ACHIEVED
        .WORD    WRITE   ;EVERYTHING THE
        .WORD    -1000.  ;ABOVE TABLE DID, BUT
        .WORD    WRTBUF  ;IN A CLEANER FORMAT
        .BYTE    3,5
        .WORD    400,ERRTB5,0
  
```

9.5 RH/RP REGISTERS

MNEMONIC	INDEX
RPCS1	0
RPWC	2
RPBA	4
RPJA	6
RPCS2	10
RPDS	12
RPER1	14
RPAS	16

RPLA	20
RPDB	22
RPMR1	24
RPDT	26
RPSN	30
RPOF	32
RPDC	34
RPHR	36
RPMR2	40
RPER2	42
RPEC1	44
RPEC2	46
RPBAE	50*
RPCS3	52*

* RH70 CONTROLLER REGISTERS

9.6 COMMANDS PERFORMED BY THE DRIVER

COMMAND -----	CODE ----	COMMAND TYPE -----
SEEK	105	P
RECALIRATE	107	P
DRIVE CLEAR	111	N
RELEASE	113	N
OFFSET	115	P
RETURN TO CENTER	117	P
READIN PRESET	121	N
PACK ACKNOWLEDGE	123	N
SEARCH	131	P
GET REGISTER(S)	141	S
SET FORMAT	143	S
SELECT DRIVE	145	S
WRITE CHECK DATA	151	D
WRITE CHK HEADER & DATA	153	D
WRITE DATA	161	D
WRITE HEADER & DATA	163	D
READ DATA	171	D
READ HEADER & DATA	173	D

N = HOUSEKEEPING
 P = POSITIONING
 D = DATA TRANSFER
 S = SPECIAL PROVIDED BY THE DRIVER

9.7 DPB STATUS/ERROR INDICATOR WORD

THIS INDICATOR WILL INFORM THE USER OF THE RESULTS OF THE REQUEST.
 THIS IS ACCOMPLISHED BY SETTING VARIOUS BITS OF THE INDICATOR TO
 A ONE.

BIT NO. -----	MEANING IF ON A '1' -----
15	ERROR OCCURRED

DONE (BIT07=0); BITS 14-9 SPECIFIES TYPE
DONE (BIT07=1); BITS 6-3 SPECIFIES TYPE

14(1) USER MADE A REQUEST FOR A FUNCTION TO BE PERFORMED ON AN OFFLINE OR UNSAFE DRIVE

13(1) USER MADE A REQUEST FOR A FUNCTION TO BE PERFORMED ON A DRIVE THAT HAS AN UNLOAD REQUEST IN QUEUE.

12(2) PERSISTENT UNSAFE CONDITION EXIST.

11(2) UNCORRECTABLE PARITY ERROR OCCURRED

10(2)(4) FATAL PARITY ERROR. A MASSBUS CLEAR WAS PERFORMED, ALL QUEUES WERE EMPTIED, AND ALL DRVACT'S SET TO THE IDLE STATE

9(3)(4) SOFTWARE TIMEOUT OCCURRED ON THIS DRIVE

8(4) SOFTWARE TIMEOUT OCCURRED ON ANOTHER DRIVE

7 DONE

6(2) ERROR OCCURRED DURING AN I/O OPERATION

5(2) ERROR OCCURRED DURING AN OPERATION OTHER THAN I/O.

4(2) CORRECTABLE UNSAFE CONDITION OCCURRED

3(2) DRIVE ERROR OCCURRED THAT CAUSED AN AUTOMATIC "RECALIBRATE" SEQUENCE

2 PORT REQUEST TIMEOUT. THE DRIVER REQUESTED THE DRIVE BUT THE OPPOSITE PORT DID NOT RELEASE THE DRIVE WITHIN 15. SECONDS.

1 NON-EXISTENT DRIVE REQUESTED. USER MADE A REQUEST FOR A NON-EXISTENT DRIVE.

NOTES FOR ABOVE

(1) = REQUEST WASN'T PUT IN QUEUE. (RH/RP REGISTERS WERE NOT SAVED)

(2) = REQUEST QUEUE HAS BEEN EMPTIED. THE DRIVER ISSUED A "DRIVE CLEAR" TO THE DRIVE. NOTE: ALL RH/RP REGISTERS ARE SAVED AS PER DPB+14 BEFORE THE "DRIVE CLEAR".

(3) = REQUEST QUEUE HAS BEEN EMPTIED. THE DRIVER ISSUED A MASSBUS INIT. ALL RH/RP REGISTERS FOR THE DRIVE WERE SAVED AS PER DPB+14 BEFORE THE INIT.

(4) = A 'RECALIBRATE' SHOULD BE ISSUED
 BEFORE ANY OTHER COMMAND.

9.8 ERROR CALLS MADE BY THE DRIVER.

THERE ARE A FEW ERRORS THAT CAN OCCUR THAT CAN NOT BE INDICATED IN A DPB.

WHEN THIS TYPE OF ERROR IS DETECTED BY THE DRIVER IT WILL MAKE
 AN ERROR CALL OF THE FORM 'ERROR N', WHERE 'N' IS THE ERROR
 NUMBER AND THE ERROR WILL BE AN EMT INSTRUCTION.

N	TYPE	DATA AVAILABLE
-	----	-----
1	RH/RP INTERRUPT OCCURRED (RHAS=0)	*R4= RPCS1'S ADDRESS
2	UNEXPECTED ATTENTION OCCURRED	R1= DRIVE NUMBER R3= ATA BIT *R4= RPCS1'S ADDRESS R5= (RMAS) RPERRS =RPDS RPERRS+2=RPER1 RPERRS+4=RPER2 RPERRS+6=RPMR2
3	MASSBUS PARITY ERROR (MCPE=1)	RD.ADR= ADDRESS OF REG. READ RD.WRD= WORD READ
4	MASSBUS PARITY ERROR (PAR=1)	WRT.AD= ADDRESS OF REG. WRITTEN WRT.WD= WORD WRITTEN RD.WRD= WORD READ BACK
5	ADDRESS PLUG CHANGE BIT SET ('OPE' ERROR)	R1= DRIVE NUMBER R3= ATA BIT *R4= RPCS1'S ADDRESS R5= (RMAS) RPERRS =RPDS RPERRS+2=RPER1 RPERRS+4=RPER2 RPERRS+6=RPMR2

* THIS IS THE ACTUAL UNIBUS ADDRESS (176700)

.REM @

VERSION (CZRJO-A-0)

1. THIS VERSION IS THE STARTING POINT FOR CX DIAGNOSTIC SUPPORT OF
THE RP07 DISK DRIVE.

@

1
 2
 85
 87
 88
 89
 90
 91
 92
 93
 94
 95
 96
 97
 98
 99
 100
 101
 102
 103
 104

```

;*LAST REVISION 01-JAN-83

.TITLE CZRJOAO RP07 PERF EXER
;*COPYR.GHT (C) 1983
;*DIGITAL EQUIPMENT CORPORATION
;*CCLORADO SPGS., CO. 80919
;*
;*PROGRAM BY MIKE LEAVITT
;*
;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
;*PACKAGE (MAINDEC-11-DZQAC-C5), 18-MAR-81
;*
.SBTTL OPERATIONAL SWITCH SETTINGS
;*
;*      SWITCH      USE
;*      -----      -
;*      15          HALT ON ERROR
;*      13          INHIBIT ERROR TYPEOUTS
;*      10          BELL ON ERROR
;*      8           INHIBIT END OF PASS MESSAGES
;*      7           DISPLAY ALL DATA COMPARE ERRORS
;*      6           DON'T CHANGE PARAMETERS (LOOP ON PRESENT VALUES)
;*      5           A. PARTIAL REGISTER DISPLAY IF ERROR
;*                  B. NO ECC CORRECTION RESULTS DISPLAYED IF ERROR
;*      4           A. DO NOT CHECK FOR MAXIMUM ERROR COUNTS
;*                  B. DO NOT DROP DRIVE AT END OF TEST
;*      3           A. DISPLAY ERROR SECTOR IF 'DCK' DTE', OR 'WCF' ERROR
;*                  B. DISPLAY SECTOR IF 'DCK' ERR U RRECTABLE AFTER
;*                     28TH RETRY
;*                  C. IF DATA COMPARE ERROR & SW07 1, DISPLAY
;*                     REMAINDER OF BUFFER
;*      2           A. DO NOT TYPE DRIVE STATUS AT PROGRAM START
;*                  B. DO NOT TYPE PERFORMANCE REPORT AFTER SPECIFIED TIME
;*      1           INHIBIT DATA COMPARE AFTER READ W/O 'DCK' ERROR
;*      0           READ ONLY MODE

.SBTTL BASIC DEFINITIONS

;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
001100 STACK = 1100
104000 ERROR = EMT          ;;BASIC DEFINITION OF ERROR CALL
000004 SCOPE = IOT         ;;BASIC DEFINITION OF SCOPE CALL

;*MISCELLANEOUS DEFINITIONS
000011 HT = 11             ;;CODE FOR HORIZONTAL TAB
000012 LF = 12             ;;CODE FOR LINE FEED
000015 CR = 15             ;;CODE FOR CARRIAGE RETURN
000200 CRLF = 200          ;;CODE FOR CARRIAGE RETURN-LINE FEED
177776 PS = 177776        ;;PROCESSOR STATUS WORD
177776 PSW=PS
177774 STKLMT = 177774     ;;STACK LIMIT REGISTER
177772 PIRQ = 177772       ;;PROGRAM INTERRUPT REQUEST REGISTER
177570 DSWR = 177570       ;;HARDWARE SWITCH REGISTER
177570 DDISP = 177570      ;;HARDWARE DISPLAY REGISTER

;*GENERAL PURPOSE REGISTER DEFINITIONS
000000 R0 = %0             ;;GENERAL REGISTER
  
```


000001	R1	=	%1	:: GENERAL REGISTER
000002	R2	=	%2	:: GENERAL REGISTER
000003	R3	=	%3	:: GENERAL REGISTER
000004	R4	=	%4	:: GENERAL REGISTER
000005	R5	=	%5	:: GENERAL REGISTER
000006	R6	=	%6	:: GENERAL REGISTER
000007	R7	=	%7	:: GENERAL REGISTER
000006	SP	=	%6	:: STACK POINTER
000007	PC	=	%7	:: PROGRAM COUNTER

.*PRIORITY LEVEL DEFINITIONS

000000	PR0	=	0	:: PRIORITY LEVEL 0
000040	PR1	=	40	:: PRIORITY LEVEL 1
000100	PR2	=	100	:: PRIORITY LEVEL 2
000140	PR3	=	140	:: PRIORITY LEVEL 3
000200	PR4	=	200	:: PRIORITY LEVEL 4
000240	PR5	=	240	:: PRIORITY LEVEL 5
000300	PR6	=	300	:: PRIORITY LEVEL 6
000340	PR7	=	340	:: PRIORITY LEVEL 7

.*"SWITCH REGISTER" SWITCH DEFINITIONS

100000	SW15	=	100000
040000	SW14	=	40000
020000	SW13	=	20000
010000	SW12	=	10000
004000	SW11	=	4000
002000	SW10	=	2000
001000	SW09	=	1000
000400	SW08	=	400
000200	SW07	=	200
000100	SW06	=	100
000040	SW05	=	40
000020	SW04	=	20
000010	SW03	=	10
000004	SW02	=	4
000002	SW01	=	2
000001	SW00	=	1
001000	SW9=SW09		
000400	SW8=SW08		
000200	SW7=SW07		
000100	SW6=SW06		
000040	SW5=SW05		
000020	SW4=SW04		
000010	SW3=SW03		
000004	SW2=SW02		
000002	SW1=SW01		
000001	SW0=SW00		

.*DATA BIT DEFINITIONS (BIT00 TO BIT15)

100000	BIT15	=	100000
040000	BIT14	=	40000
020000	BIT13	=	20000
010000	BIT12	=	10000
004000	BIT11	=	4000
002000	BIT10	=	2000
001000	BIT09	=	1000
000400	BIT08	=	400

```

000200 BIT07 = 200
000100 BIT06 = 100
000040 BIT05 = 40
000020 BIT04 = 20
000010 BIT03 = 10
000004 BIT02 = 4
000002 BIT01 = 2
000001 BIT00 = 1
001000 BIT9=BIT09
000400 BIT8=BIT08
000200 BIT7=BIT07
000100 BIT6=BIT06
000040 BIT5=BIT05
000020 BIT4=BIT04
000010 BIT3=BIT03
000004 BIT2=BIT02
000002 BIT1=BIT01
000001 BIT0=BIT00
    
```

```

;*BASIC "CPU" TRAP VECTOR ADDRESSES
000004 ERRVEC = 4 ;;TIME OUT AND OTHER ERRORS
000010 RESVEC = 10 ;;RESERVED AND ILLEGAL INSTRUCTIONS
000014 TBITVEC = 14 ;;"T" BIT
000014 TRTVEC = 14 ;;TRACE TRAP
000014 BPTVEC = 14 ;;BREAKPOINT TRAP (BPT)
000020 IOTVEC = 20 ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
000024 PWRVEC = 24 ;;POWER FAIL
000030 EMTVEC = 30 ;;EMULATOR TRAP (EMT) **ERROR**
000034 TRAPVEC = 34 ;;"TRAP" TRAP
000060 TKVEC = 60 ;;TTY KEYBOARD VECTOR
000064 TPVEC = 64 ;;TTY PRINTER VECTOR
000240 PIRQVEC = 240 ;;PROGRAM INTERRUPT REQUEST VECTOR
    
```

105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129

.SBTTL RP07 REGISTERS

;CONTROL AND STATUS REGISTER 1 (RPCS1)

```

000100 IE = 100 ;INTERRUPT ENABLE (BIT #6)
000200 RDY = 200 ;READY (BIT #7)
000400 A16 = 400 ;HIGH ORDER BUS ADDRESS BIT (BIT #8)
001000 A17 = 1000 ;HIGH ORDER BUS ADDRESS BIT (BIT #9)
002000 PSEL = 2000 ;PORT SELECT (BIT #10)
020000 MCPE = 20000 ;MASSBUSS PARITY ERROR (BIT #13)
040000 TRE = 40000 ;TRANSFER ERROR (BIT #14)
;SC = 100000 ;SPECIAL CONDITION (BIT #15)
    
```

;WORD COUNT REGISTER (RPWC)
 ;(EACH BIT IS CALLED BY BIT NUMBER)

;BUS ADDRESS REGISTER (RPBA)
 ;(EACH BIT IS CALLED BY BIT NUMBER)

;CONTROL AND STATUS REGISTER 2 (RPCS2)

```

000001 US1 = 1 ;UNIT SELECT (BIT #0)
000002 US2 = 2 ;UNIT SELECT (BIT #1)
000004 US4 = 4 ;UNIT SELECT (BIT #2)
    
```

```

130      000010      BAI      = 10      ;BUS ADDRESS INCREMENT INHIBIT (BIT #3)
131      000020      PAT      = 20      ;MASSBUS PARITY TEST (BIT #4)
132      000040      CLR      = 40      ;CLEAR (BIT #5)
133      000100      IR       = 100     ;INPUT READY (BIT #6)
134      000200      OR       = 200     ;OUTPUT READY (BIT #7)
135      000400      MDPE     = 400     ;MASS BUS PARITY ERROR (BIT #8)
136      001000      MXF      = 1000    ;MISSED TRANSFER ERROR (BIT #9)
137      002000      PGE      = 2000    ;PROGRAM ERROR (BIT #10)
138      004000      NEM      = 4000    ;NON EXISTENT MEMORY (BIT #11)
139      010000      NED      = 10000   ;NON EXISTENT DRIVE (BIT #12)
140      020000      UPE      = 20000   ;UNIBUS PARITY ERROR (BIT #13)
141      040000      WCE      = 40000   ;WRITE CHECK ERROR (BIT #14)
142      100000      DLT      = 100000  ;DATA LATE (BIT #15)
143
144      ;DATA BUFFER REGISTER (RPDB)
145      ;(EACH BIT IS CALLED BY BIT NUMBER)
146
147      .SBTTL  RP07 REGISTERS
148
149      ;CONTROL AND STATUS 1 REGISTER. (#00)
150
151      000001      GO       = 1       ;GO BIT (BIT #0)
152      000002      FO       = 2       ;FUNCTION CODE BIT #1
153      000004      F1       = 4       ;FUNCTION CODE BIT #2
154      000010      F2       = 10      ;FUNCTION CODE BIT #3
155      000020      F3       = 20      ;FUNCTION CODE BIT #4
156      000040      F4       = 40      ;FUNCTION CODE BIT #5
157      004000      DVA      = 4000    ;DEVICE AVAILABLE (BIT #11)
158
159      ;DRIVE STATUS REGISTER (RPDS1) (#01)
160
161      000001      OFFON    = 1       ;OFFSET ON (BIT #0)
162      000002      EWN      = 2       ;EARLY WARNING (BIT #1)
163      000004      ILV      = 4       ;INTERLEAVE (BIT #2)
164      000100      VV       = 100     ;VOLUME VALID (BIT #6)
165      000200      DRY      = 200     ;DRIVE READY (BIT #7)
166      000400      DPR      = 400     ;DRIVE PRESENT (BIT #8)
167      001000      PGM      = 1000    ;PROGRAMABLE (BIT #9)
168      002000      LBT      = 2000    ;LAST SECTOR TRANSFERRED (BIT #10)
169      004000      WRL      = 4000    ;WRITE LOCK (BIT #11)
170      010000      MOL      = 10000   ;MEDIUM ON-LINE (BIT #12)
171      020000      PIP      = 20000   ;POSITIONING OPERATION IN PROGRESS (BIT #13)
172      040000      ERR      = 40000   ;COMPOSITE ERROR (BIT #14)
173      100000      ATA      = 100000  ;ATTENTION ACTIVE (BIT #15)
174
175      ;ERROR REGISTER #01 (RPER1) (#02)
176
177      000001      ILF      = 1       ;ILLEGAL FUNCTION (BIT #0)
178      000002      ILR      = 2       ;ILLEGAL REGISTER (BIT #1)
179      000004      RMR      = 4       ;REGISTER MODIFICATION REFUSED (BIT #2)
180      000010      PAR      = 10      ;PARITY ERROR (BIT #3)
181      000020      FER      = 20      ;FORMAT ERROR (BIT #4)
182      000040      WCF      = 40      ;WRITE CLOCK FAIL (BIT #5)
183      000100      ECH      = 100     ;ECC HARD ERROR (BIT #6)
184      000200      HCE      = 200     ;HEADER COMPARE ERROR (BIT #7)
185      000400      HCRC     = 400     ;HEADER CRC ERROR (BIT #8)
186      001000      AOE      = 1000    ;ADDRESS OVERFLOW ERROR (BIT #9)

```

187	002000	IAE	= 2000	:INVALID ADDRESS ERROR (BIT #10)
188	004000	WLE	= 4000	:WRITE LOCK ERROR (BIT #11)
189	010000	DTE	= 10000	:DRIVE TIMING ERROR (BIT #12)
190	020000	OPI	= 20000	:OPERATION INCOMPLETE (BIT #13)
191	040000	UNS	= 40000	:DRIVE UNSAFE (BIT #14)
192	100000	DCK	= 100000	:DATA CHECK ERROR (BIT 15)
193				
194				:MAINTENANCE REGISTER (RPMR1)(#03)
195				
196				:ATTENTION SUMMARY PSEUDO-REGISTER (RPAS) (#04)
197				
198	000001	AT0	= 1	:DEVICE 0 (BIT #0)
199	000002	AT1	= 2	:DEVICE 1 (BIT #1)
200	000004	AT2	= 4	:DEVICE 2 (BIT #2)
201	000010	AT3	= 10	:DEVICE 3 (BIT #3)
202	000020	AT4	= 20	:DEVICE 4 (BIT #4)
203	000040	AT5	= 40	:DEVICE 5 (BIT #5)
204	000100	AT6	= 100	:DEVICE 6 (BIT #6)
205	000200	AT7	= 200	:DEVICE 7 (BIT #7)
206				
207				:DESIRED SECTOR/TRACK ADDRESS REGISTER (RPDA) (#05)
208				
209				:DRIVE TYPE REGISTER (RPDT) (#06)
210				
211	000001	DT00	= 1	:DRIVE TYPE NUMBER BIT 1
212	000002	DT01	= 2	:DRIVE TYPE NUMBER BIT 2
213	000004	DT02	= 4	:DRIVE TYPE NUMBER BIT 3
214	000010	DT03	= 10	:DRIVE TYPE NUMBER BIT 4
215	000020	DT04	= 20	:DRIVE TYPE NUMBER BIT 5
216	000040	DT05	= 40	:DRIVE TYPE NUMBER BIT 6
217	000100	DT06	= 100	:DRIVE TYPE NUMBER BIT 7
218	000200	DT07	= 200	:DRIVE TYPE NUMBER BIT 8
219	000400	DT08	= 400	:DRIVE TYPE NUMBER BIT 9
220	004000	DRQ	= 4000	:DRIVE REQUEST REQUIRED (BIT #11)
221	020000	MOH	= 20000	:MOVING HEAD (BIT #13)
222	040000	TAP	= 40000	:TAPE DRIVE (BIT #14)
223	100000	NSA	= 100000	:NOT SECTOR ADDRESSED (BIT #15)
224				
225				:LOOK-AHEAD REGISTER (RPLA) (#07)
226				
227	000100	SC1	= 100	:SECTOR COUNT FIELD 0 (BIT #6)
228	000200	SC2	= 200	:SECTOR COUNT FIELD 1 (BIT #7)
229	000400	SC04	= 400	:SECTOR COUNT FIELD 2 (BIT #8)
230	001000	SC10	= 1000	:SECTOR COUNT FIELD 3 (BIT #9)
231	002000	SC20	= 2000	:SECTOR COUNT FIELD 4 (BIT #10)
232	004000	SC40	= 4000	:SECTOR COUNT FIELD 5 (BIT #11)
233	010000	SC100	= 10000	:SECTOR COUNT FIELD 6 (BIT #12)
234				
235				:SERIAL NUMBER REGISTER (RPSN) (#10)
236				: (EACH IS CALLED BY BIT NUMBER)
237				
238				:OFFSET REGISTER (RPOF) (#11)
239				
240	000200	OFFDIR	= 200	:RPC7 OFFSET DIRECTION
241	002000	HCI	= 2000	:HEADER COMPARE INHIBIT (BIT #10)
242	004000	ECI	= 4000	:ERROR CORRECTION CODE INHIBIT (BIT #11)
243	010000	FMT16	= 10000	:FORMAT BIT (BIT #12)

```

244
245      ;DESIRED CYLINDER ADDRESS (RPDC) (#12)
246      ;(EACH BIT IS CALLED BY BIT NUMBER)
247
248      ;CURRENT CYLINDER ADDRESS (RPCC) (#13)
249      ;(EACH BIT IS CALLED BY BIT NUMBER)
250
251      ;RP07 ERROR REGISTER #2 (RPER2) (#14)
252
253      000400      WRYUNS = 400      ;WRITE READY UNSAFE (BIT 8)
254      001000      WOR      = 1000     ;WRITE OVERRUN (BIT 9)
255      002000      RWU1    = 2000     ;READ/WRITE UNSAFE #1 (BIT 10)
256      004000      RWU2    = 4000     ;READ/WRITE UNSAFE #2 (BIT 11)
257      010000      RWU3    = 10000    ;READ/WRITE UNSAFE #3 (BIT 12)
258      100000      PGE      = 100000   ;PROGRAM ERROR (BIT 15)
259
260      ;RP07 ERROR REGISTER #3 (RPER3) (#15)
261
262      000001      DGE      = 1      ;DIAGNOSTIC ERROR (BIT 0)
263      000002      TPE      = 2      ;TEMPERATURE WARNING ERROR (BIT 1)
264      000004      AIR      = 4      ;AIR SYSTEM WARNING ERROR (BIT 2)
265      000010      DPE      = 10     ;DATA PARITY ERROR (BIT 3)
266      000020      BPE      = 20     ;BUFFER PARITY ERROR (BIT 4)
267      000040      DCU      = 40     ;DC UNSAFE (BIT 5)
268      000100      IXU      = 100    ;INDEX UNSAFE (BIT 6)
269      000200      DVC      = 200    ;DEVICE CHECK (BIT 7)
270      000400      TCF      = 400    ;TACH CALIBRATION FAILURE (BIT 8)
271      001000      LCE      = 1000   ;LOSS OF CYLINDER ERROR (BIT 9)
272      002000      LBC      = 2000   ;LOSS OF BIT CLOCK (BIT 10)
273      040000      SKI      = 40000  ;SEEK INCOMPLETE (BIT 14)
274      100000      BSE      = 100000 ;BAD SECTOR ERROR (BIT 15)
275
276      ;ECC POSITION REGISTER (RPEC1) (#16)
277      ;(EACH BIT IS CALLED BY BIT NUMBER)
278
279      ;ECC PATTERN REGISTER (RPEC2) (#17)
280      ;(EACH BIT IS CALLED BY BIT NUMBER)
281
282      .SBTTL  RP07 DRIVER COMMANDS
283
284      000101      NOOP     = 101     ;NO OPERATION
285      000105      SEEK     = 105     ;SEEK
286      000107      RECAL    = 107     ;RECALIBRATE
287      000111      DRVCLR   = 111     ;DRIVE CLEAR
288      000113      RELSE    = 113     ;RELEASE
289      000117      RTC      = 117     ;RETURN TO CENTER LINE
290      000121      READIN   = 121     ;READ IN PRESET
291      000121      SEARCH   = 131     ;SEARCH
292      000131      WCKD     = 151     ;WRITE CHECK DATA
293      000131      WCKHD    = 153     ;WRITE CHECK HEADER & DATA
294      000151      WRTDAT   = 161     ;WRITE DATA
295      000161      FMTRK    = 163     ;FORMAT TRACK
296      000163      RDDAT    = 171     ;READ DATA
297      000171      RDHD     = 173     ;READ HEADER & DATA
298
299
300
301
302
303
304
305
306      176700      ABASE    = 176700
307      000254      AVECT1   = 254
308

```



```
1          .SBTTL TRAP CATCHER
          000000          .=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
000174 000174          .=174
000174 000000          DISPREG: .WORD 0          ;;SOFTWARE DISPLAY REGISTER
000176 000000          SWREG:   .WORD 0          ;;SOFTWARE SWITCH REGISTER

          .SBTTL STARTING ADDRESS(ES)
000200 000137 003532          JMP      @#START1          ;;JUMP TO STARTING ADDRESS OF PROGRAM
2
3
7
8          000204 000137 003522          JMP      @#START          ;CHANGE THE RHXX/RP07 ADDRESS

          .SBTTL ACT11 HOOKS
          ;*****
          ;HOOKS REQUIRED BY ACT11
          000210          $SVPC=.          ;SAVE PC
          000046          000046          .=46
          000046 031754          $ENDAD          ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
          000052          000052          .=52
          000052 040000          .WORD 40000          ;;2)SET LOC.52 TO 40000
          000210          .= $SVPC          ;; RESTORE PC

9
10          001100          .=1100
11          .SBTTL APT PARAMETER BLOCK
          ;*****
          ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
          ;*****
          000024          001100          .$X=.          ;;SAVE CURRENT LOCATION
          000024          000024          .=24          ;;SET POWER FAIL TO POINT TO START OF PROGRAM
          000044          000200          200          ;;FOR APT START UP
          000044          000044          .=44          ;;POINT TO APT INDIRECT ADDRESS PNTR.
          000044          001100          $APTHDR          ;;POINT TO APT HEADER BLOCK
          001100          .=.$X          ;;RESET LOCATION COUNTER
          ;*****
          ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
          ;INTERFACE SPEC.

001100          $APTHD:
001100 000000          $HIBTS: .WORD 0          ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
001102 001206          $MBADR: .WORD $MAIL          ;;ADDRESS OF APT MAILBOX (BITS 0-15)
001104 014234          $TSTM:  .WORD 6300.          ;;RUN TIM OF LONGEST TEST
001106 014234          $PASTM: .WORD 6300.          ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
001110 014234          $UNITM: .WORD 6300.          ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDED UNIT
001112 000032          .WORD $ETEND-$MAIL/2          ;;LENGTH MAILBOX-ETABLE(WORDS)
12          001114          TAB.XY=.          ;CMTAGSTARING ADDRESS
13
```

0

.SBTTL COMMON TAGS

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

001114	001114			\$CMTAG:	.=TAB.XY		::START OF COMMON TAGS
001114	000000				.WORD	0	
001116	000			\$TSTNM:	.BYTE	0	::CONTAINS THE TEST NUMBER
001117	000			\$ERFLG:	.BYTE	0	::CONTAINS ERROR FLAG
001120	000000			\$ICNT:	.WORD	0	::CONTAINS SUBTEST ITERATION COUNT
001122	000000			\$LPADR:	.WORD	0	::CONTAINS SCOPE LOOP ADDRESS
001124	000000			\$LPERR:	.WORD	0	::CONTAINS SCOPE RETURN FOR ERRORS
001126	000000			\$ERTTL:	.WORD	0	::CONTAINS TOTAL ERRORS DETECTED
001130	000			\$ITEMB:	.BYTE	0	::CONTAINS ITEM CONTROL BYTE
001131	001			\$ERMAX:	.BYTE	1	::CONTAINS MAX. ERRORS PER TEST
001132	000000			\$ERRPC:	.WORD	0	::CONTAINS PC OF LAST ERROR INSTRUCTION
001134	000000			\$GDADR:	.WORD	0	::CONTAINS ADDRESS OF 'GOOD' DATA
001136	000000			\$BDADR:	.WORD	0	::CONTAINS ADDRESS OF 'BAD' DATA
001140	000000			\$GDDAT:	.WORD	0	::CONTAINS 'GOOD' DATA
001142	000000			\$BDDAT:	.WORD	0	::CONTAINS 'BAD' DATA
001144	000000				.WORD	0	::RESERVED--NOT TO BE USED
001146	000000				.WORD	0	
001150	000			\$AUTOB:	.BYTE	0	::AUTOMATIC MODE INDICATOR
001151	000			\$INTAG:	.BYTE	0	::INTERRUPT MODE INDICATOR
001152	000000				.WORD	0	
001154	177570			\$WR:	.WORD	DSWR	::ADDRESS OF SWITCH REGISTER
001156	177570			\$DISPLAY:	.WORD	DDISP	::ADDRESS OF DISPLAY REGISTER
001160	177560			\$TKS:	177560		::TTY KBD STATUS
001162	177562			\$TKB:	177562		::TTY KBD BUFFER
001164	177564			\$TPS:	177564		::TTY PRINTER STATUS REG. ADDRESS
001166	177566			\$TPB:	177566		::TTY PRINTER BUFFER REG. ADDRESS
001170	000			\$NULL:	.BYTE	0	::CONTAINS NULL CHARACTER FOR FILLS
001171	002			\$FILLS:	.BYTE	2	::CONTAINS # OF FILLER CHARACTERS REQUIRED
001172	012			\$FILLC:	.BYTE	12	::INSERT FILL CHARS. AFTER A 'LINE FEED'
001173	000			\$TP/LG:	.BYTE	0	::'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
001174	000000			\$TMPO:	.WORD	0	::USER DEFINED
001176	207	377	377	\$BELL:	.ASCIZ	<207><377><377>	::CODE FOR BELL
001202	077			\$QUES:	.ASCII	/?/	::QUESTION MARK
001203	015			\$CRLF:	.ASCII	<15>	::CARRIAGE RETURN
001204	012	000		\$LF:	.ASCIZ	<12>	::LINE FEED

.SBTTL APT MAILBOX-ETABLE

001206				.EVEN			
001206	000000			\$MAIL:			::APT MAILBOX
001210	000000			\$MSGTY:	.WORD	AMSGTY	::MESSAGE TYPE CODE
001212	000000			\$FATAL:	.WORD	AFATAL	::FATAL ERROR NUMBER
001214	000000			\$TESTN:	.WORD	ATESTN	::TEST NUMBER
001216	000000			\$PASS:	.WORD	APASS	::PASS COUNT
001220	000000			\$DEVCT:	.WORD	ADEVCT	::DEVICE COUNT
001222	000000			\$UNIT:	.WORD	AUNIT	::I/O UNIT NUMBER
001224	000000			\$MSGAD:	.WORD	AMSGAD	::MESSAGE ADDRESS
001226	000000			\$MSGLG:	.WORD	AMSGLG	::MESSAGE LENGTH
				\$ETABLE:			::APT ENVIRONMENT TABLE

001226	000	\$ENV:	.BYTE	AENV	::ENVIRONMENT BYTE
001227	000	\$ENVM:	.BYTE	AENVM	::ENVIRONMENT MODE BITS
001230	000000	\$SWREG:	.WORD	ASWREG	::APT SWITCH REGISTER
001232	000000	\$USWR:	.WORD	AUSWR	::USER SWITCHES
001234	000000	\$CPUOP:	.WORD	ACPUOP	::CPU TYPE,OPTIONS
		.*			BITS 15-11=CPU TYPE
		.*			11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
		.*			11/70=06,PDQ=07,Q=10
		.*			BIT 10=REAL TIME CLOCK
		.*			BIT 9=FLOATING POINT PROCESSOR
		.*			BIT 8=MEMORY MANAGEMENT
001236	000	\$MAMS1:	.BYTE	AMAMS1	::HIGH ADDRESS,M.S. BYTE
001237	000	\$MTYP1:	.BYTE	AMTYP1	::MEM. TYPE,BLK#1
		.*			MEM. TYPE BYTE -- (HIGH BYTE)
		.*			900 NSEC CORE=001
		.*			300 NSEC BIPOLAR=002
		.*			500 NSEC MOS=003
001240	000000	\$MADR1:	.WORD	AMADR1	::HIGH ADDRESS,BLK#1
		.*			MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
001242	000	\$MAMS2:	.BYTE	AMAMS2	::HIGH ADDRESS,M.S. BYTE
001243	000	\$MTYP2:	.BYTE	AMTYP2	::MEM. TYPE,BLK#2
001244	000000	\$MADR2:	.WORD	AMADR2	::MEM.LAST ADDRESS,BLK#2
001246	000	\$MAMS3:	.BYTE	AMAMS3	::HIGH ADDRESS,M.S.BYTE
001247	000	\$MTYP3:	.BYTE	AMTYP3	::MEM. TYPE,BLK#3
001250	000000	\$MADR3:	.WORD	AMADR3	::MEM.LAST ADDRESS,BLK#3
001252	000	\$MAMS4:	.BYTE	AMAMS4	::HIGH ADDRESS,M.S.BYTE
001253	000	\$MTYP4:	.BYTE	AMTYP4	::MEM. TYPE,BLK#4
001254	000000	\$MADR4:	.WORD	AMADR4	::MEM.LAST ADDRESS,BLK#4
001256	000254	\$VECT1:	.WORD	AVECT1	::INTERRUPT VECTOR#1,BUS PRIORITY#1
001260	000000	\$VECT2:	.WORD	AVECT2	::INTERRUPT VECTOR#2BUS PRIORITY#2
001262	176700	\$BASE:	.WORD	ABASE	::BASE ADDRESS OF EQUIPMENT UNDER TEST
001264	000000	\$DEVN:	.WORD	ADEVN	::DEVICE MAP
001266	000000	\$CDW1:	.WORD	ACDW1	::CONTROLLER DESCRIPTION WORD#1
001270	000000	\$CDW2:	.WORD	ACDW2	::CONTROLLER DESCRIPTION WORD#2
001272		\$ETEND:			
		.MEXIT			

.SBTTL USER DEFINED TAGS

001272	176700	\$RPADR:	.WORD	176700	:FIRST ADDRESS OF RHXX/RPO7 REGISTERS
001274	000254	\$RPVEC:	.WORD	254	:RPO7 VECTOR ADDRESS
001276	172540	\$LKCSR:	.WORD	172540	:ADDR OF KW11-P STATUS REGISTER
001300	172542	\$LKCSB:	.WORD	172542	:ADDR OF KW11-P COUNTER BUFFER
001302	000104	\$LPVEC:	.WORD	104	:ADDR OF KW11-P VECTOR
001304	177546	\$LKS:	.WORD	177546	:ADDR OF KW11-L STATUS REGISTER
001306	000100	\$LLVEC:	.WORD	100	:ADDR OF KW11-L VECTOR
001310	000000	CLKFLG:	.WORD	0	:NO CLOCK= 0, P-CLOCK= 1 OR L-CLOCK= -1
001312	000074	HERTZ:	.WORD	60.	:60HZ SYSTEM= 60. OR 50HZ SYSTEM= 50.
001314	000000	STATIN:	.WORD	0	: 'TYPE STATISTICS' INDICATOR
	001220	DRIVE	= \$UNIT		:DRIVE # STORAGE: ERRORS 1-5 & 10
					:SAME AS USED IN APT
001316	000000	ATTN:	.WORD	0	:ATTN REG STORAGE: ERRORS 1-5 & 10
001320	000000	DRVNO:	.WORD	0	:DRIVE # STORAGE FOR PRINTOUT
001322	000000	MASK:	.WORD	0	:ERROR RETRY REGISTER MASK
001324	000	RETRY:	.BYTE	0,0	:ERROR RETRY LIMIT IN THE LOWER BYTE
					:RETRY COUNT IN THE UPPER BYTE
001326	000003	FAIRNS:	.WORD	3	:MAXIMUM TIME IN QUEUE VALUE
001330	000000	LSTAD:	.WORD	0	:STORE LAST MEMORY ADDRESS HERE
001332	000000	CHGADR:	.WORD	0	:CHANGE RHXX/RPO7 UNIBUS ADDRESS FLAG
001334	000000	CFLAG:	.WORD	0	: 'CONTROL C' FLAG
001336	000000	BADSEC:	.WORD	0	:BAD TRACK/SECTOR FLAG
001340	000000	HOUR:	.WORD	0	:HOUR COUNT STORED HERE
001342	000000	MINUTE:	.WORD	0	:MINUTE'S COUNT STORED HERE
001344	000000	SECOND:	.WORD	0	:SECOND'S COUNT STORED HERE
001346	000000	ONESEC:	.WORD	0	:TIMER ROUTINE COUNTER (FOR ONE SECOND)
001350	177777	ZROIND:	.WORD	-1	:ZERO INDICATOR FOR THE DATA COMPARE ROUTINE
001352	000	FRSTER:	.BYTE	0	:DATA COMPARE ERROR FLAG
					:IF > 0, PROCESSING 'DCKER' OR CAN'T MATCH PATTERN
					:IF < 0, MISCOMPARSION FOUND
001353	000		.BYTE	0	:MISCOMPARSION OR CAN'T MATCH PATTERN FLAG
					:IF < 0, ERROR IN BUFFER
001354	000000	SAVPOS:	.WORD	0	:SAVE WORD POSITION IN COMPARE ROUTINE
001356	000000	SAVER1:	.WORD	0	:SAVE R1 HERE
001360	000000	SAVER5:	.WORD	0	:SAVE P5 HERE
001362	000000	ERCTR:	.WORD	0	:NUMBER OF ERRORS
001364	000000	LIMIT:	.WORD	0	:DISPLAY LIMIT
001366	000000	CMCNT:	.WORD	0	:WORD COUNT
001370	000000	CMCYL:	.WORD	0	:CYLINDER ADDRESS
001372	000	CMSEC:	.BYTE	0	:SECTOR ADDRESS
001374	000	CMTRK:	.BYTE	0	:TRACK ADDRESS
001376	000000	WRDPOS:	.WORD	0	:WORD POSITION IN COMPARE ROUTINE
001400	000000	ECBIT:	.WORD	0	:ERROR BURST BIT OFFSET
001402	000000	ECSEC:	.WORD	0	:ERROR BURST WORD OFFSET (RELATIVE TO SECTOR)
001404	000000	ECMSK0:	.WORD	0	:CORRECTION MASK FOR FIRST ERROR WORD
001406	000000	ECMSK1:	.WORD	0	:CORRECTION MASK FOR SECOND ERROR WORD
001410	000000	ECWRD:	.WORD	0	:LOCATION OF FIRST ERROR WORD
001412	000000	ECGD:	.WORD	0	:GOOD (CORRECTED) DATA, FIRST WORD
001414	000000	ECBAD0:	.WORD	0	:BAD (RECEIVED) DATA, FIRST WORD
001416	000000	ECWRD1:	.WORD	0	:LOCATION OF SECOND ERROR WORD
001420	000000	ECGD1:	.WORD	0	:GOOD (CORRECTED) DATA, SECOND WORD
001422	000000	ECBAD1:	.WORD	0	:BAD (RECEIVED) DATA, SECOND WORD
001424	000000	TSTANY:	.WORD	0	:TEST ON FE CYLINDER ONLY =0, TEST ANYWHERE ON MEDIA 1
		RONLY:	.WORD	0	:READ/WRITE MODE=0, READ MODE ONLY=1

```

001426 000000      DRVPAR: .WORD 0      ;WHEN DRIVES ARE BEING ASSIGNED,
                                ;0=CHANGE DRIVE PARAMETERS
001430 000000      XXDP:   .WORD 0      ;1=DO NOT CHANGE DRIVE PARAMETERS
                                ;THE LOW BYTE CONTAINS THE DRIVE NUMBER FROM WHICH
                                ;THE PROGRAM WAS LOADED. THE HIGH BYTE CONTAINS THE
                                ;'XXDP' DEVICE CODE FOR THE RP07.

001432 200          .LIST  BEX
001433 103          REV:   .ASCII <CRLF>
001436 112          132   122      .ASCII /CZRJO/ ;PROGRAM ID
001440 055          .ASCII /-/
001441 101          .ASCII /A/      ;REVISION LEVEL
001442 055          .ASCII /-/
001443 060          000      .ASCIIZ /0/      ;PATCH LEVEL
                                .NLIST BEX
                                .EVEN

                                .SBTTL COMMON PARAMETERS
                                ;THE FOLLOWING TWO LOCATIONS CONTAIN THE SOFT ERROR RATE WORDS USED TO
                                ;DETERMINE END OF PASS WHEN THE PROGRAM IS DATA BIASED.
                                ;IT WILL TAKE APPROXIMATELY 2.4 PASSES TO REACH THE SOFT ERROR RATE OF
                                ;1 ERROR IN 10^10 BITS (6.25 X 10^8 WORDS) READ OR 10. PASSES TO REACH THE 90%
                                ;CONFIDENCE LEVEL OF 4.128 X 10^10 BITS (2.58 X 10^9 WORDS) READ.
                                ;ENDCON= LSB AND ENDCON+2= MSB

001446 142200      ENDCON: .WORD 142200 ;(2.58 X 10^8 WORDS) OR (4.128 X 10^9 BITS) READ
001450 007540      .WORD 007540

                                ;THE FOLLOWING TWO LOCATIONS CONTAIN THE SEEK ERROR RATE WORDS USED TO
                                ;DETERMINE END OF PASS WHEN THE PROGRAM IS SEEK BIASED.
                                ;IT WILL TAKE 1 PASS TO REACH A SEEK ERROR RATE OF 1 ERROR IN 10^6 SEEKS OR 3
                                ;PASSES TO REACH A 90% CONFIDENCE LEVEL OF 3 X 10^6 SEEKS.
                                ;ENDSEK= LSB AND ENDSEK+2= MSB

001452 041100      ENDSEK: .WORD 041100 ;(1 X 10^6 SEEKS)
001454 000017      .WORD 000017

001456 000062      MAXER:  .WORD 50.      ;MAXIMUM ERRORS ALLOWED PER DRIVE
001460 000004      CMPLMT: .WORD 4        ;NUMBER OF COMPARE ERRORS TYPED OUT
001462 000017 000017 CMPTIM: .WORD 15.,15. ;FIRST WORD IS THE INTERVAL BETWEEN DATA COMPARES.
                                ;(IN MINUTES). SECOND WORD IS THE INTERVAL COUNTER.
                                ;IF FIRST WORD IS ZERO, THEN DATA COMPARE IS ALWAYS ON.
                                ;THIS TIMER HAS NO AFFECT IF SW01=1.

001466 03100C      WRDCNT: .WORD 12800. ;MAXIMUM WORD COUNT (6-12800.)
001470 000000 000000 INTRVL: .WORD 0,0 ;FIRST WORD IS THE PERFORMANCE TYPEOUT INTERVAL
                                ;(IN MINUTES). SECOND WORD IS THE INTERVAL COUNTER.
                                ;IF FIRST WORD IS ZERO, NO REPORT IS TYPED.

001474 000001      PASSES: .WORD 1      ;NUMBER OF PASSES TO END OF TEST [THIS PARAMETER IS
                                ;NOT USED WHEN PROGRAM IS OPERATING IN AUTO RUN(CHAIN)
                                ;MODE].

001476 000000      PATTERN: .WORD 0      ;IF EQ 0, RANDOMLY SELECT DATA PATTERN
                                ;IF NOT EQ 0, SELECT ONE SET OF PATTERN
                                ;POINTED BY THE 'PATTERN'.

001500 000000      RANDWC: .WORD 0      ;IF EQ TO 0, GENERATE A RANDOM WORD COUNT
                                ;FOR THE OPERATION.
                                ;IF NOT EQ TO 0, USE THE VALUE IN 'WRDCNT' FOR
  
```

```

001502 000003          RATIO: .WORD 3          ;THE WORD COUNT
;READ/WRITE RATIO [RANGE 0 - 7]
;0 - 15/1          (READ/WRITE)
;1 - 7/1
;2 - 6/2
;3 - 5/3
;4 - 4/4
;5 - 3/5
;6 - 2/6
;7 - 1/7
001504 000001          ENDING: .WORD 1          ;IF NOT EQ 0, END OF PASS DETERMINED
;BY THE 'WORDS READ' COUNT. (2.58 X 10^8 WORDS)
;IF EQ 0, END OF PASS DETERMINED
;BY THE SEEK COUNT. (1 X 10^6 SEEKS)
001506 000001          WRTCHK: .WORD 1          ;IF NOT EQ 0, DO AN APPROPRIATE WRITE
;CHECK AFTER EACH WRITE COMMAND.
;IF EQ 0, SELECT WRITE CHECK COMMANDS
;RANDOMLY.
001510 000000          RANDOM: .WORD 0          ;IF EQ TO 0, RANDOMLY SELECT DATA BLOCK
;ADDRESS. IF NOT EQU 0, SEQUENTIALLY
;SELECT DATA BLOCK ADDRESS

.SBTTL VALUES FOR FIRST OPERATION
001512 000010          BEGPAT: .WORD 8.          ;STARTING PATTERN CODE [RANGE 1 - 15.]
001514 000004          BEGCOD: .WORD 4          ;STARTING COMMAND CODE [RANGE 0 - 5]
;0 = WRITE CHECK DATA ('WCKD')
;1 = WRITE CHECK HEADER & DATA ('WCKHD' - NOT USED)
;2 = WRITE DATA ('WRDAT')
;3 = FORMAT TRACK ('FMTRK' - NOT USED)
;4 = READ DATA ('RDDAT')
;5 = READ HEADER & DATA ('RDHD')
001516 000400          BEGWC: .WORD 256.          ;STARTING WRD CNT [RANGE 6 - WRDCNT]

.SBTTL TABLES, CONSTANTS, AND VARIABLE LOCATIONS
;LIST OF DRIVES PERFORMING COMMANDS
001520 000000          ORDERQ: .WORD 0
001522 000000          .WORD 0
001524 000000          .WORD 0
001526 000000          .WORD 0
001530 000000          .WORD 0
001532 000000          .WORD 0
001534 000000          .WORD 0
001536 000000          .WORD 0
001540 000000          .WORD 0
001542 000000          ASNLST: .WORD 0          ;A BIT SET IS AN ASSIGNED DRIVE

;ADDRESSES OF DRIVES TO BE DROPPED
001544 000000          DDRVS: .WORD 0
001546 000000          .WORD 0
001550 000000          .WORD 0
001552 000000          .WORD 0
001554 000000          .WORD 0
001556 000000          .WORD 0
001560 000000          .WORD 0

```



```

001766 0C0000 000000
001772 000000 000000
001776 000000 000000
002002 000000 000000
002006 000000 000000
002012 000000 000000
002016 000000 000000
002022 000000 000000
002026 000000 000000
002032 000000 000000
002036 000000 000000
002042 000000 000000
002046 000000 000000
002052 000000 000000
    
```

```

.WORD 0.0
.WORD 0.0
.WORD 0.0
.WORD 0.0
.WORD 0.0
.WORD 0.0
.WORD 0.0
.WORD 0.0
.WORD 0.0
.WORD 0.0
.WORD 0.0
.WORD 0.0
.WORD 0.0
.WORD 0.0
.WORD 0.0
    
```

:DRIVE PARAMETER BLOCK(DPB) POINTER TABLE

```

002056 046234
002060 046476
002062 046740
002064 047202
002066 047444
002070 047706
002072 050150
002074 050412
    
```

```

BLKADR: .WORD DRIVE0 :ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 0
        .WORD DRIVE1 :ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 1
        .WORD DRIVE2 :ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 2
        .WORD DRIVE3 :ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 3
        .WORD DRIVE4 :ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 4
        .WORD DRIVE5 :ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 5
        .WORD DRIVE6 :ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 6
        .WORD DRIVE7 :ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 7
    
```

:DRIVER COMMAND CONTROL TABLE (USED IN RP DRIVER)

```

002076 151
002077 377
002100 161
002101 377
002102 171
002103 173
    
```

```

COM'BL: .BYTE WCKD :WRITE CHECK DATA
        .BYTE -1 :WRITE CHECK HEADER AND DATA (NOT USED)
        .BYTE WRDAT :WRITE DATA
        .BYTE -1 :FORMAT TRACK (NOT USED)
        .BYTE RDDAT :READ DATA
        .BYTE RDHD :READ HEADER AND DATA
    
```

:FUNCTION(COMMAND) CODE CONTROL TABLE

```

002104 004
002105 006
002106 010
002107 012
002110 016
002111 020
002112 022
002113 030
002114 050
002115 052
002116 060
002117 062
002120 070
002121 072
002122 377
    
```

```

OPTBL: .BYTE 4 :SEEK
        .BYTE 6 :RECAL
        .BYTE 10 :DRIVE CLEAR
        .BYTE 12 :RELEASE
        .BYTE 16 :RETURN TO CENTERLINE
        .BYTE 20 :READIN PRESET
        .BYTE 22 :PACK ACKNOWLEDGE
        .BYTE 30 :SEARCH
        .BYTE 50 :WRITE CHECK DATA
        .BYTE 52 :WRITE CHECK HEADER AND DATA
        .BYTE 60 :WRITE DATA
        .BYTE 62 :FORMAT TRACK
        .BYTE 70 :READ DATA
        .BYTE 72 :READ HEADER AND DATA
        .BYTE -1 :TERMINATOR
    
```

.EVEN

:MESSAGE CONTROL TABLE FOR 'OPTBL' TABLE

```

002124 123 105 105
002134 122 105 103
002144 104 122 126
002154 122 105 114
002164 122 124 103
    
```

```

MNTBL: .ASCIZ /SEEK /
        .ASCIZ /RECAL /
        .ASCIZ /DRVCLR /
        .ASCIZ /RELSE /
        .ASCIZ /RTC /
    
```

002174	122	105	101	.ASCIZ	/READIN /
002204	120	101	103	.ASCIZ	/PACK /
002214	123	105	101	.ASCIZ	/SEARCH /
002224	127	103	113	.ASCIZ	/WCKD /
002234	127	103	113	.ASCIZ	/WCKHD /
002244	127	122	124	.ASCIZ	/WRDAT /
002254	106	115	124	.ASCIZ	/FMTRK /
002264	122	104	104	.ASCIZ	/RDDAT /
002274	122	104	110	.ASCIZ	/RDHD /
002304	116	117	116	.ASCIZ	/NONE /

```

;STANDARD DATA PATTERN POINTER TABLE
STNDAT: .WORD DATA0 ;STANDARD DATA PATTERN 0
        .WORD DATA1 ;STANDARD DATA PATTERN 1
        .WORD DATA2 ;STANDARD DATA PATTERN 2
        .WORD DATA3 ;STANDARD DATA PATTERN 3
        .WORD DATA4 ;STANDARD DATA PATTERN 4
        .WORD DATA5 ;STANDARD DATA PATTERN 5
        .WORD DATA6 ;STANDARD DATA PATTERN 6
        .WORD DATA7 ;STANDARD DATA PATTERN 7
        .WORD DATA8 ;STANDARD DATA PATTERN 8
        .WORD DATA9 ;STANDARD DATA PATTERN 9
        .WORD DATA10 ;STANDARD DATA PATTERN 10
        .WORD DATA11 ;STANDARD DATA PATTERN 11
        .WORD DATA12 ;STANDARD DATA PATTERN 12
        .WORD DATA13 ;STANDARD DATA PATTERN 13
        .WORD DATA14 ;STANDARD DATA PATTERN 14
        .WORD DATA15 ;STANDARD DATA PATTERN 15
        .WORD ZEROS ;ALL 0'S PATTERN
        .WORD ONES ;ALL 1'S PATTERN

002314 002360
002316 002420
002320 002460
002322 002520
002324 002560
002326 002620
002330 002660
002332 002720
002334 002760
002336 003020
002340 003060
002342 003120
002344 003160
002346 003220
002350 003260
002352 003320
002354 003360
002356 003262

002360
002360 000000
002362 000000
002364 000000
002366 000000
002370 000000
002372 000000
002374 000000
002376 000000
002400 000000
002402 000000
002404 000000
002406 000000
002410 000000
002412 000000
002414 000000
002416 000000

ZEROS:
DATA0: .WORD 0 ;ALL 0'S 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

002420 000001
002422 000003
002424 000007
002426 000017
002430 000037
002432 000077
002434 000177
002436 000377
002440 000777
002442 001777
002444 003777
002446 007777
002450 017777
002452 037777
002454 077777
002456 177777

DATA1: .WORD 000001 ;STANDARD PATTERN 1
        .WORD 000003
        .WORD 000007
        .WORD 000017
        .WORD 000037
        .WORD 000077
        .WORD 000177
        .WORD 000377
        .WORD 000777
        .WORD 001777
        .WORD 003777
        .WORD 007777
        .WORD 017777
        .WORD 037777
        .WORD 077777
        .WORD 177777

002460 177776
002462 177774

DATA2: .WORD 177776 ;STANDARD PATTERN 2
        .WORD 177774
    
```


002464	177770	.WORD	177770
002466	177760	.WORD	177760
002470	177740	.WORD	177740
002472	177700	.WORD	177700
002474	177600	.WORD	177600
002476	177400	.WORD	177400
002500	177000	.WORD	177000
002502	176000	.WORD	176000
002504	174000	.WORD	174000
002506	170000	.WORD	170000
002510	160000	.WORD	160000
002512	140000	.WORD	140000
002514	100000	.WORD	100000
002516	000000	.WORD	000000

002520	000000	DATA3: .WORD	000000	; STANDARD PATTERN 3
002522	000000	.WORD	000000	
002524	000000	.WORD	000000	
002526	177777	.WORD	177777	
002530	177777	.WORD	177777	
002532	177777	.WORD	177777	
002534	000000	.WORD	000000	
002536	000000	.WORD	000000	
002540	177777	.WORD	177777	
002542	177777	.WORD	177777	
002544	000000	.WORD	000000	
002546	177777	.WORD	177777	
002550	000000	.WORD	000000	
002552	177777	.WORD	177777	
002554	000000	.WORD	000000	
002556	177777	.WORD	177777	

002560	133331	DATA4: .WORD	133331	; STANDARD PATTERN 4
002562	133331	.WORD	133331	
002564	133331	.WORD	133331	
002566	133331	.WORD	133331	
002570	133331	.WORD	133331	
002572	133331	.WORD	133331	
002574	133331	.WORD	133331	
002576	133331	.WORD	133331	
002600	133331	.WORD	133331	
002602	133331	.WORD	133331	
002604	133331	.WORD	133331	
002606	133331	.WORD	133331	
002610	133331	.WORD	133331	
002612	133331	.WORD	133331	
002614	133331	.WORD	133331	
002616	133331	.WORD	133331	

002620	052525	DATA5: .WORD	052525	; STANDARD PATTERN 5
002622	052525	.WORD	052525	
002624	052525	.WORD	052525	
002626	125252	.WORD	125252	
002630	125252	.WORD	125252	
002632	125252	.WORD	125252	
002634	052525	.WORD	052525	
002636	052525	.WORD	052525	

002640	125252	.WORD	125252
002642	125252	.WORD	125252
002644	052525	.WORD	052525
002646	125252	.WORD	125252
002650	052525	.WORD	052525
002652	125252	.WORD	125252
002654	052525	.WORD	052525
002656	125252	.WORD	125252

002660	147556	DATA6: .WORD	147556	;STANDARD PATTERN 6 (NOT WORST CASE)
002662	147556	.WORD	147556	
002664	147556	.WORD	147556	
002666	147556	.WORD	147556	
002670	147556	.WORD	147556	
002672	147556	.WORD	147556	
002674	147556	.WORD	147556	
002676	147556	.WORD	147556	
002700	147556	.WORD	147556	
002702	147556	.WORD	147556	
002704	147556	.WORD	147556	
002706	147556	.WORD	147556	
002710	147556	.WORD	147556	
002712	147556	.WORD	147556	
002714	147556	.WORD	147556	
002716	147556	.WORD	147556	

002720	155555	DATA7: .WORD	155555	;STANDARD PATTERN 7
002722	133333	.WORD	133333	
002724	155555	.WORD	155555	
002726	133333	.WORD	133333	
002730	155555	.WORD	155555	
002732	133333	.WORD	133333	
002734	155555	.WORD	155555	
002736	133333	.WORD	133333	
002740	155555	.WORD	155555	
002742	133333	.WORD	133333	
002744	155555	.WORD	155555	
002746	133333	.WORD	133333	
002750	155555	.WORD	155555	
002752	133333	.WORD	133333	
002754	155555	.WORD	155555	
002756	133333	.WORD	133333	

002760	030221	DATA8: .WORD	030221	;STANDARD PATTERN 8 (WORST CASE)
002762	030221	.WORD	030221	
002764	030221	.WORD	030221	
002766	030221	.WORD	030221	
002770	030221	.WORD	030221	
002772	030221	.WORD	030221	
002774	030221	.WORD	030221	
002776	030221	.WORD	030221	
003000	030221	.WORD	030221	
003002	030221	.WORD	030221	
003004	030221	.WORD	030221	
003006	030221	.WORD	030221	
003010	030221	.WORD	030221	
003012	030221	.WORD	030221	

003014	030221	.WORD	030221	
003016	030221	.WORD	030221	
003020	000001	DATA9: .WORD	000001	;STANDARD PATTERN 9
003022	000002	.WORD	000002	
003024	000004	.WORD	000004	
003026	000010	.WORD	000010	
003030	000020	.WORD	000020	
003032	000040	.WORD	000040	
003034	000100	.WORD	000100	
003036	000200	.WORD	000200	
003040	000400	.WORD	000400	
003042	001000	.WORD	001000	
003044	002000	.WORD	002000	
003046	004000	.WORD	004000	
003050	010000	.WORD	010000	
003052	020000	.WORD	020000	
003054	040000	.WORD	040000	
003056	100000	.WORD	100000	
003060	177776	DATA10: .WORD	177776	;STANDARD PATTERN 10
003062	177775	.WORD	177775	
003064	177773	.WORD	177773	
003066	177767	.WORD	177767	
003070	177757	.WORD	177757	
003072	177737	.WORD	177737	
003074	177677	.WORD	177677	
003076	177577	.WORD	177577	
003100	177377	.WORD	177377	
003102	176777	.WORD	176777	
003104	175777	.WORD	175777	
003106	173777	.WORD	173777	
003110	167777	.WORD	167777	
003112	157777	.WORD	157777	
003114	137777	.WORD	137777	
003116	077777	.WORD	077777	
003120	172666	DATA11: .WORD	172666	;STANDARD PATTERN 11
003122	155555	.WORD	155555	
003124	172666	.WORD	172666	
003126	155555	.WORD	155555	
003130	172666	.WORD	172666	
003132	155555	.WORD	155555	
003134	172666	.WORD	172666	
003136	155555	.WORD	155555	
003140	172666	.WORD	172666	
003142	155555	.WORD	155555	
003144	172666	.WORD	172666	
003146	155555	.WORD	155555	
003150	172666	.WORD	172666	
003152	155555	.WORD	155555	
003154	172666	.WORD	172666	
003156	155555	.WORD	155555	
003160	077777	DATA12: .WORD	077777	;STANDARD PATTERN 12
003162	137777	.WORD	137777	
003164	157777	.WORD	157777	

003166	167777	.WORD	167777
003170	173777	.WORD	173777
003172	175777	.WORD	175777
003174	176777	.WORD	176777
003176	177377	.WORD	177377
003200	177577	.WORD	177577
003202	177677	.WORD	177677
003204	177737	.WORD	177737
003206	177757	.WORD	177757
003210	177767	.WORD	177767
003212	177773	.WORD	177773
003214	177775	.WORD	177775
003216	177776	.WORD	177776

003220	153333	DATA13: .WORD	153333	;STANDARD PATTERN 13
003222	066667	.WORD	066667	
003224	153333	.WORD	153333	
003226	066667	.WORD	066667	
003230	153333	.WORD	153333	
003232	066667	.WORD	066667	
003234	153333	.WORD	153333	
003236	066667	.WORD	066667	
003240	153333	.WORD	153333	
003242	066667	.WORD	066667	
003244	153333	.WORD	153333	
003246	066667	.WORD	066667	
003250	153333	.WORD	153333	
003252	066667	.WORD	066667	
003254	153333	.WORD	153333	
003256	066667	.WORD	066667	

003260	000000	DATA14: .WORD	000000	;STANDARD PATTERN 14
003262	177777	ONES: .WORD	177777	;ALL 1'S DATA PATTERN
003264	177777	.WORD	177777	
003266	177777	.WORD	177777	
003270	177777	.WORD	177777	
003272	177777	.WORD	177777	
003274	177777	.WORD	177777	
003276	177777	.WORD	177777	
003300	177777	.WORD	177777	
003302	177777	.WORD	177777	
003304	177777	.WORD	177777	
003306	177777	.WORD	177777	
003310	177777	.WORD	177777	
003312	177777	.WORD	177777	
003314	177777	.WORD	177777	
003316	177777	.WORD	177777	

003320	177777	DATA15: .WORD	177777	;STANDARD PATTERN 15
003322	000000	.WORD	000000	
003324	000000	.WORD	000000	
003326	000000	.WORD	000000	
003330	000000	.WORD	000000	
003332	000000	.WORD	000000	
003334	000000	.WORD	000000	
003336	000000	.WORD	000000	
003340	000000	.WORD	000000	

003342	000000	.WORD	000000
003344	000000	.WORD	000000
003346	000000	.WORD	000000
003350	000000	.WORD	000000
003352	000000	.WORD	000000
003354	000000	.WORD	000000
003356	000000	.WORD	000000

.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

1	003360				
2					
3					
4	003360	050744	EM1		:RH CONTROLLER INTERRUPT OCCURRED (RPAS = 0)
5	003362	053751	DH1		
6	003364	054424	DT1		
7	003366	000000	0		
8					
9					
10					
11	003370	051017	EM2		:UNEXPECTED ATTENTION OCCURRED
12	003372	053756	DH2		
13	003374	054430	DT2		
14	003377	000000	0		
15					
16					
17					
18	003400	051055	EM3		:MASSBUS PARITY ERROR (MCPE=1)
19	003402	054033	DH3		
20	003404	054434	DT3		
21	003406	000000	0		
22					
23					
24					
25	003410	051113	EM4		:MASSBUS PARITY ERROR (PAR=1)
26	003412	054061	DH4		
27	003414	054440	DT4		
28	003416	000000	0		
29					
30					
31					
32	003420	051150	EM5		:NOT USED
33	003422	053756	DH2		
34	003424	054430	DT2		
35	003426	000000	0		
36					
37					
38					
39	003430	051204	EM6		:RHXX DIDN'T RESPOND TO ADDRESSING
40	003432	054120	DH6		
41	003434	054444	DT6		
42	003436	000000	0		

```

1          ;THIS ROUTINE HANDLES UNEXPECTED TIMEOUTS
2
3 003440 011600 BADTMO: MOV      (SP),R0          ;SAVE PC WHERE THE TIME OUT OCCURED
4 003442 005740      TST      -(R0)           ;ADJUST PC -2
5 003444 022626      CMP      (SP)+,(SP)+       ;RESTORE STACK POINTER
6 003446 104401 003454 TYPE      ,65$          ;:TYPE ASCIZ STRING
   003452 000417      BR       64$           ;:GET OVER THE ASCIZ
   ;:65$: .ASCIZ <CRLF><LF>/UNEXPECTED BUS TIMEOUT, PC=/
   64$:
7 003512 010J46      MOV      R0,-(SP)        ;SETUP FOR TYPING OUT PC
8 003514 104402      TYPOC
9 003516 000240      NOP
   ;PUT 'HALT(0)' INSTRUCTION HERE IF YOU WISH
   ;TO STOP ON UNEXPECTED TIMEOUT.
10
11 003520 000404      BR       START1        ;BRANCH TO START1
12
13          .SBTTL  START OF PROGRAM
14
15 003522 012737 177777 001332 START:  MOV      #-1,CHGADR      ;ALLOW RHXX/RP07 ADDRESS TO BE CHANGED
16 003530 000407      BR       START2        ;START THE PROGRAM
17
18 003532 012737 000400 001332 START1: MOV      #400,CHGADR      ;CLEAR RHXX/RP07 ADDRESS CHANGE FLAG
19          ;*****
   003540 000240      TST1:  NOP
   003542 012737 000001 001212      MOV      #1,$TESTN          ;:SET TEST NUMBER IN APT MAIL BOX
20
21 003550 005227 000000      START2: INC      #0           ;TTY LOOP, WAIT FOR INCREMENT
22 003554 001375      BNE     .-4           ;OF WORD
23 003556 000005      RESET          ;CLEAR THE WORLD
24
25          .SBTTL  INITIALIZE THE COMMON TAGS
   ;:CLEAR THE COMMON TAGS ($CMTAG) AREA
   MOV      #$CMTAG,R6          ;:FIRST LOCATION TO BE CLEARED
   CLR     (R6)+              ;:CLEAR MEMORY LOCATION
   CMP     #SWR,R6            ;:DONE?
   BNE    .-6                 ;:LOOP BACK IF NO
   MOV     #STACK,SP          ;:SETUP THE STACK POINTER
   ;:INITIALIZE A FEW VECTORS
   MOV     #$ERROR,@#EMTVEC    ;:EMT VECTOR FOR ERROR ROUTINE
   MOV     #340,@#EMTVEC+2     ;:LEVEL 7
   MOV     #$TRAP,@#TRAPVEC    ;:TRAP VECTOR FOR TRAP CALLS
   MOV     #340,@#TRAPVEC+2    ;:LEVEL 7
   MOV     #$PWRDN,@#PWRVEC    ;:POWER FAILURE VECTOR
   MOV     #340,@#PWRVEC+2     ;:LEVEL 7
   MOV     #176543,$HINUM      ;:PRIME THE RANDOM NUMBER GENERATOR
   MOV     #123456,$LONUM      ;:BOTH HIGH AND LOW WORDS
   ;:SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
   ;:EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
   MOV     @#ERRVEC,-(SP)      ;:SAVE ERROR VECTOR
   MOV     #64$,@#ERRVEC      ;:SET UP ERROR VECTOR
   MOV     #DSWR,SWR           ;:SETUP FOR A HARDWARE SWICH REGISTER
   MOV     #DDISP,DISPLAY      ;:AND A HARDWARE DISPLAY REGISTER
   CMP     #-1,@SWR           ;:TRY TO REFERENCE HARDWARE SWR
   BNE    66$                 ;:BRANCH IF NO TIMEOUT TRAP OCCURRED
   ;:AND THE HARDWARE SWR IS NOT = -1
   BR     65$                 ;:BRANCH IF NO TIMEOUT
   64$:  MOV     #65$,(SP)      ;:SET UP FOR TRAP RETURN
   RTI

```

```

003726 012737 000176 001154 65$: MOV #SWREG,SWR ;;POINT TO SOFTWARE SWR
003734 012737 000174 001156 MOV #DISPREG,DISPLAY
003742 012637 000004 66$: MOV (SP)+,@#ERRVEC ;;RESTORE ERROR VECTOR

003746 005037 001214 CLR $PASS ;;CLEAR PASS COUNT
003752 132737 000200 001227 BITB #APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
003760 001403 BEQ 67$ ;;YES,USE NON-APT SWITCH
003762 012737 001230 001154 MOV #$$SWREG,SWR ;;NO,USE APT SWITCH REGISTER
003770

26 ;:SETUP "TIMEOUT" TRAP VECTOR FOR UNEXPECTED BUS TIMEOUTS
27 003770 012737 003440 000004 MOV #BADTMO,ERRVEC ;;SETUP FOR UNEXPECTED TIMEOUT
28 003776 012737 000300 000006 MOV #PR6,ERRVEC+2 ;;LEVEL 6
29
30 004004 104401 001432 TYPE ,REV ;TYPE PROGRAM ID, REV LEVEL & PATCH LEVEL
31 .SBTTL TYPE PROGRAM NAME
;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
004010 005227 177777 INC #-1 ;;FIRST TIME?
004014 001022 BNE 68$ ;;BRANCH IF NO
004016 104401 004024 TYPE ,69$ ;;TYPE ASCIZ STRING
004022 000417 BR 68$ ;;GET OVER THE ASCIZ
;;69$: .ASCIZ <CRLF>@RP07 PERFORMANCE EXERCISER@<CRLF>
68$:
.SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
004062 005737 000042 TST @#42 ;;ARE WE RUNNING UNDER XXDP/ACT?
004066 001012 BNE 70$ ;;BRANCH IF YES
004070 123727 001226 000001 CMPB $ENV,#1 ;;ARE WE RUNNING UNDER APT?
004076 001406 BEQ 70$ ;;BRANCH IF YES
004100 023727 001154 000176 CMP SWR,#SWREG ;;SOFTWARE SWITCH REG SELECTED?
004106 001005 BNE 71$ ;;BRANCH IF NO
004110 104406 GTSWR ;;GET SOFT-SWR SETTINGS
004112 000403 BR 71$
004114 112737 000001 001150 70$: MOVB #1,$AUTOB ;;SET AUTO-MODE INDICATOR
004122 71$:

32
33 ;:THE FOLLOWING FINDS OUT THE PROGRAM CONTROL MODE:
34 ;:PAPER TAPE (MANUAL), ACT11, XXDP CHAIN OR DUMP
35
36 004122 005037 001430 CLR XXDP ;;CLEAR 'XXDP' LOAD DEVICE STORAGE
37 004126 122737 177777 000041 CMPB #-1,@#41 ;;LOADED FROM AN RP07 ? (UNKNOWN)
38 004134 001121 BNE 3$ ;;BR IF NOT
39 004136 013737 000040 001430 MOV @#40,XXDP ;;GET DEVICE INDICATOR AND NUMBER
40 004144 122737 000007 001430 CMPB #7,XXDP ;;IS IT A VALID NUMBER ?
41 004152 103002 BHIS 1$ ;;YES
42 004154 105037 001430 CLRB XXDP ;;NO, DEFAULT TO DRIVE 0
43 004160 005737 000042 1$: TST @#42 ;;CHAIN MODE OR ACT11 AUTO ACCEPT ?
44 004164 001425 BEQ 2$ ;;BR IF NEITHER
45 004166 104401 004174 TYPE ,73$ ;;TYPE ASCIZ STRING
004172 000417 BR 72$ ;;GET OVER THE ASCIZ
;;73$: .ASCIZ <CRLF>/NOT TESTING DRIVE /
72$:

004220
46 004220 005046 CLR -(SP) ;;CLEAR WORD ON STACK
47 004222 113716 001430 MOVB XXDP,(SP) ;;GET DRIVE ADDRESS
48 004226 104403 TYPOS ;;TYPE THE ADDRESS
49 004230 00 .BYTE 1 ;;ONLY 1 CHARACTER
50 004231 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
51 004232 104401 001203 TYPE ,SCRLF ;;CR-LF
52 004236 000460 BR 3$ ;;GET NUMBER OF DRIVES
    
```


53									
54	004240	005227	177777		2\$:	INC	#-1		:FIRST TIME THRU HERE ?
55	004244	001055				BNE	3\$:NO
56	004246	104401	004254			TYPE	75\$:TYPE ASCIZ STRING
	004252	000410				BR	74\$:GET OVER THE ASCIZ
	004274				75\$:	.ASCIZ	<CRLF>/TO TEST DRIVE /		
	004274	005046			74\$:	CLR	-(SP)		:CLEAR WORD ON STACK
57	004274	113716	001430			MOVB	XXDP,(SP)		:GET DRIVE ADDRESS
58	004276	104403				TYPOS			:TYPE DRIVE ADDRESS
59	004302	001				.BYTE	1		:ONLY 1 CHARACTER
60	004304	000				.BYTE	0		:SUPRESS LEADING ZEROS
61	004305	104401	004314			TYPE	76\$:TYPE ASCIZ STRING
62	004306	000432				BR	3\$:GET OVER THE ASCIZ
	004312				76\$:	.ASCIZ	/, HALT PROGRAM, CLEAR LOC. 40 AND RESTART PROGRAM./<CRLF>		
	004400				3\$:				
66	004400	004737	033430			JSR	PC,\$TKINT		:TURN ON THE KEYBOARD INTERRUPT
67	004404	105737	001226			TSTB	\$ENV		:RUN UNDER APT MODE
68	004410	001415				BEQ	5\$:NO,DO NOT BOTHER
69	004412	105737	001256			TSTB	\$VECT1		:NEW VECTOR ?
70	004416	001403				BEQ	4\$:NOT LOAD IF = 0
71	004420	113737	001256	001274		MOVB	\$VECT1,\$RPVEC		:NEW VECTOR
72	004426	005737	001262		4\$:	TST	\$BASE		:NEW BASE ADDRESS ?
73	004432	001411				BEQ	6\$:NO
74	004434	013737	001262	001272		MOV	\$BASE,\$RPADR		:NEW BASE ADDRESS
75	004442	000405				BR	6\$		
76									
77	004444	105737	001150		5\$:	TSTB	\$AUTOB		:RUNNING IN AUTO MODE ?
78	004450	001002				BNE	6\$:YES
79	004452	004737	062436			JSR	PC,BUSADR		:CHECK RHXX/RP07 BUS ADDRESS
80	004456	013737	001272	040560	6\$:	MOV	\$RPADR,RPADR		:LOAD ADDRESS INTO DRIVER
81	004464	013737	001274	040562		MOV	\$RPVEC,RPVEC		:LOAD VECTOR INTO DRIVER
82	004472	005037	001314			CLR	STATIN		:CLEAR PERFORMANCE SUMMARY TYPEDOUT FLAG
83	004476	012705	001520			MOV	#ORDERQ,R5		:START OF AREA TO CLEAR
84	004502	005025			7\$:	CLR	(R5)+		
85	004504	022705	002056			CMP	#BLKADR,R5		:LOOK FOR END OF CLEAR AREA
86	004510	001374				BNE	7\$:BR IF NOT FINISHED
87	004512	012706	001100			MOV	#STACK,SP		:SETUP THE STACK POINTER
88	004516	005037	177776			CLR	PS		:CLEAR THE PROCESSOR STATUS WORD
89	004522	013737	001312	001346		MOV	HERTZ,ONESEC		:RESTORE ONE SECOND COUNTER VALUE
90	004530	005037	001340			CLR	HOUR		:CLEAR THE HOUR'S COUNTER
91	004534	005037	001342			CLR	MINUTE		:CLEAR THE MINUTE'S COUNTER
92	004540	005037	001344			CLR	SECOND		:CLEAR THE SECOND'S COUNTER
93	004544	005037	001472			CLR	INTRVL+2		:CLEAR INTERVAL COUNTER
94	004550	013737	001462	001464		MOV	CMPTIM,CMPTIM+2		:INIT COMPARE TIME COUNTER
95	004556	005037	001334			CLR	CFLAG		:CLEAR THE 'CONTROL C' FLAG
98	004562	005037	046366			CLR	DRIVE0+\$FIRST		:RESET \$FIRST FLAG FOR DRIVE 0
	004566	005037	046630			CLR	DRIVE1+\$FIRST		:RESET \$FIRST FLAG FOR DRIVE 1
	004572	005037	047072			CLR	DRIVE2+\$FIRST		:RESET \$FIRST FLAG FOR DRIVE 2
	004576	005037	047334			CLR	DRIVE3+\$FIRST		:RESET \$FIRST FLAG FOR DRIVE 3
	004602	005037	047576			CLR	DRIVE4+\$FIRST		:RESET \$FIRST FLAG FOR DRIVE 4
	004606	005037	050040			CLR	DRIVE5+\$FIRST		:RESET \$FIRST FLAG FOR DRIVE 5
	004612	005037	050302			CLR	DRIVE6+\$FIRST		:RESET \$FIRST FLAG FOR DRIVE 6
	004616	005037	050544			CLR	DRIVE7+\$FIRST		:RESET \$FIRST FLAG FOR DRIVE 7
103	004622	005037	001424			CLR	RONLY		:ASSUME READ/WRITE CONDITION
104	004626	032777	000001	174320		BIT	#SWO,@SWR		:IS EXERCISER IN 'READ ONLY' MODE ?
105	004634	001402				BEQ	8\$:BR IF NO

```

106 004636 005237 001424          INC      RONLY          ;LOCK PROGRAM IN 'READ ONLY' MODE
107 004642          8$:
109
110          ;SEE IF OPERATOR WANTS HELP TEXT PRINTED
111
112 004642 105737 001150          TSTB    $AUTOB          ;AUTO MODE ?
113 004646 001035          BNE     13$             ;BR IF YES
114 004650 005227 177777          INC     #-1             ;FIRST TIME THRU HERE ?
115 004654 001032          BNE     13$             ;BR IF NO
116
117 004656 104401 064330          9$:    TYPE    ,MSHELP          ;TYPE 'TYPE HELP MESSAGE (L) N ? '
118 004662 104411          RDLIN
119 004664 012600          MOV     (SP)+,RO        ;READ THE ENTRY
120 004666 005737 001334          TST    CFLAG            ;SAVE ADDRESS OF RESPONSE
121 004672 001013          BNE     10$             ;WAS (^C) TYPED?
122 004674 105710          TSTB   (RO)             ;BR IF YES
123 004676 001414          BEQ    11$             ;WAS RESPONSE A CARRIAGE RETURN ?
124 004700 005760 000001          TSTB   1(RO)           ;BR IF YES (DEFAULT)
125 004704 001006          BNE     10$             ;WAS IT TERMINATED WITH CARRIAGE RETURN ?
126 004706 122710 000131          CMPB   #'Y',(RO)        ;BR IF NO
127 004712 001411          BEQ    12$             ;WAS IT A 'Y' RESPONSE ?
128 004714 122710 000116          CMPB   #'N',(RO)        ;BR IF YES
129 004720 001410          BEQ    13$             ;WAS IT A 'N' RESPONSE ?
130 004722 104401 060103          10$:   TYPE    ,BADENT          ;BR IF YES
131 004726 000753          BR     9$              ;TYPE BAD ENTRY MESSAGE
132 004730 104401 057017          11$:   TYPE    ,NODFLT          ;TRY AGAIN
133 004734 000750          BR     9$              ;TYPE 'NO DEFAULT'
134 004736 104401 064440          12$:   TYPE    ,HELPTX          ;TRY AGAIN
135 004742          13$:
136
137          ;AUTO SIZE FOR RH70 CONTROLLER AND DETERMINE IF IT IS
138          ;JUMPED FOR 22 OR 32 REGISTERS
139
140 004742 005037 040566          SIZE70: CLR    RHEXT          ;CLEAR RPBAE OFFSET
141 004746 042737 174000 001234          BIC    #174000,$CPUOP   ;CLEAR CPU TYPE REGISTER
142 004754 013746 000004          MOV    ERRVEC,-(SP)     ;SAVE CONTENTS OF ERROR VECTOR
143 004760 012737 005032 000004          MOV    #2$,ERRVEC      ;SETUP 'TRAP' RETURN ADDRESS
144 004766 013700 001272          MOV    $RPADR,RO        ;GET RPCS1 ADDRESS
145 004772 062700 000050          ADD    #50,RO           ;GET REGISTER OFFSET FOR RH70
146 004776 012701 000012          MOV    #10,,R1          ;GET NUMBER OF REGISTERS TO CHECK
147 005002 005720          TST    (RO)+            ;TRAP IF NOT A VALID RPBAE
148 005004 005720          TST    (RO)+            ;TRAP IF NOT A VALID RPCS3
149 005006 012737 000050 040566          MOV    #50,RHEXT        ;LOAD OFFSET FOR RPBAE (22 REGISTER RH)
150 005014 005720          1$:    TST    (RO)+            ;TRAP IF NOT A VALID REGISTER
151 005016 005301          DEC    R1                ;DONE WITH ALL 32 REGISTERS ?
152 005020 001375          BNE     1$              ;BR IF NO
153 005022 012737 000074 040566          MOV    #74,RHEXT        ;LOAD OFFSET FOR RPBAE (32 REGISTER RH)
154 005030 000403          BR     3$              ;SETUP RETURN ADDRESS
155 005032 012716 005040          2$:    MOV    #3$,(SP)
156 005036 000002          RTI
157
158 005040 013700 001272          3$:    MOV    $RPADR,RO        ;GET RPCS1 REGISTER
159 005044 013701 040566          MOV    RHEXT,R1         ;GET RPBAE REGISTER OFFSET
160 005050 001416          BEQ    4$              ;BR IF NONE
161 005052 060001          ADD    R0,R1            ;GET RPBAE REGISTER
162 005054 052710 001400          BIS    #A17!A16,(RO)    ;SET EXTENDED ADDRESS BITS IN RPCS1
163 005060 022711 000003          CMP    #3,(R1)          ;ARE THE EXTENDED BITS SET IN RPBAE ?

```

```

164 005064 001010      BNE      4$          ;BR IF NO
165 005066 005011      CLR      (R1)       ;CLEAR EXTENDED ADDRESS BITS IN RPBAE
166 005070 011046      MOV      (R0),-(SP) ;SAVE RPCS1 REG CONTENTS
167 005072 042726 176377 BIC      #^C<A17!A16>,(SP)+ ;ARE THE EXTEND BITS CLEAR IN RPCS1 ?
168 005076 001003      BNE      4$          ;BR IF NO
169 005100 052737 030000 001234 BIS      #BIT13!BIT12,$CPUOP ;SET THE 11/70 CPU TYPE CODE
170 005106 012637 000004 4$:      MOV      (SP)+,ERRVEC ;RESTORE CONTENTS OF ERROR VECTOR
171
172 ;ROUTINE TO DETERMINE BUFFER MAX WORD COUNT
173
174 005112 005227 177777  SIZMEM: INC      #-1          ;FIRST TIME THRU HERE ?
175 005116 001005      BNE      1$          ;BR IF NO
176 005120 004737 062304 JSR      PC,$SIZE    ;SEE HOW MUCH MEMORY ON SYSTEM
177 005124 013737 062434 001330 MOV      $LSTAD,LSTAD ;SAVE THE LAST ADDRESS
178 005132 012737 000001 001654 1$:      MOV      #1,BUFTBL  ;LOAD NUMBER OF BUFFERS
179 005140 012737 064330 001656 MOV      #ENDPGM,BUFTBL+2 ;STARTING ADDRESS OF BUFFER
180 005146 013737 001330 001660 MOV      LSTAD,BUFTBL+4 ;LAST ADDR TO BUFFER ALLOCATION TABLE
181 005154 023727 001330 160000 CMP      LSTAD,#160000 ;OVER 28K ?
182 005162 101403      BLOS    2$          ;NO
183 005164 012737 160000 001660 MOV      #160000,BUFTBL+4 ;XXDF MAX MEMORY 28K
184 005172 162737 064330 001660 2$:      SUB      #ENDPGM,BUFTBL+4 ;SUBTRACT PROGRAM SPACE
185 005200 000241      CLC          ;CLEAR THE 'C' BIT
186 005202 006037 001660 ROR      BUFTBL+4    ;CONVERT TO WORD COUNT
187 005206 105737 001150 TSTB    $AUTOB      ;RUNNING IN AUTO MODE ?
188 005212 001403      BEQ      3$          ;BR IF NO
189 005214 162737 003000 001660 SUB      #1536.,BUFTBL+4 ;SUBTRACT 'XXDP' LOADER SIZE (1.5K WORDS)
190 005222 023737 001466 001660 3$:      CMP      WRDCNT,BUFTBL+4 ;IS MAX WORD COUNT TOO LARGE ?
191 005230 003406      BLE      4$          ;BR IF NO
192 005232 013737 001660 001466 MOV      BUFTBL+4,WRDCNT ;USE MAX AVAIL MEMORY AS MAX WRD CNT
193 005240 013737 001466 060606 4$:      MOV      WRDCNT,PARLST+2 ;VALUE FOR THE PARAMETER TABLE
194 005246
195
196 ;SEE IF THE OPERATOR WANTS TO CHANGE ANY PARAMETERS
197
198 005246 005737 040020  LKPAR: TST      PWRFLG    ;RETURNING FROM POWER FAIL ?
199 005252 001154      BNE      SETVEC     ;BRANCH IF YES
205 005260 105737 001150 TSTB    $AUTOB      ;RUNNING IN AUTO MODE ?
209 005264 001404      BEQ      1$          ;BR IF NO
210 005266 032777 000004 173660 BIT      #SW02,@SWR   ;DOES USER WANT MANUAL INTERVENTION ?
211 005274 001467      BEQ      8$          ;BR IF YES
212
213 005276 005037 001334  1$:      CLR      CFLAG      ;CLEAR CONTROL C FLAG
214 005302 104401 060226 TYPE    ,MESFE      ;TYPE 'DO YOU WANT TO WRITE ANYWHERE ON MEDIA (L) N ?'
215 005306 104411      RDLIN    ;READ THE ENTRY
216 005310 012600      MOV      (SP)+,RO   ;SAVE ADDRESS OF RESPONSE
217 005312 005737 001334 TST      CFLAG      ;WAS IT CONTROL C ?
218 005316 001353      BNE      LKPAR     ;BR IF YES
219 005320 105710      TSTB    (R0)       ;WAS RESPONSE A CARRIAGE RETURN ?
220 005322 001454      BEQ      8$        ;BR IF YES
221 005324 105760 000001 TSTB    1(R0)      ;WAS IT TERMINATED WITH CARRIAGE RETURN ?
222 005330 001006      BNE      2$        ;BR IF NO
223 005332 122710 000131 CMPB    #'Y,(R0)    ;WAS IT A 'Y' RESPONSE ?
224 005336 001406      BEQ      3$        ;BR IF YES
225 005340 122710 000116 CMPB    #'N,(R0)    ;WAS IT A 'N' RESPONSE ?
226 005344 001443      BEQ      8$        ;BR IF YES
227 005346 104401 060103 2$:      TYPE    ,BADENT    ;TYPE BAD ENTRY MESSAGE
228 005352 000735      BR      LKPAR     ;TRY AGAIN
    
```

229	005354	005737	001424	3\$:	TST	RONLY	:PROGRAM RUNNING IN READ ONLY MODE ?
230	005360	001032			BNE	7\$:BR IF YES
231							
232	005362	005037	001334	4\$:	CLR	CFLAG	:CLEAR CONTROL C FLAG
233	005366	1044C1	060307		TYPE	,OVRWRT	:TYPE ' ! CUSTOMER DATA WILL BE OVERWRITTEN !
234							:-----
235							:CONTINUE (L) ?'
236	005372	104411			RDLIN		:READ THE ENTRY
237	005374	012600			MOV	(SP)+,RO	:SAVE ADDRESS OF RESPONSE
238	005376	005737	001334		TST	CFLAG	:WAS IT CONTROL C ?
239	005402	001321			BNE	LKPAR	:BR IF YES
240	005404	105710			TSTB	(RO)	:WAS RESPONSE A CARRIAGE RETURN ?
241	005406	001414			BEQ	6\$:BR IF YES
242	005410	105760	000001		TSTB	1(RO)	:WAS IT TERMINATED WITH CARRIAGE RETURN ?
243	005414	001006			BNE	5\$:BR IF NO
244	005416	122710	000131		CMPB	#'Y,(RO)	:WAS IT A 'Y' RESPONSE ?
245	005422	001411			BEQ	7\$:BR IF YES
246	005424	122710	000116		CMPB	#'N,(RO)	:WAS IT A 'N' RESPONSE ?
247	005430	001411			BEQ	8\$:BR IF YES
248	005432	104401	060103	5\$:	TYPE	,BADENT	:TYPE BAD ENTRY MESSAGE
249	005436	000751			BR	4\$:TRY AGAIN
250	005440	104401	057017	6\$:	TYPE	,NODFLT	:TYPE 'NO DEFAULT'
251	005444	000746			BR	4\$:TRY AGAIN
252	005446	005237	001422	7\$:	INC	TSTANY	:ENABLE TEST ANYWHERE OPTION
253	005452	000402			BR	9\$	
254	005454	104401	060451	8\$:	TYPE	,FEONLY	:TYPE '* TESTING WILL OCCUR ON FE CYLINDER ONLY *'
255							
256	005460	005737	001424	9\$:	TST	RONLY	:IS PROGRAM LOCKED IN 'READ ONLY' MODE ?
257	005464	001402			BEQ	13\$:BR IF NO
258	005466	104401	060526		TYPE	,MREAD	:TYPE '* PROGRAM IS LOCKED IN 'READ ONLY' MODE *'
280							
281	005472	005037	001334	13\$:	CLR	CFLAG	:CLEAR CONTROL C FLAG
283	005476	105737	001150		TSTB	\$AUTOB	:RUNNING IN AUTO MODE ?
284	005502	001040			BNE	SETVEC	:BR IF YES
286	005504	104401	060706		TYPE	,ASKPAR	:TYPE 'CHANGE PARAMETERS ?'
287	005510	104411			RDLIN		:READ THE ENTRY
288	005512	012600			MOV	(SP)+,RO	:SAVE ADDRESS OF RESPONSE
289	005514	005737	001334		TST	CFLAG	:WAS (^C) TYPED?
290	005520	001364			BNE	13\$:BR IF YES
291	005522	105710			TSTB	(RO)	:WAS RESPONSE A CARRIAGE RETURN (DEFAULT 'N')?
292	005524	001427			BEQ	SETVEC	:BR IF YES
293	005526	105760	000001		TSTB	1(RO)	:WAS IT TERMINATED WITH CARRIAGE RETURN ?
294	005532	001006			BNE	14\$:BR IF NO
295	005534	122710	000131		CMPB	#'Y,(RO)	:WAS IT A 'Y' RESPONSE ?
296	005540	001406			BEQ	ENTPR	:BR IF YES
297	005542	122710	000116		CMPB	#'N,(RO)	:WAS IT A 'N' RESPONSE ?
298	005546	001416			BEQ	SETVEC	:BR IF YES
299	005550	104401	060103	14\$:	TYPE	,BADENT	:TYPE BAD ENTRY MESSAGE
300	005554	000746			BR	13\$:TRY AGAIN
301	005556			15\$:			
302							
303	005556	012703	060604	ENTPR:	MOV	#PARLST,R3	:PARAMETER TABLE ADDRESS
304	005562	004737	031110		JSR	PC,PARENT	:GET THE PARAMETER ENTRY
305	005566	023727	001466	000006	CMP	WRDCNT,#6	:IS THE 'WRDCNT' VALUE OK ?
306	005574	103003			BHIS	SETVEC	:BR IF IT IS
307	005576	012737	000006	001466	MOV	#6,WRDCNT	:SET 'WRDCNT' TO THE MINIMUM VALUE
308							

```

309          ;DISPLAY DRIVE STATUS AND SET UP THE OTHER SYSTEM DEVICES THAT
310          ;THE PROGRAM WILL USE. PROGRAM RETURN HERE ON POWER FAIL
311
312 005604 004737 024514          SETVEC: JSR      PC,CKCLK          ;START THE CLOCK
313 005610 004737 040572          JSR      PC,RPINIT        ;INITIALIZE THE RP07 DRIVER
317 005614 012737 177777 040522  MOV      #-1,SAVEFG      ;SET THE SAVE REGISTERS FLAG
318 005622 005227 177777          INC      #-1              ;FIRST TIME THRU ?
319 005626 001407          BEQ      1$               ;BR IF YES
320 005630 005737 040020          TST     PWRFLG           ;RETURNING FROM POWER FAIL ?
321 005634 001004          BNE     1$               ;BRANCH IF YES
322 005636 032777 000004 173310  BIT     #SW02,@SWR       ;TYPEOUT THE DRIVE STATUS TABLE ?
323 005644 001113          BNE     12$              ;BR IF NOT
324 005646 005004          1$: CLR     R4              ;DRIVE TABLE POINTER
325 005650 104401 057033          TYPE   ,DRSTAT          ;TYPE 'DRIVE STATUS'
326 005654 104401 001203          2$: TYPE   ,%CRLF         ;CR-LF
327 005660 010446          MOV     R4,-(SP)         ;SAVE R4 FOR TYPEOUT
                                ;TYPE DRIVE NUMBER
                                ;GO TYPE--OCTAL ASCII
                                ;TYPE 2 DIGIT(S)
                                ;SUPPRESS LEADING ZEROS
                                ;TYPE 4 BLANKS
005662 104403          TYPOS          ;CHECK DRIVE'S STATUS
005664 002          .BYTE 2              ;BR IF UNSAFE
005665 000          .BYTE 0              ;BR IF ONLINE
328 005666 104401 056526          TYPE   ,BLNKS4         ;SEE IF OFFLINE OR NONEXISTENT
329 005672 105764 040454          TSTB   DRVSTA(R4)      ;BR IF NONEXISTENT
330 005676 100416          BMI   5$               ;BR IF OFFLINE
331 005700 001020          BNE   6$               ;DRIVE NOT AN RP07
332          ;CHECK NEXT DRIVE
333 005702 105764 040464          TSTB   DRVSTYP(R4)     ;SEE IF OFFLINE OR NONEXISTENT
334 005706 001404          BEQ   3$               ;BR IF NONEXISTENT
335 005710 100006          BPL   4$               ;BR IF OFFLINE
336 005712 104401 056700          TYPE   ,NOTRP          ;DRIVE NOT AN RP07
337 005716 000460          BR    11$              ;CHECK NEXT DRIVE
338
339 005720 104401 056715          3$: TYPE   ,NOTPRS      ;DRIVE NOT PRESENT
340 005724 000455          BR    11$              ;CHECK NEXT DRIVE
341
342 005726 104401 056607          4$: TYPE   ,UNTOFF      ;DRIVE OFFLINE
343 005732 000416          BR    8$               ;PRINT DRIVE TYPE
344
345 005734 104401 056751          5$: TYPE   ,NOTSAF      ;DRIVE UNSAFE
346 005740 000413          BR    8$               ;PRINT DRIVE TYPE
347
348 005742 005737 001430          6$: TST     XXDP          ;LOADED FROM THIS DEVICE ?
349 005746 001406          BEQ   7$               ;BR IF NO
350 005750 123704 001430          CMPB   XXDP,R4          ;LOADED FROM THIS DRIVE ?
351 005754 001003          BNE   7$               ;BR IF NO
352 005756 104401 056761          TYPE   ,LODEV          ;TYPE 'LOAD DEVICE'
353 005762 000436          BR    11$              ;CHECK NEXT DRIVE
354 005764 104401 056620          7$: TYPE   ,UNTON       ;DRIVE ONLINE
355 005770 104401 056530          8$: TYPE   ,BLNKS2      ;TYPE 2 BLANKS
356 005774 005000          CLR   R0
357 005776 116400 040464          MOVB   DRVSTYP(R4),R0  ;GET DRIVE TYPE
358 006002 012737 057075 006026  MOV     #RPO7,10$       ;ASSUME ADDRESS OF RP07 MESSAGE
359 006010 122700 000005          CMPB   #5,R0           ;IS DEVICE AN RP07 ?
360 006014 001403          BEQ   9$               ;TYPE IT IF YES
361 006016 012737 057102 006026  MOV     #RPO7P,10$      ;ASSUME ADDRESS OF RP07+ MESSAGE
362
363 006024 104401          9$: TYPE   ;TYPE THE DRIVE TYPE MESSAGE
364 006026 000000          10$: .WORD 0           ;MESSAGE ADDRESS HERE
    
```

```
365  
366 006030 006304 ASL R4 ;CHANGE TO WORD INDEX  
367 006032 016402 002056 MOV LKADR(R4),R2 ;GET BASE ADDRESS  
368 006036 006204 ASR R4 ;CHANGE TO BYTE INDEX  
369 006040 004737 045262 JSR PC,SVRHXX ;SAVE ALL REGISTERS  
370 006044 032762 000004 000200 BIT #1LV,$RPDS(R2) ;INTERLEAVE SECTOR SET ?  
371 006052 001002 BNE 11$ ;BR IF YES  
372 006054 104401 056776 TYPE ,NINLEV ;TYPE NON-INTERLEAVED MESSAGE  
373  
374 006060 005204 11$: INC R4 ;INCREMENT DRIVE NUMBER/TABLE POINTER  
375 006062 020427 000010 CMP R4,#8. ;FINISHED ?  
376 006066 001272 BNE 2$ ;BR IF NOT  
377 006070 104401 001203 TYPE ,$CRLF ;CR-LF  
378 006074 12$:
```

```

1          ;INITIALIZE PROGRAM PARAMETERS FOR STARTUP
2
3 006074 004737 033430 STA: JSR PC,$TKINT ;INITIALIZE THE KEYBOARD INTERRUPT HANDLER
4 006100 012737 142200 001446 MOV #142200,ENDCON ;INITIALIZE NORMAL XFER COUNT(LSB)
5 006106 012737 007540 001450 MOV #7540,ENDCON+2 ;(MSB)
6 006114 032777 001000 173032 BIT #SW9,@SWR ;DO YOU WANT SHORT RUN TIME (SW9=1) ?
7 006122 001406 BEQ 1$ ;BR IF NO
8 006124 012737 030440 001446 MOV #030440,ENDCON ;INITIALIZE (1/4 OF NORMAL) XFER COUNT(LSB)
9 006132 012737 001730 001450 MOV #1730,ENDCON+2 ;(MSB)
10
11 006140 105737 001226 1$: TSTB $ENV ;APT SCRIPT MODE ?
12 006144 001406 BEQ 2$ ;NO
13 006146 012737 006110 001446 MOV #006110,ENDCON ;INITIALIZE (1/16 OF NORMAL) XFER COUNT(LSB)
14 006154 012737 000366 001450 MOV #366,ENDCON+2 ;(MSB)
15
16 006162 105737 001150 2$: TSTB $AUTOB ;RUNNING IN AUTO MODE ?
17 006166 001003 BNE 3$ ;BR IF YES
18 006170 005737 001332 TST CHGADR ;START AT 200 ?
19 006174 003454 BLE 8$ ;NO
20
21 006176 005001 3$: CLR R1 ;DRIVE #
22 006200 005002 CLR R2 ;AVAIL TABLE INDEX
23 006202 005003 CLR R3 ;DRIVE# * 2
24 006204 005737 001430 4$: TST XXDP ;LOADED FROM THIS DEVICE ?
25 006210 001403 BEQ 5$ ;BR IF NO
26 006212 123701 001430 CMPB XXDP,R1 ;LOADED FROM THIS DRIVE ?
27 006216 001433 BEQ 7$ ;BR IF YES
28 006220 105761 040454 5$: TSTB DRVSTA(R1) ;DRIVE ON LINE ?
29 006224 003430 BLE 7$ ;NO
30 006226 016300 002056 MOV BLKADR(R5),R0 ;LOAD DPB ADDRESS
31 006232 004737 030104 JSR PC,CLRDPB ;CLEAR DPB BLOCK
32 006236 004737 031010 JSR PC,GETID ;GET DRIVE SERIAL NUMBER
33 006242 032760 000004 000200 BIT #ILV,$RPDS(R0) ;INTERLEAVE SECTOR SET ?
34 006250 001402 BEQ 6$ ;BR IF NO
35 006252 010062 001566 MOV R0,NEWUNT(R2) ;LOAD DPB ADDRESS TO ABAIL QUEUE
36 006256 004737 030322 6$: JSR PC,DRVPRM ;SETUP DRIVE PARAMETER LIMITS
37 006262 005737 040020 TST PWRFLG ;RETURNING FROM POWER FAIL ?
38 006266 001007 BNE 7$ ;BRANCH IF YES
39 006270 005060 000132 CLR $FIRST(R0) ;RESET $FIRST FLAG FOR FIRST 204 START
40 006274 112760 177776 000026 MOVB #-2,$PACK(R0) ;SETUP COMMAND 'WT' (WRITE DATA AND TEST)
41 006302 004737 020250 JSR PC,SEQPAR ;SETUP SEQUENTIAL PARAMETERS FOR WRITE
42
43 006306 022322 7$: CMP (R3)+,(R2)+ ;INCREMENT INDEX
44 006310 005201 INC R1 ;NEXT DRIVE
45 006312 020127 000007 CMP #1,#7 ;ALL DRIVES ASSIGNED ?
46 006316 003732 BLE 4$ ;NO
47 006320 005037 001332 CLR CHGADR ;CLEAR START FLAG
48 006324 000403 BR 9$
49
50 006326 012737 000001 001334 8$: MOV #1,CFLAG ;DUMMY 'CONTROL C' FLAG
51 006334 005037 040020 9$: CLR PWRFLG ;CLEAR POWER FAIL FLAG
    
```

```

1          .SBTTL  MAIN PROGRAM
2
3 006340 005737 001334  MAIN:  TST    CFLAG      ;WAS (^C) TYPED?
4 006344 001407          BEQ     3$        ;BR IF NO
5 006346 005737 001520  1$:   TST    ORDERQ     ;ANY DRIVES IN ORDER QUE ?
6 006352 001402          BEQ     2$        ;BR IF NO, ELSE
7 006354 000137 007234  JMP    IDLE         ;LET ALL DRIVES FINISH ORDER
8 006360 004737 026454  2$:   JSR    PC,KSR   ;SERVICE THE KEYBOARD
9 006364          3$:
10
11          ;CHECK FOR DRIVES TO BE DROPPED
12
13 006364 012703 000010  MAINDA: MOV    #8.,R3      ;DRIVE COUNTER
14 006370 012705 001544  MOV    #DDRVS,R5     ;ADDRESS OF 'DROPPED DRIVE' TABLE
15 006374 005715          1$:   TST    (R5)        ;SEE IF ENTRY AT PRESENT POSITION
16 006376 001004          BNE    3$          ;BR IF THERE IS ONE
17 006400 005725          2$:   TST    (R5)+      ;INCREMENT TO NEXT TABLE POSITION
18 006402 005303          DEC    R3           ;DECREMENT DRIVE COUNTER
19 006404 001373          BNE    1$          ;BR IF MORE TO CHECK
20 006406 000437          BR     MAIN1       ;GO CHECK FOR NEW ASSIGNED DRIVES
21
22 006410 012701 001610  3$:   MOV    #AVAIL,R1   ;ADDRESS OF 'AVAILABLE DRIVES' TABLE
23 006414 005711          4$:   TST    (R1)        ;IF AT END OF 'AVAIL' TABLE ?
24 006416 001404          BEQ    5$          ;BR IF YES
25 006420 021115          CMP    (R1),(R5)   ;IS DRIVE IN 'AVAIL' THE TABLE ?
26 006422 001412          BEQ    7$          ;BR IF YES
27 006424 005721          TST    (R1)+      ;NO, INCREMENT 'AVAIL' TABLE ADDRESS
28 006426 000772          BR     4$          ;AND CONTINUE LOOKING
29
30 006430 012701 001632  5$:   MOV    #WAIT,R1    ;ADDRESS OF THE 'WAIT' BUFFER TABLE
31 006434 005711          6$:   TST    (R1)        ;AT THE END OF 'WAIT' TABLE ?
32 006436 001760          BEQ    2$          ;BR IF YES
33 006440 021115          CMP    (R1),(R5)   ;IS DRIVE IN THE 'WAIT' TABLE ?
34 006442 001402          BEQ    7$          ;BR IF YES
35 006444 005721          TST    (R1)+      ;NO, INCREMENT 'WAIT' TABLE ADDRESS
36 006446 000772          BR     6$          ;AND CONTINUE LOOKING
37
38 006450 011100          7$:   MOV    (R1),R0     ;PUT THE DRIVE'S BLOCK ADDRESS IN R0
39 006452 104401 001203  TYPE   ,SCLF        ;CR-LF
40 006456 004737 026146  JSR    PC,$TIME     ;TYPE ELAPSED TIME
41 006462 104401 057321  TYPE   ,DASH5       ;TYPE '-----'
42 006466 104401 057357  TYPE   ,MSGDRP      ;TYPE 'DROPPED, '
43 006472 004737 025042  JSR    PC,ONESUM    ;TYPE ONE DRIVE SUMMARY
44 006476 005015          CLR    (R5)        ;CLEAR THE 'DROP DRIVE' TABLE ENTRY
45 006500 004737 022200  JSR    PC,CMPRES    ;COMPRESS THE RESPECTIVE TABLE
46 006504 000735          BR     2$          ;SEE IF ANY MORE DRIVES
47
48          ;LOOK FOR DRIVES TO BE ASSIGNED
49
50 006506 012703 000010  MAIN1: MOV    #8.,R3      ;DRIVE COUNT
51 006512 005001          CLR    R1          ;ASSIGN LIST INDEX
52 006514 005002          CLR    R2          ;'AVAIL' INDEX
53 006516 005005          CLR    R5          ;NEW DRIVE INDEX
54 006520 005765 001566  1$:   TST    NEWUNT(R5) ;NEW DRIVE IN THIS POSITION
55 006524 001005          BNE    3$          ;BR IF THERE IS
56 006526 005725          2$:   TST    (R5)+      ;INCREMENT R5
57 006530 005201          INC    R1          ;INCREMENT ASSIGN INDEX

```



```

58 006532 005303          DEC      R3          :DECREMENT DRIVE COUNT
59 006534 001371          BNE     1$          :BR IF MORE DRIVES
60 006536 000442          BR      MAIN2       :START OPERATIONS FOR THE AVAILABLE DRIVES
61
62 006540 104401 001203   3$:  TYPE     ,%CRLF      :CR-LF
63 006544 104401 056601   TYPE     ,DRVMSG     :'DRIVE'
64 006550 010146          MOV     R1,-(SP)    :SAVE R1 FOR TYPEOUT
                                :TYPE DRIVE NUMBER
                                :GO TYPE--OCTAL ASCII
                                :TYPE 2 DIGIT(S)
                                :SUPPRESS LEADING ZEROS
        006552 104403          TYPOS
        006554      002          .BYTE  2
        006555      000          .BYTE  0
65 006556 104401 056573   TYPE     ,COMMA      :TYPE ','
66 006562 104401 056531   TYPE     ,BLNKS1    :TYPE 1 BLANK
67 006566 016500 001566   MOV     NEWUNT(R5),R0 :PUT ADDRESS OF DPB IN R0
68 006572 004737 032770   JSR    PC,TYPDRV    :TYPE DRIVE SERIAL NUMBER
69 006576 104401 057623   TYPE     ,ASGND     :'STARTED'
70 006602 005762 001610   4$:  TST     AVAIL(R2) :AT END OF AVAILABLE TABLE
71 006606 001402          BEQ     5$          :BR IF YES
72 006610 005722          TST     (R2)+       :INCREMENT AVAILABLE TABLE INDEX
73 006612 000773          BR      4$          :CONTINUE LOOKING FOR END OF TABLE
74 006614 016562 001566 001610 5$:  MOV     NEWUNT(R5),AVAIL(R2) :MOVE ADDR OF DRIVE INTO AVAIL LST
75 006622 005065 001566          CLR     NEWUNT(R5)  :TAKE DRIVE OUT OF NEW DRIVE TABLE
76 006626 156137 040550 001542  BISB    ATABIT(R1),ASNLST :SET DRIVE ASSIGNED INDICATOR
77 006634 005037 032020          CLR     AUTLST     :CLEAR AUTO ASSIGN
78 006640 005722          TST     (R2)+       :INCREMENT AVAILABLE TABLE POINTER
79 006642 000731          BR      2$          :LOOK FOR MORE DRIVES
80
81                          :GET PARAMETERS, BUFFER SPACE, AND START ORDERS FOR DRIVES IN
82                          :THE 'AVAILABLE' QUEUE
83
84 006644 005002          MAIN2: CLR     R2          :START FROM THE FIRST LOCATION
85 006646 105737 001542   TSTB   ASNLST      :ANY DRIVES ACTIVE ?
86 006652 001030          BNE     2$          :BR IF YES
87 006654 105737 001150   TSTB   $AUTOB     :RUNNING IN AUTO MODE ?
88 006660 001023          BNE     1$          :BR IF YES
89 006662 012737 000001 001334  MOV     #1,CFLAG   :DUMMY 'CONTROL C' FLAG
90 006670 013737 001312 001346  MOV     HERTZ,ONESEC :RESTORE ONE SECOND COUNTER VALUE
91 006676 005037 001340          CLR     HOUR       :CLEAR THE HOUR'S COUNTER
92 006702 005037 001342          CLR     MINUTE     :CLEAR THE MINUTE'S COUNTER
93 006706 005037 001344          CLR     SECOND     :CLEAR THE SECOND'S COUNTER
94 006712 005037 001472          CLR     INTRVL+2   :CLEAR INTERVAL COUNTER
95 006716 013737 001462 001464  MOV     CMPTIM,CMPTIM+2 :INIT COMPARE TIME COUNTER
96 006724 104401 060124          TYPE     ,NODRVS   :TYPE 'NO DRIVES ASSIGNED'
97 006730 000137 031744   1$:  JMP     $GET42    :GIVE CONTROL TO MONITOR
98
99 006734 005762 001632   2$:  TST     WAIT(R2)   :ANY DRIVES WAITING FOR THE BUFFER ?
100 006740 001435          BEQ     5$          :BR IF NO
101 006742 016200 001632   MOV     WAIT(R2),R0 :LOAD R0 WITH THE DPB ADDRESS
102 006746 005046          CLR     -(SP)      :CLEAR THE STACK FOR BUFFER REQ
103 006750 004737 017512   JSR    PC,GETBUF   :CALL TO GET THE BUFFER RT.
104 006754 012660 000006   MOV     (SP)+,$BUF(RC) :IF 0,BUFFER IS STILL NOT AVAILABLE
105 006760 001423          BEQ     4$          :BRANCH IF NO BUFFER AVAILABLE
106 006762 005060 000130          CLR     $NEXT(R0)  :CLEAR PARAMETER SELECT FLAG
107 006766 005060 000116          CLR     $FAIR(R0)  :CLEAR THE FAIR FLAG
108
109 006772 004737 020102          JSR    PC,FILBUF   :FILL THE BUFFER
110 006776 004737 020160          JSR    PC,GODRIV   :SET COMMAND AND GO
    
```

111	007002	012705	001520		MOV	#ORDERQ,R5	:PUT THE WAIT QUE INTO ORDER QUE
112	007006	005725		3\$:	TST	(R5)+	:QUE AVAILABLE ?
113	007010	001376			BNE	3\$:BR IF NO
114	007012	010045			MOV	RO,-(R5)	:LOAD THE DPB ADDRESS INTO THE ORDER QUE
115	007014	012701	001632		MOV	#WAIT,R1	:REMOVE THE DRIVE FROM THE 'WAIT' QUE
116	007020	060201			ADD	R2,R1	:OFFSET THE QUE POSITION
117	007022	004737	022200		JSR	PC,COMPRES	:COMPRESS THE QUE
118	007026	000742			BR	2\$:BRANCH IF DONE
119	007030	005722		4\$:	TST	(R2)+	:CHECK THE NEXT QUE
120	007032	000740			BR	2\$:LOOPING BACK
121							
122	007034	005737	001520	5\$:	TST	ORDERQ	:ANY OUTSTANDING ORDERS ?
123	007040	001075			BNE	IDLE	:BR IF YES
124							
125	007042	005002			CLR	R2	:CLEAR DRIVE TABLE POINTER
126	007044	005762	001610	6\$:	TST	AVAIL(R2)	:ANY DRIVES WAITING FOR PARAMETERS
127	007050	001002			BNE	7\$:BRANCH IF ANY
128	007052	000137	007234		JMP	IDLE	:BRANCH IF NONE
129	007056	016200	001610	7\$:	MOV	AVAIL(R2),RO	:CONTROL BLOCK ADDR IN RO
130	007062	005760	000130		TST	\$NEXT(RO)	:PARAMETERS BEEN SELECTED ?
131	007066	001010			BNE	9\$:BR IF THEY HAVE
132	007070	105760	000026		TSTB	\$PACK(RO)	: 'T' COMMAND FOR THIS DRIVE ?
133	007074	001403			BEQ	8\$:BR IF YES
134	007076	004737	020250		JSR	PC,SEQPAR	:GET SEQUENTIAL PARAMETERS FOR READ OR WRITE
135	007102	000404			BR	10\$:GET THE BUFFER
136							
137	007104	004737	020612	8\$:	JSR	PC,GENPAR	:GO GENERATE THE PARAMETERS
138	007110	004737	021662	9\$:	JSR	PC,LODPAR	:LOAD THE PARAMETERS JUST GENERATED
139							
140	007114	005046		10\$:	CLR	-(SP)	:MAKE ROOM ON THE STACK FOR THE BUFFER ADDR
141	007116	004737	017512		JSR	PC,GETBUF	:GET BUFFER
142	007122	012660	000006		MOV	(CP)+,\$BUF(RO)	:MOVE BUFFER ADDR TO DPB
143	007126	001416			BEQ	12\$:BR IF '0' ADDR (NO BUFFER)
144	007130	004737	020102		JSR	PC,FILBUF	:FILL THE BUFFER
145	007137	005060	000130		CLR	\$NEXT(RO)	:CLEAR PARAMETER SELECT FLAG
146	007140	005060	000116		CLR	\$FAIR(RO)	:CLEAR THE 'FAIRNESS' COUNT
147							
148	007144	004737	020160		JSR	PC,GODRIV	:PUT CURRENT DPB IN DRIVER
149	007150	012705	001520		MOV	#ORDERQ,R5	:ADDRESS OF ORDER QUE IN R5
150	007154	005725		11\$:	TST	(R5)+	:END OF QUE ?
151	007156	001376			BNE	11\$:BR IF NOT
152	007160	010045			MOV	RO,-(R5)	:PUT BLOCK ADDRESS INTO QUE
153	007162	000416			BR	15\$:CONTINUE LOOKING
154							
155	007164	005260	000116	12\$:	INC	\$FAIR(RO)	:INCREMENT THE FAIR COUNT
156	007170	022760	000003	000116	CMPL	#3,\$FAIR(RO)	:THREE TIMES,BUFFER IS NOT AVAILABLE?
157	007176	101006			BH!	14\$:BRANCH IF NOT OVER THREE TIMES
158	007200	012705	001632		MOV	#WAIT,R5	:LOAD INTO THE WAIT QUE
159	007204	005725		13\$:	TST	(R5)+	:AN AVAILABLE LOCATION ?
160	007206	001376			BNE	13\$:BRANCH IF NOT
161	007210	010045			MOV	RO,-(R5)	:LOAD INTO WAIT QUE
162	007212	000402			BR	15\$:REMOVE THE DPB FROM AVAILABLE QUE
163							
164	007214	005722		14\$:	TST	(R2)+	:INCREMENT INDEX
165	007216	000712			BR	6\$:BRANCH BACK TO FIRE NEXT DRIVE
166	007220	012701	001610	15\$:	MOV	#AVAIL,R1	: 'AVAILABLE' TABLE ADDRESS
167	007224	060201			ADD	R2,R1	:FORM ADDRESS OF LAST ENTRY


```

1          ;PROCESS THE COMMAND TERMINATION
2
3          PROCES: MOVB    (R0),DRVNO      ;DRIVE NUMBER FOR ANY ERROR MESSAGES
7          TST     $STATUS(R0)          ;SEE IF DRIVER SIGNALLED AN ERROR
8          BMI     ERPROC                ;BR IF ERROR
9          BIT     #BIT15,$RPCS1(R0)     ;SEE IF 'SC' SET
10         BEQ     1$                    ;BR IF NOT SET
11         BIT     #BIT14,$RPCS1(R0)     ;SEE IF 'TRE' SET
12         BNE     ERPROC                ;BR IF SET
13         BIT     #BIT14,$RPDS(R0)     ;SEE IF 'ERR' SET
14         BNE     ERPROC                ;BR IF SET
15
16         ;NO ERRORS DETECTED IN REGISTERS, DO SOME CHECKING ANYWAY
17
18         1$:     JSR     PC,CKERR        ;CHECK ERROR BITS
19         JSR     PC,CKBUS              ;CHECK BUS ADDRESS & WORD COUNT
20         BIT     #SW01,@SWR           ;INHIBIT DATA COMPARE (SW01=1) ?
21         BNE     3$                    ;BR IF YES
22         TST     CMPTIM                ;IS COMPARE DATA ALWAYS ON ?
23         BEQ     2$                    ;BR IF YES
24         CMP     CMPTIM+2,CMPTIM      ;TIME TO COMPARE DATA ?
25         BLT     3$                    ;BR IF NO
26         MOV     CMPTIM,-(SP)         ;GET CURRENT INTERVAL TIME AND
27         ADD     #1,(SP)              ;ADD ONE MINUTE
28         CMP     CMPTIM+2,(SP)+      ;STILL COMPARING DATA ?
29         BLT     2$                    ;BR IF YES
30         CLR     CMPTIM+2            ;CLEAR INTERVAL TIMER FOR NEXT COMPARE
31         JSR     PC,CMPAR              ;NO 'DCK' ERROR, BUT COMPARE DATA ANYWAY
32         2$:     JSR     PC,CMPAR
33         3$:     RTS     PC            ;RETURN
34
35         ;COMMAND TERMINATED WITH AN ERROR - PROCESS THE ERROR
36         ERPROC: BIT     #BIT07,$STATUS(R0) ;DONE BIT SET ?
44         BEQ     ERPRC1                ;BR IF NO, ORDER DIDN'T COMPLETE NORMALLY
45         JMP     DONE                  ;PROCESS ERROR WITH 'DONE' BIT SET
47
48         ;PROCESS ORDER COMPLETION WITH 'ERROR' & 'DONE' NOT SET
49
50         ERPRC1:
51         BIT     #BIT12,$STATUS(R0)    ;SEE IF DRIVE WAS UNSAFE
52         BNE     PUNSAF                ;BR IF YES
53         BIT     #BIT11,$STATUS(R0)    ;PARITY ERROR OCCURRED
54         BNE     UCPAR                 ;BR IF IT DID
55         BIT     #BIT10,$STATUS(R0)    ;FATAL PARITY ERROR?
56         BNE     FALPAR                ;BR IF THERE IS ONE
57         BIT     #BIT09,$STATUS(R0)    ;TIMEOUT?
58         BNE     SWTIM                 ;BR IF YES
59         BIT     #BIT14!BIT01,$STATUS(R0) ;DRIVE WENT OFFLINE ?
60         BNE     OFLIN                 ;BR IF IT DID
61         BIT     #BIT2,$STATUS(R0)     ;PORT REQUEST TIME OUT ?
62         BNE     PRTIM                 ;BR IF IT DID
63         RTS     PC                    ;ERROR. RETURN
64
65         ;DRIVE IS PERSISTENTLY UNSAFE
66
67         PUNSAF: JSR     PC,LINE1       ;PRINT LINE 1 OF ERROR MESSAGE
68         DISPLY ,EM12                 ;PERSISTENT DEVICE UNSAFE MESSAGE
    
```



```

122 010150 004737 022214          PRTIM JSR PC,LINE1      ;TYPE LINE 1 OF THE ERROR MESSAGE
123 010154 104414 051503          DISPLY ,EM15      ;PRINT PORT TIME OUT MESSAGE
126 010160 004737 022262          JSR PC,LINE2      ;TYPE LINE 2 OF THE ERROR MESSAGE
    010164 004737 022716          JSR PC,LINE3      ;TYPE LINE 3 OF THE ERROR MESSAGE
    010170 004737 023366          JSR PC,LINE4      ;TYPE LINE 4 OF THE ERROR MESSAGE
127 010174 004737 026122          JSR PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
128 010200 004737 024056          JSR PC,LINE7      ;TYPE LINE 7 OF THE ERROR MESSAGE
129 010204 000207          RTS PC           ;RETURN
130
131          ;PROCESS ORDER COMPLETION WITH 'ERROR' & 'DONE' BIT SET
132
139 010206 000240          DONE: NOP
141 010210 032760 000030 000016 1$: BIT #BIT04,BIT03,STATUS(RO) ;UNSAFE OCCURRED
142 010216 001402          BEQ 2$           ;BR IF NOT
143 010220 000137 013340          JMP UNSAF        ;REPORT UNSAFE
144
145 010224 032760 040000 000202 2$: BIT #BIT14,$RPER1(RO) ;SEE IF 'UNS' SET
146 010232 001402          BEQ 3$           ;BR IF NO
147 010234 000137 013340          JMP UNSAF        ;REPORT UNSAFE
148
149 010240 032760 000002 000200 3$: BIT #BIT1,$RPDS(RO) ;SEE IF 'EWN' SET
150 010246 001402          BEQ 4$           ;BR IF NOT
151 010250 000137 013432          JMP EWNERR       ;REPORT EARLY WARNING
152
153 010254 032760 040000 000176 4$: BIT #BIT14,$RPCS2(RO) ;IS 'WCE' SET ?
154 010262 001402          BEQ 5$           ;BRANCH IF NOT SET
155 010264 000137 011234          JMP WCKER        ;WRITE CHECK ERROR
156
157 010270 032760 040000 000166 5$: BIT #BIT14,$RPCS1(RO) ;CHECK 'TRE'
158 010276 001002          BNE 6$           ;BR IF SET
159 010300 000137 013104          JMP TRFER        ;PROCESS 'TRE'
160
161 010304 032760 000400 000202 6$: BIT #BIT08,$RPER1(RO) ;'HCRC' SET?
162 010312 001402          BEQ 7$           ;BR IF NOT
163 010314 000137 011624          JMP HCRCER       ;PROCESS 'HCRC'
164
165 010320 032760 000020 000202 7$: BIT #BIT04,$RPER1(RO) ;'FMT' SET?
166 010326 001402          BEQ 8$           ;BR IF NOT SET
167 010330 000137 012010          JMP CKFMT        ;CHECK FORMAT ERROR
168
169 010334 032760 000200 000202 8$: BIT #BIT07,$RPER1(RO) ;'HCE' SET?
170 010342 001402          BEQ 9$           ;BR IF NOT SET
171 010344 000137 012172          JMP CKHCE        ;CHECK 'HCE' ERROR
172
173 010350 032760 020000 000202 9$: BIT #BIT13,$RPER1(RO) ;'OPI' SET?
174 010356 001402          BEQ 10$          ;BR IF NOT SET
175 010360 000137 012460          JMP OPIER        ;REPORT 'OPI'
176
177 010364 032760 000010 000202 10$: BIT #BIT3,$RPER1(RO) ;'PAR' SET?
178 010372 001402          BEQ 11$          ;BR IF NOT SET
179 010374 000137 012576          JMP PARER        ;REPORT 'PAR'
180
181 010400 032760 000040 000202 11$: BIT #BIT5,$RPER1(RO) ;'WCF' SET?
182 010406 001402          BEQ 12$          ;BR IF NOT SET
183 010410 000137 013242          JMP WCFER        ;REPORT 'WCF'
184
185 010414 032760 002000 000202 12$: BIT #BIT10,$RPER1(RO) ;'IAE' SET?
    
```

```

186 010422 001402          BEQ    13$          :BR IF NOT SET
187 010424 000137 012670  JMP    IAEER          :REPORT 'IAE'
188
189 010430 032760 004000 000202 13$:  BIT    #BIT11,$RPER1(RO)  : 'WLE' SET?
190 010436 001402          BEQ    14$          :BR IF NOT SET
191 010440 000137 012722  JMP    WLEER          :REPORT 'WLE'
192
193 010444 032760 001000 000202 14$:  BIT    #BIT9,$RPER1(RO)   : 'AOE' SET?
194 010452 001405          BEQ    15$          :BR IF NOT SET
195 010454 032760 002000 000200  BIT    #BIT10,$RPDS(RO)  : 'LBT' SET?
196 010462 001401          BEQ    15$          :BR IF NOT SET
197 010464 000207          RTS    PC            : 'AOE' & 'LBT' SET, EXIT
198
199 010466 032760 010000 000202 15$:  BIT    #BIT12,$RPER1(RO)  :SEE IF 'DTE' SET
200 010474 001402          BEQ    16$          :BR IF NOT
201 010476 000137 012562  JMP    DTEER          :REPORT 'DTE' ERROR
202
203 010502 005760 000202          16$:  TST    $RPER1(RO)      :SEE IF 'DCK' SET
204 010506 100002          BPL    17$          :BR IF NOT
205 010510 000137 010546  JMP    DCKER          :PROCESS 'DCK'
206
207 010514 032760 040000 000230 17$:  BIT    #BIT14,$RPER3(RO)  : 'SKI' SET
208 010522 001006          BNE    18$          :BRANCH IF SKI, OCYL SET
209 010524 032760 100000 000230  BIT    #BIT15,$RPER3(RO)  :BAD SPOT ?
210 010532 001004          BNE    19$          :BRANCH IF SO
211 010534 000137 011756  JMP    DRVER          :REPORT ERROR
212
213 010540 000137 013204          18$:  JMP    SKIER          :REPORT SKI ERROR
214 010544 000207          19$:  RTS    PC            :EXIT FROM ERROR ANALYSIS ROUT.
    
```

```

1          ;PROCESS DATA ('DCK') CHECK ERROR
2
8 010546 032760 000100 000202 DCKER: BIT #ECH,$RPER1(R0) ;ECC ERROR SET ?
10 010554 001411 BEQ 1$ ;BR IF NO
11 010556 022760 010040 000232 CMP #10040,$RPEC1(R0) ;OTHERWISE RPEC1=10040
12 010564 001024 BNE 2$ ;REPORT ECC LOGICAL FAILURE
13 010566 004737 022214 JSR PC,LINE1 ;FIRST LINE OF ERROR MESSAGE
14 010572 104414 053456 DISPLY ,EM52 ;ECC ERROR - ECC UNCORRECTABLE
15 010576 000451 BR 7$
16
17 010600 026027 000232 010040 1$: CMP $RPEC1(R0),#10040 ;IS POSITION COUNT OVER MAXIMUM ?
18 010606 101013 BHI 2$ ;BR IF YES
19 010610 005760 001232 TST $RPEC1(R0) ;POSITION COUNT 0 ?
20 010614 001410 BEQ 2$ ;BR IF YES
21 010616 005760 000234 TST $RPEC2(R0) ;VALUE IN PATTERN REGISTER ?
22 010622 001033 BNE 6$ ;BR IF YES
23 010624 004737 022214 JSR PC,LINE1 ;TYPE FIRST LINE OF ERROR MESSAGE
24 010630 104414 053300 DISPLY ,EM47 ;TYPE 'ECC LOGIC ERROR'
25 010634 000404 BR 3$
26 010636 004737 022214 2$: JSR PC,LINE1 ;TYPE FIRST LINE OF ERROR MESSAGE
27 010642 104414 053140 DISPLY ,EM45 ;TYPE 'ECC LOGIC ERROR'
28 010646 004737 022262 3$: JSR PC,LINE2 ;TYPE LINE 2 OF ERROR MESSAGE
29 010652 004737 026122 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
30 010656 012737 000003 001324 MOV #3,RETRY ;RETRY COUNT
31 010664 004737 017040 JSR PC,$RETRY ;RETRY THE ORDER
32 010670 000403 BR 4$ ;RETRY WAS NOT SUCCESSFUL
33 010672 004737 024016 JSR PC,LINE6C ;PRINT 'CORRECTED ON N RETRY(S)'
34 010676 000402 BR 5$ ;FINISH THE ERROR REPORT
35
36 010700 004737 024024 4$: JSR PC,LINE6D ;PRINT 'UNCORRECTABLE AFTER N RETRY(S)'
37 010704 004737 024056 5$: JSR PC,LINE7 ;TYPE LINE 7 OF ERROR MESSAGE
38 010710 000207 RTS PC ;RETURN
39
40          ;THE VALUES IN THE ECC REGISTERS ARE CORRECT, REPORT 'DCK' ERROR
41
42 010712 004737 022214 6$: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
43 010716 104414 051560 DISPLY ,EM21 ;DATA CHECK ERROR
44 010722 7$:
45
46 010722 004737 022262 DCKER1: JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
47 010726 004737 022716 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
48 010732 004737 023366 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
49 010736 004737 016356 JSR PC,PRTBAD ;SEE IF BAD SECTOR TO BE PRINTED
50 010742 012737 110100 001322 MOV #BIT15!BIT12!BIT06,MASK ;LOAD ERROR MASK
51 010750 032760 010000 000202 BIT #BIT12,$RPER1(R0) ;IS IT 'DTE' ?
52 010756 001012 BNE 2$ ;BR IF YES
53 010760 032760 000100 000202 BIT #BIT06,$RPER1(R0) ;IS IT 'ECH' ?
54 010766 001403 BEQ 1$ ;BR IF NO
55 010770 004737 011174 JSR PC,13$ ;COMPARE BAD DATA
56 010774 000403 BR 2$
57 010776 004737 023776 1$: JSR PC,LINE6 ;PRINT 'SECTOR IS ECC CORRECTABLE'
58 011002 000460 BR 10$ ;FINISH THE ERROR REPORT
59 011004 012737 000012 001324 2$: MOV #10,RETRY ;RETRY COUNT
63
64 011012 004737 020160 3$: JSR PC,GODRIV ;RETRY
65 011016 005760 000016 4$: TST $STATUS(R0) ;TEST FOR DONE
66 011022 001775 BEQ 4$ ;BR IF NOT DONE
    
```



```

67 011024 100057          BPL      12$          ;BR IF NOT ERROR
68 011026 032760 000200 000016  BIT      #BIT7,$STATUS(R0) ;SEE IF ORDER TERMINATED NORMALLY
69 011034 001006          BNE      5$          ;BR IF NOT
70 011036 004737 026122  JSR      PC,INCTOT      ;INCREMENT TOTAL ERROR COUNT
71 011042 104414 055570  DISPLY   ,LIN8M        ;'DIFFERENT ERROR DURING RETRY'
72 011046 000137 007636  JMP      ERPRC1        ;SEE WHICH ERROR
73
74 011052 033760 001322 000202 5$:  BIT      MASK,$RPER1(R0) ;LOOK AT CURRENT ERROR
75 011060 001412          BEQ      7$          ;BR IF DIFFERENT ERROR
76 011062 032760 010100 000202  BIT      #BIT12:BIT6,$RPER1(R0) ;'ECH' OR 'DTE' STILL SET ?
77 011070 001421          BEQ      9$          ;BR IF NEITHER SET
78 011072 105237 001325  INCB     RETRY+1        ;INCREMENT RETRY COUNT
79 011076 123737 001324 001325  CMPB    RETRY,RETRY+1  ;DONE ?
80 011104 001342          BNE      3$          ;BR IF NOT
91 011106 004737 024274          7$:  JSR      PC,LINE8      ;PRINT LINE 8 OF ERROR MESSAGE
92 011112 004737 026026          8$:  JSR      PC,INCHRD     ;INCREMENT 'HARD' ERROR COUNT
93 011116 004737 026122          JSR      PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
94 011122 004737 024056          JSR      PC,LINE7     ;PRINT LINE 7 OF ERROR MESSAGE
95 011126 004737 016356          JSR      PC,PRTBAD    ;PRINT THE BAD SECTOR
96 011132 000436          BR       15$         ;CLEAN UP AND RETURN
97
98 011134 004737 023776          9$:  JSR      PC,LINE6     ;PRINT 'SECTOR IS ECC CORRECTABLE'
99 011140 004737 023734          JSR      PC,LINE5B    ;PRINT LINE 5B OF THE ERROR MESSAGE
100 011144 004737 026002         10$:  JSR      PC,INCSOF    ;INCREMENT 'SOFT' ERROR COUNT
101 011150 004737 015534          JSR      PC,ECC       ;CORRECT THE ERROR USING ECC AND CHECK IT
102 011154 000407          BR       13$         ;COMPARE THE BUFFER
103
104 011156 004737 024024         11$:  JSR      PC,LINE6D   ;PRINT 'UNCORRECTABLE AFTER N RETRY(S)'
105 011162 000753          BR       8$          ;INCREMENT ERROR COUNT
106
107 011164 004737 024010         12$:  JSR      PC,LINE6A   ;PRINT 'SECTOR READ CORRECTLY AFTER N RETRY(S)'
108 011170 004737 026002          JSR      PC,INCSOF    ;INCREMENT 'SOFT' ERROR COUNT
109 011174 012737 000001 001352 13$:  MOV      #1,FRSTER     ;SET PROCESSING 'DCKER' INDICATOR
110 011202 004737 014136          JSR      PC,COMPARD   ;COMPARE THE BUFFER
111 011206 105737 001353          TSTB    FRSTER+1     ;ERROR IN COMPARE ?
112 011212 100406          BMI     15$         ;BRANCH IF ERROR
113 011214 004737 026122          JSR      PC,INCTOT   ;INCREMENT TOTAL ERROR COUNT
114 011220 104414 056037          DISPLY  ,LIN9G        ;'DATA COMPARE OK' MESSAGE
115 011224 004737 024056         14$:  JSR      PC,LINE7   ;PRINT LINE 7 OF ERROR MESSAGE
116 011230 000240         15$:  NOP
117 011232 000207          RTS      PC          ;RETURN
118
119          ;WRITE CHECK ERROR PROCESSING
120
121 011234 032760 100000 000202 WCKER: BIT      #BIT15,$RPER1(R0) ;SEE IF 'DCK' SET ALSO
122 011242 001034          BNE     2$          ;BR IF IT IS
123 011244 004737 022214          JSR      PC,LINE1    ;PRINT LINE 1 OF ERROR MESSAGE
124 011250 104414 051664          DISPLY  ,EM23        ;PRINT WCE & DCK NOT SET
125 011254 005037 001322          CLR     MASK        ;CLEAR ERROR MASK
128 011260 004737 022262          JSR      PC,LINE2    ;PRINT LINE 2 OF ERROR MESSAGE
      011264 004737 022716          JSR      PC,LINE3    ;PRINT LINE 3 OF ERROR MESSAGE
      011270 004737 023366          JSR      PC,LINE4    ;PRINT LINE 4 OF ERROR MESSAGE
      011274 004737 023456          JSR      PC,LINE5    ;PRINT LINE 5 OF ERROR MESSAGE
129 011300 004737 026122          JSR      PC,INCTOT   ;INCREMENT TOTAL ERROR COUNT
130 011304 012737 000003 001324  MOV      #3,RETRY     ;RETRY LIMIT
131 011312 004737 017040          JSR      PC,$RETRY   ;RETRY THE OPERATION
132 011316 000403          BR       1$          ;RETRY UNSUCCESSFUL
    
```

```

133
134 011320 004737 024016      JSR    PC,LINE6C      ;PRINT 'CORRECTED ON N RETRY(S)'
135 011324 000532              BR      15$          ;FINISH PROCESSING THE ERROR
136
137 011326 004737 024024      1$:   JSR    PC,LINE6D      ;PRINT 'UNCORRECTABLE AFTER N RETRY(S)'
138 011332 000527              BR      15$          ;FINISH PROCESSING THE ERROR
139
144 011334 012737 000012 001324 2$:   MOV    #10.,RETRY      ;RETRY LIMIT
145 011342 004737 022214      JSR    PC,LINE1       ;PRINT LINE 1 OF ERROR MESSAGE
146 011346 012737 051611 011474      MOV    #EM22,8$       ;ASSUME THAT EM22 WILL BE PRINTED
147 011354 032760 040000 000176      BIT    #BIT14,$RPCS2(R0) ;DID 'WCK' ALSO SET ?
148 011362 001003              BNE    3$             ;BR IF IT DID
149 011364 012737 052512 011474      MOV    #EM37,8$       ;MESSAGE FOR 'DCK' AND 'WCK' NOT DURING
150                                ;WRITE CHECK
151 011372 032760 000100 000202 3$:   BIT    #ECH,$RPER1(R0) ;WRITE CHECK ERROR WITH , ECH SET ?
152 011400 001410              BEQ    4$             ;BRANCH IF NOT
153 011402 012737 053456 011474      MOV    #EM52,8$       ;REPORT 'ECH' ERROR
154 011410 022760 010040 000232      CMP    #10040,$RPEC1(R0) ;OTHERWISE RPEC1=10040
155 011416 001017              BNL    5$             ;REPORT ECC LOGICAL FAILURE
156 011420 000424              BR      7$
157
158 011422 026027 000232 010040 4$:   CMP    $RPEC1(R0),#10040 ;IS POSITION COUNT OVER MAXIMUM ?
159 011430 001012              BHI    5$             ;BR IF YES
160 011432 005760 000232              TST    $RPEC1(R0)     ;POSITION COUNT 0 ?
161 011436 001407              BEQ    5$             ;BR IF YES
162 011440 005760 000234              TST    $RPEC2(R0)     ;VALUE IN PATTERN REGISTER ?
163 011444 001007              BNE    6$             ;BR IF YES
164 011446 012737 053300 011474      MOV    #EM47,8$       ;ELSE, REPORT THE ECC LOGIC FAILURE
165 011454 000403              BR      6$
166 011456 012737 053140 011474 5$:   MOV    #EM45,8$       ;ELSE, REPORT THE ECC LOGIC FAILURE
167 011464 012737 000003 001324 6$:   MOV    #3,RETRY       ;RETRY LIMIT
168
169 011472 104414              7$:   DISPLY      ;TYPE THE ERROR MESSAGE
170 011474 000000      8$:   .WORD    0           ;MESSAGE ADDRESS GOES HERE
173 011476 004737 022262      JSR    PC,LINE2       ;PRINT LINE 2 OF ERROR MESSAGE
      011502 004737 022716      JSR    PC,LINE3       ;PRINT LINE 3 OF ERROR MESSAGE
      011506 004737 023366      JSR    PC,LINE4       ;PRINT LINE 4 OF ERROR MESSAGE
      011512 004737 023456      JSR    PC,LINES       ;PRINT LINE 5 OF ERROR MESSAGE
174
179 011516 004737 020160      9$:   JSR    PC,GODRIV     ;RETRY THE ORDER
180 011522 005760 000016      10$:  TST    $IAIUS(R0)     ;ORDER FINISHED ?
181 011526 001775              BEQ    10$            ;BR IF NOT
182 011530 100405              BMI    11$            ;BR IF ERROR ON ORDER
183 011532 105237 001325      INCB   RETRY+1        ;INCREMENT RETRY COUNT
184 011536 004737 024016      JSR    PC,LINE6C     ;PRINT 'CORRECTED ON N RETRY(S)'
185 011542 000416              BR      13$          ;FINISH ERROR PROCESSING
186
187 011544 105237 001325      11$:  INCB   RETRY+1        ;INCREMENT RETRY COUNT
188 011550 123737 001324 001325      CMPB   RETRY,RETRY+1 ;DONE ?
189 011556 001663              BEQ    1$             ;BR IF AT RETRY LIMIT
190 011560 032760 100000 000202      BIT    #BIT15,$RPER1(R0) ;'DCK' SET
191 011566 001401              BEQ    12$            ;BR IF NOT - DIFFERENT ERROR
192 011570 000752              BR      9$            ;CONTINUE RETRY
193
200 011572 004737 024274      12$:  JSR    PC,LINE8       ;PRINT LINE 8 - 'DIFFERENT ERROR '
201 011576 000405              BR      15$
202 011600 004737 026002      13$:  JSR    PC,INCSOF     ;COUNT AS A SOFT ERROR
    
```

```

203 011604 000402          BR      15$          ;EXIT
204
205 011606 004737 026026  14$:   JSR      PC,INCHRD  ;COUNT AS A HARD ERROR
206 011612 004737 026122  15$:   JSR      PC,INCTOT  ;INCREMENT TOTAL ERROR COUNT
207 011616 004737 024056          JSR      PC,LINE7    ;FINISH THE ERROR MESSAGE
208 011622 000207          RTS      PC      ;RETURN
209
210          ;REPORT 'HCRC' ERROR
211
217 011624 004737 024364  HCRCER: JSR      PC,READDR  ;READ ERROR SECTOR HEADER
219 011630 004737 016746          JSR      PC,READHD  ;GET THE HEAD INFORMATION
220 011634 004737 022214          JSR      PC,LINE1   ;PRINT LINE 1 OF ERROR MESSAGE
221 011640 104414 051537          DISPLY  ,EM20      ;REPORT 'HCRC'
222 011644 004737 022262          JSR      PC,LINE2   ;PRINT LINE 2 OF ERROR MESSAGE
223 011650 004737 022716          JSR      PC,LINE3   ;PRINT LINE 3 OF ERROR MESSAGE
224 011654 004737 023366          JSR      PC,LINE4   ;PRINT LINE 4 OF ERROR MESSAGE
225 011660 004737 023666          JSR      PC,LINESA  ;PRINT THE HEADER INFORMATION
226 011664 032760 040000 000176  BIT      #BIT14,$RPCS2(R0) ;'WCE' ERROR ALSO ?
227 011672 001402          BEQ      1$          ;BR IF NOT
228 011674 004737 023456          JSR      PC,LINES  ;DISPLAY WORDS WHICH CAUSED 'WCE'
229 011700 004737 026002  1$:   JSR      PC,INCSOF  ;INCREMENT 'SOFT' ERROR COUNT
230 011704 004737 026122          JSR      PC,INCTOT  ;INCREMENT TOTAL ERROR COUNT
231 011710 012737 000400 001322  MOV      #BIT8,MASK  ;SET ERROR MASK
232 011716 012737 000003 001324  MOV      #3,RETRY   ;RETRY LIMIT
233 011724 004737 017040          JSR      PC,$RETRY  ;RETRY ORDER
234 011730 000405          BR      2$          ;RETRY NOT SUCCESSFUL
235 011732 004737 024016          JSR      PC,LINE6C  ;PRINT 'CORRECTED ON N RETRY(S)'
236 011736 004737 024056          JSR      PC,LINE7   ;PRINT LINE 7 OF ERROR MESSAGE
237 011742 000404          BR      3$          ;EXIT
238 011744 004737 024024  2$:   JSR      PC,LINE6D  ;PRINT 'UNCORRECTABLE AFTER N RETRY(S)'
239 011750 004737 024056          JSR      PC,LINE7   ;PRINT LINE 7 OF ERROR MESSAGE
240 011754 000207          RTS      PC      ;RETURN
241
242          ;REPORT DRIVE ERROR
243
244 011756 004737 022214  DRVER:  JSR      PC,LINE1  ;PRINT LINE 1 OF ERROR MESSAGE
245 011762 104414 052154          DISPLY  ,EM30      ;REPORT DRIVE ERROR
246 011766 004737 022262          JSR      PC,LINE2   ;PRINT LINE 2 OF ERROR MESSAGE
247 011772 004737 022716          JSR      PC,LINE3   ;PRINT LINE 3 OF ERROR MESSAGE
248 011776 004737 026122          JSR      PC,INCTOT  ;INCREMENT TOTAL ERROR COUNT
249 012002 004737 024056          JSR      PC,LINE7   ;PRINT LINE 7 OF ERROR MESSAGE
250 012006 000207          RTS      PC      ;RETURN
251
252          ;PROCESS FORMAT ('FER') ERROR
253
254 012010 032760 000400 000202  CKFMT:  BIT      #BIT8,$RPER1(R0) ;'HCRC' SET ON ORIGINAL ERROR ?
255 012016 001402          BEQ      1$          ;BR IF NOT SET
256 012020 000137 011624          JMP      HCRCER     ;REPORT HCRC ERROR
257 012024 004737 024364  1$:   JSR      PC,READDR  ;GET CORRECTED TRACK & SECTOR ADDRSSES
258 012030 004737 016746          JSR      PC,READHD  ;READ HEADER
259 012034 032737 000400 050710  BIT      #BIT8,GENREG+RPER1 ;'HCRC' SET WHEN HEADER READ?
260 012042 001002          BNE      2$          ;BR IF 'HCRC' SET
261 012044 000137 012750          JMP      FMTER     ;NO, ERROR IS 'FMT' ONLY
262
268 012050 004737 022214  2$:   JSR      PC,LINE1  ;PRINT LINE 1 OF ERROR MESSAGE
270 012054 104414 051743          DISPLY  ,EM24      ;HEADER READ ERROR - FMT BIT DROPPED UP
271 012060 004737 022262          JSR      PC,LINE2   ;PRINT LINE 2 OF ERROR MESSAGE
    
```

```

272 012064 004737 022716 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
273 012070 004737 023366 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
274 012074 032760 040000 000176 BIT #BIT14,$RPCS2(R0) ;'WCE' ERROR ALSO ?
275 012102 001402 BEQ 3$ ;BR IF NOT
276 012104 004737 023456 JSR PC,LINES ;DISPLAY WORDS WHICH CAUSED 'WCE'
277 012110 004737 023666 3$: JSR PC,LINESA ;DISPLAY HEADER
278 012114 004737 026002 JSR PC,INCSOF ;INCREMENT SOFT ERROR COUNT
279 012120 004737 026122 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
280 012124 012737 000020 001322 MOV #BIT4,MASK ;SET ERROR MASK
281 012132 012737 000003 001324 MOV #3,RETRY ;RETRY LIMIT
282 012140 004737 017040 JSR PC,$RETRY ;RETRY THE ORDER
283 012144 000405 BR 4$ ;RETRY NOT SUCCESSFUL
284 012146 004737 024016 JSR PC,LINE6C ;PRINT 'CORRECTED ON N RETRY(S)'
285 012152 004737 024056 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
286 012156 000404 BR 5$ ;EXIT
287 012160 004737 024024 4$: JSR PC,LINE6D ;PRINT 'UNCORRECTABLE AFTER N RETRY(S)'
288 012164 004737 024056 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
289 012170 000207 5$: RTS PC ;RETURN
290
291 ;PROCESS HEADER COMPARE ('HCE') ERROR
292
293 012172 032760 000400 000202 CKHCE: BIT #BIT8,$RPER1(R0) ;HCRC SET ON ORIGINAL ERROR ?
294 012200 001402 BEQ 1$ ;BR IF NOT SET
295 012202 000137 011624 JMP HCRCER ;REPORT HEADER CRC ERROR
296 012206 004737 024364 1$: JSR PC,READDR ;GET CURRENT SECTOR & TRACK ADDRS
297 012212 004737 016746 JSR PC,READHD ;READ HEADER OF CURRENT SECTOR
298 012216 032737 000400 050710 BIT #BIT8,GENREG+RPER1 ;'HCRC' SET ?
299 012224 001017 BNE 3$ ;BR IF SET
300 012226 013746 063324 MOV CYLNDR,-(SP) ;:PUSH CYLNDR ON STACK
301 012232 042737 150000 063324 BIC #150000,CYLNDR ;CLEAR FORMAT, MFG AND USER BITS FROM HEADER
302 012240 026037 000222 063324 CMP $RPDC(R0),CYLNDR ;CORRECT CYLINDER ?
303 012246 001402 BEQ 2$ ;BR IF IT IS
304 012250 000137 012406 JMP POSER ;REPORT POSITIONING ERROR
305 012254 2$:
306 012254 012637 063324 MOV (SP)+,CYLNDR ;:POP STACK INTO CYLNDR
307 012260 000137 013026 JMP HCEER ;REPORT 'HCE' ERROR
308
309 012264 004737 022214 3$: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
310 012270 104414 052011 DISPLY ,EMC5 ;HEADER READ ERROR - 'HCE' SET
311 012274 004737 022262 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
312 012300 004737 022716 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
313 012304 004737 023366 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
314 012310 032760 040000 000176 BIT #BIT14,$RPCS2(R0) ;'WCE' ERROR ALSO ?
315 012316 001402 BEQ 4$ ;BR IF NOT
316 012320 004737 023456 JSR PC,LINES ;DISPLAY WORDS WHICH CAUSED 'WCE'
317 012324 004737 023666 4$: JSR PC,LINESA ;PRINT LINE 5 OF ERROR MESSAGE
318 012330 004737 026002 JSR PC,INCSOF ;INCREMENT SOFT ERROR COUNT
319 012334 004737 026122 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
320 012340 012737 000200 001322 MOV #BIT7,MASK ;SET ERROR MASK
321 012346 012737 000003 001324 MOV #3,RETRY ;RETRY LIMIT
322 012354 004737 017040 JSR PC,$RETRY ;RETRY THE ORDER
323 012360 000405 BR 5$ ;RETRY NOT SUCCESSFUL
324 012362 004737 024016 JSR PC,LINE6C ;PRINT 'CORRECTED ON N RETRY(S)'
325 012366 004737 024056 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
326 012372 000404 BR 6$ ;EXIT
327 012374 004737 024024 5$: JSR PC,LINE6D ;PRINT 'UNCORRECTABLE AFTER N RETRY(S)'
328 012400 004737 024056 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
    
```

```

334 012404 000207      6$:   RTS      PC      ;RETURN
335
336                   ;REPORT POSSIBLE POSITIONING ERROR
337
338 012406 004737 022214 POSER: JSR      PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
339 012412 104414 053413      DISPLY ,EM51      ;PROGRAM DETECTED POSITIONING ERROR
340 012416 004737 022262      JSR      PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
341 012422 004737 022744      JSR      PC,LINE3C     ;PRINT LINE 3C OF ERROR MESSAGE
342 012426 012637 063324      MOV      (SP)+,CYLNR   ;POP STACK INTO CYLNR
343 012432 004737 023666      JSR      PC,LINE5A    ;PRINT LINE 5A OF THE ERROR MESSAGE
344 012436 004737 026076      JSR      PC,INCMIS    ;INCREMENT MISPOSITIONING COUNT
345 012442 004737 026122      JSR      PC,INCTOT    ;INCREMENT TOTAL ERROR COUNT
346 012446 004737 024176      JSR      PC,LINE7A    ;PRINT LINE 7A OF ERROR MESSAGE
347 012452 004737 016630      JSR      PC,RECALT    ;RECALIBRATE
348 012456 000207      RTS      PC      ;EXIT
349
350                   ;REPORT 'OPI' ERROR
351
357 012460 004737 022214 OPIER: JSR      PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
359 012464 104414 052206      DISPLY ,EM31      ;'OPI' ERROR
360 012470 004737 022262      JSR      PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
361 012474 004737 022716      JSR      PC,LINE3      ;PRINT LINE 3 OF ERROR MESSAGE
362 012500 004737 023366      JSR      PC,LINE4      ;PRINT LINE 4 OF ERROR MESSAGE
363 012504 004737 026002      JSR      PC,INCSOF    ;COUNT 'OPI' AS A SOFT DATA ERROR
364
365 012510 004737 026122      JSR      PC,INCTOT    ;INCREMENT TOTAL ERROR COUNT
366 012514 012737 020000 001322 MOV      #BIT13,MASK ;ERROR MASK
367 012522 012737 000003 001324 OPIER1: MOV      #3,RETRY ;RETRY LIMIT
368 012530 004737 017040      JSR      PC,$RETRY    ;RETRY THE ORDER
369 012534 000405      BR      1$           ;RETRY UNSUCCESSFUL
370 012536 004737 024016      JSR      PC,LINE6C    ;PRINT 'CORRECTED ON N RETRY(S)'
371 012542 004737 024056      JSR      PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
372 012546 000207      RTS      PC      ;EXIT
373 012550 004737 024024 1$:   JSR      PC,LINE6D    ;PRINT 'UNCORRECTABLE AFTER N RETRY(S)'
374 012554 004737 024056      JSR      PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
375 012560 000207      RTS      PC      ;RETURN
376
377                   ;REPORT 'DTE' ERROR
378
385 012562 004737 022214 DTEER: JSR      PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
387 012566 104414 052251      DISPLY ,EM32      ;'DTE' ERROR
388 012572 000137 010722      JMP      DCKER1      ;FINISH PROCESSING THE 'DTE' ERROR
389
390                   ;REPORT 'PAR' ERROR
391
392 012576 004737 022214 PARER: JSR      PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
393 012602 104414 052304      DISPLY ,EM33      ;REPORT 'PAR'
394 012606 004737 022262      JSR      PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
395 012612 004737 023022      JSR      PC,LINE3E    ;PRINT LINE 3E OF ERROR MESSAGE
396 012616 004737 023366      JSR      PC,LINE4      ;PRINT LINE 4 OF ERROR MESSAGE
397 012622 004737 026122      JSR      PC,INCTOT    ;INCREMENT TOTAL ERROR COUNT
398 012626 012737 000010 001322 MOV      #BIT03,MASK ;ERROR MASK
399 012634 012737 000003 001324 MOV      #3,RETRY ;RETRY LIMIT
400 012642 004737 017040      JSR      PC,$RETRY    ;RETRY ORDER
401 012646 000405      BR      2$           ;RETRY UNSUCCESSFUL
402 012650 004737 024016      JSR      PC,LINE6C    ;PRINT 'CORRECTED ON N RETRY(S)'
403 012654 004737 024056 1$:   JSR      PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
    
```

```

404 012660 000207
405 012662 004737 024024
406 012666 000772
407
408
409
410 012670 004737 022214
411 012674 104414 052423
412 012700 004737 022262
413 012704 004737 023110
414 012710 004737 026122
415 012714 004737 024056
416 012720 000207
417
418
419
420 012722 004737 022214
421 012726 104414 052461
422 012732 004737 022262
423 012736 004737 026122
424 012742 004737 024056
425 012746 000207
426
427
428
429 012750 004737 022214
430 012754 104414 052072
431 012760 004737 022262
432 012764 004737 022716
433 012770 004737 023366
434 012774 032760 040000 000176
435 013002 001402
436 013004 004737 023456
437 013010 004737 023666
438 013014 004737 026122
439 013020 004737 024056
440 013024 000207
441
442
443
444 013026 004737 022214
445 013032 104414 052117
446 013036 004737 022262
447 013042 004737 022716
448 013046 004737 023366
449 013052 032760 040000 000176
450 013060 001402
451 013062 004737 023456
452 013066 004737 023666
453 013072 004737 026122
454 013076 004737 024056
455 013102 000207
456
457
458
459 013104 004737 022214
460 013110 104414 052564

```

```

                RTS      PC
2$:             JSR      PC,LINE6D
                BR       1$
                ;EXIT
                ;PRINT 'UNCORRECTABLE AFTER N RETRY(S)'
                ;FINISH ERROR MESSAGE

;REPORT 'IAE' ERROR
IAEER:         JSR      PC,LINE1
                DISPLY  ,EM35
                ;PRINT LINE 1 OF ERROR MESSAGE
                ;REPORT 'IAE'
                JSR      PC,LINE2
                ;PRINT LINE 2 OF ERROR MESSAGE
                JSR      PC,LINE3F
                ;PRINT LINE 3F OF ERROR MESSAGE
                JSR      PC,INCTOT
                ;INCREMENT TOTAL ERROR COUNT
                JSR      PC,LINE7
                ;PRINT LINE 7 OF ERROR MESSAGE
                RTS      PC
                ;RETURN

;REPORT 'WLE' ERROR
WLEER:         JSR      PC,LINE1
                DISPLY  ,EM36
                ;PRINT LINE 1 OF ERROR MESSAGE
                ;REPORT 'WLE'
                JSP      PC,LINE2
                ;PRINT LINE 2 OF ERROR MESSAGE
                JSR      PC,INCTOT
                ;INCREMENT TOTAL ERROR COUNT
                JSR      PC,LINE7
                ;PRINT LINE 7 OF ERROR MESSAGE
                RTS      PC
                ;RETURN

;REPORT FORMAT ERROR
FMTER:         JSR      PC,LINE1
                DISPLY  ,EM26
                ;PRINT LINE 1 OF ERROR MESSAGE
                ;FORMAT ERROR
                JSR      PC,LINE2
                ;PRINT LINE 2 OF ERROR MESSAGE
                JSR      PC,LINE3
                ;PRINT LINE 3 OF ERROR MESSAGE
                JSR      PC,LINE4
                ;PRINT LINE 4 OF ERROR MESSAGE
                BIT      #BIT14,$RPCS2(R0) ;'WCE' ERROR ALSO ?
                BEQ      1$
                ;BR IF NOT
                JSR      PC,LINES5
                ;DISPLAY WORDS WHICH CAUSED 'WCE'
1$:            JSR      PC,LINE5A
                ;PRINT LINE 5A OF ERROR MESSAGE
                JSR      PC,INCTOT
                ;INCREMENT TOTAL ERROR COUNT
                JSR      PC,LINE7
                ;PRINT LINE 7 OF ERROR MESSAGE
                RTS      PC

;REPORT HEADER COMPARE ERROR
HCEER:         JSR      PC,LINE1
                DISPLY  ,EM27
                ;PRINT LINE 1 OF ERROR MESSAGE
                ;HEADER COMPARE ERROR
                JSR      PC,LINE2
                ;PRINT LINE 2 OF ERROR MESSAGE
                JSR      PC,LINE3
                ;PRINT LINE 3 OF ERROR MESSAGE
                JSR      PC,LINE4
                ;PRINT LINE 4 OF ERROR MESSAGE
                BIT      #BIT14,$RPCS2(R0) ;'WCE' ERROR ALSO ?
                BEQ      1$
                ;BR IF NOT
                JSR      PC,LINES5
                ;DISPLAY WORDS WHICH CAUSED 'WCE'
1$:            JSR      PC,LINE5A
                ;PRINT LINE 5A OF ERROR MESSAGE
                JSR      PC,INCTOT
                ;INCREMENT TOTAL ERROR COUNT
                JSR      PC,LINE7
                ;PRINT LINE 7 OF ERROR MESSAGE
                RTS      PC
                ;RETURN

;PROCESS CONTPOL/INTERFACE TRANSFER ERROR
TRFER:         JSR      PC,LINE1
                DISPLY  ,EM40
                ;PRINT LINE 1 OF ERROR MESSAGE
                ;RHXX OR UNIBUS TRANSFER ERROR

```

```

461 013114 004737 022262      JSR      PC,LINE2      :PRINT LINE 2 OF ERROR MESSAGE
462 013120 004737 022716      JSR      PC,LINE3      :PRINT LINE 3 OF ERROR MESSAGE
463 013124 004737 023366      JSR      PC,LINE4      :PRINT LINE 4 OF ERROR MESSAGE
464 013130 004737 026122      JSR      PC,INCTOT     :INCREMENT TOTAL ERROR COUNT
465 013134 032760 121400 000176 BIT      #BIT15!BIT13!BIT9!BIT8,$RPCS2(R0) ;'DLT','UPE','MXF','MDPE' SET ?
466 013142 001415              BEQ      2$            :BR IF NONE SET
467 013144 012737 000003 001324 MOV      #3,RETRY     :RETRY LIMIT
468 013152 005037 001322      CLR      MASK         :CLEAR ERROR MASK
469 013156 004737 017040      JSR      PC,$RETRY    :RETRY THE OPERATION
470 013162 000403              BR       1$           :RETURN HERE IF RETRY UNSUCCESSFUL
471 013164 004737 024016      JSR      PC,LINE6C    :PRINT 'CORRECTED ON N RETRY(S)'
472 013170 000402              BR       2$           :FINISH THE ERROR REPORT
473 013172 004737 024024 1$:      JSR      PC,LINE6D    :PRINT 'UNCORRECTABLE AFTER N RETRY(S)'
474 013176 004737 024056 2$:      JSR      PC,LINE7     :PRINT LINE 7 OF ERROR MESSAGE
475 013202 000207      RTS      PC
476
477      ;PROCESS 'SKI' ERRORS
478
479 013204 004737 022214  SKIER:  JSR      PC,LINE1      :PRINT LINE 1 OF ERROR MESSAGE
480 013210 104414 053355      DISPLY  ,EM50         :'SKI' ERROR
481 013214 004737 022262      JSR      PC,LINE2      :PRINT LINE 2 OF ERROR MESSAGE
482 013220 004737 022732      JSR      PC,LINE3B     :PRINT LINE 3B OF ERROR MESSAGE
483 013224 004737 026122      JSR      PC,INCTOT     :INCREMENT TOTAL ERROR COUNT
484 013230 004737 026052      JSR      PC,INCSKI    :INCREMENT 'SKI' OR 'OCYL' ERROR COUNT
485 013234 004737 024176      JSR      PC,LINE7A     :PRINT LINE 7A OF ERROR MESSAGE
486 013240 000207      RTS      PC
487
488      .REPORT WRITE CLOCK FAILURE ('WCF')
489
490 013242 004737 022214  WCFER:  JSR      PC,LINE1      :PRINT LINE 1 OF ERROR MESSAGE
491 013246 104414 052361      DISPLY  ,EM34         :REPORT WRITE CLOCK FAILURE
492 013252 004737 022262      JSR      PC,LINE2      :PRINT LINE 2 OF ERROR MESSAGE
493 013256 004737 022724      JSR      PC,LINE3A     :PRINT LINE 3A OF ERROR MESSAGE
494 013262 004737 023366      JSR      PC,LINE4      :PRINT LINE 4 OF ERROR MESSAGE
495 013266 004737 026122      JSR      PC,INCTOT     :INCREMENT TOTAL ERROR COUNT
496 013272 004737 016356      JSR      PC,PRTBAD     :SEE IF BAD SECTOR TO BE PRINTED
497 013276 012737 000003 001324 MOV      #3,RETRY     :RETRY COUNT
498 013304 012737 000040 001322 MOV      #BIT05,MASK  :ERROR MASK
499 013312 004737 017040      JSR      PC,$RETRY    :RETRY THE ORDER
500 013316 000405              BR       2$           :RETURN HERE IF RETRY UNSUCCESSFUL
501 013320 004737 024016      JSR      PC,LINE6C    :PRINT 'CORRECTED ON N RETRY(S)'
502 013324 004737 024056 1$:      JSR      PC,LINE7     :PRINT LINE 7 OF ERROR MESSAGE
503 013330 000207      RTS      PC
504 013332 004737 024024 2$:      JSR      PC,LINE6D    :PRINT 'UNCORRECTABLE AFTER N RETRY(S)'
505 013336 000772      BR       1$
506
507      ;PROCESS DRIVE UNSAFE ERROR
508
509 013340 004737 022214  UNSAF:  JSR      PC,LINE1      :PRINT LINE 1 OF ERROR MESSAGE
510 013344 104414 053522      DISPLY  ,EM60         :REPORT DRIVE UNSAFE
511 013350 004737 022262      JSR      PC,LINE2      :PRINT LINE 2 OF ERROR MESSAGE
512 013354 004737 022716      JSR      PC,LINE3      :PRINT LINE 3 OF ERROR MESSAGE
513 013360 004737 026122      JSR      PC,INCTOT     :INCREMENT TOTAL ERROR COUNT
514 013364 012737 040000 001322 MOV      #BIT14,MASK  :RETRY MASK
515 013372 012737 000003 001324 MOV      #3,RETRY     :RETRY COUNT
516 013400 004737 017040      JSR      PC,$RETRY    :RETRY THE ORDER
517 013404 000403              BR       1$           :RETRY WAS UNSUCCESSFUL
    
```

```

518 013406 004737 024016          JSR    PC,LINE6C      ;PRINT 'CORRECTED ON N RETRY(S)'
519 013412 000404                    BR     2$             ;CONTINUE WITH ERROR REPORT
520 013414 004737 024024          1$:  JSR    PC,LINE6D      ;PRINT 'UNCORRECTABLE AFTER N RETRY(S)'
521 013420 000137 031302          JMP    DROP           ;DROP UNSAFE DRIVE
522
523 013424 004737 024056          2$:  JSR    PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
524 013430 000207                    RTS     PC             ;RETURN
525
526                                ;REPORT EARLY WARNING ERROR
527
528 013432 004737 022214          EWNERR: JSR    PC,LINE1      ;SKIP LINES
529 013436 000240                    NOP
530 013440 022760 000002 000230    BIT    #BIT1,$RPER3(RO) ;'TPE' SET?
531 013446 001403                    BEQ    1$             ;BR IF NO
532 013450 104414 053545          DISPLY ,EM70          ;PRINT TEMPERATURE WARNING ERROR
533 013454 000411                    BR     3$             ;TYPE REMAINING ERROR MESSAGE
534 013456 032760 000004 000230    1$:  BIT    #BIT2,$RPER3(RO) ;'AIR' SET?
535 013464 001403                    BEQ    2$             ;BR IF NO
536 013466 104414 053624          DISPLY ,EM71          ;'AIR SYSTEM WARNING ERROR'
537 013472 000402                    BR     3$
538 013474 104414 053702          2$:  DISPLY ,EM72          ;'EARLY WARNING ERROR,AIR TPE NOT SET'
539 013500 004737 022262          3$:  JSR    PC,LINE2      ;ERROR MESSAGE
540 013504 004737 022716          JSR    PC,LINE3      ;PRINT OUT
541 013510 004737 026122          JSR    PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
542 013514 012737 000007 001322    MOV    #7,MASK        ;SET UP ERROR MASK
543 013522 012737 000003 001324    MOV    #3,RETRY       ;RETRY COUNT
544 013530 004737 017040          JSR    PC,$RETRY     ;RETRY THE ORDER
545 013534 000403                    BR     4$             ;RETRY UNSUCCESSFUL RETURN
546 013536 004737 024016          JSR    PC,LINE6C      ;PRINT 'CORRECTED ON N RETRY(S)'
547 013542 000207                    RTS     PC             ;RETURN
548 013544 004737 024024          4$:  JSR    PC,LINE6D      ;PRINT 'UNCORRECTABLE AFTER N RETRY(S)'
549 013550 000137 031302          JMP    DROP           ;DROP DRIVE
550
551                                ;REPORT AN 'UNKNOWN' DATA PATTERN
552
553                                ;R1 = POINTER FIRST WORD OF DATA BUFFER
554
555 013554 005737 001334          NOMTCH: TST    CFLAG      ;WAS (^C) TYPED?
556 013560 001060                    BNE    5$             ;BR IF YES
557 013562 105737 001352          1$:  TSTB   FRSTER       ;FIRST ERROR IN THE SECTOR ?
558 013566 001013                    BNE    2$             ;BR IF NOT OR IF PROCESSING 'DCKER'
559 013570 004737 022214          JSR    PC,LINE1      ;TYPE LINE 1 OF ERROR MESSAGE
560 013574 104414 052747          DISPLY ,EM43          ;'CAN'T MATCH DATA WITH PATTERN'
561 013600 004737 022262          JSR    PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
562 013604 004737 022724          JSR    PC,LINE3A     ;PRINT LINE 3A OF ERROR MESSAGE
563 013610 004737 023366          JSR    PC,LINE4      ;PRINT LINE 4 OF ERROR MESSAGE
564 013614 000404                    BR     3$             ;CONTINUE PROCESSING ERROR
565 013616 104414 052747          2$:  DISPLY ,EM43          ;'CAN'T MATCH DATA WITH PATTERN'
566 013622 104414 001203          DISPLY ,%CRLF         ;CR-LF
567 013626 104414 055735          3$:  DISPLY ,LIN91       ;HEADER FOR DATA PRINTOUT
568 013632 005737 001334          4$:  TST    CFLAG      ;WAS (^C) TYPED?
569 013636 001031                    BNE    5$             ;BR IF YES
570 013640 010146                    MOV    R1,-(SP)       ;ADDRESS OF WORD POSITION
571 013642 004737 024306          JSR    PC,LIN0CT     ;TYPE WORD NUMBER
572 013646 005237 001374          INC    WRDPOS        ;INC WORD POSITION
573 013652 013745 001374          MOV    WRDPOS,-(SP)  ;PUT THE WORD POS ON THE STACK
574 013656 004737 033332          JSR    PC,$SB2D      ;CONVERT IT TO DECIMAL

```



```

581 013662 004537 032662      JSR      R5,$REPLZ      ;TYPE IT (REPLACE LEADING ZEROS)
582 013666 000006      .WORD      6           ;TYPE 6 DIGITS
583 013670 104414 057711      DISPLY    ,PERIOD      ;TYPE ' '
584 013674 104414 056530      DISPLY    ,BLNKS2     ;TYPE 2 BLANKS
585 013700 012146      MOV      (R1)+,-(SP)   ;ADDRESS OF DATA WORD
586 013702 004737 024306      JSR      PC,LINOC1    ;TYPE DATA
587 013706 104414 001203      DISPLY    ,$CRLF      ;CR-LF
588 013712 023727 001374 000003      CMP      WRDPOS,#3    ;DONE ALL WORDS ?
589 013720 001344      BNE      4$          ;BR IF NO
590 013722 062701 000770      5$:      ADD      #<252 *2.>,R1 ;INCREMENT BUFFER POINTER - 4 MATCHING WORDS
591 013726 005002      CLR      R2          ;CLEAR 'WORDS TO COMPARE' COUNT IN R2
592 013730 012737 177777 001352      MOV      #-1,FRSTER   ;SET ERROR FOUND INDICATOR
593 013736 013737 001460 001364      MOV      CMLMT,LIMIT  ;RESET THE COMPARE ERROR TIMEOUT LIMIT
594 013744 000207      RTS      PC          ;RETURN
595
596      ;CHECK ERROR BITS IN THE RMX & RP07 REGISTERS
597
598 013746 032760 060000 000166      CKERR:   BIT      #60000,$RPCS1(R0) ;SEE IF 'TRE' OR 'MCPE' SET
599 013754 001015      BNE      1$          ;BR IF EITHER SET
600 013756 032760 177400 000176      BIT      #177400,$RPCS2(R0) ;SEE IF ERROR BITS IN CS2 SET
601 013764 001011      BNE      1$          ;BR IF ANY SET
602 013766 005760 000202      TST      $RPER1(R0)  ;ANY BITS SET IN ER1
603 013772 001006      BNE      1$          ;BR IF ANY SET
604 013774 005760 000230      TST      $RPER3(R0)  ;ANY BITS SET IN ER3 ?
605 014000 001003      BNE      1$          ;BR IF YES
606
607 014002 005760 000226      TST      $RPER2(R0)  ;ANY BIT SET IN ER2
608 014006 000416      BR       2$          ;BR IF NO
609
610 014010 004737 022214      1$:      JSR      PC,LINE1    ;PRINT LINE 1 OF ERROR MESSAGE
611 014014 104414 053043      DISPLY    ,EM44      ;ERROR BITS SET, BUT NO ERROR BITS SET
612 014020 004737 022262      JSR      PC,LINE2    ;PRINT LINE 2 OF ERROR MESSAGE
613 014024 004737 022716      JSR      PC,LINE3    ;PRINT LINE 3 OF ERROR MESSAGE
614 014030 004737 023366      JSR      PC,LINE4    ;PRINT LINE 4 OF ERROR MESSAGE
615 014034 004737 026122      JSR      PC,INCTOT   ;INCREMENT TOTAL ERROR COUNT
616 014040 004737 024056      JSR      PC,LINE7    ;PRINT LINE 7 OF ERROR MESSAGE
617 014044 000207      2$:      RTS      PC          ;RETURN
618
619      ;CHECK BUS ADDRESS REGISTER & WORD COUNT REGISTER
620
621 014046 005760 000170      CKBUS:   TST      $RPWC(R0) ;CHECK WORD COUNT
622 014052 001010      BNE      1$          ;BR IF NOT ZERO
623 014054 016046 000020      MOV      $WRDL(R0),-(SP) ;WORD LENGTH
624 014060 006316      ASL      (SP)        ;CHANGE INTO BYTE COUNT
625 014062 066016 000006      ADD      $BUF(R0),(SP) ;ADD THE STARTING LOCATION
626 014066 022660 000172      CMP      (SP)+,$RPBA(R0) ;BUFFER ADDRESS PROPER ?
627 014072 001416      BEQ      2$          ;BR IF OK
628 014074 004737 022214      1$:      JSR      PC,LINE1    ;PRINT LINE 1 OF ERROR MESSAGE
629 014100 104414 052622      DISPLY    ,EM41      ;BUS ADDRESS OR WORD COUNT INCORRECT
630 014104 004737 022262      JSR      PC,LINE2    ;PRINT LINE 2 OF ERROR MESSAGE
631 014110 004737 022754      JSR      PC,LINE3D   ;PRINT LINE 3D OF ERROR MESSAGE
632 014114 004737 023366      JSR      PC,LINE4    ;PRINT LINE 4 OF ERROR MESSAGE
633 014120 004737 026122      JSR      PC,INCTOT   ;INCREMENT TOTAL ERROR COUNT
634 014124 004737 024056      JSR      PC,LINE7    ;PRINT LINE 7 OF ERROR MESSAGE
635 014130 000207      2$:      RTS      PC
    
```

```

1          ;COMPARE THE BUFFER
2
3 014132 005037 001352      CMPAR: CLR      FRSTER      ;CLEAR 'FIRST ERROR' AND 'NO DCK' INDICATOR
4
5 014136 132760 000004 000024  CMPARD: BITB   #4,$CODE(R0) ;SEE IF READ COMMAND
6 014144 001001          BNE      1$          ;BR IF IT IS
7 014146 000207          RTS      PC          ;RETURN
8
9 014150 005037 001362      1$:    CLR      ERCTR      ;CLEAR THE ERROR COUNTER
10 014154 016001 000006      MOV     $BUF(R0),R1    ;BUFFER ADDRESS
11 014160 016037 000020 001366  MOV     $WRDL(R0),CMCNT ;WORD COUNT TO WORKING LOCATION
12 014166 066037 000170 001366  ADD     $RPWC(R0),CMCNT ;CALCULATE WORDS TRANSFERED
13 014174 001001          BNE      2$
14 014176 000207          RTS      PC          ;EXIT--NO WORDS XFERED
15
16 014200 016037 000012 001370  2$:    MOV     $CYL(R0),CMCYL ;CYLINDER ADDRESS WORKING LOCATION
17 014206 052737 150000 001370  BIS     #150000,CMCYL  ;SET MFG, USER AND FMT BITS
18 014214 016037 000010 001372  MOV     $SEC(R0),CMSEC ;SECTOR & TRACK ADDRESSES TO WORKING LOCNS
19 014222 013737 001460 001364  MOV     CMPLMT,LIMIT  ;DISPLAY LIMIT
20 014230 005237 001364      INC     LIMIT        ;CONVERT PARAMETER INTO LIMIT VALUE
21 014234 012737 177777 001350  CMSTR: MOV     #-1,ZROIND  ;CLEAR THE 'ZERO'S' INDICATOR
22 014242 005037 001356      CLR     SAVER1       ;CLEAR THE R1 SAVE WORD
23 014246 005037 001360      CLR     SAVER5       ;CLEAR THE R5 SAVE WORD
24 014252 023760 001366 000022  CMP     CMCNT,$SSEC(R0) ;IS BUFFER SIZE GREATER THAN ONE SECTOR ?
25 014260 101005          BHI     1$          ;BR IF IT IS
26 014262 013702 001366      MOV     CMCN1,R2     ;LESS THAN, USE REMAINING BUFFER
27 014266 005037 001366      CLR     CMCNT        ;SET COUNTER TO 0
28 014272 000405          BR      2$
29 014274 016002 000022      1$:    MOV     $SSEC(R0),R2 ;COMPARE SECTOR
30 014300 166037 000022 001366  SUB     $SSEC(R0),CMCNT ;DECREMENT WORD COUNT
31 014306 126027 000024 000005  2$:    CMPB   $CODE(R0),#5 ;READ HEADER & DATA?
32 014314 001034          BNE     CMDAT        ;BR IF NOT
33
34          ;COMPARE HEADER WORDS
35
36 014316 012705 001370      CMHED: MOV     #CMCYL,R5 ;ADDRESS OF COMPARING CYLINDER
37 014322 052711 150000      BIS     #150000,(R1)  ;SET BITS INCASE BAD SECTOR ENCOUNTER
38 014326 022521          CMP     (R5)+,(R1)+  ;CHECK CYLINDER
39 014330 001405          BEQ     1$          ;BR IF COMPARE OK
40 014332 012737 177777 001374  MOV     #-1,WRDPOS   ;INDICATE WORD POSITION IS HEADER WRD 1
41 014340 004737 014374      JSR    PC,CMSTR2    ;REPORT ERROR
42 014344 022521          CMP     (R5)+,(R1)+  ;COMPARE SECTOR/TRACK
43 014346 001405          BEQ     2$          ;BR IF EQ
44 014350 012737 177776 001374  MOV     #-2,WRDPOS   ;INDICATE WORD POSITION IS HEADER WRD 2
45 014356 004737 014374      JSR    PC,CMSTR2    ;REPORT ERROR
46 014362 162702 000002      2$:    SUB     #2,R2     ;SUBTRACT HEADER LENGTH FROM SIZE
47 014366 003007          BGT     CMDAT        ;BR IF NOT FINISHED
48 014370 000137 015010      JMP     CMPRX       ;COMPARE THE DATA PORTION
49
50 014374 005237 001362      CMSTR2: INC     ERCTR   ;INCREMENT THE ERROR COUNT
51 014400 004737 015016      JSR    PC,CMPRT    ;REPORT THE COMPARISON ERROR
52 014404 000207          RTS      PC        ;CHECK THE REST OF THE HEADER
53
54          ;COMPARE DATA FIELD
55
56 014406 012737 177777 001374  CMDAT: MOV     #-1,WRDPOS ;INITIALIZE WORD POSITION
57 014414 004737 015444      JSR    PC,MATCH    ;FIND THE PATTERN

```

58	014420	000403				BR	1\$:FOUND A PATTERN
59	014422	004737	013554			JSR	PC,NOMTCH		:RETURN HERE IF NO MATCH WITH PATTERN MADE
60	014426	000463				BR	7\$:BYPASS COMPARE ROUTINE
61									
62	014430	011405			1\$:	MOV	(R4),R5		:ADDRESS OF PATTERN ADDRESS IN R4
63	014432	012703	000020			MOV	#16.,R3		:R3 IS PATTERN POS COUNTER
64	014436	005237	001374		2\$:	INC	WRDPOS		:INCREMENT TO CURRENT WORD POSITION
65	014442	022125				CMP	(R1)+,(R5)+		:COMPARE BUFFER WITH PATTERN
66	014444	001016				BNC	4\$:BR IF NOT EQUAL
67	014446	005737	001362			TST	ERCTR		:ERRORS DETECTED ?
68	014452	001406				BEQ	3\$:BR IF NO ERRORS
69	014454	032777	000010	164472		BIT	#SW3,@SWR		:SWITCH 3 SET ?
70	014462	001402				BEQ	3\$:BR IF NOT SET
71	014464	004737	015016			JSR	PC,CMPRT		:DISPLAY THE WORD
72									
73	014470	005302			3\$:	DEC	R2		:DECREMENT SIZE COUNT
74	014472	003441				BLE	7\$:BR WHEN AT END
75	014474	005303				DEC	R3		:DECREMENT PATT POS COUNT
76	014476	001357				BNE	2\$:BR IF NOT AT END OF PATT
77	014500	000753				BR	1\$:RESTART THE PATTERN
78									
79	014502	005761	177776		4\$:	TST	-2(R1)		:IS MISCOMPARED CHARACTER=0
80	014506	001410				BEQ	5\$:BR IF YES
81	014510	012737	177777	001350		MOV	#-1,ZROIND		:SET NON-ZERO MISCOMPARED INDICATOR
82	014516	005237	001362			INC	ERCTR		:INCREMENT THE ERROR COUNTER
83	014522	004737	015016			JSR	PC,CMPRT		:REPORT ERROR
84	014526	000760				BR	3\$:CONTINUE COMPARE
85									
86	014530	105737	001352		5\$:	TSTB	FRSTER		:FIRST ERROR?
87	014534	100412				BMI	6\$:BR IF NOT
88	014536	005037	001350			CLR	ZROIND		:SET THE ZERO INDICATOR
89	014542	010137	001356			MOV	R1,SAVER1		:SAVE CURRENT R1
90	014546	010537	001360			MOV	R5,SAVER5		:SAVE CURRENT R5
91	014552	013737	001374	001354		MOV	WRDPOS,SAVPOS		:SAVE CURRENT WORD POSITION
92	014560	000743				BR	3\$:CONTINUE COMPARE
93	014562	005737	001350		6\$:	TST	ZROIND		:ANY MISCOMPARISONS NOT ZEROS ?
94	014566	001740				BEQ	3\$:BR IF NONE-ALL ERRORS=ZERO
95	014570	004737	015016			JSR	PC,CMPRT		:REPORT ERROR
96	014574	000735				BR	3\$:CONTINUE COMPARING
97									
98	014576	126027	000024	000005	7\$:	CMPB	\$CODE(R0),#5		:READ HEAD AND DATA ?
99	014604	001414				BEQ	9\$:YES
100	014606	013702	001366		8\$:	MOV	CMCNT,R2		:SET COUNTER = REMAIN BUFFER LENGTH
101	014612	020227	000004			CMP	R2,#4		:IS THERE AT LEAST 4 WORDS TO MATCH PATTERN ?
102	014616	002474				BLT	CMPRX		:BR IF NO
103	014620	162737	000400	001366		SUB	#256.,CMCNT		:GREATER THAN A SECTOR ?
104	014626	003667				BLE	CMDAT		:NO,RETURN TO COMPARE LOOP
105	014630	012702	000400			MOV	#256.,R2		:SET COUNTER =SECTOR SIZE
106	014634	000664				BR	CMDAT		:RETURN TO COMPARE LOOP
107									
108	014636	023727	001366	000002	9\$:	CMP	CMCNT,#2		:IS THERE AT LEAST 2 WORDS TO COMPARE HEADER ?
109	014644	002461				BLT	CMPRX		:BR IF NO
110	014646	105237	001372			INCB	CMSEC		:INCREMENT COUNTER
111	014652	123760	001372	000152		CMPB	CMSEC,SECLMT(R0)		:MAX SECTOR # ?
112	014660	101424				BLOS	10\$:NO
113	014662	105037	001372			CLRB	CMSEC		:RESET SECTOR #
114	014666	105237	001373			INCB	CMTRK		:INCREMENT TRACK #

```

115 014672 123760 001373 000154      CMPB      CMTRK,TRKLMT(RO)      ;MAX TRACK # ?
116 014700 101414      BLOS      10$                  ;NO
117 014702 105037 001373      CLR      CMTRK                  ;RESET TRACK #
118 014706 005237 001370      INC      CMCYL                  ;INCREMENT CYLINDER NUMBER
119 014712 013746 001370      MOV      CMCYL, -(SP)          ;GET COMPARING CYLINDER
120 014716 042716 150000      BIC      #150000, (SP)        ;SAVE ONLY THE CYLINDER BITS
121 014722 022660 000150      CMP      (SP)+, CYLLMT(RO)    ;LAST CYLINDER ?
122 014726 101401      BLOS      10$                  ;NO
123 014730 000427      BR       CMPRX                 ;NORMAL RETURN, NOT WRAP AROUND
124
125 014732 012705 001370      10$:    MOV      #CMCYL, R5        ;ADDRESS OF COMPARING CYLINDER
126 014736 052711 150000      BIS      #150000, (R1)        ;SET BITS INCASE BAD SECTOR ENCOUNTER
127 014742 022521      CMP      (R5)+, (R1)+        ;COMPARE 1ST HEADER WORD
128 014744 001405      BEQ      11$                  ;MATCH
129 014746 012737 177777 001374      MOV      #-1, WRDPOS          ;INDICATE WORD POSITION IS HEADER WRD 1
130 014754 004737 014374      JSR      PC, CMSTR2          ;NOT MATCH
131 014760 022521      11$:    CMP      (R5)+, (R1)+        ;SECOND WORD OF HEADER
132 014762 001405      BEQ      12$                  ;MATCH
133 014764 012737 177776 001374      MOV      #-2, WRDPOS          ;INDICATE WORD POSITION IS HEADER WRD 2
134 014772 004737 014374      JSR      PC, CMSTR2          ;NOT MATCH
135
136 014776 162737 000002 001366      12$:    SUB      #2, CMCNT        ;ADJUST WORD COUNT
137 015004 003401      BLE      CMPRX                ;COMPARE IS DONE
138 015006 000677      BR       8$                   ;RETURN TO COMPARE LOOP
139
146 015010 004737 015376      CMPRX:  JSR      PC, ENDCMP    ;PRINT LAST LINE IF ERRORS
147 015014 000207      RTS      PC
149
150      ;TYPE DATA COMPARE ERRORS
151
157 015016 000240      CMPRT:  NOP
159
160 015020 005737 001356      1$:    TST      SAVER1             ;PRINT SAVED VALUES ?
161 015024 001005      BNE      2$                   ;BR IF YES
162 015026 004737 015124      JSR      PC, 4$                ;PRINT INITIAL MESSAGE INFO
163 015032 004737 015206      JSR      PC, 7$                ;PRINT REMAINDER OF MESSAGE
164 015036 000431      BR       3$                   ;EXIT
165 015040      2$:    MOV      R1, -(SP)           ;; PUSH R1 ON STACK
166 015040 010146      MOV      R5, -(SP)           ;; PUSH R5 ON STACK
167 015042 010546      MOV      WRDPOS, -(SP)       ;; PUSH WRDPOS ON STACK
168 015044 013746 001374      MOV      SAVER1, R1          ;DISPLAY SAVED R1
169 015050 013701 001356      MOV      SAVER5, R5          ;DISPLAY SAVED R5
170 015054 013705 001360      MOV      SAVPOS, WRDPOS      ;DISPLAY SAVED WORD POSITION
171 015060 013737 001354 001374      JSR      PC, 4$                ;PRINT INITIAL MESSAGE INFO
172 015066 004737 015124      JSR      PC, 7$                ;PRINT SAVED VALUES
173 015072 004737 015206      CLR      SAVER1              ;CLEAR SAVED REGISTER INDICATORS
174 015076 005037 001356      CLR      SAVER5              ;CLEAR THE OTHER ONE
175 015102 005037 001360      MOV      (SP)+, WRDPOS       ;; POP STACK INTO WRDPOS
176 015106 012637 001374      MOV      (SP)+, R5           ;; POP STACK INTO R5
177 015112 012605      MOV      (SP)+, R1           ;; POP STACK INTO R1
178 015114 012601      JSR      PC, 7$                ;PRINT REMAINDER OF MESSAGE
179 015116 004737 015206      3$:    RTS      PC                ;RETURN
180
177 015124 105737 001352      4$:    TSTB     FRSTER            ;FIRST ERROR ?
178 015130 100443      BMI      9$                   ;BR IF NOT
179 015132 001013      BNE      5$                   ;BR IF FIRST ERROR AND PROCESSING 'DCK' ERROR,

```

```

180
181 015134 004737 022214 JSR PC,LINE1 ;ELSE, FIRST ERROR AND COMPARE ERROR W/O 'DCK'
182 015140 104414 052666 DISPLY ,EM42 ;PRINT LINE 1 OF ERROR MESSAGE
183 015144 004737 022262 JSR PC,LINE2 ;DATA COMPARE ERROR W/O 'DCK' ERROR
184 015150 004737 022724 JSR PC,LINE3A ;PRINT LINE 2 OF ERROR MESSAGE
185 015154 004737 023366 JSR PC,LINE4 ;PRINT LINE 3A OF ERROR MESSAGE
186 015160 000404 BR 6$ ;PRINT LINE 4 OF ERROR MESSAGE
187 ;GO TO TYPE HEADER
188 015162 104414 055613 5$: DISPLY ,LIN9B ;PRINT 'DATA COMPARISON ERRORS'
189 015166 104414 001203 DISPLY ,$CRLF ;CR-LF
190 015172 104414 055642 6$: DISPLY ,LIN9H ;PRINT '
191 ; ADDR WORD EXPCTD RECEVD
192 015176 012737 177777 001352 MOV #-1,FRSTER ;SET FIRST ERROR FLAG
193 015204 000207 RTS PC ;RETURN
194
195 015206 005737 001334 7$: TST CFLAG ;WAS (^C) TYPED?
196 015212 100412 BMI 9$ ;BR IF YES
197 015214 005737 001364 TST LIMIT ;TIMEOUT LIMIT REACHED ?
198 015220 001403 BEQ 8$ ;BR IF IT HAS
199 015222 005337 001364 DEC LIMIT ;DECREMENT LIMIT COUNTER
200 015226 001005 BNE 10$ ;BR IF NOT AT LIMIT
201 015230 032777 000200 163716 8$: BIT #SW07,@SWR ;PRINT ALL DATA COMPARE ERRORS ?
202 015236 001001 BNE 10$ ;BR IF YES
203 015240 000207 9$: RTS PC ;RETURN
204
205 015242 010146 10$: MOV R1,-(SP) ;BUFFER ADDRESS
206 015244 162716 000002 SUB #2,(SP) ;ADJUST ADDRESS
207 015250 004737 024306 JSR PC,LINOC T ;TYPE IT
208 015254 004737 015312 JSR PC,11$ ;GET WORD POSITION
209 015260 016546 177776 MOV -2(R5),-(SP) ;PUT EXPECTED DATA ON THE STACK
210 015264 004737 024306 JSR PC,LINOC T ;TYPE IT
211 015270 104414 056530 DISPLY ,BLNKS2 ;TYPE 2 BLANKS
212 015274 016146 177776 MOV -2(R1),-(SP) ;RECEIVED DATA
213 015300 004737 024306 JSR PC,LINOC T ;TYPE IT
214 015304 104414 001203 DISPLY , $CRLF ;CR-LF
215 015310 000207 RTS PC ;RETURN
216
217 015312 023727 001374 177777 11$: CMP WRDPOS,#-1 ;IS IT HEADER WORD 1 ?
218 015320 001003 BNE 12$ ;BR IF NO
219 015322 104414 057055 DISPLY ,MSHW1 ;TYPE ' HW1 '
220 015326 000420 BR 14$
221 015330 023727 001374 177776 12$: CMP WRDPOS,#-2 ;IS IT HEADER WORD 2 ?
222 015336 001003 BNE 13$ ;BR IF NO
223 015340 104414 057065 DISPLY ,MSHW2 ;TYPE ' HW2 '
224 015344 000411 BR 14$
225 015346 013746 001374 13$: MOV WRDPOS, -(SP) ;PUT THE WORD POS ON THE STACK
226 015352 004737 033332 JSR PC,$SB2D ;CONVERT IT TO DECIMAL
227 015356 004537 032662 JSR R5,$REPLZ ;TYPE IT (REPLACE LEADING ZEROS)
228 015362 000006 .WORD 6 ;TYPE 6 DIGITS
229 015364 104414 057711 DISPLY ,PERIOD ;TYPE '.'
230 015370 104414 056530 14$: DISPLY ,BLNKS2 ;TYPE 2 BLANKS
231 015374 000207 RTS PC
232
233 ;LAST LINE OF COMPARE ERROR REPORTING
234
235 015376 105737 001353 ENDCMP: TSTB FRSTER+1 ;ANY COMPARE ERRORS FOUND ?
236 015402 001417 BEQ 2$ ;BR IF NOT
    
```

237	015404	005737	001362	TST	ERCTR	:SEE HOW MANY ERRORS
238	015410	001410		BEQ	1\$:BR IF ONLY CAN'T MATCH PATTERN
239	015412	104414	056010	DISPLY	,LINE	: 'NUMBER OF ERRORS='
240	015416	013746	001362	MOV	ERCTR,-(SP)	:NUMBER OF ERRORS
241	015422	004737	024340	JSR	PC,LINDEC	:TYPE IT
242	015426	104414	001203	DISPLY	,\$CRLF	:CR-LF
243	015432	004737	026122	1\$: JSR	PC,INCTOT	:INCREMENT TOTAL ERROR COUNT
244	015436	004737	024056	JSR	PC,LINE7	:PRINT LINE 7 OF ERROR MESSAGE
245	015442	000207		2\$: RTS	PC	:RETURN

```

1          ;ROUTINE TO MATCH THE DATA WITH A PATTERN, ONLY WHEN LOCATION 'PATTERN'
2          ;IS EQUAL TO 0 (RANDOM DATA PATTERN MODE). OTHERWISE, THIS ROUTINE WILL
3          ;RETURN THE ADDRESS OF THE EXPECTED FIXED DATA PATTERN IN R4.
10         ;CALL:
11         :      MOV      #BUFFER,R1      ;BUFFER ADDRESS
12         :      JSR      PC,MATCH
13         :      RETURN1
14         :      RETURN2
15
16 015444 010146 MATCH: MOV      R1,-(SP)      ;SAVE R1 ON THE STACK
21 015446 013704 001476 MOV      PATTERN,R4      ;WAS RANDOM PATTERN ENABLED ?
22 015452 001402 BEQ      1$              ;BR IF YES
23 015454 006304 ASL      R4              ;* 2
24 015456 000416 BR       4$              ;USE KNOWN PATTERN
25 015460 012704 000044 1$: MOV     #4,R4          ;PATTERN TABLE INDEX
26 015464 011601 2$: MOV     (SP),R1      ;RELOAD R1
27 015466 162704 000002 SUB     #2,R4          ;DECREMENT INDEX
28 015472 001413 BEQ      5$              ;BR IF PATTERN NOT MATCH
29 015474 016405 002314 MOV     STNDAT(R4),R5   ;ADDRESS OF PATTERN ADDRESS
30 015500 012703 000004 MOV     #4,R3          ;NUMBER OF LOCATIONS TO CHECK
31 015504 022125 3$: CMP     (R1)+,(R5)+   ;COMPARE THE BUFFER AGAINST THE PATTERN
32 015506 001366 BNE     2$              ;BR IF NOT EQUAL, TRY NEXT PATTERN
33 015510 005303 DEC     R3              ;FINISHED CHECKING?
34 015512 001374 BNE     3$              ;BR IF NOT FINISHED
35 015514 062704 002314 4$: ADD     #STNDAT,R4      ;MAKE PATTERN ADDRESS ABSOLUTE
36 015520 000403 BR       6$              ;EXIT
37 015522 062766 000002 5$: ADD     #2,2(SP)      ;INCREMENT RETURN ADDRESS
38 015530 012601 6$: MOV     (SP)+,R1      ;RESTORE R1
39 015532 000207 RTS      PC          ;RETURN
40
41         ;USE ECC TO CORRECT THE DATA ERROR
42
43 015534 016037 000172 001400 ECC: MOV     $RPBA(RO),ECSEC ;ADDRESS OF LAST LOCN XFERED
44 015542 016046 000170 MOV     $RPWC(RO),-(SP) ;ACT WORDS XFERED (2'S COMP)
45 015546 066016 000020 ADD     $WRDL(RO),(SP) ;ADD WORDS REQUESTED
46 015552 001002 BNE     1$
47 015554 005726 TST     (SP)+          ;RESTORE STACK
48 015556 000207 RTS      PC          ;EXIT--NO WORDS XFERED
49 015560 005043 1$: CLR     -(SP)          ;CLEAR NEXT STACK LOCN
50 015562 016046 000022 MOV     $SSEC(RO),-(SP) ;SECTOR SIZE
51 015566 004737 032046 JSR     PC,$DIV        ;DIVIDE WORDS XFERED BY SECTOR SIZE
52 015572 005716 TST     (SP)          ;PARTIAL SECTOR XFERED ?
53 015574 001413 BEQ      2$              ;BR IF NOT
54 015576 006316 ASL     (SP)          ;CONVERT INTO NUMBER OF BYTES
55 015600 161637 001400 SUB     (SP),ECSEC     ;SUBTRACT SECTOR RESIDUE
56 015604 122760 000005 000024 CMPB   #5,$CODE(RO)   ;WAS OPERATION, READ HEAD & DATA
57 015612 001007 BNE     3$              ;BR IF NOT
58 015614 062737 000004 001400 ADP    #4,ECSEC       ;ADD HEADER SIZE (IN BYTES) BACK IN
59 015622 000403 BR       3$              ;GO ADJUST THE STACK POINTER
60 015624 162737 001000 001400 2$: SUB     #256.*2,ECSEC ;SUBTRACT SECTOR DATA FIELD SIZE (IN BYTES)
61 015632 062706 000004 3$: ADD     #4,SP          ;ADJUST THE STACK POINTER
62 015636 016037 000232 001376 MOV     $RPEC1(RO),ECBIT ;ECC POSITION COUNT
63 015644 005337 001376 DEC     ECBIT          ;ADJUST BIT POSITION
64 015650 013737 001376 001406 MOV     ECBIT,ECWRD    ;LOAD THE WORD COUNT LOCATION
65 015656 042737 177760 001376 BIC    #*C17,ECBIT    ;SAVE THE BIT OFFSET COUNT
66 015664 042737 000017 001406 BIC    #17,ECWRD      ;CLEAR THE BIT OFFSET
67 015672 006237 001406 ASR     ECWRD          ;CHANGE TO BYTE COUNT(DIVIDE BY 2)
    
```

68	015676	006237	001406			ASR	ECWRD	:CHANGE TO BYTE COUNT(DIVIDE BY 4)
69	015702	006237	001406			ASR	ECWRD	:CHANGE TO BYTE COUNT(DIVIDE BY 8.)
70	015706	104414	056065			DISPLY	,LIN10A	: 'ERROR BURST BEGINS AT'
71	015712	013746	001406			MOV	ECWRD,-(SP)	:PUT THE WORD COUNT ON THE STACK
72	015716	006216				ASR	(SP)	:GET STARTING WORD FOR MESSAGE(DIVIDE BY 16.)
73	015720	011637	001374			MOV	(SP),WRDPOS	:GET CURRENT WORD POSITION
74	015724	004737	033332			JSR	PC,\$SB2D	:CONVERT IT TO DECIMAL
75	015730	004737	032446			JSR	PC,\$SUPR1	:TYPE IT (LEFT JUSTIFIED)
76	015734	104414	057711			DISPLY	,PERIOD	:TYPE ' . '
77	015740	104414	056121			DISPLY	,LIN10B	: ' IN DATA FIELD OF ERROR SECTOR'
78	015744	063737	001400	001406		ADD	ECSEC,ECWRD	:FIND THE BEGINNING OF THE ERROR BURST
79	015752	026037	000172	001406		CMPI	\$RPBA(RO),ECWRD	:SEE IF BURST WAS IN DATA READ
80	015760	101002				BHI	4\$:BR IF IN DATA READ
81	015762	000137	016344			JMP	ECC2	:NOT IN DATA READ - REPORT 'T
82								
83	015766	016037	000234	001402	4\$:	MOV	\$RPEC2(RO),ECMSK0	:GET THE ERROR BIT MASK
84	015774	005037	001404			CLR	ECMSK1	:CLEAR THE UPPER MASK WORD
85	016000	005337	001376		5\$:	DEC	ECBIT	:DECREMENT THE BIT OFFSET COUNT
86	016004	002405				BLT	6\$:BR IF DONE
87	016006	006337	001402			ASL	ECMSK0	:SHIFT THE ERROR MASK
88	016012	006137	001404			ROL	ECMSK1	:SHIFT THE LOWER INTO THE UPPER
89	016016	000770				BR	5\$:CONTINUE THE SHIFT
90								
91	016020	017737	163362	001412	6\$:	MOV	@ECWRD,ECBAD0	:SAVE THE INCORRECT WORD
92	016026	013746	001402			MOV	ECMSK0,-(SP)	:PUT LOWER MASK ON STACK
93	016032	047716	163350			BIC	@ECWRD,(SP)	:CLEAR ERRONEOUS ONE BITS FROM MASK
94	016036	043777	001402	163342		BIC	ECMSK0,@ECWRD	:CLEAR ERRONEOUS ONE BITS FROM BAD WORD
95	016044	052677	163336			BIS	(SP)+,@ECWRD	:SET DROPPED BITS
96								
97	016050	005737	001404			TST	ECMSK1	:DOES ERROR GO INTO NEXT WORD ?
98	016054	001415				BEQ	7\$:BR IF NO
99	016056	013737	001406	001414		MOV	ECWRD,ECWRD1	:DUPLICATE ADDRESS
100	016064	062737	000002	001414		ADD	#2,ECWRD1	:INCREMENT ERROR ADDRESS
101	016072	026037	000172	001414		CMPI	\$RPBA(RO),ECWRD1	:IS NEXT WORD IN THE BUFFER ?
102	016100	101006				BHI	8\$:BR IF YES, ELSE,
103	016102	005737	001402			TST	ECMSK0	:WAS ERROR IN FIRST WORD ?
104	016106	001516				BEQ	ECC2	:BR IF NO
105	016110	005037	001414		7\$:	CLR	ECWRD1	:CLEAR 2ND WORD ADDRESS
106	016114	000414				BR	ECC1	:PRINT WORD CORRECTED
107								
108	016116	017737	163272	001420	8\$:	MOV	@ECWRD1,ECBAD1	:SAVE THE SECOND BAD WORD
109	016124	013746	001404			MOV	ECMSK1,-(SP)	:PUT THE UPPER MASK ON THE STACK
110	016130	047716	163260			BIC	@ECWRD1,(SP)	:CLEAR ERRONEOUS ONE BITS FROM UPPER MASK
111	016134	043777	001404	163252		BIC	ECMSK1,@ECWRD1	:CLEAR ERRONEOUS ONE BITS FROM DATA WORD
112	016142	052677	163246			BIS	(SP)+,@ECWRD1	:SET DROPPED BITS
113								
114	016146	104414	056265		ECC1:	DISPLY	,LIN10H	:HEADER
118	016152	013746	001406			MOV	ECWRD,-(SP)	:PUT ECWRD ON THE STACK
	016156	004737	024306			JSR	PC,LIN0CT	:TYPE ECWRD
119	016162	013746	001374			MOV	WRDPOS,-(SP)	:PUT THE WORD POS ON THE STACK
120	016166	004737	033332			JSR	PC,\$SB2D	:CONVERT IT TO DECIMAL
121	016172	004537	032662			JSR	R5,\$REPLZ	:TYPE IT (REPLACE LEADING ZEROS)
122	016176	000006				.WORD	6	:TYPE 6 DIGITS
123	016200	104414	057711			DISPLY	,PERIOD	:TYPE ' . '
124	016204	104414	056530			DISPLY	,BLNKS2	:TYPE 2 BLANKS
129	016210	013746	001412			MOV	ECBAD0,-(SP)	:PUT ECBAD0 ON THE STACK
	016214	004737	024306			JSR	PC,LIN0CT	:TYPE ECBAD0

	016220	104414	056530	DISPLY	,BLNKS2	:TYPE 2 BLANKS
	016224	017746	163156	MOV	@ECWRD,-(SP)	:PUT @ECWRD ON THE STACK
	016230	004737	024306	JSR	PC,LINOLT	:TYPE @ECWRD
	016234	104414	056530	DISPLY	,BLNKS2	:TYPE 2 BLANKS
130						
131	016240	005737	001414	TST	ECWRD1	:PRINT THE NEXT WORD ?
132	016244	001441		BEQ	ECCX	:BR IF NOT
133	016246	104414	001203	DISPLY	,\$CRLF	:CR-LF
137	016252	013746	001414	MOV	ECWRD1,-(SP)	:PUT ECWRD1 ON THE STACK
	016256	004737	024306	JSR	PC,LINOLT	:TYPE ECWRD1
138	016262	013746	001374	MOV	WRDPOS,-(SP)	:PUT THE WORD POS ON THE STACK
139	016266	005216		INC	(SP)	:INC TO SECOND WORD OF BURST
140	016270	004737	033332	JSR	PC,\$SB2D	:CONVERT IT TO DECIMAL
141	016274	004537	032662	JSR	R5,\$REPLZ	:TYPE IT (REPLACE LEADING ZEROS)
142	016300	000006		.WORD	6	:TYPE 6 DIGITS
143	016302	104414	057711	DISPLY	,PERIOD	:TYPE '.'
144	016306	104414	056530	DISPLY	,BLNKS2	:TYPE 2 BLANKS
149	016312	013746	001420	MOV	ECBAD1,-(SP)	:PUT ECBAD1 ON THE STACK
	016316	004737	024306	JSR	PC,LINOLT	:TYPE ECBAD1
	016322	104414	056530	DISPLY	,BLNKS2	:TYPE 2 BLANKS
	016326	017746	163062	MOV	@ECWRD1,-(SP)	:PUT @ECWRD1 ON THE STACK
	016332	004737	024306	JSR	PC,LINOLT	:TYPE @ECWRD1
	016336	104414	056530	DISPLY	,BLNKS2	:TYPE 2 BLANKS
150	016342	000402		BR	ECCX	:EXIT
151						
152	016344	104414	056161	ECC2: DISPLY	,LIN10C	:ERROR BURST WAS NOT TRANSFERED TO MEMORY
153	016350	104414	001203	ECCX: DISPLY	,\$CRLF	:CR-LF
154	016354	000207		RTS	PC	:RETURN

```

1          ;ROUTINE TO DISPLAY THE SECTOR WHICH GAVE THE HARD ERROR
2
3 016356 032777 000010 162570 PRTBAD: BIT    #SW3,@SWR    ;PRINT THE BAD SECTOR ?
4 016364 001520          BEQ    8$          ;BR IF NOT
5 016366 016001 000172          MOV    $RPBA(R0),R1    ;PUT THE END ADDRESS INTO R1
6 016372 016046 000020          MOV    $WRDL(R0),-(SP) ;FIND THE BEGINNING OF THE SECTOR
7 016376 066016 000170          ADD    $RPWC(R0),(SP) ;SUBTRACT THE WORDS NOT TRANSFERED
8 016402 001002          BNE    1$          ;
9 016404 005726          TST    (SP)+          ;RESTORE STACK
10 016406 000207          RTS    PC          ;EXIT--NO WORDS XFERRED
11 016410 005046          1$: CLR    -(SP)          ;MAKE THE UPPER DIVIDEND 0
12 016412 016046 000022          MOV    $SSEC(R0),-(SP) ;DIVIDE THE WORDS XFERED BY THE SECTOR SIZE
13 016416 004737 032046          JSR    PC,$DIV          ;DIVIDE
14 016422 005716          TST    (SP)          ;REMAINDER = 0 ?
15 016424 001403          BEQ    2$          ;BR IF IT IS - COMPLETE SECTOR TRANSFERED
16 016426 006316          ASL    (SP)          ;CONVERT THE RESIDUAL SECTOR INTO BYTE COUNT
17 016430 161601          SUB    (SP),R1          ;SUBTRACT IT FROM THE END ADDRESS
18 016432 000410          BR    3$          ;FINISH THE SIZING
19 016434 162701 001000          2$: SUB    #256,*2,R1    ;SUBTRACT FULL SECTOR FROM END ADDR (IN BYTES)
20 016440 122760 000005 000024          CMPB   #5,$CODE(R0)    ;WAS OPERATION READ HEADER & DATA ?
21 016446 001002          BNE    3$          ;BR IF NOT
22 016450 162701 000004          SUB    #4,R1          ;SUBTRACT HEADER SIZE FROM ADDR
23 016454 062706 000004          3$: ADD    #4,SP          ;RESTORE THE STACK POINTER
24 016460 104414 001203          DISPLY , $CRLF          ;CR-LF
25 016464 104414 056413          DISPLY , LIN1H          ;PRINT THE HEADER
26 016470 122760 000005 000024          CMPB   #5,$LODE(R0)    ;WAS OPERATION READ HEADER & DATA ?
27 016476 001021          BNE    4$          ;BR IF NOT
28 016500 104414 056466          DISPLY , LIN1          ;TYPE 'ADDR  HEADER'
29 016504 010146          MOV    R1,-(SP)          ;PUT THE ADDRESS ON THE STACK
30 016506 004737 024306          JSR    PC,LIN0CT        ;TYPE THE ADDRESS
31 016512 104414 056527          DISPLY , BLNK3          ;TYPE 3 BLANKS
32 016516 012146          MOV    (R1)+,-(SP)        ;PUT WORD ON STACK
33 016520 004737 024306          JSR    PC,LIN0CT        ;TYPE THE 1ST HEADER WORD
34 016524 104414 056531          DISPLY , BLNK1          ;TYPE 1 BLANK
35 016530 012146          MOV    (R1)+,-(SP)        ;PUT WORD ON STACK
36 016532 004737 024306          JSR    PC,LIN0CT        ;TYPE THE 2ND HEADER WORD
37 016536 104414 001203          DISPLY , $CRLF          ;CR-LF
38
39 016542 104414 056507          4$: DISPLY , LIN1A          ;TYPE 'ADDR  DATA'
40 016546 012702 000010          5$: MOV    #8.,R2          ;8. DATA WORDS PER LINE
41 016552 010146          MOV    R1,-(SP)          ;PUT THE ADDRESS ON THE STACK
42 016554 004737 024306          JSR    PC,LIN0CT        ;TYPE THE ADDRESS
43 016560 104414 056530          DISPLY , BLNK2          ;TYPE 2 BLANKS
44 016564 020160 000172          6$: CMP    R1,$RPBA(R0)    ;PRINTED ALL THE SECTOR ?
45 016570 001412          BEQ    7$          ;BR IF ALL PRINTED
46 016572 104414 056531          DISPLY , BLNK1          ;TYPE 1 BLANK
47 016576 012146          MOV    (R1)+,-(SP)        ;PUT THE DATA ON THE STACK
48 016600 004737 024306          JSR    PC,LIN0CT        ;TYPE THE DATA
49 016604 005302          DEC    R2          ;DECREMENT THE HORIZONTAL COUNT
50 016606 001366          BNE    6$          ;BR IF NOT AT THE END OF THE LINE
51 016610 104414 001203          DISPLY , $CRLF          ;CR-LF
52 016614 000754          BR    5$          ;RESTORE THE WORDS/LINE COUNT
53 016616 104414 001203          7$: DISPLY , $CRLF          ;CR-LF
54 016622 104414 001203          DISPLY , $CRLF          ;CR-LF
55 016626 000207          8$: RTS    PC          ;RETURN
56
57          ;ROUTINE TO DO A RECALIBRATE USING ACTIVE DPB
    
```

```

58          ;CALL:
59          :      MOV      #DPB,RO      ;DPB ADDRESS
60          :      JSP      PC,RECALT
61          :      RETURN
62
63 016630 010037 016654          RECALT: MOV      RO,2$      ;LOAD THE DPB ADDRESS
64 016634 116060 000166 000027 MOVB     $RPCS1(RO),$PREVO(RO) ;SAVE THE PREVIOUS COMMAND
65 016642 112760 000107 000002 MOVB     #RECAL,$COMND(RO)    ;LOAD THE NEW COMMAND
66 016650 004037 041304          1$: JSR      RO,RPO7      ;START THE RECALIBRATE
67 016654 000000          2$: .WORD 0      ;DPB ADDRESS
68 016656 000774          BR      1$      ;DRIVER DIDN'T ACCEPT THE COMMAND
69 016660 005760 000016          3$: TST     $STATUS(RO)    ;SEE IF FINISHED
70 016664 001775          BEQ     3$      ;IF EQ NO
71 016666 004737 024364          JSR     PC,READDR    ;DECREMENT THE ADDRESSES
72 016672 012660 000034          MOV     (SP)+,$PREVA+2(RO) ;MOVE THE CYLINDER ADDRESS
73 016676 112660 000033          MOVB   (SP)+,$PREVA+1(RO) ;MOVE THE TRACK ADDRESS
74 016702 112660 000032          MOVB   (SP)+,$PREVA(RO)   ;MOVE THE SECTOR ADDRESS
75 016706 005060 000012          CLR     $CYL(RO)        ;CLEAR THE CURRENT CYLINDER ADDRESS
76 016712 005060 000010          CLR     $SEC(RO)        ;CLEAR THE CURRENT TRK/SEC ADDRESS
77 016716 000207          RTS      PC            ;RETURN
78
79          ;ROUTINE TO A RECAL WITH NO JPB ACTIVE
80          ;CALL:
81          :      MOVB     #DRIVE,GENDPB ;DRIVE ADDRESS
82          :      JSR      PC,RECALO
83          :      RETURN
84
85 016720 112737 000107 050656 RECALO: MOVB     #RECAL,GENDPB+$COMND ;RECALIBRATE COMMAND
86 016726 004037 041304          1$: JSR      RO,RPO7      ;DRIVER ENTRANCE
87 016732 050654          GENDPB ;DPB ADDRESS FOR COMMAND
88 016734 000774          BR      1$      ;DRIVER DIDN'T ACCEPT THE COMMAND
89 016736 005737 050672          2$: TST     GENDPB+$ IUS ;SEE IF FINISHED
90 016742 001775          BEQ     2$      ;BR IF NOT FINISHED
91 016744 000207          RTS      PC
108
109          ;UTILITY READ HEADER ROUTINE
110          ;CALL:
111          :      MOV      #DPB,RO      ;DPB ADDRESS
112          :      MOV      #SECTOR,-(SP) ;SECTOR ADDRESS
113          :      MOV      #TRACK,-(SP)  ;TRACK ADDRESS
114          :      MOV      #CYLINDER,-(SP);CYLINDER ADDRESS
115          :      JSR      PC,READDR
116          :      RETURN
117
118 016746 116637 000004 050665 READHD: MOVB     4(SP),GENDPB+$TRK ;TRACK ADDRESS
119 016754 116637 000006 050664          MOVB     6(SP),GENDPB+$SEC ;SECTOR ADDRESS
120 016762 016637 000002 050666          MOV      2(SP),GENDPB+$CYL ;CYLINDER ADDRESS
121 016770 111037 050654          MOVB     (RO),GENDPB ;DRIVE NUMBER
122 016774 112737 000173 050656          MOVB     #RDHD,GENDPB+$COMND ;COMMAND
123 017002 012737 177776 050660          MOV      #-2,GENDPB+$WCNT ;WORD CTR = 2
124 017010 004037 041304          1$: JSR      RO,RPO7      ;DRIVER ENTRANCE
125 017014 050654          GENDPB ;DPB ADDRESS FOR COMMAND
126 017016 000774          BR      1$      ;DRIVER DIDN'T ACCEPT COMMAND
127 017020 005737 050672          2$: TST     GENDPB+$STATUS ;FINISHED?
128 017024 001775          BEQ     2$      ;BR IF NOT
129 017026 011666 000006          MOV     (SP),6(SP) ;ADJUST STACK FOR RETURN
130 017032 062706 000006          ADD     #6,SP ;ADJUST RETRUN POINTER
    
```

```

131 017036 000207          RTS      PC          ;RETURN
132
133          ;RETRY THE PRESENT OPERATION
134          ;CALL:
135          ;      MOV      #COUNT,RETRY      ;RETRY COUNT
136          ;      JSR      PC,$RETRY
137          ;      RETURN1
138          ;      RETURN2
139          ;
140          ;
141
142 017040 004737 020160  $RETRY: JSR      PC,GODRIV      ;RE-START ORDER
143 017044 005760 000016  1$:    TST      $STATUS(R0)      ;ORDER FINISHED?
144 017050 001775          BEQ      1$              ;BR IF NOT
145 017052 100405          BMI      2$              ;BR IF ERROR
146 017054 105237 001325  INCB     RETRY+1          ;INCREMENT RETRY COUNT
147 017060 062716 000002  ADD      #2,(SP)         ;INCREMENT RETURN
148 017064 000436          BR       6$              ;GO TO EXIT
149
150 017066 032760 000200 000016  2$:    BIT      #BIT7,$STATUS(R0) ;DID ORDER TERMINATE NORMALLY ?
151 017074 001441          BEQ      8$              ;BR IF NOT
152 017076 005757 001322  TST      MASK            ;IS ERROR MASK 0 ?
153 017102 001004          BNE     3$              ;BR IF NOT
154 017104 005760 000202  TST      $RPER1(R0)      ;MAKE SURE THAT THE DRIVE ERROR REG IS CLEAR
155 017110 001025          BNE     7$              ;BR IF NOT
156 017112 000415          BR       5$              ;CONTINUE RETRY
157
158 017114 022737 000007 001322  3$:    CMP      #7,MASK        ;EWN ERROR?
159 017122 001005          BNE     4$              ;BR IF NO
160 017124 032760 000002 000200  BIT      #BIT1,$RPDS(R0) ;'EWN' STILL SET?
161 017132 001414          BEQ      7$              ;NO, REPORT DIFFERENT ERROR
162 017134 000404          BR       5$              ;SET, RETRY
163
164 017136 033760 001322 000202  4$:    BIT      MASK,$RPER1(R0) ;SAME ERROR?
165 017144 001407          BEQ      7$              ;BR IF NOT
166 017146 105237 001325          INCB     RETRY+1          ;INCREMENT RETRY COUNT
167 017152 123737 001324 001325  5$:    CMPB    RETRY,RETRY+1      ;DONE ?
168 017160 001327          BNE     $RETRY          ;BR IF NOT DONE
169 017162 000207          6$:    RTS      PC          ;RETURN
170
171 017164 004737 024274          7$:    JSR      PC,LINE8      ;REPORT DIFFERENT ERROR
172 017170 004737 024056          JSR      PC,LINE7      ;PRINT LINE 7
173 017174 005726          TST      (SP)+          ;ADJUST STACK POINTER FOR DIRECT RETURN
174 017176 000207          RTS      PC          ;RETURN
175
176 017200 104414 055570          8$:    DISPLY  ,LIN8M      ;'DIFFERENT ERROR DURING RETRY'
177 017204 000137 007636          JMP      ERPRC1        ;REPORT THE ERROR
    
```

```

1          ;ROUTINE TO UPDATE THE PERFORMANCE SUMMARY STATISTICS
2          ;CALL:
3          ;
4          ;       MOV      #DPB,RO           ;DPB ADDRESS
5          ;       JSR      PC,STATIS
6          ;       RETURN
7 017210   032760   000300   000016   STATIS: BIT      #BIT07!BIT06,STATUS(RO) ;CHECK FOR DATA TERMINATION
8 017216   001533           BEQ      7$                ;BR IF NOT DATA TERMINATION
9 017220   016037   000172   017510   MOV      $RPBA(RO),FACTOR ;STORE THE FINAL BUFFER ADDRESS
10 017226   166037   000006   017510   SUB      $BUF(RO),FACTOR ;SUBTRACT THE INITIAL ADDRESS
11 017234   001524           BEQ      7$                ;BR IF NO DATA TRANSFER
12 017236   006237   017510   ASR      FACTOR           ;CONVERT TO A WORD COUNT
13
14 017242   122760   000002   000024   CMPB    #2,$CODE(RO)     ;SEE IF COMMAND WAS A WRITE
15 017250   001404           BEQ      1$                ;BRANCH IF YES
16 017252   122760   000000   000024   CMPB    #0,$CODE(RO)     ;PRESENT OPERATION AN AUTO WRITE CHECK ?
17 017260   001043           BNE     4$                ;BR IF NO
18 017262   063760   017510   000042   1$:    ADD      FACTOR,$WTPAS(RO) ;ADD WORDS WRITTEN PER PASS
19 017270   005560           ADC     $WTPAS+2(RO)       ;ADD ANY CARRY
20 017274   063760   017510   000062   ADD     FACTOR,$WTCTL+4(RO) ;ADD WORDS WRITTEN TO REP COUNTER
21 017302   005560   000064           ADC     $WTOTL+6(RO)       ;ADD ANY CARRY
22 017306   016046   000062           MOV     $WTOTL+4(RO),-(SP) ;GET LOW DIVIDEND
23 017312   016046   000064           MOV     $WTOTL+6(RO),-(SP) ;GET HIGH DIVIDEND
24 017316   012746   041100           MOV     #041100,-(SP)     ;GET LOW DIVISOR
25 017322   012746   000017           MOV     #17,-(SP)        ;GET HIGH DIVISOR
26 017326   004737   032114           JSR     PC,$DBDIV        ;DIVIDE BY 1 X10^6
27 017332   005766   000004           TST     4(SP)            ;DID REP COUNTER REACH LIMIT YET ?
28 017336   001412           BEQ     2$                ;BR IF NO
29 017340   012660   000064           MOV     (SP)+,$WTOTL+6(RO) ;GET HIGH REMAINDER
30 017344   012660   000062           MOV     (SP)+,$WTOTL+4(RO) ;GET LOW REMAINDER
31 017350   062760   000001   000056   ADD     #1,$WTOTL(RO)     ;UPDATE THE TOTAL WRDS WRITTEN
32 017356   005560   000060           ADC     $WTOTL+2(RO)       ;ADD ANY CARRY
33 017362   000401           BR      3$
34 017364   022626           2$:    CMP     (SP)+,(SP)+     ;GET HIGH & LOW REMAINDER OFF STACK
35 017366   005726           3$:    TST     (SP)+           ;GET QUOTIENT OFF STACK
36
37 017370   122760   000002   000024   4$:    CMPB    #2,$CODE(RO)     ;SEE IF COMMAND WAS A WRITE
38 017374   001443           BEQ     7$                ;BRANCH IF YES
39 017400   063760   017510   000036   ADD     FACTOR,$RDPAS(RO) ;ADD WORDS READ PER PASS
40 017406   005560   000040           ADC     $RDPAS+2(RO)       ;ADD ANY CARRY
41 017412   063760   017510   000072   ADD     FACTOR,$RTOTL+4(RO) ;ADD WORDS READ TO REP COUNTER
42 017420   005560   000074           ADC     $RTOTL+6(RO)       ;ADD ANY CARRY
43 017424   016046   000072           MOV     $RTOTL+4(RO),-(SP) ;GET LOW DIVIDEND
44 017430   016046   000074           MOV     $RTOTL+6(RO),-(SP) ;GET HIGH DIVIDEND
45 017434   012746   041100           MOV     #041100,-(SP)     ;GET LOW DIVISOR
46 017440   012746   000017           MOV     #17,-(SP)        ;GET HIGH DIVISOR
47 017444   004737   032114           JSR     PC,$DBDIV        ;DIVIDE BY 1 X10^6
48 017450   005766   000004           TST     4(SP)            ;DID REP COUNTER REACH LIMIT YET ?
49 017454   001412           BEQ     5$                ;BR IF NO
50 017456   012660   000074           MOV     (SP)+,$RTOTL+6(RO) ;GET HIGH REMAINDER
51 017462   012660   000072           MOV     (SP)+,$RTOTL+4(RO) ;GET LOW REMAINDER
52 017466   062760   000001   000066   ADD     #1,$RTOTL(RO)     ;UPDATE THE TOTAL WRDS READ
53 017474   005560   000070           ADC     $RTOTL+2(RO)       ;ADD ANY CARRY
54 017500   000401           BR      6$
55 017502   022626           5$:    CMP     (SP)+,(SP)+     ;GET HIGH & LOW REMAINDER OFF STACK
56 017504   005726           6$:    TST     (SP)+           ;GET QUOTIENT OFF STACK
57 017506   000207           7$:    RTS     PC
    
```

58
59 017510 000000

FACTOR: .WORD 0

;USED FOR WORDS TRANSFERED

```

1      ;ROUTINE TO GET A BUFFER
2      :CALL:
3      :
4      :       MOV      #DPB,R0          ;DPB ADDRESS
5      :       CLR      -(S?)          ;CLEAR THE STACK
6      :       JSR      PC,GETBUF
7      :       RETURN
8      :
9      GETBUF: MOV      R1,-(SP)        ;SAVE R1
10     MOV      R2,-(SP)        ;SAVE R2
11     MOV      R3,-(SP)        ;SAVE R3
12     MOV      BUFTBL,R2      ;NUMBER OF SEPARATE BUFFERS
13     BEQ      5$             ;BR IF NONE AVAILABLE
14     MOV      #BUFTBL+2,R1    ;FIRST ADDRESS OF ALLOCATION TABLE
15     CMP      $WRDL(R0),2(R1) ;SEE IF THERE IS A BLOCK LARGE ENOUGH
16     BLOS    2$             ;BRANCH IF IT IS
17     DEC      R2             ;DECREMENT TABLE COUNT
18     BEQ      5$             ;BR IF THROUGH TABLE
19     ADD      #4,R1          ;INCREMENT TABLE POINTER
20     BR       1$             ;CONTINUE LOOKING
21     MOV      (R1),10(SP)     ;BUFFER ADDRESS TO STACK
22     SUB      $WRDL(R0),2(R1) ;ADJUST BUFFER WRD CNT
23     BEQ      3$             ;BR IF DIFFERENCE IS ZERO
24     ASL      $WRDL(R0)       ;CONVERT # WORDS TO BYTES
25     ADD      $WRDL(R0),(R1)  ;MAKE NEW STARTING ADDRESS
26     ASR      $WRDL(R0)       ;RETURN # BYTES TO WORDS
27     BR       5$             ;RETURN
28     3$: DEC      BUFTBL      ;DECREMENT ENTRY COUNT
29     BEQ      5$             ;BR IF ALLOCATION TABLE EMPTY
30     DEC      R2             ;DECREMENT TABLE COUNT
31     BEQ      5$             ;BR IF ITEM WERE LAST ENTRY
32     MOV      R1,R3          ;MOVE TABLE POINTER
33     ADD      #4,R3          ;POINT TO NEXT ENTRY
34     4$: MOV      (R3)+,(R1)+ ;MOVE ITEMS
35     MOV      (R3)+,(R1)+
36     DEC      R2             ;DECREMENT TABLE COUNT
37     BNE     4$             ;CONTINUE IF NOT AT END OF TABLE
38     5$: MOV      (SP)+,R3    ;RESTORE R3
39     MOV      (SP)+,R2      ;RESTORE R2
40     MOV      (SP)+,R1      ;RESTORE R1
41     RTS      PC            ;RETURN
42
43     ;ROUTINE TO PUT BUFFER BACK IN TABLE
44     :CALL:
45     :
46     :       MOV      #DPB,R0          ;DPB ADDRESS
47     :       JSR      PC,RELBUF
48     :       RETURN
49     RELBUF: MOV      R1,-(SP)        ;SAVE R1
50     MOV      R2,-(SP)        ;SAVE R2
51     MOV      R3,-(SP)        ;SAVE R3
52     MOV      R4,-(SP)        ;SAVE R4
53     MOV      R5,-(SP)        ;SAVE R5
54     MOV      #BUFTBL+2,R1    ;BEGINNING OF TABLE
55     MOV      BUFTBL,R2      ;ENTRY COUNT
56     BEQ      2$             ;BR IF EMPTY TABLE
57     MOV      $HLDWC(R0),R3   ;TRIAL ADDRESS

```

```

58 017676 006303          ASL      R3          ;CHANGE TO BYTE COUNT
59 017700 066003 000006    ADD      $BUF(R0),R3 ;ADDRESS OF HIGHER ADJACENT BLOCK
60 017704 021103          CMP      (R1),R3    ;UPPER ADJACENT BLOCK
61 017706 001424          BEQ      3$         ;BR IF YES
62 017710 062701 000004    ADD      #4,R1      ;INCREMENT POINTER
63 017714 005302          DEC      R2         ;DECREMENT ENTRY COUNT
64 017716 001372          BNE     1$         ;CONTINUE SEARCHING
65 017720 016011 000006    MOV      $BUF(R0),(R1) ;PUT THE BUFFER BLOCK INTO THE TABLE
66 017724 016061 000120 000002  MOV      $HLDWC(R0),2(R1) ;BLOCK WRD CNT
67 017732 005237 001654    INC      BUFTBL     ;INCREMENT ENTRY COUNT
68 017736 005202          INC      R2         ;INCREMENT R2 FOR USE LATER
69 017740 000414          BR      4$         ;SEE IF A LOWER ADJACENT BLOCK IS IN THE TABLE
70 017742 016021 000006    MOV      $BUF(R0),(R1)+ ;BLOCK ADDRESS TO TABLE
71 017746 016021 000120    MOV      $HLDWC(R0),(R1)+ ;WRD CNT TO TABLE
72 017752 005237 001654    INC      BUFTBL     ;INCREMENT ENTRY COUNT
73 017756 000443          BR      8$         ;EXIT
74 017760 016011 000006    MOV      $BUF(R0),(R1) ;RELEASED BUFFER IS LOWER ADJACENT
75 017764 066061 000120 000002  ADD      $HLDWC(R0),2(R1) ;INCREMENTED WRD CNT
76 017772 010246          MOV      R2,-(SP)   ;SAVE R2
77 017774 013702 001654    MOV      BUFTBL,R2  ;ENTRY COUNT
78 020000 012705 001656    MOV      #BUFTBL+2,R5 ;BEGINNING OF TABLE
79 020004 016504 000002 5$:      MOV      2(R5),R4   ;BLOCK SIZE (IN WORDS)
80 020010 006304          ASL      R4         ;CHANGE TO BYTE COUNT
81 020012 061504          ADD      (R5),R4   ;ADD BLOCK BEGINNING ADDRESS
82 020014 020411          CMP      R4,(R1)   ;R1 STILL POINTS TO INSERTED ENTRY
83 020016 001406          BEQ      6$         ;LOWER ADJACENT 'V' TABLE
84 020020 062705 000004    ADD      #4,R5     ;INCREMENT POINT R
85 020024 005302          DEC      R2         ;DECREMENT ENTP COUNT
86 020026 001366          BNE     5$         ;CONTINUE LOOKING
87 020030 005726          TST     (SP)+      ;RESTORE STACK POINTER
88 020032 000415          BR      8$         ;END
89 020034 012602          MOV      (SP)+,R2  ;RESTORE R2
90 020036 066165 000002 000002  ADD      2(R1),2(R5) ;INCREMENT LOWER BLOCK LENGTH
91 020044 005337 001654    DEC      BUFTBL     ;DECREMENT ENTRY COUNT
92 020050 010105          MOV      R1,R5     ;GET READY TO COMPRESS
93 020052 062705 000004    ADD      #4,R5     ;INCREMENT TO NEXT ENTRY
94 020056 012521 7$:      MOV      (R5)+,(R1)+ ;COMPRESS TABLE
95 020060 012521          MOV      (R5)+,(R1)+ ;MOVE SIZE FIELD DOWN
96 020062 005302          DEC      R2         ;DECREMENT ENTRY COUNT
97 020064 001374          BNE     7$         ;BR IF NOT FINISHED
98 020066 012605 8$:      MOV      (SP)+,R5  ;RESTORE R5
99 020070 012604          MOV      (SP)+,R4  ;RESTORE R4
100 020072 012603          MOV      (SP)+,R3  ;RESTORE R3
101 020074 012602          MOV      (SP)+,R2  ;RESTORE R2
102 020076 012601          MOV      (SP)+,R1  ;RESTORE R1
103 020100 000207          RTS      PC        ;RETURN
104
105          ;FILL THE ASSIGNED BUFFER (IF WR - OR WRITE CHECK COMMAND)
106          ;CALL:
107          ;
108          ;      MOV      #DPB,R0          ;DPB ADDRESS
109          ;      MOV      #BUFADR,$BUF(R0) ;LOAD BUFFER ADDRESS INTO THE DPB
110          ;      MOVB   #PATTERN,$PATT(R0) ;PATTERN CODE
111          ;      JSR      PC,FILBUF
112          ;
113 020102 104412          FILBUF: SAVREG     ;SAVE THE REGISTERS
114 020104 132760 000004 000024  BITB   #4,$CODE(R0) ;SEE IF READ COMMAND
    
```



```

115 020112 001020
116 020114 016001 000006
117 020120 016002 000020
118 020124 116004 000030
119 020130 016405 002314
120 020134 012703 000020
121 020140 012521
122 020142 005302
123 020144 003403
124 020146 005303
125 020150 001373
126 020152 000766
127 020154 104413
128 020156 000207
129
130
131
132
133
134
135
136 020160 010046
137 020162 010037 020172
138 020166 004037 041304
139 020172 000000
140 020174 000000
141 020176 012600
142 020200 062760 000001 000052
143 020206 005560 000054
144 020212 026060 000034 000012
145 020220 001412
146 020222 062760 000001 000046
147 020230 005560 000050
148 020234 062760 000001 000076
149 020242 005560 000100
150 020246 000207

1$: BNE 4$ :BR IF READ
MOV $BUF(R0),R1 :BUFFER ADDRESS
MOV $WRDL(R0),R2 :POSITIVE WORD COUNT
MOVB $PATT(R0),R4 :RELATIVE PATTERN ADDRESS
2$: MOV $NDAT(R4),R5 :PATTERN ADDRESS
MOV #16,R3 :PATTERN COUNT
3$: MOV (R5)+,(R1)+ :MOVE THE PATTERN INTO THE BUFFER
DEC R2 :DECREMENT THE WORD COUNT
BLE 4$ :BR IF DONE (WORD COUNT = 0)
DEC R3 :DECREMENT THE PATTERN COUNT
BNE 3$ :BR IF MORE PATTERN
BR 2$ :CONTINUE DISTRIBUTING THE PATTERN
4$: RESREG :RESTORE THE REGISTERS
RTS PC :RETURN

:START THE COMMAND FOR THE DPB IN R0
:CALL:
: MOV #DPB,R0 :DPB ADDRESS
: JSR PC,GODRIV
: RETURN

GODRIV: MOV R0,-(SP) :SAVE R0
MOV R0,2$ :CURRENT DPB ADDRESS
1$: JSR R0,RPG7 :CALL THE DRIVE HANDLER
2$: .WORD 0 :DRIVE BLOCK ADDRESS GOES HERE
HALT :DRIVER REJECTED REQUEST
MOV (SP)+,R0 :RESTORE R0
ADD #1,$OPERC(R0) :INCREMENT THE OPERATION COUNT
ADC $OPERC+2(R0)
CMP $PREVA+2(R0),$CYL(R0) :DID COMMAND REQUIRE A CYLINDER CHANGE ?
3$: BEQ 3$ :BR IF NO
ADD #1,$SEEKS(R0) :INCREMENT SEEK COUNT PER PASS
ADC $SEEKS+2(R0) :ADD ANY CARRY
ADD #1,$STOTL(R0) :INCREMENT SEEK COUNT TOTAL
ADC $STOTL+2(R0) :ADD ANY CARRY
3$: RTS PC
    
```

MAIN PROGRAM

```

1          ;ROUTINE TO SETUP PARAMETERS FOR A SEQUENTIAL READ OR WRITE OF THE DISK
2          ;CALL
3          MOV     #DPB,R0           ;DPB ADDRESS
4          MOV     #NN,$PACK(R0)    ;GET COMMAND NUMBER
5          JSR     PC,SEQPAR        ;CALL ROUTINE
6          RETURN
7
8          ;WHERE 'NN' CAN BE ONE OF THE FOLLOW NUMBERS:
9          'WT'  COMMAND= -2
10         'W'   COMMAND= -1
11         'T'   COMMAND= 0      (NOT USED IN ROUTINE)
12         'R'   COMMAND= 1
13
14 020250 005760 000054  SEQPAR: TST     $OPERC+2(R0)    ;IS THIS THE FIRST OPERATION ?
15 020254 001010          BNE     1$           ;BR IF NO
16 020256 005760 000052  TST     $OPERC(R0)    ;IS THIS THE FIRST OPERATION ?
17 020262 001005          BNE     1$           ;BR IF NO
18 020264 012704 000010  MOV     #$SEC,R4      ;GET INDEX TO SECTOR ADDR STORAGE IN DPB
19 020270 004737 021332  JSR     PC,CKLMTS    ;GO CHECK DISK ADDRESS LIMITS
20 020274 000453          BR      3$           ;BR IF NOT AT END OF SEQUENTIAL ADDRESSING
21
22 020276 116060 000166 000027 1$:  MOVB   $RPCS1(R0),$PREVO(R0) ;SAVE CURRENT PARAMETERS
23 020304 016060 000010 000032  MOV     $SEC(R0),$PREVA(R0) ;SAVE PREVIOUS TRACK/SECTOR ADDRESS
24 020312 016060 000012 000034  MOV     $CYL(R0),$PREVA+2(R0) ;SAVE PREVIOUS CYLINDER ADDRESS
25 020320 016060 000174 000010  MOV     $RPDA(R0),$SEC(R0) ;CURRENT SECTOR & TRACK ADDRESS
26 020326 016060 000222 000012  MOV     $RPDC(R0),$CYL(R0) ;CURRENT CYLINDER ADDRESS
27
28 020334 012704 000010 2$:  MOV     #$SEC,R4      ;GET INDEX TO SECTOR ADDR STORAGE IN DPB
29 020340 004737 021332  JSR     PC,CKLMTS    ;GO CHECK DISK ADDRESS LIMITS
30 020344 000427          BR      3$           ;BR IF NOT AT END OF SEQUENTIAL ADDRESSING
31
32 020346 116060 000146 000010  MOVB   MINSEC(R0),$SEC(R0) ;RESET SECTOR ADDRESS
33 020354 116060 000142 000011  MOVB   MINTRK(R0),$TRK(R0) ;RESET TRACK ADDRESS
34 020362 016060 000136 000012  MOV     MINCYL(R0),$CYL(R0) ;RESET CYLINDER ADDRESS
35 020370 112760 000004 000024  MOVB   #4,$CODE(R0) ;SET CODE TO READ DATA
36 020376 122760 177776 000026  CMPB   #-2,$PACK(R0) ;WAS WRITE DATA IN PROGRESS ?
37 020404 001473          BEQ     8$           ;BR IF YES (START TESTING, 'T' COMMAND)
38 020406 004737 031420  JSR     PC,$EOP      ;THIS IS THE END OF PASS
39 020412 032777 000020 160534 BIT     #SW04,@SWR    ;DO NOT DROP DRIVE AT EOT (SW04=1) ?
40 020420 001467          BEQ     9$           ;BR IF NO
41 020422 000744          BR      2$           ;MUST SURE ADDRESS IS NOT 'BSF' TRACK
42
43 020424 012704 000010 3$:  MOV     #$SEC,R4      ;GET INDEX TO SECTOR STORAGE
44 020430 013705 001466  MOV     WRDCNT,R5    ;WORD COUNT IS MAXIMUM
45 020434 004737 021512  JSR     PC,CHKWC     ;CHECK WORD COUNT FOR MAXCYL/MAXTRK
46 020440 010560 000020  MOV     R5,$WRDL(R0) ;GET WORD COUNT
47 020444 042760 000377 000020 BIC     #377,$WRDL(R0) ;IS IT LESS THAN ONE SECTOR WORD COUNT ?
48 020452 001002          BNE     4$           ;BR IF NO
49 020454 105260 000021  INCB   $WRDL+1(R0) ;SET TO ONE SECTOR
50 020460 016060 000020 000004 4$:  MOV     $WRDL(R0),$WCNT(R0) ;STORE FOR 2'S COMPLEMENT WORD
51 020466 016060 000020 000120  MOV     $WRDL(R0),$HLDWC(R0) ;HOLD WORD FOR 'REBUF' ROUTINE
52 020474 005460 000004  NEG     $WCNT(R0) ;CHANGE WORD COUNT TO 2'S COMP.EMENT
53 020500 012760 000400 000022  MOV     #256.,$SEC(R0) ;SECTOR SIZE FOR READ OR WRITE
54
55 020506 105760 000026  TSTB   $PACK(R0) ;'R' OR 'W' COMMAND FOR THIS DRIVE ?
56 020512 100407          BMI     6$           ;BR IF WRITE
57 020514 112760 000004 000024 5$:  MOVB   #4,$CODE(R0) ;CODE FOR READ DATA

```

```
58 020522 112760 000171 000002      MOVB  #RDDAT,$COMND(R0)      ;DRIVE CODE FOR OPERATION
59 020530 000415                      BR      7$                  ;SET UP FOR EXIT
61 020532 005737 001424      6$:  TST  RONLY                ;LOCKED IN 'READ ONLY' MODE ?
62 020536 001366                      BNE     5$                  ;BR IF YES
63 020540 112760 000002 000024      MOVB  #2,$CODE(R0)          ;CODE FOR WRTDAT
67 020546 112760 000161 000002      MOVB  #WRTDAT,$COMND(R0)    ;OP CODE
68 020554 004737 021272      JSR   PC,GETPAT            ;GET PATTERN CODE
69 020560 110560 000030      MOVB  R5,$PATT(R0)         ;PATTERN CODE
70 020564 012760 177777 000130 7$:  MOV   #-1,$NEXT(R0)        ;SET PARAMETERS SELECTED INDICATOR
71 020572 000207                      RTS                          ;RETURN
72
73 020574 105060 000026      8$:  CLRB  $PACK(R0)          ;SET 'T' COMMAND FLAG IN DPB TABLE
74 020600 005060 000130      9$:  CLR   $NEXT(R0)          ;CLEAR 'PARAMETER SELECTED' INDICATOR
75 020604 005726                      TST   (SP)+                ;CLEAR STACK LEVEL
76 020606 000137 006340      JMP   MAIN                 ;JUMP TO MAIN BACKGROUND LOOP
```

```

1          ;GENERATE PARAMETERS FOR THE OPERATION
2          ;CALL:
3          ;
4          ;       MOV      #DPB,RO          ;DPB ADDRESS
5          ;       JSR      PC,GENPAR
6          ;       RETURN
7 020612 004737 037126          GENPAR: JSR      PC,$RAND          ;CYCLE THE RANDOM NUMBER GENERATOR
9 020616 005737 001424          TST      RONLY          ;LOCKED IN "READ ONLY" MODE ?
10 020622 001016                BNE     1$              ;BR IF YES
12 020624 032777 000001 160322 BIT     #SW0,@SWR      ;SEE IF SW0 SET
13 020632 001012                BNE     1$              ;BR IF SET - READ ONLY
14 020634 012705 000010          MOV     #8,R5          ;READ/WRITE SELECTION DIVISOR
15 020640 004737 032022          JSR     PC,GETREM      ;GET SELECTION VALUE
16 020644 020537 001502          CMP     R5,RATIO      ;DETERMINE IF READ OR WRITE
17 020650 103003                BHS    1$              ;BR IF READ
18 020652 012705 000002          MOV     #2,R5          ;SELECT WRITE DATA COMMAND
19 020656 000407                BR     2$              ;SELECT ADDRESS
20
21 020660 013705 037226          1$:    MOV     $LONUM,R5 ;SELECT READ OPERATION CODE
22 020664 000305                SWAB   R5              ;SWAP BYTES IN R5
23 020666 042705 177776          BIC    #^C1,R5        ;MASK OUT ALL BUT BIT 0
24 020672 062705 000004          ADD     #4,R5          ;TABLE OFFSET FOR READ CODE
25 020676 110560 000122          2$:    MOVB   R5,$NCODE(RO) ;COMMAND SELECTION CODE TO CONTROL BLOCK
26 020702 016060 000174 000124 MOV     $RPDA(RO),$NSEC(RO) ;SECTOR AND TRACK
27 020710 016060 000222 000126 MOV     $RPDC(RO),$NCTL(RO) ;CYLINDER NUMBER
28
29 020716 005737 001510          THEAD: TST     RANDOM    ;ENABLE RANDOM ADDRESS SELECT ?
30 020722 001427                BEQ    RANCYL          ;YES
31 020724 005760 000054          TST     $OPERC+2(RO)  ;IS THIS THE FIRST OPERATION ?
32 020730 001003                BNE    1$              ;BR IF NO
33 020732 005760 000052          TST     $OPERC(RO)   ;IS THIS THE FIRST OPERATION ?
34 020736 001405                BEQ    2$              ;BR IF YES
35
36 020740 012704 000124          1$:    MOV     #$NSEC,R4  ;GET INDEX TO SECTOR ADDR STORAGE IN DPB
37 020744 004737 021332          JSR     PC,CKLMTS    ;GO CHECK DISK ADDRESS LIMITS
38 020750 000412                BR     3$              ;BR IF NOT AT END OF SEQUENTIAL ADDRESSING
39 020752 116060 000146 000124 2$:    MOVB   MINSEC(RO),$NSEC(RO) ;RESET SECTOR ADDRESS
40 020760 116060 000142 000125 MOVB   MINTRK(RO),$NTRK(RO) ;RESET TRACK ADDRESS
41 020766 016060 000136 000126 MOV     MINCYL(RO),$NCTL(RO) ;RESET CYLINDER ADDRESS
42 020774 000761                BR     1$              ;RE-CHECK FOR 'BSF' TRACK
43 020776 000137 021150          3$:    JMP     RANSIZ      ;GO CHECK FOR RANDOM WORD SIZE
44

```

```

1          ;GENERATE A RANDOM CYLINDER ADDRESS BETWEEN VALUES 'MINCYL' & 'MAXCYL'
2
3 021002 016005 000134 RANCYL: MOV     MAXCYL(R0),R5 ;GET MAXIMUM CYLINDER ADDRESS
4 021006 026005 000136      CMP     MINCYL(R0),R5 ;'MINCYL' AND 'MAXCYL' THE SAME ?
5 021012 001407          BEQ     1$ ;BR IF THEY ARE
6 021014 166005 000136      SUB     MINCYL(R0),R5 ;GET NUMBER OF ALLOWABLE CYLINDERS
7 021020 005205          INC     R5 ;INCREMENT DIFFERENCE TO USE AS DIVISOR
8 021022 004737 032022      JSR     PC,GETREM ;GET THE RANDOM AUGMENT
9 021026 066005 000136      ADD     MINCYL(R0),R5 ;NEW CYLINDER ADDRESS
10 021032 010560 000126     1$:  MOV     R5,$NCYL(R0) ;STORE CYLINDER ADDRESS IN DPB
11
12          ;GENERATE A RANDOM TRACK ADDRESS BETWEEN VALUES 'MINTRK' & 'MAXTRK'
13
14 021036 016005 000140 RANTRK: MOV     MAXTRK(R0),R5 ;GET MAXIMUM TRACK ADDRESS
15 021042 026005 000142      CMP     MINTRK(R0),R5 ;'MINTRK' AND 'MAXTRK' THE SAME ?
16 021046 001407          BEQ     1$ ;BR IF THEY ARE
17 021050 166005 000142      SUB     MINTRK(R0),R5 ;GET NUMBER OF ALLOWABLE TRACKS
18 021054 005205          INC     R5 ;INCREMENT DIFFERENCE TO USE AS DIVISOR
19 021056 004737 032022      JSR     PC,GETREM ;GET THE RANDOM AUGMENT
20 021062 066005 000142      ADD     MINTRK(R0),R5 ;NEW TRACK ADDRESS
21 021066 110560 000125     1$:  MOV     R5,$NTRK(R0) ;STORE TRACK ADDRESS IN DPB
22
23          ;GENERATE A RANDOM SECTOR ADDRESS BETWEEN VALUES 'MINSEC' & 'MAXSEC'
24
25 021072 016005 000144 RANSEC: MOV     MAXSEC(R0),R5 ;GET MAXIMUM SECTOR ADDRESS
26 021076 026005 000146      CMP     MINSEC(R0),R5 ;'MINSEC' AND 'MAXSEC' THE SAME ?
27 021102 001407          BEQ     1$ ;BR IF THEY ARE
28 021104 166005 000146      SUB     MINSEC(R0),R5 ;GET NUMBER OF ALLOWABLE SECTORS
29 021110 005205          INC     R5 ;INCREMENT DIFFERENCE TO USE AS DIVISOR
30 021112 004737 032022      JSR     PC,GETREM ;GET THE RANDOM AUGMENT
31 021116 066005 000146      ADD     MINSEC(R0),R5 ;NEW SECTOR ADDRESS
32 021122 110560 000124     1$:  MOV     R5,$NSEC(R0) ;STORE SECTOR ADDRESS IN DPB
33
34          ;MAKE SURE ADDRESS JUST GENERATED IS NOT 'BAD SECTOR FILE'
35
36 021126 012704 000124      MOV     #$NSEC,R4 ;GET INDEX TO SECTOR ADDR STORAGE IN DPB
37 021132 004737 021332      JSR     PC,CKLMTS ;GO CHECK DISK ADDRESS LIMITS
38 021136 000404          BR     2$ ;BR IF NOT AT END OF SEQUENTIAL ADDRESSING
39 021140 004737 037126      JSR     PC,$RAND ;CYCLE THE RANDOM NUMBER GENERATOR
40 021144 000137 021002      JMP     RANCYL ;GO GENERATE NEW ADDRESS
41 021150
42
43          ;GENERATE A RANDOM BUFFER LENGTH BETWEEN 6 & THE VALUE IN 'WRDCNT'
44
45 021150 013705 001466 RANSIZ: MOV     WRDCNT,R5 ;GET MAX WORD COUNT
46 021154 005737 001500      TST     RANDWC ;SELECT A RANDOM WORD COUNT ?
47 021160 001011          BNE     2$ ;BR IF NOT
48 021162 005205          INC     R5 ;INCREMENT THE MAXIMUM WRD CNT
49 021164 00737 032022      JSR     PC,GETREM ;DIVIDE BY MAX VALUE
50 021170 020527 000006      CMP     R5,#6 ;WORD COUNT LESS THAN 6 ?
51 021174 002003          BGE     2$ ;BR IF NO
52 021176 004737 037126     1$:  JSR     PC,$RAND ;CYCLE THE RANDOM NUMBER GENERATOR
53 021202 000762          BR
54
55 021204 012704 000124     2$:  MOV     #$NSEC,R4 ;GET INDEX TO SECTOR STORAGE
56 021210 004737 021512      JSR     PC,CHKWC ;SEE IF WORD COUNT IS TOO LARGE TO FIT
57          ;IN REMAINDER OF TRACK. IF SO, THEN ADJUST

```

```

58
59 021214 122760 000002 000122 3$:   CMPB   #2,$NCODE(RO)   ;WORD COUNT TO FIT ON TRACK.
60 021222 001005                BNE    4$              ;WRITE OPERATION ?
61 021224 042705 000377          BIC    #377,R5         ;BR IF NO
62 021230 001002                BNE    4$              ;WRITING PARTIAL SECTOR ?
63 021232 012705 000400          MOV    #256.,R5        ;BR IF NO, ELSE,
64 021236 010560 000020 4$:   MOV    R5,$WRDL(RO)   ;WRITE AT LEAST ONE SECTOR
65                                     ;WORD COUNT
66                                     ;GET A RANDOM PATTERN NUMBER
67
68 021242 122760 000002 000122 RANPAT: CMPB   #2,$NCODE(RO)   ;WRITE OPERATION ?
69 021250 001004                BNE    RANXIT          ;BR IF NO
70 021252 004737 021272          JSR    PC,GETPAT       ;GET PATTERN CODE
71 021256 110560 000123          MOVB   R5,$NPATC(RO)   ;MOVE PATTERN CODE TO CONTROL BLOCK
72 021262 012760 177777 000130 RANXIT: MOV    #-1,$NEXT(RO)   ;SET PARAMETERS SELECTED INDICATOR
73 021270 000207                RTS    PC              ;RETURN
74
75                                     ;ROUTINE TO SELECT A PATTERN
76
77 021272 012705 000020          GETPAT: MOV   #16.,R5   ;SELECT PATTERN
78 021276 005737 001476          TST    PATTERN        ;ENABLE RANDOM PATTERN SELECTION ?
79 021302 001403                BEQ    1$              ;YES
80 021304 013705 001476          MOV    PATTERN,R5     ;USE INDEXED PATTERN
81 021310 000406                BR    2$              ;NO
82 021312 004737 037126 1$:   JSR    PC,$RAND        ;CYCLE THE RANDOM NUMBER GENERATOR
83 021316 004737 032022          JSR    PC,GETREM      ;GET CODE
84 021322 005705                TST    R5              ;WAS PATTERN ZERO SELECTED ?
85 021324 001762                BEQ    GETPAT          ;BR IF YES
86 021326 006305 2$:   ASL    R5              ;MAKE CODE INTO TABLE INDEX
87 021330 000207                RTS    PC
    
```

```

1      ;THIS ROUTINE IS USED TO CHECK ADDRESS LIMITS BEFORE THE NEXT COMMAND
2      ;IS PERFORMED. ALSO, IT WILL CHECK FOR MAXIMUM ADDRESS LIMITS TO LOOK
3      ;FOR AN END TO THE SEQUENTIAL ADDRESSING AND WILL MAKE SURE THE CURRENT
4      ;ADDRESS IS NOT THE 'BAD SECTOR FILE'
5      ;CALL:
6      ;
7      ;      MOV      #DPB,R0          ;DPB ADDRESS
8      ;      MOV      #POINTER,R4      ;POINTER TO SECTOR STORAGE ($SEC OR $NSEC) IN DPB
9      ;      JSR      PC,CKLMTS        ;CALL ADDRESS LIMITS ROUTINE
10     ;      BR       ???              ;RETURN HERE IF NOT END OF SEQUENTIAL ADDRESSING
11     ;      -----                  ;ELSE, RETURN HERE TO RESET DISK ADDRESS
12     ;
13     ;R0 = DPB ADDRESS BEFORE CALLING THE ROUTINE
14     ;R4 = POINTER TO SECTOR STORAGE BEFORE CAL.ING THE ROUTINE
15 021332 060004          CKLMTS: ADD      R0,R4          ;POINT TO SECTOR STORAGE POINT IN DPB
16 021334 026460 000002 000136  CMP      2(R4),MINCYL(R0) ;IS CYLINDER ADDRESS BELOW MIN. ?
17 021342 002003          BGE      1$              ;BR IF NO
18 021344 016064 000136 000002  MOV      MINCYL(R0),2(R4) ;RESET CYLINDER TO MIN.
19 021352 126460 000001 000142 1$:  CMPB   1(R4),MINTRK(R0) ;IS TRACK ADDRESS BELOW MIN. ?
20 021360 002003          BGE      2$              ;BR IF NO
21 021362 116064 000142 000001  MOVB   MINTRK(R0),1(R4) ;RESET TRACK TO MIN.
22 021370 121460 000146          2$:  CMPB   (R4),MINSEC(R0) ;IS SECTOR ADDRESS BELOW MIN. ?
23 021374 002002          BGE      3$              ;BR IF NO
24 021376 116014 000146          MOVB   MINSEC(R0),(R4) ;RESET SECTOR TO MIN.
25
26     ;LOOK FOR MAXIMUM LIMITS, FOR END OF SEQUENTIAL ADDRESSING AND
27     ;ALSO CHECK THAT ADDRESS IS NOT 'BAD SECTOR FILE'
28
29 021402 121460 000146          3$:  CMPB   (R4),MAXSEC(R0) ;IS SECTOR ADDRESS AT MAXIMUM ?
30 021406 003404          BLE      5$              ;BR IF NO
31 021410 116014 000146          MOVB   MINSEC(R0),(R4) ;RESET SECTOR ADDRESS
32 021414 105264 000001          4$:  INCB   1(R4)          ;INCREMENT TO NEXT TRACK ADDRESS
33 021420 126460 000001 000140  5$:  CMPB   1(R4),MAXTRK(R0) ;IS TRACK ADDRESS OVER MAXIMUM ?
34 021426 003407          BLE      6$              ;BR IF NO
35 021430 116014 000146          MOVB   MINSEC(R0),(R4) ;RESET SECTOR ADDRESS
36 021434 116064 000142 000001  MOVB   MINTRK(R0),1(R4) ;RESET TRACK ADDRESS
37 021442 005264 000002          INC      2(R4)          ;INCREMENT TO NEXT CYLINDER ADDRESS
38 021446 026460 000002 000134  6$:  CMP      2(R4),MAXCYL(R0) ;IS CYLINDER ADDRESS OVER MAXIMUM ?
39 021454 003403          BLE      7$              ;BR IF NO
40 021456 062716 000002          ADD     #2,(SP)        ;ADJUST RETURN TO RESET DISK ADDRESS PARAMETERS
41 021462 000412          BR       8$
42
43 021464 016046 000156          7$:  MOV     FE1(R0),-(SP) ;CHECK NOT TO READ OR WRITE BAD SECTOR TRACK
44 021470 005316          DEC     (SP)          ;GET 1ST FE CYLINDER (LAST CYL+1)
45 021472 026426 000002          CMP     2(R4),(SP)+   ;LOOK AT LAST USER CYLINDER
46 021476 001004          BNE     8$            ;ARE WE ON LAST USER CYLINDER ?
47 021500 126460 000001 000154  8$:  CMPB   1(R4),TRKLMT(R0) ;IS THIS THE BAD SECTOR TRACK ?
48 021506 001742          BEQ     4$            ;BR IF YES
49 021510 000207          RTS      PC          ;RETURN
    
```

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16 021512 060004
17 021514 105760 000146
18 021520 001023
19 021522 126060 000144 000152
20 021530 001017
21 021532 105760 000142
22 021536 001010
23 021540 126060 000140 000154
24 021546 001004
25
26
27
28 021550 026064 000134 000002
29 021556 001022
30 021560 126064 000140 000001 1$:
31 021566 001016
32
33 021570 111404
34 021572 016046 000144
35 021576 160416
36 021600 005004
37 021602 062704 000400
38 021606 005316
39 021610 002374
40 021612 005726
41 021614 020504
42 021616 003420
43 021620 010405
44 021622 000416
45
46 021624 016046 000156
47 021630 005316
48 021632 026426 000002
49 021636 001010
50 021640 016046 000154
51 021644 005316
52 021646 005046
53 021650 116416 000001
54 021654 022626
55 021656 001744
56 021660 000207

;THIS ROUTINE IS USED TO CALCULATE AND CHECK THE WORD COUNT FOR THE
;DRIVE THAT IS TO DO A DATA TRANSFER ON THE MAXIMUM TRACK OF THE MAXIMUM
;CYLINDER. IF THE CALCULATED MAXIMUM WORD COUNT, EXCEEDS THE DESIRED WORD
;COUNT (CONTENTS OF R5), THEN THE DESIRED WORD COUNT IS CHANGED, SO THAT
;THE WORD COUNT WILL NOT CAUSE A TRACK OVERFLOW DURING THE TRANSFER.
;CALL:
;      MOV      #DPB,R0          ;DPB ADDRESS
;      MOV      #POINTER,R4     ;POINTER TO SECTOR STORAGE ($SEC OR $NSEC) IN DPB
;      JSR      PC,CHKWC        ;CALL CHECK WORD COUNT ROUTINE
;      RETURN                     ;RETURN WITH R5 CONTAINING THE DESIRED WORD COUNT

;R0 = DPB ADDRESS BEFORE CALLING THE ROUTINE
;R4 = POINTER TO SECTOR STORAGE BEFORE CALLING THE ROUTINE
;R5 = DESIRED WORD COUNT BEFORE CALLING THE ROUTINE

CHKWC:  ADD      R0,R4           ;POINT TO SECTOR STORAGE POINT IN DPB
        TSTB    MINSEC(R0)     ;ALLOW SPIRAL RD/WRT ?
        BNE     2$              ;BR IF NO
        CMPB    MAXSEC(R0),SECLMT(R0) ;ALLOW SPIRAL RD/WRT ?
        BNE     2$              ;BR IF NO
        TSTB    MINTRK(R0)     ;ALLOW SPIRAL RD/WRT ?
        BNE     1$              ;BR IF NO
        CMPB    MAXTRK(R0),TRKLMT(R0) ;ALLOW SPIRAL RD/WRT ?
        BNE     1$              ;BR IF NO
;      WHEN SPIRAL RD/WRT IS ALLOWED, THEN CHECK
;      TO MAKE SURE YOU DO NOT SPIRAL OVER MAXIMUM
;      TRACK ON MAXIMUM CYLINDER
        CMP     MAXCYL(R0),2(R4) ;ON MAXIMUM CYLINDER ?
        BNE     4$              ;BR IF NO
        CMPB    MAXTRK(R0),1(R4) ;ON MAXIMUM TRACK ?
        BNE     4$              ;BR IF NO
;      ADJUST WORD COUNT TO FIT ON REMAINDER OF TRACK
;      GET STARTING SECTOR ($SEC OR $NSEC) ADDRESS
        MOVB    (R4),R4         ;GET MAXIMUM SECTOR
        MOV     R4,(SP)         ;GET NUMBER SECTORS TO BE XFERD
        CLR     R4              ;CLEAR R4
        ADD     #256.,R4        ;ADD 1 SECTOR OF WORDS TO R4
        DEC     (SP)           ;DONE ALL SECTORS YET ?
        BGE     3$              ;BR IF NO
        TST     (SP)+          ;RESTORE STACK
        CMP     R5,R4          ;TOO MANY WORDS FOR TRACK ?
        BLE     5$              ;BR IF NO
        MOV     R4,R5          ;YES, CHANGE WORD COUNT
        BR      5$

4$:     MOV     FE1(R0),-(SP)    ;GET 1ST FE CYLINDER (LAST CYL+1)
        DEC     (SP)           ;LOOK AT LAST USER CYLINDER
        CMP     2(R4),(SP)+     ;LAST USER CYLINDER ?
        BNE     5$              ;BR IF NO
        MOV     TRKLMT(R0),-(S) ;GET LAST TRACK
        DEC     (SP)           ;LOOK AT NEXT TO LAST TRACK
        CLR     -(SP)          ;PUSH STACK
        MOVB    1(R4),(SP)     ;GET CURRENT TRACK
        CMP     (SP)+,(SP)+    ;IS IT TRACK BEFORE BAD SECTOR TRACK ?
        BEQ     2$              ;BR IF YES (DON'T ALLOW SPIRAL TO BAD SEC TRK)
        BR      2$

5$:     RTS     PC
    
```


MAIN PROGRAM

```

1          ;ROUTINE TO GET THE PREVIOUSLY SELECTED PARAMETER VALUES
2          ;CALL:
3          :
4          :
5          :
6          :
7          :
8 021662   010546
9 021664   105760   000026
10 021670   001127
11 021672   116060   000166   000027
12 021700   142760   177701   000027
13 021706   132760   000006   000122
14 021714   001007
15 021716   016060   000012   000034
16 021724   016060   000010   000032
17 021732   000410
18 021734   004737   024364
19 021740   012660   000034
20 021744   112660   000033
21 021750   112660   000032
22
23 021754   005337   022176
24 021760   002007
25 021762   013737   022174   022176
26 021770   032777   000100   157156
27 021776   001413
28 022000   122760   000151   000002
29 022006   001060
30 022010   112760   000002   000024
31 022016   112760   000161   000002
32 022024   000451
33
34 022026   116060   000122   000024
35 022034   116005   000122
36 022040   116560   002076   000002
37 022046   122760   000151   000002
38 022054   001012
39 022056   122760   000060   000027
40 022064   001431
41 022066   112760   000171   000002
42 022074   112760   000004   000024
43
44 022102   116060   000123   000030
45 022110   016060   000124   000010
46 022116   016060   000126   000012
47 022124   012760   000400   000022
48 022132   132760   000001   000024
49 022140   001403
50 022142   062760   000002   000022
51 022150   016060   000020   000004
52 022156   016060   000020   000120
53 022164   005460   000004
54 022170   012605
55 022172   000207
56
57 022174   000000   000000

```

```

:ROUTINE TO GET THE PREVIOUSLY SELECTED PARAMETER VALUES
:CALL:
:
:
:
:
:
LODPAR: MOV    R5,-(SP)          ;SAVE R5
        TSTB   $PAK(R0)        ;'T' COMMAND FOR THIS DRIVE ?
        BNE    6$              ;BR IF NO
        MOVB   $RPCS1(R0),$PREV0(R0) ;SAVE CURRENT PARAMETERS
        BICB   #^C76,$PREV0(R0) ;STRIP GO,AND IE BITS
        BITB   #6,$NCODE(R0)   ;SEE IF NEXT OPERATION IS READ OR WRITE
        BNE    1$              ;BR IF EITHER
        MOV    $CYL(R0),$PREVA+2(R0) ;SAVE STARTING CYLINDER
        MOV    $SEC(R0),$PREVA(R0)  ;SAVE STARTING SECTOR AND TRACK
        BR     2$
1$:     JSR    PC,READDR        ;GET THE DECREMENTED SECTOR AND TRACK ADDRESSES
        MOV    (SP)+,$PREVA+2(R0)   ;CYLINDER ADDRESS
        MOVB   (SP)+,$PREVA+1(R0)   ;TRACK ADDRESS
        MOVB   (SP)+,$PREVA(R0)    ;SECTOR ADDRESS
2$:     DEC    ITCNT+2          ;REPEAT THIS COMMAND AGAIN ?
        BGE    3$              ;BR IF YES
        MOV    ITCNT,ITCNT+2      ;RESTORE ITERATION COUNT
        BIT    #SW06,@SWR        ;LOOP ON PREVIOUS VALUES (SW6=1) ?
        BEQ    4$              ;BR IF NO
3$:     CMPB   #WCKD,$COMND(R0)   ;IS COMMAND A WRITE CHECK DATA ?
        BNE    6$              ;BR IF NO
        MOVB   #2,$SCODE(R0)      ;CODE FOR WRTDAT
        MOVB   #WRTDAT,$COMND(R0) ;PUT WRITE DATA COMMAND BACK FOR LOOP
        BR     6$
4$:     MOVB   $NCODE(R0),$SCODE(R0) ;LOGICAL CODE FOR OPERATION
        MOVB   $NCODE(R0),R5      ;LOAD R5 FOR USE AS TABLE INDEX
        MOVB   COMTBL(R5),$COMND(R0) ;COMMAND CODE
        CMPB   #WCKD,$COMND(R0)   ;IS NEW COMMAND A WRITE CHECK DATA ?
        BNE    5$              ;BR IF NO
        CMPB   #60,$PREV0(R0)    ;WAS PREVIOUS COMMAND A WRITE DATA ?
        BEQ    6$              ;BR IF YES
        MOVB   #RDDAT,$COMND(R0)  ;CHANGE WRITE CHECK TO READ DATA COMMAND
        MOVB   #4,$SCODE(R0)      ;CODE NUMBER CHANGED TO READ DATA
5$:     MOVB   $NPATC(R0),$PATT(R0) ;PATTERN CODE
        MOV    $NSEC(R0),$SEC(R0)  ;TRACK AND SECTOR ADDRESSES
        MOV    $NICYL(R0),$CYL(R0) ;CYLINDER ADDRESS
        MOV    #256,$SSEC(R0)     ;INITIAL VALUE OF SECTOR SIZE
        BITB   #1,$SCODE(R0)      ;HEADER OPERATION ?
        BEQ    6$              ;BR IF NOT
        ADD    #2,$SSEC(R0)       ;ADD HEADER SIZE
6$:     MOV    $WRDL(R0),$WCNT(R0)  ;GET WORD COUNT AND
        MOV    $WRDL(R0),$HLDWC(R0) ;HOLD WORD FOR 'RELBUF' ROUTINE
        NEG    $WCNT(R0)          ;MAKE IT 2'S COMPLEMENT
        MOV    (SP)+,R5           ;RESTORE R5
        RTS    PC                 ;RETURN
ITCNT:  .WORD 0,0                ;'ITCNT' CONTAINS THE NUMBER OF TIMES TO

```



```

1          .SBTTL  ERROR MESSAGE GENERATION ROUTINES
2
3          :PRINT LINE 1 OF ERROR MESSAGE:
4          : 'HH:MM:SS'
5
6 022214 032777 002000 156732 LINE1: BIT      #SW10,@SWR      :SWITCH 10 SET ?
7 022222 001402                BEQ      1$              :BR IF NOT
8 022224 104401 001176                TYPE    ,SBELL          :RING THE BELL
9 022230 032777 020000 156716 1$: BIT      #SW13,@SWR      :INHIBIT TYPEOUT (SW13=1) ?
10 022236 001010                BNE     2$              :BR IF YES
11 022240 104401 001203                TYPE    ,SCRLF         :CR-LF
12 022244 104401 001203                TYPE    ,SCRLF         :CR-LF
13 022250 004737 026146                JSR     PC,$TIME       :TYPE ELAPSED TIME
14 022254 104401 056531                TYPE    ,BLNKS1       :TYPE 1 BLANK
15 022260 000207                2$:   RTS      PC        :RETURN & TYPE DESCRIPTION
16
17          :PRINT LINE 2 OF ERROR MESSAGE
18          : 'PRESENT ORDER = XXXX PREVIOUS ORDER = XXXX'
19          : '* ERROR AT BAD TRACK/SECTOR'
20          : 'DRIVE  RPCS1  RPCS2  RPDS1  RPER1  RPER2  RPER3  RPEC1'
21          : 'RPEC2  RPWC  RPBA  RPDA  RPAS  RPLA  RPDB  RPMR1'
22          : 'RPDT  RPSN  RPOF  RPDC  RPCC  STATUS'
23          : 'RPBAE  RPCS3' (RH70 ONLY)
24          : 'BUS ADDRESS OR WORD COUNT NOT CONSISTENT'
25          : 'RPA = XXXXXX  RPWC = XXXXXX'
26          : 'BUFFER ADR = XXXXXX  WRD CNT = XXXX  NMBR WRDS XFRD = XXX'
27
28          LINE2:
29          022262 010346  MOV     R3,-(SP)      ::PUSH R3 ON STACK
30          022262 010446  MOV     R4,-(SP)      ::PUSH R4 ON STACK
31          022266 010546  MOV     R5,-(SP)      ::PUSH R5 ON STACK
32          022270 104414 001203  DISPLY  ,SCRLF        :CR-LF
33          022274 005037 022422  CLR     4$           :CLEAR MESSAGE ADDRESS STORAGE
34          022300 005004  CLR     R4           :WORKING REGISTER
35          022302 012737 054536 022422  MOV     #LIN2C,4$     :ADDRESS OF 'PRESENT ORDER = ' MSG
36          022310 116004 000166  MOVB   $RPCS1(R0),R4 :GET THE OPCODE
37          022314 042704 177701  BIC    #^C76,R4     :SAVE ONLY SIGNIFICANT BITS
38          022320 004737 022356  JSR    PC,1$        :TYPE THE FIRST MNEMONIC
39          022324 005737 022426  TST    5$           :SEE IF MNEMONIC ENTRY FOUND
40          022330 001440  BEQ    LINE2A       :BR IF NOT
41          022332 012737 054556 022422  MOV     #LIN2P,4$     :ADDRESS OF 'PREVIOUS ORDER = ' MSG
42          022340 116004 000027  MOVB   $PREVO(R0),R4 :PREVIOUS OPERATION CODE
43          022344 042704 177701  BIC    #^C76,R4     :SAVE ON SIGNIFICANT BITS
44          022350 004737 022356  JSR    PC,1$        :TYPE THE PREVIOUS MNEMONIC
45          022354 000426  BR     LINE2A       :CONTINUE
46          022356 005005  1$:   CLR     R5           :CLEAR THE TABLE INDEX
47          022360 126504 002104  2$:   CMPB   OPTBL(R5),R4 :LOOK FOR THE OPCODE
48          022364 001405  BEQ    3$           :BR WHEN OPCODE COUNT = OPCODE
49          022366 105765 002104  TSTB   OPTBL(R5)    :LOOK FOR END OF TABLE
50          022372 100402  BMI    3$           :BR IF END
51          022374 005205  INC    R5           :INCREMENT THE POINTER
52          022376 000770  BR     2$           :CONTINUE - NOT END OF TABLE
53          022400 006305  3$:   ASL    R5           :SHIFT INDEX
54          022402 006305  ASL    R5           :SHIFT THE INDEX
55          022404 006305  ASL    R5           :SHIFT THE INDEX
56          022406 012737 002124 022426  MOV     #MNTBL,5$    :ADDRESS OF ASCII TEXT TABLE
57          022414 060537 022426  ADD    R5,5$        :ADD THE INDEX

```

55	022420	104414			DISPLY				:TYPE IT
56	022422	000000		4\$:	.WORD	0			:ADDRESS OF 'PRESENT' OR 'PREVIOUS' MESSAGE
57	022424	104414			DISPLY				:TYPE THE OPERATION MNEMONIC
58	022426	000000		5\$:	.WORD	0			:ADDRESS OF MESSAGE
59	022430	000207			RTS	PC			:RETURN TO MAIN ROUTINE
60									
61	022432	005737	001336	LINE2A:	TST	BADSEC			:PRINT THE BAD SECTOR LINE ?
62	022436	001404			BEQ	LINE2B			:BR IF NOT
63	022440	104414	001203		DISPLY	,\$CRLF			:CR-LF
64	022444	104414	054577		DISPLY	,\$LIN2S			:ERROR ADDRESS DEFINED AS BAD AREA
65	022450	104414	001203	LINE2B:	DISPLY	,\$CRLF			:CR-LF
66	022454	104414	054126		DISPLY	,\$DH14			:STANDARD RP07 REGISTER HEADER
67	022460	013746	001320		MOV	DRVNO,-(SP)			:PUT THE DRIVE NUMBER ON THE STACK
68	022464	004737	024306		JSR	PC,\$LINOCT			:TYPE IT
69	022470	104414	056530		DISPLY	,\$BLNKS2			:TYPE 2 BLANK
70	022474	012705	054450		MOV	,\$DT14,\$R5			:REGISTER INDEXES
71	022500	004737	022662		JSR	PC,\$3\$:PRINT THE REGISTERS
72									
73	022504	032777	000040	156442	BIT	,\$SW05,\$@SWR			:PRINT THE OPTIONAL REGISTERS ?
74	022512	001031			BNE	1\$:BR IF NOT
75	022514	104414	054225		DISPLY	,\$DH15			:STANDARD RP07 REGISTER HEADER
76	022520	012705	054470		MOV	,\$DT15,\$R5			:SECOND DATA LINE
77	022524	004737	022662		JSR	PC,\$3\$:PRINT THEM
78									
79	022530	104414	054324		DISPLY	,\$DH16			:STANDARD RP07 REGISTER HEADER
80	022534	012705	054512		MOV	,\$DT16,\$R5			:THIRD DATA LINE
81	022540	004737	022662		JSR	PC,\$3\$:PRINT THE REGISTERS
82									
83	022544	013746	001234		MOV	,\$CPUOP,-(SP)			:CHECK THE CPU (RH) TYPE
84	022550	042716	003777		BIC	,\$*C174000,(SP)			:LEAVE THE CPU BITS
85	022554	022726	030000		CMP	,\$#30000,(SP)+			:SEE IF RH70
86	022560	001006			BNE	1\$:BR IF NO
87	022562	104414	054404		DISPLY	,\$DH17			:STANDARD RP07 REGISTER HEADER
88	022566	012705	054530		MOV	,\$DT17,\$R5			:OPTIONAL FOURTH DATA LINE
89	022572	004737	022662		JSR	PC,\$3\$:PRINT THE REGISTERS
90	022576	032760	000100	000016	1\$:	BIT	,\$#BIT6,\$STATUS(R0)		:DATA ERROR ?
91	022604	001422			BEQ	2\$:BR IF NOT
92	022606	016046	000020		MOV	,\$WRDL(R0),-(SP)			:TRANSFER SIZE
93	022612	066016	000170		ADD	,\$RPWC(R0),(SP)			:ADD REMAINING WORD COUNT
94	022616	006316			ASL	(SP)			:CONVERT TO AN BYTE INCREMENT
95	022620	066016	000006		ADD	,\$BUF(R0),(SP)			:BUFFER STARTING ADDRESS
96	022624	022660	000172		CMP	(SP)+,\$RPBA(R0)			:CORRECT BUFFER ADDRESS ?
97	022630	001410			BEQ	2\$:BR IF YES
98	022632	104414	053226		DISPLY	,\$EM46			: 'BUS ADDRESS AND WORD COUNT ARE NOT CONSISTENT'
99	022636	104414	001203		DISPLY	,\$CRLF			:CR-LF
100	022642	004737	022754		JSR	PC,\$LINE3D			:PRINT LINE 3D OF ERROR MESSAGE
101	022646	004737	023366		JSR	PC,\$LINE4			:PRINT LINE 4 OF ERROR MESSAGE
102	022652			2\$:					
	022652	012605			MOV	(SP)+,\$R5			::POP STACK INTO R5
	022654	012604			MOV	(SP)+,\$R4			::POP STACK INTO R4
	022656	012603			MOV	(SP)+,\$R3			::POP STACK INTO R3
103	022660	000207			RTS	PC			:RETURN TO ERROR PROCESSING ROUTINE
104									
105	022662	012546		3\$:	MOV	(R5)+,-(SP)			:PUT THE REGISTER INDEX ON THE STACK
106	022664	060016			ADD	R0,(SP)			:ADD DRIVE'S TABLE ADDRESS
107	022666	017646	000000		MOV	@(SP),-(SP)			:VALUE
108	022672	004737	024306		JSR	PC,\$LINOCT			:TYPE IT

```

109 022676 005726
110 022700 104414 056530
111 022704 005715
112 022706 001365
113 022710 104414 001203
114 022714 000207
115
116
117
118
119 022716 104414 054633
120 022722 000517
121
122
123
124
125 022724 104414 054651
126 022730 000514
127
128
129
130
131 022732 004737 023270
132 022736 104414 001203
133 022742 000207
134
135
136
137
138 022744 004737 023270
139 022750 000137 023322
140
141
142
143
144 022754 032777 000040 156172
145 022762 001416
146 022764 104414 055010
147 022770 016046 000172
148 022774 004737 024306
149 023000 104414 055017
150 023004 016046 000170
151 023010 004737 024306
152 023014 104414 001203
153 023020 000207
154
155
156
157
158 023022 104414 054712
159 023026 016046 000012
160 023032 004737 024340
161 023036 104414 056530
162 023042 104414 055030
163 023046 005046
164 023050 116016 000011
165 023054 004737 024340

      TST      (SP)+      ;CORRECT THE STACK POINTER
      DISPLY   ,BLNKS2    ;TYPE 2 BLANKS
      TST      (R5)      ;AT END OF LINE ?
      BNE     3$         ;BR IF NOT
4$:    DISPLY   ,%CRLF    ;CR-LF
      RTS     PC         ;RETURN

;PRINT LINE 3 OF ERROR MESSAGE
;'ERROR AT CCC TT SS PREVIOUS ADR = CCC TT SS'
LINE3: DISPLY   ,LINM3    ;LINE 3 ENTRANCE
      BR      LIN3.1     ;FINISH PRINTOUT

;PRINT LINE 3A OF ERROR MESSAGE
;'START CYL= CCC END CYL= CCC'
LINE3A: DISPLY  ,LINN3    ;LINE 3A ENTRANCE
      BR      LIN3.1     ;FINISH ERROR LINE

;PRINT LINE 3B OF ERROR MESSAGE
;'START CYL= CCC END CYL= CCC ACTUAL CYL= CCC'
LINE3B: JSR     PC,LIN3.3 ;LINE 3B ENTRANCE
      DISPLY   ,%CRLF    ;
      RTS     PC

;PRINT LINE 3C OF ERROR MESSAGE
;'START CYL= CCC END CYL= CCC ACTUAL CYL= CCC TRK= TT'
LINE3C: JSR     PC,LIN3.3 ;LINE 3C ENTRANCE
      JMP     LIN3.4     ;FINISH MESSAGE

;PRINT LINE 3D OF ERROR MESSAGE
;'RPBA = XXXXXX RPWC = XXXXXX'
LINE3D: BIT     #SW05,@SWR ;SWITCH 5 SET ?
      BEQ     1$         ;BR IF IT IS
      DISPLY  ,LINB3    ;'RPBA = '
      MOV     $RPBA(RO),-(SP) ;BUFFER ADDR REG CONTENTS
      JSR     PC,LINOCT  ;CONVERT TO OCTAL AND TYPE IT
      DISPLY  ,LINW3    ;' RPWC = '
      MOV     $RPWC(RO),-(SP) ;WORD COUNT REGISTER CONTENTS
      JSR     PC,LINOCT  ;CONVERT TO OCTAL AND TYPE IT
1$:    DISPLY  ,%CRLF    ;
      RTS     PC

;PRINT LINE 3E OF ERROR MESSAGE
;'START CYL = CCC START TRK = TT START SEC = SS'
LINE3E: DISPLY  ,LINS3    ;'START CYL = '
      MOV     $CYL(RO),-(SP) ;MOVE CYL TO STACK
      JSR     PC,LINDEC  ;TYPE IT IN DECIMAL
      DISPLY  ,BLNKS2    ;TYPE 2 BLANKS
      DISPLY  ,LINST3    ;'START TRK = '
      CLR     -(SP)      ;CLEAR STACK
      MOVB   $TRK(RO),(SP) ;TRACK TO STACK
      JSR     PC,LINDEC  ;TYPE IT IN DECIMAL
    
```



```

223 023322 104414 054742          LIN3.4: DISPLY ,LINA3          ;PRINT 'ACTUAL CYL= '
224 023326 013746 063324          MOV      CYLNDR,-(SP)        ;ACTUAL CYLINDER
225 023332 042716 010000          BIC      #BIT12,(SP)        ;CLEAR THE FORMAT BIT
226 023336 004737 024340          JSR      PC,LINDEC          ;TYPE IT IN DECIMAL
227 023342 104414 054761          DISPLY  ,LINT3             ;PRINT 'TRK= '
228 023346 005046                   CLR      -(SP)              ;CLEAR STACK WORD
229 023350 116016 000175          MOV      $RPDA+1(RO),(SP)   ;PUT TRACK ON STACK
230 023354 004737 024340          JSR      PC,LINDEC          ;TYPE IT IN DECIMAL
231 023360 104414 001203          DISPLY  ,%CRLF             ;
232 023364 000207          RTS      PC
233
234          ;PRINT LINE 4 OF ERROR MESSAGE
235          ;'BUFFER ADR= XXXXXX WRD CNT= XXXX NMBR WRDS XFRD= XXX'
236
237 023366 032760 000100 000016  LINE4: BIT      #BIT06,$STATUS(RO) ;DATA ERROR ?
238 023374 001427          BEQ      1$                ;BR IF NOT
239 023376 104414 055060          DISPLY  ,LINM4             ;PRINT 'BUFFER ADR= '
240 023402 016046 000006          MOV      $BUF(RO),-(SP)    ;BUFFER ADDR ON STACK
241 023406 004737 024306          JSR      PC,LINOCT         ;CONVERT TO OCTAL & PRINT
242 023412 104414 055076          DISPLY  ,LINS4             ;PRINT 'WRD CNT= '
243 023416 016046 000020          MOV      $WRDL(RO),-(SP)   ;BUFFER SIZE
244 023422 004737 024340          JSR      PC,LINDEC          ;TYPE IT IN DECIMAL
245 023426 104414 055112          DISPLY  ,LINX4             ;' NMBR WRDS XFRD = '
246 023432 016046 000172          MOV      $RPBA(RO),-(SP)   ;VALUE IN BUFFER ADDR REGISTER
247 023436 166016 000006          SUB      $BUF(RO),(SP)     ;SUBTRACT STARTING ADDRESS
248 023442 006216          ASR      (SP)              ;CONVERT INTO A WORD COUNT
249 023444 004737 024340          JSR      PC,LINDEC          ;TYPE IT IN DECIMAL
250 023450 104414 001203          DISPLY  ,%CRLF             ;CR-LF
251 023454 000207          1$:      RTS      PC          ;RETURN
252
253          ;PRINT LINE 5 OF ERROR MESSAGE
254          ;'EXPCTD DATA= XXXXXX RECEVD DATA= XXXXXX WORD POS= XXX'
255
256 023456 104414 055135          LINE5: DISPLY ,LIND5          ;PRINT 'EXPCTD DATA'
257 023462 162760 000002 000172  SUB      #2,$RPBA(RO)        ;BACK THE ADDRESS UP
258 023470 013746 001234          MOV      $CPUOP,-(SP)      ;CHECK THE CPU (RH) TYPE
259 023474 042716 003777          BIC      #^C17400,(SP)     ;LEAVE THE CPU BITS
260 023500 022726 030000          CMP      #30000,(SP)+      ;SEE IF RH70
261 023504 001012          BNE      1$                ;BR IF NO
262 023506 162760 000004 000172  SUB      #4,$RPBA(RO)        ;BACKUP THE BUFFER POINTER
263 023514 032760 004000 000240  BIT      #BIT11,$RPCS3(RO)   ;SEE WHICH WORD HALF DIDN'T COMPARE
264 023522 001403          BEQ      1$                ;IF EQ, EVEN HALF DIDN'T COMPARE
265 023524 162760 000002 000172  SUB      #2,$RPBA(RO)        ;BACKUP THE BUFFER POINTER AGAIN
266 023532 017046 000172          1$:      MOV      @,$RPBA(RO),-(SP) ;'EXPCTD' DATA - AT THE BUFFER LOCATION
267 023536 004737 024306          JSR      PC,LINOCT         ;TYPE IT
268 023542 104414 055153          DISPLY  ,LIND5             ;PRINT 'RECEVD DATA'
269 023546 016046 000210          MOV      $RPDB(RO),-(SP)   ;RECEVD DATA FROM BUFFER
270 023552 004737 024306          JSR      PC,LINOCT         ;TYPE IT
271 023556 016046 000170          MOV      $RPWC(RO),-(SP)   ;WORD LENGTH ON STACK
272 023562 066016 000020          ADD      $WRDL(RO),(SP)    ;MAKE INTO A POSITIVE NUMBER
273 023566 005046          CLR      -(SP)              ;UPPER DIVIDEND TO ZERO
274 023570 016046 000022          MOV      $SSEC(RO),-(SP)   ;SECTOR SIZE ON THE STACK
275 023574 004737 032046          JSR      PC,$DIV           ;DIVIDE WORDS XFERED BY SECTOR SIZE
276 023600 012616          MOV      (SP)+,(SP)        ;MOVE REMAINDER UP THE STACK
277 023602 005316          DEC      (SP)              ;DECREMENT WORD POSITION BY 1
278 023604 032760 040000 000176  BIT      #BIT14,$RPCS2(RO)   ;IS 'WCE' SET ?
279 023612 001416          BEQ      2$                ;BR IF NO
    
```

```

280 023614 013746 001234      MOV      $CPUOP,-(SP)      ;CHECK THE CPU (RH) TYPE
281 023620 042716 003777      BIC      #^C174000,(SP)   ;LEAVE THE CPU BITS
282 023624 022720 030000      CMP      #30000,(SP)+    ;SEE IF RH70
283 023630 001007              BNE      2$              ;BR IF NO
284 023632 162716 000002      SUB      #2,(SP)         ;SUBTRACT 2 FOR A DOUBLE WORD
285 023636 032760 004000 000240 BIT      #BIT11,$RPCS3(RO) ;SEE WHICH WORD HALF DIDN'T COMPARE
286 023644 001401              BEQ      2$              ;IF EQ, EVEN HALF DIDN'T COMPARE
287 023646 005316              DEC      (SP)            ;SUBTRACT 1 FOR AN ODD WORD
288 023650 104414 055173      2$:     DISPLY  ,LINS5      ;PRINT 'WORD POS'
289 023654 004737 024340      JSR      PC,LINDEC      ;TYPE THE POSITION
290 023660 104414 001203      DISPLY  ,$CRLF
291 023664 000207      RTS      PC
292
293      ;PRINT LINE 5A OF THE ERROR MESSAGE
294      ;'HEADER FROM ERROR SECTOR  XXXXXX  XXXXXX  XXXXXX  XXXXXX'
295
296 023666      LINE5A:
297 023666 104414 055210      DISPLY  ,LINS5          ;'HEADER CONTENTS OF ERROR SECTOR'
302 023672 013746 063324      MOV      (CYLNR,-(SP)    ;HEADER POSITION
      023676 004737 024306      JSR      PC,LINOC1      ;TYPE IT
      023702 104414 056530      DISPLY  ,BLNKS2        ;TYPE 2 BLANKS
      023706 013746 063326      MOV      (CYLNR+2,-(SP) ;HEADER POSITION +2
      023712 004737 024306      JSR      PC,LINOC1      ;TYPE IT
      023716 104414 056530      DISPLY  ,BLNKS2        ;TYPE 2 BLANKS
303 023722 104414 056533      DISPLY  ,LINS5
304 023726 104414 001203      DISPLY  ,$CRLF
305 023732 000207      RTS      PC
306
307      ;PRINT LINE 5B OF ERROR MESSAGE
308      ;'RPEC1 = XXXXXX  RPEC2 = XXXXXX'
309
310 023734 104414 055243      LINE5B: DISPLY  ,LINE55   ;'RPEC1 = '
311 023740 016046 000232      MOV      $RPEC1(RO),-(SP) ;PUT REGISTER CONTENTS ON THE STACK
312 023744 004737 024306      JSR      PC,LINOC1      ;TYPE IT
313 023750 10 14 056530      DISPLY  ,BLNKS2        ;TYPE 2 BLANKS
314 023754 10 14 055253      DISPLY  ,LINE05        ;' RPEC2 = '
315 023760 01 146 000234      MOV      $RPEC2(RO),-(SP) ;PUT REGISTER CONTENTS ON THE STACK
316 023764 00 137 024306      JSR      PC,LINOC1      ;TYPE IT
317 023770 104 14 001203      DISPLY  ,$CRLF
318 023774 000207      RTS      PC              ;RETURN
319
320      ;PRINT LINE 6 OF ERROR MESSAGE
321      ;'SECTOR IS ECC CORRECTABLE'
322
323 023776 104414 055265      LINE6:  DISPLY  ,LINB6   ;PRINT 'SECTOR IS ECC CORRECTABLE '
324 024002 104414 001203      DISPLY  ,$CRLF
325 024006 000207      RTS      PC
326
327      ;PRINT LINE 6A OF THE ERROR MESSAGE
328      ;'SECTOR READ CORRECTLY AFTER N RETRY(S)'
329
330 024010 104414 055320      LINE6A: DISPLY  ,LINC6   ;PRINT 'SECTOR READ CORRECTLY AFTER N RETRY(S)'
331 024014 000406      BR      LIN6.2         ;TYPE THE REST OF THE LINE
332
333      ;PRINT LINE 6C OF THE ERROR MESSAGE
334      ;'CORRECTED ON N RETRY(S)'
335
    
```



```

336 024016 104414 055355 LINE6C: DISPLY ,LING6 ;'CORRECTED ON N RETRY(S)'  

337 024022 000403 BR LIN6.2 ;TYPE THE REST OF THE LINE  

338  

339 ;PRINT LINE 6D OF THE ERROR MESSAGE  

340 ;'UNCORRECTABLE AFTER N RETRY(S)'  

341  

342 024024 104414 055405 LINE6D: DISPLY ,LINU06 ;'UNCORRECTABLE AFTER N RETRY(S)'  

343 024030 000400 BR LIN6.2 ;FINISH  

355  

356 ;RETRY COUNT TYPEOUT  

357  

358 024032 005046 LIN6.2: CLR -(SP) ;CLEAR STACK  

359 024034 113716 001325 MOVB RETRY+1,(SP) ;RETRY COUNT  

360 024040 004737 024340 JSR PC,LINDEC ;TYPE IT IN DECIMAL  

361 024044 104414 055373 DISPLY ,LINR6 ;' RETRY(S)'  

362 024050 104414 001203 DISPLY ,%CRLF  

363 024054 000207 RTS PC  

364  

365 ;PRINT LINE 7 OF THE ERROR MESSAGE  

366 ;'TOTALS: ERRORS:XXX WRDS WRITN:XXXXX X10^6 WRDS READ:XXXXX X10^6'  

367  

377 024056 104414 055514 LINE7: DISPLY ,LIN7T ;TOTALS: ERRORS  

378 024062 016046 000102 MOV $TOTAL(RO),-(SP) ;TO STACK  

379 024066 004737 024340 JSR PC,LINDEC ;TYPE IT IN DECIMAL  

380 024072 104414 055535 DISPLY ,LIN7X ;PRINT ' WRDS WRITN'  

381 024076 012746 000056 MOV #SWTTL,-(SP) ;ADDRESS OF LOW WORD ON STACK  

382 024102 060016 ADD RO,(SP)  

383 024104 004737 037324 JSR PC,$DB2D ;CONVERT  

384 024110 004737 032446 JSR PC,$SUPRL ;PRINT  

385 024114 104414 057711 DISPLY ,PERIOD ;TYPE '.'  

386 024120 104414 057552 DISPLY ,MSGX10 ;TYPE ' X10^6'  

387 024124 104414 055553 DISPLY ,LIN7R ;PRINT ' WRDS READ'  

388 024130 012746 000066 MOV #RRTOTL,-(SP) ;LOW WORD ADDRESS  

389 024134 060016 ADD RO,(SP)  

390 024136 004737 037324 JSR PC,$DB2D ;CONVERT  

391 024142 004737 032446 JSR PC,$SUPRL ;PRINT IT  

392 024146 104414 057711 DISPLY ,PERIOD ;TYPE '.'  

393 024152 104414 057552 DISPLY ,MSGX10 ;TYPE ' X10^6'  

394 024156 104414 001203 DISPLY ,%CRLF ;CR-LF  

395 024162 032777 100000 154764 BIT #SW15,@SWR ;SEE IF 'HALT ON ERROR' - SWITCH 15  

396 024170 001401 BEQ 1$ ;BR IF NOT  

397 024172 001000 HALT ;SWITCH 15 HALT  

398 024174 000207 1$ RTS PC  

399  

400 ;PRINT LINE 7A OF ERROR MESSAGE  

401 ;'TOTALS: SEEKS= XXXXX MIS POS ERRORS= XXX SKI ERRORS= XXX'  

402  

412 024176 104414 055455 LINE7A: DISPLY ,LIN7P ;'TOTALS: SEEK' = '  

413 024202 012746 000076 MOV #SSTCTL,-(SP) ;TOTAL SEEKS  

414 024206 060016 ADD RO,(SP) ;LEVICC TABLE ADDRESS  

415 024210 004737 037324 JSR PC,$DL2D ;CONVERT THE SEEK COUNT  

416 024214 004737 032446 JSR PC,$SUPRL ;PRINT IT  

417 024220 104414 057711 DISPLY ,PERIOD ;TYPE '.'  

418 024224 104414 055432 DISPLY ,LIN7M ;' MIS POS ERRORS= '  

419 024230 016046 000112 MOV $MISPO(RO),-(SP) ;TOTAL ERRORS  

420 024234 004737 024340 JSR PC,LINDEC ;TYPE IT IN DECIMAL  

421 024240 104414 055475 DISPLY ,LIN7N ;' SKI ERRORS= '
    
```

```

422 024244 016046 000110      MOV    $SKI(RO),-(SP)  ;CONVERT & PRINT IT
423 024250 004737 024340      JSR    PC,LINDEC     ;TYPE IT IN DECIMAL
424 024254 104414 001203      DISPLY ,%CRLF        ;CR-LF
425 024260 032777 100000 154666    BIT    #SW15,@SWR    ;SEE IF HALT ON ERROR - SWITCH 15 SET
426 024266 001401          BEQ    1$            ;BR IF NOT
427 024270 000000          HALT                          ;SWITCH 15 HALT
428 024272 000207      1$:    RTS    PC
429
430      ;PRINT LINE 8 OF THE ERROR MESSAGE
431      ;'DIFFERENT ERROR DURING RETRY'
432
433 024274 104414 000170      LINE8: DISPLY ,LIN8M
434 024300 004737 022262      JSR    PC,LINE2     ;PRINT LINE 2 OF ERROR MESSAGE
435 024304 000207      RTS    PC
436
437      ;OCTAL TYPEOUT ROUTINE
438      ;CALL:
439      ;
440      ;   MOV    NUM, -(SP)      ;PUT THE NUMBER ON THE STACK
441      ;   JSR    PC,LINOCT
442      ;   RETURN
443 024306 016646 000002      LINOCT: MOV    2(SP), -(SP)    ;PUT NUMBER IN PROPER LOCATION ON STACK
444 024312 004737 033362      JSR    PC,$SB20     ;CONVERT THE NUMBER TO OCTAL
445 024316 012637 024332      MOV    (SP)+, 1$    ;GET THE ADDRESS OF THE ASCII STRING
446 024322 004737 000005 024332    ADD    #5., 1$      ;ADDRESS THE LAST 6 ASCII DIGITS
447 024330 104414          DISPLY          ;TYPE IT
448 024332 000000      1$:    .WORD    0      ;ADDRESS
449 024334 012616      MOV    (SP)+, (SP)  ;CORRECT THE STACK
450 024336 000207      RTS    PC          ;RETURN
451
452      ;ROUTINE TO CONVERT THE INPUT NUMBER TO DECIMAL AND TYPE IT
453      ;LEFT JUSTIFIED
454      ;CALL:
455      ;
456      ;   MOV    NUM, -(SP)      ;PUT THE NUMBER ON THE STACK
457      ;   JSR    PC,LINDEC
458      ;   RETURN
459 024340 016646 000002      LINDEC: MOV    2(SP), -(SP)   ;SET UP STACK FOR CONVERT
460 024344 004737 033332      JSR    PC,$SB20     ;CONVERT IT TO DECIMAL
461 024350 004737 032446      JSR    PC,$SUPRL    ;TYPE IT (LEFT JUSTIFIED)
462 024354 104414 057711      DISPLY ,PERIOD      ;TYPE '.'
463 024360 012616      MOV    (SP)+, (SP)  ;RESTORE STACK POINTER
464 024362 000207      RTS    PC
465
466      .SBTTL  GENERAL SUPPORT SUBROUTINES
467
468      ;DECREMENT THE SECTOR-TRACK ADDRESS
469      ;CALL:
470      ;
471      ;   MOV    #DPB, RO      ;DPB ADDRESS
472      ;   JSR    PC,READDR
473      ;   RETURN
474      ;ON RETURN THE STACK CONTAINS THE FOLLOWING:
475      ;   4(SP) = SECTOR ADDRESS
476      ;   2(SP) = TRACK ADDRESS
477      ;   (SP) = CYLINDER ADDRESS
478
    
```

479	024364	162706	000006		READDR: SUB	#6,SP	:DECREMENT THE STACK POINTER
480	024370	016616	000006		MOV	6(SP),(SP)	:MOVE THE RETURN ADDR DOWN THE STACK
481	024374	005066	000006		CLR	6(SP)	:CLEAR STACK FOR SECTOR
482	024400	005066	000004		CLR	4(SP)	:CLEAR STACK FOR TRACK
483	024404	116066	000174	000006	MOVB	\$RPDA(R0),6(SP)	:SECTOR ON STACK
484	024412	116066	000175	000004	MOVB	\$RPDA+1(R0),4(SP)	:TRACK ADDRESS
485	024420	016066	000222	000002	MOV	\$RPDC(R0),2(SP)	:CYLINDER ADDRESS
486	024426	005766	000006		1\$: TST	6(SP)	:SECTOR 0 ?
487	024432	001403			BEQ	2\$:BRANCH IF SO
488	024434	105366	000006		DECB	6(SP)	:DECREMENT ONE SECTOR
489	024440	000424			BR	4\$:BRANCH TO EXIT
490	024442	005766	000004		2\$: TST	4(SP)	:ALSO ON TRACK 0 ?
491	024446	001406			BEQ	3\$:BRANCH IF SO
492	024450	116066	000152	000006	MOVB	SECLMT(R0),6(SP)	:LAST SECTOR
493	024456	105366	000004		DECB	4(SP)	:DECREMENT ONE TRACK
494	024462	000413			BR	4\$:EXIT
495	024464	005766	000002		3\$: TST	2(SP)	:ALSO ON CYLINDER 0 ?
496	024470	001410			BEQ	4\$:BRANCH IF SO
497	024472	116066	000152	000006	MOVB	SECLMT(R0),6(SP)	:LAST SECTOR
498	024500	116066	000154	000004	MOVB	TRKLMT(R0),4(SP)	:GET LAST TRACK
499	024506	005366	000002		DEC	2(SP)	:DECREMENT ONE CYLINDER COUNT
500	024512	000207			4\$: RTS	PC	:RETURN

```

1          .SBTTL  KW11 CLOCK CHECK ROUTINE
2
3          ;ROUTINE TO CHECK FOR KW11-L OR KW11-P CLOCKS
4
5 024514 005037 001310          CKCLK: CLR      CLKFLG          ;ASSUME 'NO CLOCK' AVAILABLE
6 024520 013746 000004          MOV      ERRVEC, -(SP)      ;:PUSH ERRVEC ON STACK
7 024524 012737 024576 000004  MOV      #CKCLK1, ERRVEC    ;:SETUP ERROR TRAP VECTOR FOR P-CLOCK CHECK
8 024532 005777 154540          TST      @CLKCSR          ;:CHECK FOR KW11-P
9 024536 012737 000001 001310  MOV      #1, CLKFLG        ;:SET P-CLOCK FLAG
10 024544 013701 001302          MOV      SLPVEC, R1        ;:KW11-P VECTOR ADDRESS
11 024550 012721 026272          MOV      #KWSVR, (R1)+     ;:SETUP KW11-P VECTOR
12 024554 012711 000300          MOV      #PR6, (R1)        ;:SET INTERRUPT PRIORITY TO 6
13 024560 012777 174575 154512  MOV      #-1667, @CLKCSB   ;:LOAD COUNTER BUFFER WITH 16.67
14 024566 012777 000131 154502  MOV      #131, @CLKCSR     ;:SET KW11-P INTERRUPT, CNT UP, 10US, REPEAT MODE
15 024574 000442
16
17 024576 012716 024604          CKCLK1: MOV     #1$, (SP)    ;:SETUP RETURN ADDRESS
18 024602 000002          RTI
19 024604 012737 024650 000004  1$:      MOV     #CKCLK2, ERRVEC ;:SETUP ERROR TRAP VECTOR FOR L-CLOCK CHECK
20 024612 005777 154466          TST      @CLKS           ;:CHECK FOR KW11-L
21 024616 012737 177777 001310  MOV      #-1, CLKFLG      ;:SET L-CLOCK FLAG
22 024624 013701 001306          MOV      SLLVEC, R1       ;:KW11-L VECTOR ADDRESS
23 024630 012721 026272          MOV      #KWSVR, (R1)+    ;:SETUP KW11-L VECTOR
24 024634 012711 000300          MOV      #PR6, (R1)       ;:SET INTERRUPT PRIORITY TO 6
25 024640 012777 000100 154436  MOV      #100, @CLKS      ;:SET KW11-L INTERRUPT
26 024646 000415          BR
27
28 024650 012716 024656          CKCLK2: MOV     #1$, (SP)    ;:SETUP RETURN ADDRESS
29 024654 000002          RTI
30 024654 104401 057634          1$:      TYPE    ,NEDCLK      ;:'P OR L CLOCK MUST BE ON SYSTEM'
31 024662 105737 001150          TSTB    $AUTOB          ;:RUNNING IN AUTO MODE ?
32 024666 001402          BEQ     2$              ;:BR IF NOT
33 024670 000137 031744          JMP     $GET42          ;:ABORT PROGRAM
34 024674 000000          2$:      HALT
35 024676 000137 003522          JMP     START          ;:TRY AGAIN
36
37 024702 012637 000004          CKCLK3: MOV     (SP)+, ERRVEC ;:RESTORE THE ERROR VECTOR
38 024706 000207          RTS      PC
39
40          ;THIS ROUTINE IS USED TO SHUT OFF INTERRUPT TO THE SYSTEM CLOCK
41          ;CALL
42          ;
43          ;      JSR      PC,CLKOFF      ;CALL ROUTINE
44 024710 005737 001310          CLKOFF: TST     CLKFLG      ;:IS CLOCK AVAILABLE ?
45 024714 001410          BEQ     2$              ;:BR IF NO
46 024716 100404          BMI    1$              ;:BR IF L-CLOCK
47 024720 042777 000101 154350  BIC     #101, @CLKCSR     ;:SHUT OFF KW11-P INTERRUPT
48 024726 000403          BR     2$
49 024730 042777 000100 154346  1$:      BIC     #100, @CLKS    ;:SHUT OFF KW11-L INTERRUPT
50 024736 000207          2$:      RTS      PC          ;EXIT

```

```

1
2
3      ;ROUTINE TO DISPLAY STATISTICS FOR ASSIGNED DRIVES
4      ;CALL:
5      ;      JSR      PC,STATPR
6      ;      RETURN
7
8      024741 010046
9      024742 010446
10     024744 005737 001542
11     024750 001431
12     21 024752 005004
13     22 024754 104401 001203
14     23 024760 006304
15     24 024762 016400 002056
16     25 024766 006204
17     26 024770 136437 045550 001542
18     27 024776 001412
19     28 025000 104401 001203
20     29 025004 004737 026146
21     30 025010 104401 057321
22     31 025014 104401 057345
23     32 025020 004737 025070
24     33 025024 005204
25     34 025026 020427 000010
26     35 025032 001352
27     36 025034
28     025034 012604
29     025036 012600
30     37 025040 000207
31
32
33     ;ROUTINE TO TYPE THE SUMMARY FOR ONLY ONE DRIVE
34     ;CALL:
35     ;      MOV      #DPB,R0      ;DPB ADDRESS
36     ;      JSR      PC,ONESUM
37     ;      RETURN
38
39     ONESUM: MOV      R0,-(SP)      ;SAVE R0
40             MOV      R4,-(SP)      ;SAVE R4
41             CLR      R4            ;CLEAR R4 FOR DRIVE NUMBER
42             MOV      (R0),R4       ;DRIVE NUMBER
43             JSR      PC,TYF_JM     ;TYPE THE SUMMARY
44             TYPE     ,DASH13      ;TYPE '-----'
45             MOV      (SP)+,R4      ;RESTORE R4
46             MOV      (SP)+,R0      ;RESTORE R0
47             RTS      PC           ;RETURN
48
49     ;TYPE THE SUMMARY
50     ;CALL:
51     ;      MOV      #DRIVE,R4     ;DRIVE NUMBER
52     ;      MOV      #DPB,R0      ;DPB ADDRESS
53     ;      RETURN
54
55     TYF_JM: NOP
56
57     ;TYPE REST OF SUMMARY LINE 1
58     ;TYPE     ,DRVMSG             ;TYPE 'DRIVE'
59
60
61
62
63
64
65
66
67
68

```

```

69 025076 010446          MOV      R4,-(SP)          ;;SAVE R4 FOR TYPEOUT
      025100 104403          TYP0S          ;;TYPE DRIVE NUMBER
      025102      002        .BYTE      2          ;;GO TYPE--OCTAL ASCII
      025103      000        .BYTE      0          ;;TYPE 2 DIGIT(S)
70 025104 104401 056531    TYPE      ,BLNKS1      ;;SUPPRESS LEADING ZEROS
71 025110 104401 056575    TYPE      ,DASH         ;;TYPE 1 BLANK
72 025114 104401 C56531    TYPE      ,BLNKS1      ;;TYPE 1 BLANK
73 025120 012737 057075 025146 MOV      #SRP07,2$      ;;ADDRESS OF RP07 MESSAGE
74 025126 122764 000005 040464 CMPB     #5,DRV1YP(R4)  ;;IS DEVICE AN RP07 ?
75 025134 001403          BEQ       1$           ;;BR IF YES
76 025136 012737 057102 025146 MOV      #SRP07P,2$     ;;ADDRESS OF RP07+ MESSAGE
77 025144 104401          1$:      TYPE          ;;TYPE THE DRIVE TYPE MESSAGE
78 025146 000000          2$:      .WORD      0          ;;MESSAGE ADDRESS HERE
79 025150 104401 056573    TYPE      ,COMMA       ;;TYPE ','
80 025154 104401 056531    TYPE      ,BLNKS1      ;;TYPE 1 BLANK
81 025160 004737 032764    JSR      PC,TYDRV      ;;TYPE DRIVE SERIAL NUMBER
82 025164 104401 025172    TYPE      ,65$        ;;TYPE ASCII STRING
      025170 000404          BR        64$        ;;GET OVER THE ASCIIZ
      025202          ;;65$: .ASCIIZ /, PASS /
      64$:
83 025202 016046 000114    MOV      $PASSC(RO),-(SP) ;;PUT THE PASS COUNT ON THE STACK
84 025206 004737 033332    JSR      PC,$SB2D      ;;CONVERT IT
85 025212 004537 032552    JSR      R5,REPLZ      ;;TYPE IT
86 025216 000003          .WORD      3          ;;TYPE 3 DIGITS
87 025220 104401 057711    TYPE      ,PERIOD      ;;TYPE '.'
88
89
90          ;;TYPE LINE 2 OF SUMMARY
91          ;;:      TYPE      ,%CRLF          ;;CR-LF
92
93 025224 104401 025232    ;;TYPE LINE 3 OF SUMMARY
      025230 000407    TYPE      ,67$        ;;TYPE ASCIIZ STRING
      BR        66$        ;;GET OVER THE ASCIIZ
      ;;67$: .ASCIIZ <CRLF>/WRDS WRITN /
      66$:
94 025250 104401 057532    TYPE      ,MSPASS      ;;TYPE '/ PASS '
95 025254 010046          MOV      RO,-(SP)      ;;GET ADDRESS OF DPB
96 025256 062716 000042    ADD      #SWTPAS,(SP)  ;;POINT TO LOW NUMBER OF WRDS WRITTEN PER PASS
97 025262 004737 037324    JSR      PC,$DB2D      ;;CONVERT DECIMAL NUMBER
98 025266 004737 032362    JSR      PC,SUPRS      ;;SUPPRESS LEADING ZEROS AND TYPE
99 025272 104401 057711    TYPE      ,PERIOD      ;;TYPE '.'
100 025276 104401 057542   TYPE      ,MSTOTL      ;;TYPE ' TOTAL '
101 025302 010046          MOV      RO,-(SP)      ;;GET ADDRESS OF DPB
102 025304 062716 000056   ADD      #SWTOTL,(SP)  ;;POINT TO LOW NUMBER OF WRDS WRITTEN
103 025310 004737 037324    JSR      PC,$DB2D      ;;CONVERT DECIMAL NUMBER
104 025314 004737 032362    JSR      PC,SUPRS      ;;SUPPRESS LEADING ZEROS AND TYPE
105 025320 104401 057711    TYPE      ,PERIOD      ;;TYPE '.'
106 025324 104401 057552   TYPE      ,MSGX10      ;;TYPE ' x10^6 '
107
108          ;;TYPE LINE 4 OF SUMMARY
109 025330 104401 025336    TYPE      ,69$        ;;TYPE ASCIIZ STRING
      025334 000407    BR        68$        ;;GET OVER THE ASCIIZ
      ;;69$: .ASCIIZ <CRLF>/WRDS READ /
      68$:
110 025354 104401 057532    TYPE      ,MSPASS      ;;TYPE '/ PASS '
111 025360 010046          MOV      RO,-(SP)      ;;GET ADDRESS OF DPB
112 025362 062716 000036   ADD      #SRDPAS,(SP)  ;;POINT TO LOW NUMBER OF WRDS READ PER PASS
  
```

```

113 025366 004737 037324 JSR PC,$DB2D ;CONVERT DECIMAL NUMBER
114 025372 004737 032362 JSR PC,SUPRS ;SUPPRESS LEADING ZEROS AND TYPE
115 025376 104401 057711 TYPE ,PERIOD ;TYPE ' '
116 025402 104401 057542 TYPE ,MSTOTL ;TYPE ' ' TOTAL '
117 025406 010046 MOV RO,-(SP) ;GET ADDRESS OF DPB
118 025410 062716 000066 ADD #RSTOTL,(SP) ;POINT TO LOW NUMBER OF WRDS READ
119 025414 004737 037324 JSR PC,$DB2D ;CONVERT DECIMAL NUMBER
120 025420 004737 032362 JSR PC,SUPRS ;SUPPRESS LEADING ZEROS AND TYPE
121 025424 104401 057711 TYPE ,PERIOD ;TYPE ' '
122 025430 104401 057552 TYPE ,MSGX10 ;TYPE ' 'x10^6'
123
124
125 025434 104401 025442 ;TYPE LINE 5 OF SUMMARY
025440 000407 TYPE 71$ ;;TYPE ASCIZ STRING
BR 70$ ;;GET OVER THE ASCIZ
70$: .ASCIZ <CRLF>/SEEKS /
126 025460 104401 057532 TYPE ,MSPASS ;TYPE '/ PASS '
127 025464 010046 MOV RO,-(SP) ;PUT $STOTL ON THE STACK
128 025466 062716 000046 ADD #SSEKS,(SP) ;POINT TO LOW NUMBER OF SEEK COUNT
129 025472 004737 037324 JSR PC,$DB2D ;CONVERT DECIMAL NUMBER
130 025476 004737 032362 JSR PC,SUPRS ;SUPPRESS LEADING ZEROS AND TYPE
131 025502 104401 057711 TYPE ,PERIOD ;TYPE ' '
132 025506 104401 057542 TYPE ,MSTOTL ;TYPE ' ' TOTAL '
133 025512 010046 MOV RO,-(SP) ;PUT $STOTL ON THE STACK
134 025514 062716 000076 ADD #SSTOTL,(SP) ;POINT TO LOW NUMBER OF SEEK COUNT
135 025520 004737 037324 JSR PC,$DB2D ;CONVERT DECIMAL NUMBER
136 025524 004737 032362 JSR PC,SUPRS ;SUPPRESS LEADING ZEROS AND TYPE
137 025530 104401 057711 TYPE ,PERIOD ;TYPE ' '
146
147
148 025534 104401 025542 ;TYPE LINES 6 AND 7 OF SUMMARY
025540 000412 TYPE 73$ ;;TYPE ASCIZ STRING
BR 72$ ;;GET OVER THE ASCIZ
72$: .ASCIZ <CRLF>'CUMULATIVE ERRORS:/
149 025566 104401 025574 TYPE 75$ ;;TYPE ASCIZ STRING
025572 000404 BR 74$ ;;GET OVER THE ASCIZ
74$: .ASCIZ <CRLF>/SOFT /
150 025604 016046 000104 MOV $SOFT(RO),-(SP) ;;SAVE $SOFT(RO) FOR TYPEOUT
025610 104405 TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN
151 025612 104401 057711 TYPE ,PERIOD ;TYPE ' '
152 025616 104401 025624 TYPE 77$ ;;TYPE ASCIZ STRING
025622 000404 BR 76$ ;;GET OVER THE ASCIZ
76$: .ASCIZ / HARD /
153 025634 016046 000106 MOV $HARD(RO),-(SP) ;;SAVE $HARD(RO) FOR TYPEOUT
025640 104405 TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN
154 025642 104401 057711 TYPE ,PERIOD ;TYPE ' '
155 025646 104401 025654 TYPE 79$ ;;TYPE ASCIZ STRING
025652 000403 BR 78$ ;;GET OVER THE ASCIZ
78$: .ASCIZ / SKI /
156 025662 016046 000110 MOV $SKI(RO),-(SP) ;;SAVE $SKI(RO) FOR TYPEOUT
025666 104405 TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN
157 025670 104401 057711 TYPE ,PERIOD ;TYPE ' '
158 025674 104401 025702 TYPE 81$ ;;TYPE ASCIZ STRING
025700 000404 BR 80$ ;;GET OVER THE ASCIZ

```

```
      025712  
159 025712 016046 000112  
      025716 104405  
160 025720 104401 057711  
161 025724 104401 025732  
      025730 000404  
  
      025742  
162 025742 016046 000102  
165 025746 166016 000104  
      025752 166016 000106  
      025756 166016 000110  
      025762 166016 000112  
166 025766 104405  
167 025770 104401 057711  
168 025774 104401 001203  
169 026000 000207  
  
      81$: .ASCIZ / MISP /  
80$:  
      MOV $MISPO(R0),-(SP) ;:SAVE $MISPO(R0) FOR TYPEOUT  
      TYPDS ;:GO TYPE--DECIMAL ASCII WITH SIGN  
      TYPE ,PERIOD ;:TYPE '.'  
      TYPE 83$ ;:TYPE ASCII STRING  
      BR 82$ ;:GET OVER THE ASCIIZ  
  
      83$: .ASCIZ / OTHER /  
82$:  
      MOV $TOTAL(R0),-(SP) ;:CALCULATE NUMBER OF OTHER ERRORS  
      SUB $SOFT(R0),(SP) ;:SUBTRACT $SOFT FROM $TOTAL  
      SUB $SHARD(R0),(SP) ;:SUBTRACT $SHARD FROM $TOTAL  
      SUB $SKI(R0),(SP) ;:SUBTRACT $SKI FROM $TOTAL  
      SUB $MISPO(R0),(SP) ;:SUBTRACT $MISPO FROM $TOTAL  
      TYPDS ;:GO TYPE--DECIMAL ASCII WITH SIGN  
      TYPE ,PERIOD ;:TYPE '.'  
      TYPE ,$CRLF ;:CR-LF  
      RTS PC
```


1
15
16

:ROUTINE TO INCREMENT \$SOFT
:NOTE: \$SOFT WILL NOT BE INCREMENTED BEYOND 77777 (32767.)

026002	005737	001336		INCSOF: TST	BADSEC	:SEE IF BAD TRK/SEC INDICATOR SET
026006	001006			BNE	1\$:BR IF IT'S SET, DON'T INCREMENT COUNT
026010	026027	000104	077777	CMP	\$SOFT(RO),#77777	:IS \$SOFT ALREADY AT MAXIMUM?
026016	103002			BHIS	1\$:BR IF IT IS
026020	005260	000104		INC	\$SOFT(RO)	:INCREMENT \$SOFT
026024	000207			1\$: RTS	PC	:RETURN

:ROUTINE TO INCREMENT \$SHARD
:NOTE: \$SHARD WILL NOT BE INCREMENTED BEYOND 77777 (32767.)

026026	005737	001336		INCHRD: TST	BADSEC	:SEE IF BAD TRK/SEC INDICATOR SET
026032	001006			BNE	1\$:BR IF IT'S SET, DON'T INCREMENT COUNT
026034	026027	000106	077777	CMP	\$SHARD(RO),#77777	:IS \$SHARD ALREADY AT MAXIMUM?
026042	103002			BHIS	1\$:BR IF IT IS
026044	005260	000106		INC	\$SHARD(RO)	:INCREMENT \$SHARD
026050	000207			1\$: RTS	PC	:RETURN

18

:ROUTINE TO INCREMENT \$SKI
:NOTE: \$SKI WILL NOT BE INCREMENTED BEYOND 77777 (32767.)

026052	005737	001336		INCSKI: TST	BADSEC	:SEE IF BAD TRK/SEC INDICATOR SET
026056	001006			BNE	1\$:BR IF IT'S SET, DON'T INCREMENT COUNT
026060	026027	000110	077777	CMP	\$SKI(RO),#77777	:IS \$SKI ALREADY AT MAXIMUM?
026066	103002			BHIS	1\$:BR IF IT IS
026070	005260	000110		INC	\$SKI(RO)	:INCREMENT \$SKI
026074	000207			1\$: RTS	PC	:RETURN

19

:ROUTINE TO INCREMENT \$MISPO
:NOTE: \$MISPO WILL NOT BE INCREMENTED BEYOND 77777 (32767.)

026076	005737	001336		INCMIS: TST	BADSEC	:SEE IF BAD TRK/SEC INDICATOR SET
026102	001006			BNE	1\$:BR IF IT'S SET, DON'T INCREMENT COUNT
026104	026027	000112	077777	CMP	\$MISPO(RO),#77777	:IS \$MISPO ALREADY AT MAXIMUM?
026112	103002			BHIS	1\$:BR IF IT IS
026114	005260	000112		INC	\$MISPO(RO)	:INCREMENT \$MISPO
026120	000207			1\$: RTS	PC	:RETURN

20

:ROUTINE TO INCREMENT \$TOTAL
:NOTE: \$TOTAL WILL NOT BE INCREMENTED BEYOND 77777 (32767.)

026122	005737	001336		INCTOT: TST	BADSEC	:SEE IF BAD TRK/SEC INDICATOR SET
026126	001006			BNE	1\$:BR IF IT'S SET, DON'T INCREMENT COUNT
026130	026027	000102	077777	CMP	\$TOTAL(RO),#77777	:IS \$TOTAL ALREADY AT MAXIMUM?
026136	103002			BHIS	1\$:BR IF IT IS
026140	005260	000102		INC	\$TOTAL(RO)	:INCREMENT \$TOTAL
026144	000207			1\$: RTS	PC	:RETURN

```

1
2
3      ;ROUTINE TO TYPE THE ELAPSED CPU RUN TIME
4 026146 005737 001310      $TIME:  TST      CLKFLG      ;IS CLOCK AVAILABLE ?
5 026152 001446              BEQ      3$          ;BR IF NO
6 026154 012737 000002 026222  MOV     #2,2$      ;ASSUME 2 DIGITS TO TYPE
7 026162 013746 001340      MOV     HOUR,-(SP)  ;PUT 'HOURS' ON THE STACK
8 026166 021627 000144      CMP     (SP),#100.  ;100. HOURS OR MORE ?
9 026172 002407              BLT     1$          ;BR IF NO
10 026174 005237 026222     INC     2$          ;TYPE 3 DIGITS
11 026200 021627 001750     CMP     (SP),#1000. ;1000. HOURS OR MORE ?
12 026204 002402              BLT     1$          ;BR IF NO
13 026206 005237 026222     INC     2$          ;TYPE 4 DIGITS
14 026212 004737 033332     1$:   JSR     PC,$SB2D ;CONVERT TO DECIMAL
15 026216 004537 032546     JSR     R5,FILLZ   ;TYPE IT
16 026222 000000     2$:   .WORD  0      ;NUMBER OF HOUR DIGITS TO TYPE
17 026224 104401 060074     TYPE   ,COLON     ;': '
18 026230 013746 001342     MOV     MINUTE,-(SP);PUT 'MINUTES' ON THE STACK
19 026234 004737 033332     JSR     PC,$SB2D  ;CONVERT TO DECIMAL
20 026240 004537 032546     JSR     R5,FILLZ   ;TYPE IT
21 026244 000002     .WORD  2          ;TYPE 2 DIGITS
22 026246 104401 060074     TYPE   ,COLON     ;': '
23 026252 013746 001344     MOV     SECOND,-(SP);PUT SECONDS ON THE STACK
24 026256 004737 033332     JSR     PC,$SB2D  ;CONVERT TO DECIMAL
25 026262 004537 032546     JSR     R5,FILLZ   ;TYPE IT
26 026266 000002     .WORD  2          ;TYPE 2 DIGITS
27 026270 000207     3$:   RTS      PC
28
29      ;KW11 CLOCK INTERRUPT SERVICE ROUTINE
30
31 026272 005337 001346     KWSVR: DEC     ONESEC ;INCREMENT THE 1/60 SECOND COUNTER
32 026276 001037              BNE     1$          ;BR IF A SECOND NOT COUNTED
33 026300 013737 001312 001346  MOV     HERTZ,ONESEC ;RESTORE THE VALUE
34 026306 005237 001344      INC     SECOND     ;COUNT THE SECOND
35 026312 023727 001344 000074  CMP     SECOND,#60. ;AT MAXIMUM ?
36 026320 103426              BLO     1$          ;BR IF NOT
37 026322 005037 001344      CLR     SECOND     ;CLEAR THE SECOND'S COUNTER
38 026326 005237 001472      INC     INTRVL+2   ;COUNT SUMMARY INTERVAL COUNTER
39 026332 005237 001464      INC     CMPTIM+2   ;COUNT COMPARE TIME INTERVAL COUNTER
40 026336 005237 001342      INC     MINUTE     ;COUNT THE MINUTE
41 026342 023727 001342 000074  CMP     MINUTE,#60. ;AT MAXIMUM ?
42 026350 103412              BLO     1$          ;BR IF NOT
43 026352 005037 001342      CLR     MINUTE     ;CLEAR THE MINUTE'S COUNTER
44 026356 005237 001340      INC     HOUR       ;COUNT THE HOURS
45 026362 023727 001340 023417  CMP     HOUR,#9999. ;AT MAXIMUM
46 026370 101402              BLOS   1$          ;BR IF NOT
47 026372 005037 001340      CLR     HOUR       ;CLEAR THE HOURS
48 026376 012746 000024     1$:   MOV     #20,-(SP)  ;20MS ON THE STACK @ 50HZ
49 026402 023727 001312 000062  CMP     HERTZ,#50. ;CPU RUNNING @ 50HZ ?
50 026410 001402              BEQ     2$          ;BR IF YES
51 026412 012716 000020     MOV     #16,-(SP)  ;16MS ON THE STACK @ 60HZ
52 026416 004737 044322     2$:   JSR     PC,RPTMR  ;DRIVER TIMER ROUTINE
53 026422 005737 001470      TST     INTRVL     ;DISPLAY THE PERFORMANCE SUMMARY ?
54 026426 001411              BEQ     3$          ;BR IF NOT
55 026430 023737 001472 001470  CMP     INTRVL+2,INTRVL ;DISPLAY INTERVAL FINISHED ?
56 026436 002405              BLT     3$          ;BR IF NO
57 026440 012737 177777 001314  MOV     #-1,STATIN ;SET PERFORMANCE SUMMARY DISPLAY FLAG

```

58 026446 005037 001472
59 026452 000002

3S:

CLR
RTI

INTRVL+2

;CLEAR THE PERFORMANCE INTERVAL COUNTER

```

1
2
3
4
5
6
7
8
9
10 026454 005737 040546
11 026460 100375
12 026462 104412
13 026464 012737 000200 177776
14 026472 013704 040560
15 026476 012764 000040 000010
19 026504 005037 001334
20 026510 104401 001203
21 026514 004737 026146
22 026520 104401 056573
23 026524 104401 056531
24 026530 104401 034656
25 026534 017746 152414

    026540 104402
26 026542 004737 033430
27 026546 004737 024710
28 026552 104401 060001
29
30 026556 104411
31 026560 012605
32 026562 005737 001334
33 026566 001405
34 026570 005737 001542
35 026574 001141
36 026576 000137 003522
37
38 026602 004737 024514
39 026606 005205
40 026610 122715 000124
41 026614 001465
42 026616 122715 000101
43 026622 001410
44 026624 121527 000067
45 026630 101117
46 026632 121527 000060
47 026636 103514
48 026640 142715 177770
49 026644 122765 000124 177777
50 026652 001003
51 026654 004737 030030
52 026660 000507
53 026662 122765 000104 177777
54 026670 001003
55 026672 004737 027620
56 026676 000500
57 026700 122765 000123 177777
58 026706 001003

;COMMAND DECODE ROUTINE
;CALL:
;      MOV      #1,CFLAG      ;'CFLAG' IS NORMALLY SET BY THE TTY SERVICE
;                               ;ROUTINE IN INTERRUPT MODE
;      JSR      PC,KSR
;      RETURN1
;      RETURN2      ;SYSTEM BUSY RETURN
;                               ;RETURN AFTER KEYBOARD SERVICED
KSR:  TST      DTUW      ;ANY DATA TRANSFERS UNDER WAY ?
      BPL      KSR      ;BR IF YES
KSR1: SAVREG
      MOV      #PR4,PS    ;SAVE THE REGISTERS
      MOV      RPADR,R4   ;SET PRIORITY TO 4
1$:   MOV      #CLR,RPCS2(R4) ;GET RP/RH BASE ADDRESS
      CLR      CFLAG     ;CLEAR MASSBUS CONTROLLER
      TYPE    $,CRLF     ;CLEAR THE 'CONTROL C' FLAG
      JSR      PC,$TIME  ;CR-LF
      TYPE    $,COMMA   ;TYPE ELAPSED TIME
      TYPE    $,BLNKS1  ;TYPE ','
      TYPE    $,$MSWR+2 ;TYPE 'BLANK'
      MOV      @SWR,-(SP) ;TYPE 'SWR ='
;                               ;SAVE @SWR FOR TYPEOUT
;                               ;CONTENTS OF SWITCH REGISTER
;                               ;GO TYPE--OCTAL ASCII(ALL DIGITS)
      JSR      PC,$TKINT ;INITIALIZE TTY KEYBOARD
      JSR      PC,CLKOFF ;SHUT OFF CLOCK INTERRUPT WHILE WAITING
      TYPE    $,ENTCOM  ;'ENTER COMMAND'
RDLIN
30:  MOV      (SP)+,R5    ;READ THE KEYBOARD
      TST      CFLAG     ;GET ADDRESS OF INPUT STRING
      BEQ     2$        ;WAS (^C) TYPED?
      TST      ASNLST   ;BR IF NO
      BNE     13$      ;ANY DRIVES ASSIGNED ?
      JMP     START    ;BR IF YES
;                               ;JUMP TO START
2$:  JSR      PC,CKCLK   ;START SYSTEM CLOCK
      INC     R5        ;POINT TO SECOND CHARACTER
      CMPB   #'T',(R5) ;EQ TO A 'T' ?
      BEQ     9$        ;YES
      CMPB   #'A',(R5) ;EQ TO AN 'A'
      BEQ     3$        ;BR IF IT IS
      CMPB   (R5),#'7  ;DRIVE NUMBER GREATER THAN AN ASCII 7 ?
      BHI     12$      ;BR IF IT IS
      CMPB   (R5),#'0  ;DRIVE NUMBER LESS THAN AN ASCII 0 ?
      BLO     12$      ;BR IF IT IS
      BICB  #'C7,(R5)  ;LEAVE ONLY LOWER 3 BITS IF CHAR NOT 'A'
      CMPB   #'T,-1(R5) ;EQ TO 'T'
      BNE     4$        ;BR IF NOT EQ
      JSR      PC,NEWASN ;ASSIGN DRIVE FOR TEST
      BR     13$      ;EXIT
      CMPB   #'D,-1(R5) ;EQ TO 'D' ?
      BNE     5$        ;BR IF NOT EQ
      JSR      PC,DROPD ;DROCP DRIVE
      BR     13$      ;EXIT
      CMPB   #'S,-1(R5) ;EQ TO 'S'
      BNE     6$        ;BR IF NOT EQ

```

59	026710	004737	027726		JSR	PC,SCMND		:TYPE STATISTICS
60	026714	000471			BR	13\$:EXIT
61	026716	122765	000127	177777	6\$:	CMPB	#'W,-1(R5)	:EQ TO 'W'
62	026724	001012			BNE	8\$:BR IF NOT EQ
64	026726	005737	001424		TST	RONLY		:LOCKED IN 'READ ONLY' MODE ?
65	026732	001053			BNE	11\$:BR IF YES
67	026734	032777	000001	152212		BIT	#SW0,@SWR	:IS SWITCH 0 SET ?
68	026742	001047			BNE	11\$:BR IF SET, CAN'T DO 'W' COMMAND
69	026744	004737	030052		7\$:	JSR	PC,DATAPK	:WRITE DATA
70	026750	000453			BR	13\$:EXIT
71	026752	122765	000122	177777	8\$:	CMPB	#'R,-1(R5)	:EQ TO 'R' ?
72	026760	001043			BNE	12\$:BR IF NOT EQ
73	026762	004737	030040		JSR	PC,DEDAPK		:READ DATA
74	026766	000444			BR	13\$:EXIT
75	026770	122765	000127	177777	9\$:	CMPB	#'W,-1(R5)	:WT COMMAND ?
76	026776	001034			BNE	12\$:NO
78	027000	005737	001424		TST	RONLY		:LOCKED IN 'READ ONLY' MODE ?
79	027004	001026			BNE	11\$:BR IF YES
81	027006	032777	000001	152140		BIT	#SW0,@SWR	:IS SWITCH 0 SET ?
82	027014	001022			BNE	11\$:BR IF SET, CAN'T DO 'W' COMMAND
83	027016	122765	000101	000001		CMPB	#'A,' 5)	:ALL DRIVES ?
84	027024	001413			BEQ	10\$:YES
85	027026	126527	000001	000067		CMPB	1(R5),#'7	:GREAT THAN 7
86	027034	101015			BHI	12\$:YES
87	027036	126527	000001	000060		CMPB	1(R5),#'0	:LESS THAN 0
88	027044	103411			BLO	12\$:YES
89	027046	142765	177770	000001		BICB	#^C7,1(R5)	:CHOP OFF THE HIGHER BITS
90	027054	004737	030064		10\$:	JSR	PC,WATPAK	:ASSIGN DRIVES WITH WT COMMAND
91	027060	000407			BR	13\$		
92	027062	104401	057713		11\$:	TYPE	,MSWRO	:TYPE 'CAN'T WRITE IN READ ONLY MODE'
93	027066	000601			BR	1\$:TRY AGAIN
94	027070	104401	057756		12\$:	TYPE	,INVLD	:TYPE 'INVALID COMMAND' MESSAGE
95	027074	000137	026472			JMP	1\$:TRY AGAIN
96	027100	104413			13\$:	RESREG		:RESTORE R0 - R5
97	027102	005777	152054			TST	@\$TKB	:CLEAR THE TTY BUFFER
98	027106	052777	000100	152044		BIS	#BIT06,@\$TKS	:SET TTY INTERRUPT ENABLE
99	027114	005037	177776			CLR	PS	:SET PRIORITY BACK TO ZERO
100	027120	000207				RTS	PC	:RETURN
101								
102								:ROUTINE TO PROCESS THE ASSIGN REQUEST ('T', 'R', OR 'W' COMMANDS)
103								
104	027122	111504			ASSIGN:	MOVB	(R5),R4	:PUT DRIVE # IN R4
105	027124	005037	001334		1\$:	CLR	CFLAG	:CLEAR CONTROL C FLAG
106	027130	005037	001426			CLR	DRVPAR	:ASSUME CHANGING DRIVE PARAMETERS
107	027134	104401	060164			TYPE	,MSPRM	:TYPE 'CHANGE DRIVE PARAMETERS ?'
108	027140	104411				RDLIN		:READ THE ENTRY
109	027142	012600				MOV	(SP)+,R0	:SAVE ADDRESS OF RESPONSE
110	027144	005737	001334			TST	CFLAG	:WAS (^C) TYPED?
111	027150	001365				BNE	1\$:BR IF YES
112	027152	105710				TSTB	(R0)	:WAS RESPONSE A CARRIAGE RETURN (DEFAULT 'N')?
113	027154	001414				BEQ	3\$:BR IF YES
114	027156	105760	000001			TSTB	1(R0)	:WAS IT TERMINATED WITH CARRIAGE RETURN ?
115	027162	001006				BNE	2\$:BR IF NO
116	027164	122710	000131			CMPB	#'Y,(R0)	:WAS IT A 'Y' RESPONSE ?
117	027170	001410				BEQ	4\$:BR IF YES
118	027172	122710	000116			CMPB	#'N,(R0)	:WAS IT A 'N' RESPONSE ?
119	027176	001403				BEQ	3\$:BR IF YES

```

120 027200 104401 060103      2$:  TYP:  ,BADENT      ;TYPE BAD ENTRY MESSAGE
121 027204 000747              BR      1$          ;TRY AGAIN
122 027206 005237 001426      3$:  INC  DRVPAR      ;DO NOT CHANGE DRIVE PARAMETERS
123 027212 122704 000101      4$:  CMF3 #'A,R4      ;ASSIGN ALL DRIVES ?
124 027216 001426              BEQ    ASGN2        ;BR IF YES
125
126 027220 012737 056652 031276 ASGN1: MOV    #UNTASN,ASNMSG ;ERROR MESSAGE
127 027226 005737 001430      TST    XXDP        ;LOADED FROM THIS DEVICE ?
128 027232 001407              BEQ    1$          ;BR IF NO
129 027234 123704 001430      CMPB  XXDP,R4      ;LOADED FROM THIS DRIVE ?
130 027240 001004              BNE   1$          ;BR IF NO
131 027242 012737 056761 031276 MOV    #LODEV,ASNMSG ;'LOAD DEVICE' MESSAGE ADDRESS
132 027250 000407              BR     2$
133 027252 136437 040550 001542 1$:  BITB  ATABIT(R4),ASNLST ;DRIVE ALREADY ASSIGNED ?
134 027260 001003              BNE   2$          ;BR IF IT IS
135 027262 004737 027364      JSR   PC,ASGN3     ;SEE IF DRIVE ON THE SYSTEM
136 027266 000207              RTS   PC           ;RETURN
137 027270 000137 031252      JMP   ASNERR       ;EXIT ERROR
138
139 027274 005004              ASGN2: CLR   R4      ;START WITH DRIVE 0
140 027276 012737 056652 031276 1$:  MOV    #UNTASN,ASNMSG ;ERROR MESSAGE
141 027304 005737 001430      TST    XXDP        ;LOADED FROM THIS DEVICE ?
142 027310 001407              BEQ    2$          ;BR IF NO
143 027312 123704 001430      CMPB  XXDP,R4      ;LOADED FROM THIS DRIVE ?
144 027316 001004              BNE   2$          ;BR IF NO
145 027320 012737 056761 031276 MOV    #LODEV,ASNMSG ;'LOAD DEVICE' MESSAGE ADDRESS
146 027326 000413              BR     4$
147 027330 136437 040550 001542 2$:  BITB  ATABIT(R4),ASNLST ;ALREADY ASSIGNED ?
148 027336 001007              BNE   4$          ;YES
149 027340 004737 027364      JSR   PC,ASGN3     ;ASSIGN THE DRIVE
150 027344 005204              3$:  INC  R4          ;INCREMENT DRIVE #
151 027346 020427 000007      CMP   R4,#1        ;ALL DRIVE CHECKED ?
152 027352 003751              BLE   1$          ;NO
153 027354 000207              RTS   PC           ;YES
154 027356 004737 031252      JSR   PC,ASNERR    ;ERROR MESSAGE
155 027362 000770              BR     3$          ;TO LOOP
156
157 027364 136437 040550 001542 ASGN3: BITB  ATABIT(R4),ASNLST ;DRIVE ALREADY ASSIGNED ?
158 027372 001056              BNE   ASGN4        ;BR IF IT IS
159 027374 110437 050654      MOVB  R4,GENDPB     ;GET DRIVE NUMBER
160 027400 006304              ASL   R4           ;MAKE R4 WORD INDEX
161 027402 016400 002056      MOV   BLKADR(R4),R0 ;PUT BLOCK'S ADDR INTO R0
162 027406 004737 016720      JSR   PC,RECALO    ;RECALIBRATE DRIVE
163 027412 006204              ASR   R4           ;MAKE R4 BYTE INDEX
164 027414 105764 040454      TSTB  DRVSTA(R4)   ;DRIVE AVAILABLE?
165 027420 001451              BEQ   ASGN7        ;BR IF DRIVE OFFLINE OR NONEXISTENT
166 027422 100443              BMI   ASGN6        ;BR IF DRIVE UNSAFE
167 027424 004737 030104      JSR   PC,CLRDPB    ;CLEAR BLOCK FOR DRIVE JUST ASSIGNED
168 027430 004737 031010      JSR   PC,GETID     ;GET DRIVE SERIAL NUMBER
169 027434 032760 000004 000200 BIT   #ILV,$RPDS(R0) ;INTERLEAVE SECTOR SET ?
170 027442 001461              BEQ   ASGN8        ;BR IF NO
171 027444 005737 001426      TST   DRVPAR       ;CHANGE DRIVE PARAMETERS ?
172 027450 001015              BNE   1$          ;BR IF NO
173 027452 104401 001203      TYPE  ,%CRLF       ;CR-LF
174 027456 104401 056601      TYPE  ,DRVMSG      ;TYPE 'DRIVE'
175 027462 010446              MOV   R4,-(SP)     ;SAVE R4 FOR TYPEOUT
                        ;TYPE DRIVE NUMBER

```



```

1
2
3      ;'D' COMMAND (ROUTINE TO DROP A DRIVE)
4 027620 005004      DROPD: CLR      R4          ;START WITH DRIVE 0
5 027622 012703 000010      MOV      #8,R3        ;COUNTER
6 027626 122715 000101      CMPB    #'A,(R5)      ;DROP ALL DRIVES ?
7 027632 C01403      BEQ      1$          ;BR IF YES
8 027634 111504      MOVB    (R5),R4       ;GET DRIVE NUMBER
9 027636 012703 000001      MOV      #1,R3        ;SET R3 FOR ONE DRIVE
10 027642 136437 040550 001542 1$: BITB    ATABIT(R4),ASNLS  ;DRIVE ASSIGNED ?
11 027650 001417      BEQ      3$          ;BR IF NOT
12 027652 146437 040550 001542  BICB    ATABIT(R4),ASNLS  ;DELETE THE DRIVE FROM THE ASSIGN.D LIST
13 027660 146437 040550 032020  BICB    ATABIT(R4),AUTLS  ;DELETE DRIVE FROM AUTO ASSIGN LIST
14 027666 006304      ASL      R4          ;MAKE ADDR INTO A WORD INDEX
15 027670 016464 002056 001544  MOV     BLKADR(R4),DDRVS(R4) ;PUT ADDRESS IN DROP LIST
16 027676 006204      ASR      R4
17 027700 005303      2$: DEC      R3          ;ANY MORE DRIVES ?
18 027702 001410      BEQ      4$          ;BR IF NOT
19 027704 005204      INC      R4
20 027706 000755      BR      1$
21 027710 012737 056630 031276 3$: MOV     #UNTNOT,ASNMSG ;ADDR OF 'NOT ASSIGNED' MESSAGE
22 027716 004737 031252      JSR     PC,ASNERR    ;REPORT IT
23 027722 000766      BR      2$
24 027724 000207      4$: RTS      PC
25
26      ;'S' COMMAND (ROUTINE TO TYPE DRIVE PERFORMANCE SUMMARY)
27
28      SCMND:
29 027726 013746 001542      MOV     ASNLS,-(SP)   ;:PUSH ASNLS ON STACK
30 027732 122715 000101      CMPB    #'A,(R5)      ;ALL STATISTICS ?
31 027736 001416      BEQ      2$          ;BR IF YES
32 027740 111504      MOVB    (R5),R4       ;GET DRIVE NUMBER
33 027742 136414 040550      BITB    ATABIT(R4),(SP) ;IS THIS DRIVE ASSIGNED ?
34 027746 001404      BEQ      1$          ;BR IF NO
35 027750 116437 040550 001542  MOVB    ATABIT(R4),ASNLS ;GET DRIVE ASSIGN BIT
36 027756 000411      BR      3$
37 027760 012737 056630 031276 1$: MOV     #UNTNOT,ASNMSG ;ADDR OF 'NOT ASSIGNED' MSG
38 027766 004737 031252      JSR     PC,ASNERR    ;TYPE ERROR MESSAGE
39 027772 000413      BR      4$          ;EXIT
40 027774 105737 001542      2$: TSTB   ASNLS      ;ANY DRIVE ASSIGNED ?
41 030000 001410      BEQ      4$          ;BR IF NO
42 030002 004737 024740      3$: JSR     PC,STATPR   ;TYPE ALL STATISTICS
43 030006 104401 057327      TYPE    ,DASH13      ;TYPE '-----'
44 030012 104401 001203      TYPE    ,$CRLF       ;CR-LF
45 030016 104401 057513      TYPE    ,MSGCON      ;TYPE 'CONTINUING...'
46
47 030022 012637 001542      4$: MOV     (SP)+,ASNLS ;:POP STACK INTO ASNLS
48 030026 000207      RTS      PC
49
50      ;'T' COMMAND (ROUTINE TO TEST A DRIVE)
51
52 030030 005037 030102      NEWASN: CLR     PACK   ;SET 'T' COMMAND INDICATOR
53 030034 000137 027122      JMP     ASSIGN      ;GO TO THE ASSIGN ROUTINE
54
55      ;'R' COMMAND (ROUTINE TO DO SEQUENTIAL READ DATA)

```



```
56  
57 030040 012737 000001 030102 REDAPK: MOV #1,PACK ;SET 'R' COMMAND INDICATOR  
58 030046 000137 027122 JMP ASSIGN ;ASSIGN THE REQUESTED DRIVE  
59  
60 ;'W' COMMAND (ROUTINE TO DO SEQUENTIAL WRITE DATA)  
61  
62 030052 012737 177777 030102 DAT.PK: MOV #-1,PACK ;SET 'W' COMMAND INDICATOR  
63 030060 000137 027122 JMP ASSIGN ;ASSIGN REQUESTED DRIVE  
64  
65 ;'WT' COMMAND (ROUTINE TO DO WRITE DATA AND TEST A DRIVE)  
66  
67 030064 116515 000001 WATPAK: MOVB 1(R5),(R5) ;ADJUST DRIVE NUMBER ADDRESS  
68 030070 012737 177776 030102 MOV #-2,PACK ;SET 'WT' COMMAND INDICATOR  
69 030076 000137 027122 JMP ASSIGN ;JUMP TO ASSIGN ROUTINE  
70  
71 030102 000000 PACK: .WORD 0 ;TEMPORARY STORAGE FOR COMMAND INDICATOR
```

```

1
2
3
4
5
6
7
8
9 030104
030104 010146
030106 010346
030110 010446
030112 005446
10 030114 005737 040020
11 030120 001073
12 030122 010004
13 030124 062704 0000C2
14 030130 012703 000012
15 030134 005024
16 030136 162703 000002
17 030142 001374
18
19 030144 062704 000002
20 030150 012703 000114
21 030154 005024
22 030156 162703 000002
23 030162 001374
24
25 030164 062704 000026
26
27 030170 012703 000062
28 030174 005024
29 030176 162703 000002
30 030202 001374
31
32
33 030204 113760 001514 000024
34 030212 013701 001514
35 030216 116160 002076 000002
36 030220 113760 001512 000030
37 030232 106360 000030
38 030236 013760 001516 000020
39 030244 013760 001516 000004
40 030252 005460 000004
41 030256 012760 000400 000022
42 030264 012760 000001 000114
43 030272 132760 000001 000024
44 030300 001403
45 030302 062760 000002 000022
46 030310
030310 012605
030312 012604
030314 012603
030316 012601
47 030320 000207
48
49

```

:ROUTINE TO CLEAR THE DPB FOR THE ASSIGNED DRIVE

:CALL:

```

MOV #DPB,R0 ;DPB ADDRESS
JSR PC,CLRDPB
RETURN

```

:R0 = DPB ADDRESS BEFORE CALLING THE ROUTINE

CLRDPB:

```

MOV R1,-(SP) ;;PUSH R1 ON STACK
MOV R3,-(SP) ;;PUSH R3 ON STACK
MOV R4,-(SP) ;;PUSH R4 ON STACK
MOV R5,-(SP) ;;PUSH R5 ON STACK
TST PWRFLG ;RETURNING FROM POWER FAIL ?
BNE 4$ ;BRANCH IF YES
MOV R0,R4 ;GET THE DPB ADDRESS
ADD #2,R4 ;ADDRESS OF FIRST LOCN TO BE CLEARED
MOV #<$CYL-$COMND>+2,R3 ;NUMBER OF LOCNS TO BE CLEARED
1$: CLR (R4)+ ;CLEAR LOCATIONS '$COMND' - '$CYL' IN DPB
SUB #2,R3 ;DONE CLEARING YET ?
BNE 1$ ;BR IF NO

```

```

ADD #2,R4 ;SKIP OVER THE '$REG' LOCATION
MOV #<$NEXT-$STATUS>+2,R3 ;NUMBER OF LOCNS TO BE CLEARED
2$: CLR (R4)+ ;CLEAR LOCATIONS '$STATUS' - '$NEXT' IN DPB
SUB #2,R3 ;DONE CLEARING YET ?
BNE 2$ ;BR IF NO

```

```

ADD #<$DRVSN-$FIRST>,R4 ;SKIP OVER '$FIRST', MIN/MAX ADRS
;LIMITS AND 'CYL, TRK, SEC, FE1' LIMITS
MOV #<$RPCS3-$DRVSN>+2,R3 ;NUMBER OF LOCNS TO BE CLEARED
3$: CLR (R4)+ ;CLEAR LOCATIONS '$DRVSN' - '$RPCS3' IN DPB
SUB #2,R3 ;DONE CLEARING YET ?
BNE 3$ ;BR IF NO

```

```

;INITIALIZE SOME OTHER LOCATIONS
MOV B, BEGCODE($CODE(R0)) ;INITIAL COMMAND CODE
MOV R1, BEGCODE ;GET THE ACTUAL OP CODE
MOV B, COMTBL(R1), $COMND(R0) ;OPERATION CODE
MOV B, BEGPAT, $PATTC(R0) ;PATTERN CODE
ASLB $PATTC(R0) ;CONVERT CODE TO A TABLE INDEX
MOV B, BEGWC, $WRDL(R0) ;BEGINNING WORD COUNT
MOV B, BEGWC, $WCNT(R0) ;VALUE FOR DATA TRANSFER
NEG $WCNT(R0) ;MAKE IT INTO 2'S COMPLEMENT
MOV #256, $SSEC(R0) ;INITIAL VALUE OF SECTOR SIZE
MOV #1, $PASSC(R0) ;PRESET PASS COUNT TO 1
BITB #1, $CODE(R0) ;HEADER COMMAND ?
BEQ 4$ ;BR IF NOT
ADD #2, $SSEC(R0) ;ADD HEADER SIZE TO SECTOR SIZE
4$:

```

```

MOV (SP)+,R5 ;;POP STACK INTO R5
MOV (SP)+,R4 ;;POP STACK INTO R4
MOV (SP)+,R3 ;;POP STACK INTO R3
MOV (SP)+,R1 ;;POP STACK INTO R1
RTS PC ;TURN

```

:ROUTINE TO GET ADDRESS LIMITS FROM THE OPERATOR

```

50
51      :CALL:
52      :      MOV      #DPB,RO      :DPB ADDRESS
53      :      JSR      PC,DRVPRM    :CALL ROUTINE
54      :RO = DPB ADDRESS BEFORE CALLING THE ROUTINE
55
56 030322 010346  DRVPRM: MOV      R3,-(SP)      :SAVE R3
57 030324 010446  MOV      R4,-(SP)      :SAVE R4
58 030326 04737 030706 1$: JSR      PC,GETLMT    :GET ADDRESS LIMITS
59 030332 062760 177777 000132  ADD      #-1,$FIRST(RO) :SEE IF FIRST TIME STARTED
60 030340 103426  BCS      4$           :BR IF NOT
61 030342 016060 000150 000134  MOV      CYLLMT(RO),MAXCYL(RO) :LOAD MAXIMUM CYLINDER
62 030350 016060 000154 000140  MOV      TRKLMT(RO),MAXTRK(RO) :LOAD MAXIMUM TRACK
63 030356 016060 000152 000144  MOV      SECLMT(RO),MAXSEC(RO) :LOAD MAXIMUM SECTOR
64 030364 005737 001422  TST      TSTANY      :ARE YOU TESTING ANYWHERE ON MEDIA ?
65 030370 001004  BNE      2$           :BR IF YES
66 030372 016060 000156 000136  MOV      FE1(RO),MINCYL(RO) :RESET MINIMUM CYLINDER ADDRESS
67 030400 000402  BR       3$
68 030402 005060 000136 2$: CLR      MINCYL(RO)    :CLEAR MINIMUM CYLINDER
69 030406 005060 000142 3$: CLR      MINTRK(RO)   :CLEAR MINIMUM TRACK
70 030412 005060 000146  CLR      MINSEC(RO)    :CLEAR MINIMUM SECTOR
71
72
73
74
75
76
77
78 030416 105737 001150 4$: TSTB     $AUTOB      :RUNNING IN AUTO MODE ?
79 030422 001074  BNE      7$           :BR IF YES
80 030424 005737 001332  TST      CHGADR      :PROGRAM STARTED AT 200 ?
81 030430 003071  BGT      7$           :BR IF YES
82 030432 005737 001426  TST      DRVPAR      :CHANGE DRIVE PARAMETERS ?
83 030436 001066  BNE      7$           :BR IF NO
84 030440 016403 061604  MOV      TABLE(R4),R3 :PARAMETER TABLE ADDRESS
85 030444 016063 000150 000002  MOV      CYLLMT(RO),2(R3) :LOAD CYLINDER LIMIT FOR MINCYL
86 030452 016063 000150 000010  MOV      CYLLMT(RO),10(R3) :LOAD CYLINDER LIMIT FOR MAXCYL
87 030460 016063 000154 000016  MOV      TRKLMT(RO),16(R3) :LOAD TRACK LIMIT FOR MINTRK
88 030466 016063 000154 000024  MOV      TRKLMT(RO),24(R3) :LOAD TRACK LIMIT FOR MAXTRK
89 030474 016063 000152 000032  MOV      SECLMT(RO),32(R3) :LOAD SECTOR LIMIT FOR MINSEC
90 030502 016063 000152 000040  MOV      SECLMT(RO),40(R3) :LOAD SECTOR LIMIT FOR MAXSEC
91 030510 104001 060022  TYPE     ,ENTLMT      :ADDRESS LIMITS:
92 030514 004737 031110  JSR      PC,PARENT    :GET THE DRIVE'S PARAMETERS
93
94 030520 016003 000136  MOV      MINCYL(RO),P3 :STORE MINCYL VALUE
95 030524 016004 000134  MOV      MAXCYL(RO),R4 :STORE MAXCYL VALUE
96 030530 020304  CMP      R3,R4        :IS MIN. LESS THAN OR EQUAL TO MAX. ?
97 030532 003404  BLE     5$           :BR IF YES
98 030534 010360 000134  MOV      R3,MAXCYL(RO) :SWAP MIN. TO MAX.
99 030540 010460 000136  MOV      R4,MINCYL(RO) :SWAP MAX. TO MIN.
100 030544 016003 000142 5$: MOV      MINTRK(RO),R3 :STORE MINTRK VALUE
101 030550 016004 000140  MOV      MAXTRK(RO),R4 :STORE MAXTRK VALUE
102 030554 020304  CMP      R3,R4        :IS MIN. LESS THAN OR EQUAL TO MAX. ?
103 030556 003404  BLE     6$           :BR IF YES
104 030560 010360 000140  MOV      R3,MAXTRK(RO) :SWAP MIN. TO MAX.
105 030564 010460 000142  MOV      R4,MINTRK(RO) :SWAP MAX. TO MIN.
106 030570 016003 000146 6$: MOV      MINSEC(RO),R3 :STORE MINSEC VALUE
107 030574 016004 000144  MOV      MAXSEC(RO),R4 :STORE MAXSEC VALUE
108 030600 020304  CMP      R3,R4        :IS MIN. LESS THAN OR EQUAL TO MAX. ?
109 030602 003404  BLE     7$           :BR IF YES
110 030604 010360 000144  MOV      R3,MAXSEC(RO) :SWAP MIN. TO MAX.
111 030610 010460 000146  MOV      R4,MINSEC(RO) :SWAP MAX. TO MIN.
112

```

```

114 030614 005737 001422      7$:  TST      TSTANY          ;ARE YOU TESTING ANYWHERE ON MEDIA ?
115 030620 001016              BNE      9$              ;BR IF YES
116 030622 026060 000136 000156    CMP      MINCYL(R0),FE1(R0) ;IS MIN. CYLINDER < 1ST FE CYLINDER ?
117 030630 103003              BHIS     8$              ;BR IF NO
118 030632 016060 000156 000136    MOV      FE1(R0),MINCYL(R0) ;RESET MINIMUM CYLINDER ADDRESS
119 030640 026060 000134 000156    8$:  CMP      MAXCYL(R0),FE1(R0) ;IS MAX. CYLINDER < 1ST FE CYLINDER ?
120 030646 103003              BHIS     9$              ;BR IF NO
121 030650 016060 000150 000134    MOV      CYLLMT(R0),MAXCYL(R0) ;RESET MAXIMUM CYLINDER ADDRESS
122
123 030656 016060 000136 000012    9$:  MOV      MINCYL(R0),$CYL(R0) ;INITIAL CYLINDER VALUE
127 030664 116060 000142 000011    MOVB     MINTRK(R0),$TRK(R0) ;INITIAL TRACK VALUE
128 030672 116060 000146 000010    MOVB     MINSEC(R0),$SEC(R0) ;INITIAL SECTOR VALUE
129 030700 012604              MOV      (SP)+,R4        ;;POP STACK INTO R4
      030702 012603              MOV      (SP)+,R3        ;;POP STACK INTO R3
130 030704 000207              RTS      PC              ;RETURN
131
132                               ;ROUTINE TO GET THE ADDRESS LIMITS FOR THE CURRENT DRIVE TYPE
133                               ;CALL:
134                               JSR      PC,GETLMT          ;CALL ROUTINE
135                               ;
136                               ;RO = DPB ADDRESS BEFORE CALLING THE ROUTINE
137
138 030706                               GETLMT:
      030706 010146              MOV      R1,-(SP)        ;;PUSH R1 ON STACK
139 030710 005001              CLR      R1              ;START FRESH
140 030712 012760 001057 000156    MOV      #559.,FE1(R0)   ;GET ADDRESS OF 1ST FE CYLINDER
141 030720 012760 001060 000150    MOV      #560.,CYLLMT(R0) ;ASSUME CYLINDER LIMIT FOR RPO7
142 030726 012760 000035 000154    MOV      #29.,TRKLMT(R0) ;ASSUME TRACK LIMIT ""
143 030734 012760 000041 000152    MOV      #33.,SECLMT(R0) ;ASSUME SECTOR LIMIT ""
144 030742 111001              MOVB     (R0),R1         ;GET DRIVE NUMBER
145 030744 122761 000005 040464    CMPB     #5,DRVTYP(R1)   ;IS DRIVE AN RPO7 ?
146 030752 001414              BEQ      1$              ;BR IF YES
147 030754 012760 001166 000156    MOV      #630.,FE1(R0)   ;GET ADDRESS OF 1ST FE CYLINDER
148 030762 012760 001166 000150    MOV      #630.,CYLMT(R0) ;GET CYLINDER LIMIT FOR RPO7+
149 030770 012760 000037 000154    MOV      #31.,TRKLMT(R0) ;GET TRACK LIMIT ""
150 030776 012760 000061 000152    MOV      #49.,SECLMT(R0) ;GET SECTOR LIMIT ""
151 031004              1$:  MOV      (SP)+,R1        ;;POP STACK INTO R1
152 031006 000207              RTS      PC              ;RETURN

```

```

1      ;ROUTINE TO GET THE DRIVE SERIAL NUMBER FROM RPSN REGISTER
2      ;THIS NUMBERS CONTAINED IN THE REGISTER ARE ONLY THE 4 LSD'S OF THE
3      ;SERIAL NUMBER.
4      ;CALL:
5      ;       MOV     #DPB,R0           ;DPB ADDRESS
6      ;       JSR     PC,GETID         ;CALL ROUTINE
7      ;
8      ;RO = DPB ADDRESS BEFORE CALLING THE ROUTINE
9
10     GETID:
11     031010 010046      MOV     R0,-(SP)           ;;PUSH R0 ON STACK
12     031012 010146      MOV     R1,-(SP)           ;;PUSH R1 ON STACK
13     031014 010246      MOV     R2,-(SP)           ;;PUSH R2 ON STACK
14     031016 010546      MOV     R5,-(SP)           ;;PUSH R5 ON STACK
15     031020 010002      MOV     R0,R2           ;GET INDEX TO DPB
16     031022 004737 045262 JSR     PC,SVRHXX        ;SAVE ALL REGISTERS
17     031026 012702 000004 MOV     #4,R2           ;FOUR DIGITS TO STORE
18     031032 016001 000216 MOV     $RPSN(R0),R1    ;SERIAL NUMBER
19     031036 005005      1$: CLR     R5           ;ZERO
20     031040 006101      ROL     R1           ;PUT THE NEXT DIGIT
21     031042 006105      ROL     R5           ;INTO R5
22     031044 006101      ROL     R1
23     031046 006105      ROL     R5
24     031050 006101      ROL     R1
25     031052 006105      ROL     R5
26     031054 006101      ROL     R1
27     031056 006105      ROL     R5
28     031060 062705 000060 ADD     #0,R5           ;MAKE IT ASCII
29     031064 110560 000160 MOV     R5,$DRVSN(R0)   ;SAVE DRIVE SERIAL NUMBER DIGIT
30     031070 005200      INC     R0           ;GET NEXT INDEX FOR DRIVE SERIAL NUMBER
31     031072 005302      DEC     R2           ;ALL DIGITS TYPED?
32     031074 003360      BGT     1$          ;NO -- BRANCH
33     031076 012605      MOV     (SP)+,R5      ;;POP STACK INTO R5
34     031100 012602      MOV     (SP)+,R2      ;;POP STACK INTO R2
35     031102 012601      MOV     (SP)+,R1      ;;POP STACK INTO R1
36     031104 012600      MOV     (SP)+,R0      ;;POP STACK INTO R0
37     031106 000207      RTS     PC           ;RETURN
  
```

1				.SBTTL	PARAMETER ENTRY ROUTINE	
2						
3				:	PARAMETER ENTRY ROUTINE	
4				:	CALL:	
5				:	MOV	#ADR,R3 ;PARAMETER TABLE ADDRESS
6				:	JSR	PC,PARENT ;GET THE PARAMETERS
7						
8	031110	010346		PARENT:	MOV	R3,-(SP) ;SAVE THE PARAMETER TABLE ADDRESS
9	031112	005037	001334		CLR	CFLAG ;CLEAR THE 'CONTROL C' FLAG
10	031116	012337	031126	1\$:	MOV	(R3)+,2\$;ADDRESS OF PARAMETER NAME
11	031122	001451			BEQ	7\$;BR IF AT END OF TABLE
12	031124	104401			TYPE	;TYPE THE PARAMETER NAME
13	031126	000000		2\$:	.WORD	0 ;ADDRESS OF PARAMETER NAME TEXT
14	031130	012302			MOV	(R3)+,R2 ;MAXIMUM PARAMETER VALUE
15	031132	012305			MOV	(R3)+,R5 ;ADDRESS OF PARAMETER
16	031134	011546			MOV	(R5),-(SP) ;CURRENT VALUE OF PARAMETER
17	031136	004737	033332		JSR	PC,\$\$SB2D ;CONVERT IT TO DECIMAL
18	031142	004737	032346		JSR	PC,SUPRSL ;TYPE IT (LEFT JUSTIFIED)
19						;TYPE THE CURRENT VALUE OF THE PARAMETER
20	031146	104401	056531		TYPE	,BLNKS1 ;TYPE 1 BLANK
21	031152	104401	056567		TYPE	,QUES ;'?'
22	031156	104401	056531		TYPE	,BLNKS1 ;TYPE 1 BLANK
23	031162	104411			RDLIN	;READ THE KEYBOARD
24	031164	012601			MOV	(SP)+,R1 ;INPUT ASCII STRING ADDRESS
25	031166	005737	001334		TST	CFLAG ;WAS (^C) TYPED?
26	031172	001021			BNE	6\$;BR IF IT WAS
27	031174	004537	033174		JSR	R5,CK.DIG ;CHECK THE DIGIT(S)
	031200	031116			1\$;CARRIAGE RETURN ONLY ENTERED
	031202	031246			7\$;PERIOD ONLY ENTERED
	031204	031220			4\$;ILLEGAL INPUT
	031206	031214			3\$;TERMINATED WITH A CARRIAGE RETURN
	031210	031220			4\$;TERMINATED WITH A '..'
	031212	031232			5\$;TERMINATED WITH A '...'
28	031214	010215		3\$:	MOV	R2,(R5) ;MOVE NEW VALUE TO PARAMETER LOCATION
29	031216	000737			BR	1\$;GET MORE PARAMETERS
30	031220	104401	060103	4\$:	TYPE	,BADENT ;'BAD ENTRY'
31	031224	162703	000006		SUB	#6,R3 ;DECREMENT THE TABLE POINTER
32	031230	000732			BR	1\$;TRY AGAIN
33	031232	010215		5\$:	MOV	R2,(R5) ;NEW VALUE
34	031234	000404			BR	7\$;EXIT
35	031236	005037	001334	6\$:	CLR	CFLAG ;CLEAR THE 'CONTROL C' FLAG
36	031242	011603			MOV	(SP),R3 ;RELOAD THE PARAMETER TABLE ADDRESS
37	031244	000724			BR	1\$;TRY AGAIN
38	031246	005726		7\$:	TST	(SP)+ ;CORRECT THE STACK POINTER
39	031250	000207			RTS	PC ;RETURN

```

1
2
3
4
5
6
7 031252 104401 001203
8 031256 104401 056567
9 031262 104401 056601
10 031266 010446
    031270 104403
    031272 002
    031273 000
11 031274 104401
12 031276 000000
13 031300 000207
14
15
16
17
18
19
20 031302 005037 043460
24 031306 005004
25 031310 111004
26 031312 146437 040550 001542
27 031320 146437 040550 032020
28 031326 006304
29 031330 010064 001544
30 031334 104401 001203
31 031340 104401 057561
32 031344 104401 057616
33 031350 104401 056601
34 031354 006204
35 031356 010446
    031360 104403
    031362 002
    031363 000
36 031364 104401 001203
37 031370 000207
38
39
40
41 031372 032777 000020 147554
42 031400 001006
43 031402 023760 001456 000102
44 031410 101002
52 031412 000137 031302
53 031416 000207
54
55
56
;TYPEOUT ASSIGN/DROP ERROR MESSAGE
;CALL:
;      MOV      #MESADR,ASNMSG ;ERROR MESSAGE ADDRESS
;      JSR      PC,ASNERR
;      RETURN
ASNERR: TYPE      ,SCLRF           ;CR-LF
        TYPE      ,QUES           ;'?
        TYPE      ,DRVMSG         ;TYPE 'DRIVE'
        MOV      R4,-(SP)         ;SAVE R4 FOR TYPEOUT
        TYPOS                    ;TYPE DRIVE NUMBER
        .BYTE     2               ;GO TYPE--OCTAL ASCII
        .BYTE     0               ;TYPE 2 DIGIT(S)
        .BYTE     0               ;SUPPRESS LEADING ZEROS
ASNMSG: .WORD     0               ;TYPE SPECIFIC MESSAGE
        RTS      PC               ;MESSAGE ADDRESS

;DROP DRIVE IF A FATAL ERROR OCCURS
;CALL:
;      JSR      PC,DROP
;      RETURN
DROP:   CLR      PERM             ;CLR PERMENANT ERROR FLAG
        CLR      R4              ;CLEAR R4 FOR DRIVE NUMBER
        MOV      (R0),R4         ;MOVE DRIVE NUMBER TO R4
        BICB    ATABIT(R4),ASNLST ;REMOVE DRIVE FROM ASSIGNED LIST
        BICB    ATABIT(R4),AUTLST ;DELETE DRIVE FROM AUTO ASSIGN LIST
        ASL      R4              ;MAKE DRIVE NUMBER INTO A TABLE INDEX
        MOV      R0,DDRVS(R4)    ;PUT DRIVE IN DROP LIST
        TYPE      ,SCLRF           ;TYPE '?FATAL OR EXCESSIVE ERRORS'
        TYPE      ,DROPNG         ;TYPE 'ON'
        TYPE      ,MSGON          ;TYPE 'DRIVE'
        ASR      R4              ;DRIVE NUMBER
        MOV      R4,-(SP)         ;SAVE R4 FOR TYPEOUT
        TYPOS                    ;TYPE DRIVE NUMBER
        .BYTE     2               ;GO TYPE--OCTAL ASCII
        .BYTE     0               ;TYPE 2 DIGIT(S)
        .BYTE     0               ;SUPPRESS LEADING ZEROS
1$:     TYPE      ,SCLRF           ;CR-LF
        RTS      PC

;ROUTINE TO DROP DRIVE IF ERRORS BECOMES EXCESSIVE
ABNRML: BIT      #SW04,@SWR       ;SEE IF SWITCH 4 SET
        BNE     1$               ;BR IF IT'S SET
        CMP     MAXER,$TOTAL(R0) ;CHECK TOTAL ERROR VALUE
        BHI     1$               ;BR IF ERRORS DO NOT EXCEED MAX
        JMP     DROP             ;DROP THE DRIVE
1$:     RTS      PC               ;RETURN

;ROUTINE TO CHECK FOR END OF PASS AND END OF TEST

```

1

.SBTTL END OF PASS ROUTINE

 :*INCREMENT THE PASS NUMBER (\$PASS)
 :*IF THERES A MONITOR GO TO IT
 :*IF THERE ISN'T JUMP TO RETURN

```

031420                                $EOP:
031420 010446                          MOV    R4, -(SP)           ;SAVE R4
031422 111004                          1$:  MOVB   (R0),R4         ;MOVE DRIVE NUMBER

031424 105737 001150                    TSTB   $AUTOB          ;RUNNING IN AUTO MODE ?
031430 001410                          BEQ    2$              ;BR IF NO
031432 136437 040550 032020            BITB   ATABIT(R4),AUTLST ;IS DRIVE ALREADY ASSIGNED TO AUTO LIST ?
031440 001110                          BNE    6$              ;BR IF YES
031442 156437 040550 032020            BISB   ATABIT(R4),AUTLST ;ADD DRIVE TO AUTO ASSIGN LIST
031450 000441                          BR     3$

031452 026037 000114 001474 2$:      CMP    $PASSC(R0),PASSES ;SEE IF AT END OF TEST
031460 103435                          BLO    3$              ;BR IF NOT
031462 032777 000020 147464            BIT    #SW04,@SWR      ;TYPE END OF TEST MESSAGE (SW04=1) ?
031470 001031                          BNE    3$              ;BR IF NO
031472 104401 001203                    TYPE   , $CRLF          ;CR-LF
031476 104401 001203                    TYPE   , $CRLF          ;CR-LF
031502 104401 057327                    TYPE   , DASH13        ;TYPE '-----'
031506 104401 057417                    TYPE   , MSGEOT         ;TYPE 'END OF TEST-----EOT'
031512 146437 040550 001542            BICB   ATABIT(R4),ASNLST ;DELETE DRIVE FROM ASSIGNED LIST
031520 006304                          ASL    R4              ;MAKE DRIVE NUMBER INTO TABLE INDEX
031522 010064 001544                    MOV    R0,DDRVS(R4)    ;PUT BLOCK ADDRESS INTO DROP LIST
031526 105737 001542                    TSTB   ASNLST          ;ALL DRIVES ARE DROPPED ?
031532 001062                          BNE    7$              ;BR IF NO
031534 005237 001216                    INC    $DEVCT          ;INCREMENT DEVICE COUNT
031540 005237 001214                    INC    $PASS           ;INCREMENT THE PASS COUNT
031544 042737 100000 001214            BIC    #100000,$PASS   ;AVOID NEGATIVE NUMBER
031552 000452                          BR     7$

031554 032777 000400 147372 3$:      BIT    #SW08,@SWR      ;INHIBIT END OF PASS TYPEOUT (SW08=1) ?
031562 001022                          BNE    4$              ;BR IF YES
031564 104401 001203                    TYPE   , $CRLF          ;CR-LF
031570 104401 001203                    TYPE   , $CRLF          ;CR-LF
031574 104401 057327                    TYPE   , DASH13        ;TYPE '-----'
031600 104401 057371                    TYPE   , MSGEOP         ;TYPE 'END OF PASS'
031604 104401 001203                    TYPE   , $CRLF          ;CR-LF
031610 004737 026146                    JSR    PC,$TIME        ;TYPE ELAPSED TIME
031614 104401 057321                    TYPE   , DASH5         ;TYPE '-----'
031620 104401 057345                    TYPE   , MSGSUM        ;TYPE 'SUMMARY'
031624 004737 025042                    JSR    PC,ONESUM       ;TYPE ONE DRIVE SUMMARY

031630 010346                          4$:  MOV    R3, -(SP)           ;SAVE R3
031632 010004                          MOV    R0,R4           ;DRIVE'S BLOCK ADDRESS
031634 062704 000036                    ADD    # $RDPAS,R4     ;ADD THE STARTING ADDR OF SECTIONS TO CLEAR
031640 012703 000016                    MOV    #<$OPERC-$RDPAS>+2,R3 ;NUMBER OF LOCNS TO BE CLEARED
031644 005024                          5$:  CLR    (R4)+           ;CLEAR LOCATIONS '$RDPAS' - '$OPERC' IN DPB
031646 162703 000002                    SUB    #2,R3           ;DONE CLEARING YET ?
031652 001374                          BNE    5$              ;BR IF NO
031654 012603                          MOV    (SP)+,R3        ;RESTORE R3
031656 005260 000114                    INC    $PASSC(R0)     ;INCREMENT THE PASS COUNT
  
```



```

031662 105737 001150      6$:  TSTB  $AUTOB      ;RUNNING IN AUTO MODE ?
031666 001404              BEQ    7$              ;BR IF NO
031670 023737 001542 032020  CMP    ASNLST,AUTLST  ;HAVE ALL DRIVES COMPLETED PASS IN AUTO MODE ?
031676 001402              BEQ    8$              ;BR IF YES
031700 012604              7$:  MOV    (SP)+,R4      ;RESTORE R4
031702 000207              RTS    PC              ;RETURN

031704 005237 032020      8$:  INC    AUTLST      ;CLEAR AUTO ASSIGN LIST FOR NEXT PASS AND
031710 001375              BNE    8$              ;WAIT FOR TTY
031712 005237 001216      INC    $DEVCT        ;INCREMENT DEVICE COUNT
031716 005237 001214      INC    $PASS         ;INCREMENT THE PASS NUMBER
031722 042737 100000 001214  BIC    #100000,$PASS ;DON'T ALLOW A NEG. NUMBER
031730 005327              DEC    (PC)+         ;LOOP?
031732 000001      $EJPCT: .WORD 1
031734 003013              BGT    $DOAGN        ;:YES
031736 012737              MOV    (PC)+,@(PC)+ ;:RESTORE COUNTER
031740 000001      $ENDCT: .WORD 1
031742 031732              $EOPCT
031744 013700 000042      $GET42: MOV    @#42,R0    ;:GET MONITOR ADDRESS
031750 001405              BEQ    $DOAGN        ;:BRANCH IF NO MONITOR
031752 000005              RESET              ;:CLEAR THE WORLD
031754 004710      $ENDAD: JSR    PC,(R0) ;:GO TO MONITOR
031756 000240              NOP                ;:SAVE ROOM
031760 000240              NOP                ;:FOR
031762 000240              NOP                ;:ACT11
031764              $DOAGN:
031764 000137      $RTNAD: JMP    @(PC)+      ;:RETURN
031766 031770      $RTNAD: .WORD  RTURN

2 3 031770 005037 177776      RTURN: CLR    PS      ;SET PRIORITY TO 0
4 031774 012706 001100      MOV    #STACK,SP    ;RESTORE STACK
5 032000 005237 001212      INC    $TESTN       ;INCREMENT THE TEST NUMBER IN THE MAIL BOX
6 032004 004737 033430      JSR    PC,$TKINT    ;MAKE SURE KEYBOARD INTERRUPT AND
7 032010 004737 024514      JSR    PC,CKCLK     ;SYSTEM CLOCK ARE STILL ON.
8 032014 000137 006340      JMP    MAIN        ;RETURN TO LOOP
9
10 032020 000000      AUTLST: .WORD 0    ;AUTO ASSIGN LIST (USED IN AUTO RUN MODE)
11
12 ;ROUTINE TO GET THE REMAINDER OF THE RANDOM NUMBER
13 ;CALL:
14 ;
15 ;   MOV    NUMBER,R5    ;DIVISOR INTO R5
16 ;   JSR    PC,GETREM   ;REMAINDER IS IN R5
17 ;   RETURN
18 032022 013746 037226      GETREM: MOV    $LONUM,-(SP) ;STORE RANDOM NUMBER ON THE STACK FOR DIVIDE
19 032026 013746 037224      MOV    $HINUM,-(SP)   ;UPPER PART
20 032032 010546      MCV    R5,-(SP)       ;PUT THE DIVISOR ONTO THE STACK
21 032034 004737 032046      JSR    PC,$DIV        ;DIVIDE THE RANDOM NUMBERS
22 032040 012605      MOV    (SP)+,R5      ;PUT THE REMAINDER INTO R5
23 032042 005726      TST    (SP)+         ;ADJUST THE STACK POINTER
24 032044 000207      RTS    PC

25
26 .SBITL  INTEGER DIVIDE ROUTINE
27
28 ;*****
29 ;*THIS ROUTINE WILL DIVIDE A 32-BIT TWO'S COMPLEMENT INTEGER

```

```

30      ;*DIVIDEND BY A 16-BIT TWO'S COMPLEMENT INTEGER DIVISOR GIVING
31      ;*A 16-BIT TWO'S COMPLEMENT INTEGER QUOTIENT AND A 16-BIT REMAINDER.
32      ;*DIVISION WILL BE PERFORMED SO THAT THE REMAINDER IS OF THE
33      ;*SAME SIGN AS THE DIVIDEND.
34      ;*CALL:
35      ;*   MOV     LOW DIVIDEND,-(SP)      ;;THE HIGH DIVIDEND MUST BE < 1/2
36      ;*   MOV     HIGH DIVIDEND,-(SP)   ;;AS LARGE AS THE DIVISOR
37      ;*   MOV     DIVISOR,-(SP)
38      ;*   JSR     PC,$DIV
39      ;*   RETURN                                ;;QUOTIENT & REMAINDER ARE ON THE STACK
40
41      ;*   STACK  NO ERROR      OVERFLOW      DIVIDE BY ZERO
42      ;*   -----  -----
43      ;*   TOP    REMAINDER     ALL ZEROS     ALL ONES
44      ;*   +2    QUOTIENT      ALL ZEROS     ALL ONES
45
46      ;*NOTE: TH'S ROUTINE WILL LINK TO THE DIVISION SUBROUTINE ('M.DPID').
    
```

```

49 032046 104412
50 032050 016605 000026
51 032054 005004
52 032056 016602 000030
53 032062 016603 000032
54 032056 005000
55 032070 005001
56 032072 004737 032170
57 032076 010166 000030
58 032102 010366 00J032
59 032106 104413
60 032110 012616
61 032112 000207

$DIV: SAVREG          ;STORE R0 - R5
      MOV     26(SP),R5 ;DIVISOR
      CLR    R4        ;OTHER DIVISOR WORD
      MOV    30(SP),R2 ;UPPER DIVIDEND WORD
      MOV    32(SP),R3 ;LOWER DIVIDEND WORD
      CLR    R0        ;CLEAR OTHER DIVIDEND REGISTERS
      CLR    R1
      JSR    PC,M.DPID ;GO TO THE DIVIDE ROUTINE
      MOV    R1,30(SP) ;REMAINDER ON THE STACK
      MOV    R3,32(SP) ;QUOTIENT ON THE STACK
      RESREG        ;RESTORE R0 - R5
      MOV    (SP)+,(SP) ;MOVE RETURN UP THE STACK
      RTS     PC
    
```

.SBTTL DOUBLE DIVIDE ROUTINE

```

62
63
64
65      ;*****
66      ;*THIS ROUTINE WILL DIVIDE A 32-BIT TWO'S COMPLEMENT INTEGER
67      ;*DIVIDEND BY A 32-BIT TWO'S COMPLEMENT INTEGER DIVISOR GIVING
68      ;*A 16-BIT TWO'S COMPLEMENT INTEGER QUOTIENT AND A 32-BIT REMAINDER.
69      ;*DIVISION WILL BE PERFORMED SO THAT THE REMAINDER IS OF THE
70      ;*SAME SIGN AS THE DIVIDEND.
71      ;*CALL:
72      ;*   MOV     LOW DIVIDEND,-(SP)      ;;THE HIGH DIVIDEND MUST BE < 1/2
73      ;*   MOV     HIGH DIVIDEND,-(SP)   ;;AS LARGE AS THE DIVISOR
74      ;*   MOV     LOW DIVISOR,-(SP)
75      ;*   MOV     HIGH DIVISOR,-(SP)
76      ;*   JSR     PC,$DBDIV
77      ;*   RETURN                                ;;QUOTIENT & REMAINDER ARE ON THE STACK
78
79      ;*   STACK  NO ERROR      OVERFLOW      DIVIDE BY ZERO
80      ;*   -----  -----
81      ;*   TOP    REMAINDER     ALL ZEROS     ALL ONES (MSD)
82      ;*   +2    REMAINDER     ALL ZEROS     ALL ONES (LSD)
83      ;*   +4    QUOTIENT      ALL ZEROS     ALL ONES
84
85      ;*NOTE: THIS ROUTINE WILL LINK TO THE DIVISION SUBROUTINE ('M.DPID').
86
    
```

```

87
88 032114 104412          $DBDIV: SAVREG          ;STORE R0 - R5
89 032116 016604 000026  MOV      26(SP),R4      ;HIGH DIVISOR WORD
90 032122 016605 000030  MOV      30(SP),R5      ;LOW DIVISOR WORD
91 032126 016602 000032  MOV      32(SP),R2      ;UPPER DIVIDEND WORD
92 032132 016603 000034  MOV      34(SP),R3      ;LOWER DIVIDEND WORD
93 032136 005000          CLR      R0            ;CLEAR OTHER DIVIDEND REGISTERS
94 032140 005001          CLR      R1
95 032142 004737 032170  JSR      PC,M.DPID      ;GO TO THE DIVIDE ROUTINE
96 032146 010066 000030  MOV      R0,30(SP)      ;REMAINDER ON THE STACK (MSD)
97 032152 010166 000032  MOV      R1,32(SP)      ;REMAINDER ON THE STACK (LSD)
98 032156 010366 000034  MOV      R3,34(SP)      ;QUOTIENT ON THE STACK
99 032162 104413          RESREG
100 032164 012616        MOV      (SP)+,(SP)      ;RESTORE R0 - R5
101 032166 000207        RTS      PC            ;MOVE RETURN UP THE STACK
102
103          .SBTTL  DOUBLE PRECISION DIVISION SUBROUTINE
104
105          :CALL:
106          :      JSR      PC,M.DPID
107          :
108          :      DIVIDEND = R0-R1-R2-R3 (R0=MSD)
109          :      DIVISOR  = R4-R5 (R4=MSD)
110          :
111          :RETURN
112          :
113          :      REMAINDER AFTER DIVISION = R0-R1 (R0=MSD)
114          :      QUOTIENT AFTER DIVISION = R2-R3 (R2=MSD)
115          :
116 032170 012746 000040  M.DPID: MOV      #40,-(SP)    ;COUNTER FOR DIVISION CYCLES
117 032174 010446          MOV      R4,-(SP)      ;HIGH ORDER
118 032176 010546          MOV      R5,-(SP)      ;LOW ORDER DIVISOR TO THE STACK
119 032200 005466 000002  NEG      2(SP)          ;FORM NEGATIVE
120 032204 005416          NEG      @SP            ;VERSION OF THE DIVISOR
121 032206 005666 000002  SBC      2(SP)
122 032212 061601          ADD      @SP,R1
123 032214 005500          ADC      R0            ;PERFORM THE INITIAL SUBTRACTION
124 032216 066600 000002  ADD      2(SP),R0
125 032222 103445          BCS      M.DP50        ;IF CARRY THEN OVERFLOW HAS OCCURRED
126 032224 005046          CLR      -(P)          ;THIS IS A LONGER LASTING CARRY BIT
127 032226 006103          M.DP40: ROL      R3
128 032230 006102          ROL      R2
129 032232 006101          ROL      R1
130 032234 006100          ROL      R0
131 032236 005716          TST      @SP            ;TEST "CARRY" INDICATOR
132 032240 001410          BEQ      M.DP41        ;IF NO "CARRY" THEN ADD ELSE SUBTRACT
133 032242 005016          CLR      @SP            ;CLEAR UP FOR NEXT TIME
134 032244 066601 000002  ADD      2(SP),Ri
135 032250 005500          ADC      R0            ;ADD -(DIVISOR)
136 032252 005516          ADC      @SP            ;SET "CARRY"
137 032254 066600 000004  ADD      4(SP),R0;<-
138 032260 000404          BR      M.DP42
139 032262 060501          M.DP41: ADD      R5,R1
140 032264 005500          ADC      R0            ;ADD +(DIVISOR)
141 032266 005516          ADC      @SP            ;SET "CARRY"
142 032270 060400          ADD      R4,R0;<-
143 032272 005516          M.DP42: ADC      @SP            ;SET "CARRY"
    
```

```
144 032274 005716          TST    @SP          ;TEST THE UPDATE INDICATOR
145 032276 001401          BEQ    .+4          ;IF ZERO FORGET IT
146 032300 005203          INC    R3           ;NO CARRY POSSIBLE HERE
147 032302 005366 000006  DEC    6(SP)        ;DECREMENT COUNTER
148 032306 003347          BGT    M.DP40       ;BRANCH IF MORE TO DO
149 032310 006003          ROR    R3
150 032312 103404          BCS    M.DP44
151 032314 060501          ADD    R5,R1
152 032316 005500          ADC    R0
153 032320 060400          ADD    R4,R0
154 032322 000241          CLC
155 032324 006103          M.DP44: ROL    R3
156 032326 062706 000010  ADD    #10,SP      ;ADJUST STACK BY 4 WORDS
157 032332 000242          CLV
158 032334 000207          RTS    PC
159 032336 062706 000006  M.DP50: ADD    #6,SP
160 032342 000262          SEV
161 032344 000207          RTS    PC
```

```

1      .SBTTL SUPRS - TYPE ASCIZ, REPLACE LEADING 0'S WITH BLANKS
2      .SBTTL SUPRSL -TYPE ASCIZ, LEFT JUSTIFY
3
4      ::*****
5      :CALL:
6      :      MOV      #NUMADR, -(SP)      ;FIRST ADDRESS OF ASCIZ STRING
7      :      JSR      PC, SUPRS
8      :      OR
9      :      MOV      #NUMADR, -(SP)      ;FIRST ADDRESS OF ASCIZ STRING
10     :      JSR      PC, SUPRSL
11
12     032346 010046 SUPRSL: MOV      RO, -(SP)      ;SAVE RO
13     032350 016500 000004      MOV      4(SP), RO      ;GET POINTER TO MESSAGE
14     032354 005037 032436      CLR      SUPR2
15     032360 000405      BR       SUPR1
16
17     032362 010046 SUPRS:  MOV      RO, -(SP)      ;SAVF RO
18     032364 016600 000004      MOV      4(SP), RO      ;GET POINTER TO MESSAGE
19     032370 010037 032436      MOV      RO, SUPR2      ;GET POINTER FOR TYPING
20     032374
21     032374 105710 SUPR1:  1$:      TSTB     (RO)      ;TEST FOR TERMINATOR
22     032376 001406      BEQ      2$             ;YES
23     032400 122710 000060      CMPB     #'0', (RO)    ;IS THIS A '0' ?
24     032404 001006      BNE      3$             ;NO
25     032406 112770 000040      MOVB     #40, (RO)+    ;REPLACE IT WITH A 'BLANK'
26     032412 000770      BR       1$             ;NEXT CHAR.
27     032414 005300      2$:      DEC      RO           ;BACKUP 1
28     032416 112710 000060      MOVB     #'0', (RO)    ;MAKE IT '0'
29     032422 005737 032436      3$:      TST      SUPR2        ;LEFT JUSTIFY ?
30     032426 001002      BNE      4$             ;NO
31     032430 010037 032436      MOV      RO, SUPR2     ;YES
32     032434 104401      4$:      TYPE
33     032436 000000 SUPR2:  .WORD    0
34     032440 012600      MOV      (SP)+, RO     ;RESTORE RO
35     032442 012616      MOV      (SP)+, (SP)   ;RESTORE STACK
36     032444 000207      RTS      PC
  
```

```

1      .SBTTL - TYPE ASCIZ, REPLACE LEADING 0'S WITH BLANKS
2      .SBTTL $SUPRL - TYPE ASCIZ, LEFT JUSTIFY
3
4      ;*****
5
6      ;THIS ROUTINE IS SAME AS 'SUPRSL' AND 'SUPRS', EXCEPT THAT IT
7      ;WILL SUPPRESS THE ERROR TYPEOUT IF SW13=1. THIS IS ACCOMPLISHED BY
8      ;USING THE TRAP CALL 'DISPLY', INSTEAD OF 'TYPE'.
9      ;CALL:
10     MOV     #NUMADR,-(SP)    ;FIRST ADDRESS OF ASCIZ STRING
11     JSR     PC,$SUPRS
12
13     OR
14     MOV     #NUMADR,-(SP)    ;FIRST ADDRESS OF ASCIZ STRING
15     JSR     PC,$SUPRL
16
17 $SUPRL: MOV     RO,-(SP)      ;SAVE RO
18         MOV     4(SP),RO     ;GET POINTER TO MESSAGE
19         CLR     $SUPR2
20         BR      $SUPR1
21
22 $SUPRS: MOV     RO,-(SP)      ;SAVE RO
23         MOV     4(SP),RO     ;GET POINTER TO MESSAGE
24         MOV     RO,$SUPR2    ;GET POINTER FOR TYPING
25
26 $SUPR1:
27 1$:     TSTB    (RO)         ;TEST FOR TERMINATOR
28         BEQ     2$          ;YES
29         CMPB   #'0',(RO)    ;IS THIS A '0' ?
30         BNE     3$          ;NO
31         MOVB   #40,(RO)+    ;REPLACE IT WITH A 'BLANK'
32         BR      1$          ;NEXT CHAR.
33 2$:     DEC     RO          ;BACKUP 1
34         MOVB   #'0',(RO)    ;MAKE IT '0'
35 3$:     TST     $SUPR2       ;LEFT JUSTIFY ?
36         BNE     4$          ;NO
37         MOV     RO,$SUPR2    ;YES
38 4$:     DISPLY 0             ;TYPE, UNLESS SW13=1
39 $SUPR2: .WORD   0
40         MOV     (SP)+,RO     ;RESTORE RO
41         MOV     (SP)+,(SP)   ;RESTORE STACK
42         RTS     PC

```

```

1      ;ROUTINE TO REPLACE LEADING ZEROS IN A NUMERIC STRING WITH SPACES
2      ;CALL:
3      :
4      :       MOV      #ADR, -(SP)      ;ADDRESS OF NUMBER (IN ASCII)
5      :       JSR      R5, REPLZ      ;REPLACE PRECEDING ZEROS WITH BLANKS
6      :       .WORD    N                ;'N' IS NUMBER OF DIGITS TO BE TYPED
7      :
8      :       OR
9      :       MOV      #ADR, -(SP)      ;ADDRESS OF NUMBER (IN ASCII)
10     :       JSR      R5, FILLZ      ;TYPE PRECEDING ZEROS
11     :       .WORD    N                ;'N' IS NUMBER OF DIGITS TO BE TYPED
12
13     032546 005237 032654  FILLZ:  INC      FILL0      ;LEAVE ZERO'S
14
15     032552 010046  REPLZ:  MOV      R0, -(SP)      ;SAVE R0
16     032554 016600 000004  MOV      4(SP), R0      ;ADDRESS OF NUMBER TO R0
17     032560 005737 032654  TST      FILL0      ;LEAVE PRECEDING ZEROS ?
18     032564 001014  BNE      3$          ;BR IF YES
19     032566 122710 000060  1$:     CMPB     #'0', (R0)      ;BYTE EQUAL TO ASCII '0' ?
20     032572 001004  BNE      2$          ;BR IF NOT
21     032574 112710 000040  MOVB     #40, (R0)      ;REPLACE THE ZERO WITH A SPACE
22     032600 005200  INC      R0          ;INCREMENT THE BYTE ADDRESS
23     032602 000771  BR       1$          ;GO BACK AND LOOK FOR MORE LEADING ZEROS
24     032604 105710  2$:     TSTB     (R0)          ;SEE IF ZERO BYTE TERMINATOR
25     032606 001003  BNE      3$          ;BR IF NOT
26     032610 005300  DEC      R0          ;BACKUP STRING POINTER
27     032612 112710 000060  MOVB     #'0', (R0)      ;PUT A ZERO BACK IN
28     032616 016600 000004  3$:     MOV      4(SP), R0      ;PUT ADDRESS OF FIRST CHARACTER ON STACK
29     032622 105720  4$:     TSTB     (R0)+        ;SEE IF ZERO BYTE TERMINATOR
30     032624 001376  BNE      4$          ;BR IF NOT
31     032626 005300  DEC      R0          ;BACKUP STRING POINTER
32     032630 162500  SUB      (R5)+, R0      ;ADJUST ADDRESS
33     032632 010037 032640  MOV      R0, 5$        ;GET ADDRESS FOR TYPEOUT
34     032636 104401  TYPE     ;TYPE THE NUMBER
35     032640 000000  5$:     .WORD    0          ;ADDRESS OF NUMBER
36     032642 012600  MOV      (SP)+, R0      ;POP STACK INTO R0
37     032644 012616  MOV      (SP)+, (SP)    ;RESTORE STACK
38     032646 005037 032654  CLR      FILL0      ;RESET FILL FLAG
39     032652 000205  RTS      R5          ;RETURN
40
41     032654 000000  FILL0:  .WORD    0          ;IF SET, LEAVE PRECEDING ZEROS FOR TYPE
42
43     ;THIS ROUTINE IS SAME AS 'REPLZ' AND 'FILLZ', EXCEPT THAT IT
44     ;WILL SUPPRESS THE ERROR TYPEOUT IF SW13=1. THIS ACCOMPLISHED BY
45     ;USED TRAP CALL 'DISPLY', INSTEAD OF 'TYPE'.
46     ;CALL:
47     :
48     :       MOV      #ADR, -(SP)      ;ADDRESS OF NUMBER (IN ASCII)
49     :       JSR      R5, $REPLZ      ;REPLACE PRECEDING ZEROS WITH BLANKS
50     :       .WORD    N                ;'N' IS NUMBER OF DIGITS TO BE TYPED
51     :
52     :       OR
53     :       MOV      #ADR, -(SP)      ;ADDRESS OF NUMBER (IN ASCII)
54     :       JSR      R5, $FILLZ      ;TYPE PRECEDING ZEROS
55     :       .WORD    N                ;'N' IS NUMBER OF DIGITS TO BE TYPED
56
57     032656 005237 032654  $FILLZ: INC      FILL0      ;LEAVE ZERO'S
58
59     032662 010046  $REPLZ: MOV      R0, -(SP)      ;SAVE R0
60     032664 016600 000004  MOV      4(SP), R0      ;ADDRESS OF NUMBER TO R0
61     032670 005737 032654  TST      FILL0      ;LEAVE PRECEDING ZEROS ?

```

```

58 032674 001014          BNE      3$          ;BR IF YES
59 032676 122710 000060  1$: CMPB   #'0,(R0)    ;BYTE EQUAL TO ASCII '0' ?
60 032702 001004          BNE      2$          ;BR IF NOT
61 032704 112710 000040  MOVB   #40,(R0)     ;REPLACE THE ZERO WITH A SPACE
62 032710 005200          INC      R0          ;INCREMENT THE BYTE ADDRESS
63 032712 000771          BR       1$          ;GO BACK AND LOOK FOR MORE LEADING ZEROS
64 032714 105710          2$: TSTB   (R0)        ;SEE IF ZERO BYTE TERMINATOR
65 032716 001003          BNE      3$          ;BR IF NOT
66 032720 005300          DEC      R0          ;BACKUP STRING POINTER
67 032722 112710 000060  MOVB   #'0,(R0)     ;PUT A ZERO BACK IN
68 032726 016600 000004  3$: MOV    4(SP),R0    ;PUT ADDRESS OF FIRST CHARACTER ON STACK
69 032732 105720          4$: TSTB   (R0)+      ;SEE IF ZERO BYTE TERMINATOR
70 032734 001376          BNE      4$          ;BR IF NOT
71 032736 005300          DEC      R0          ;BACKUP STRING POINTER
72 032740 162500          SUB     (R5)+,R0     ;ADJUST ADDRESS
73 032742 010037 032750  MOV     R0,5$        ;GET ADDRESS FOR TYPEOUT
74 032746 104414          DISPLY  0           ;TYPE, UNLESS SW13=1
75 032750 000000          5$: .WORD  0         ;ADDRESS OF NUMBER
76 032752 012600          MOV     (SP)+,R0     ;POP STACK INTO R0
77 032754 012616          MOV     (SP)+,(SP)  ;RESTORE STACK
78 032756 005037 032654  CLR     FILL0       ;RESET FILL FLAG
79 032762 000205          RTS      R5         ;RETURN

```

```

80
91 ;ROUTINE TO TYPE THE DRIVE SERIAL NUMBER IN DECIMAL
92 ;CALL:
93     MOV     #DPB,R0          ;ADDRESS OF DRIVE PARAMETER BLOCK
94     JSR     PC,TYDRV         ;CALL ROUTINE
95     OR
96     MOV     #DPB,R0          ;ADDRESS OF DRIVE PARAMETER BLOCK
97     JSR     PC,TYPDRV        ;CALL ROUTINE(WITH NO HEADER MESSAGE)
98
99 ;R0 = DPB ADDRESS BEFORE CALLING THE ROUTINE

```

```

100
101 032764 104401 060152  TYDRV: TYPE  ,DRVSN      ;TYPE 'DRV S/N: '
102 032770 104401 057052  TYPDRV: TYPE  ,MSGPG     ;TYPE 'PG'
103 032774 010037 033010  MOV     R0,1$             ;ADDRESS OF DPB
104 033000 062737 000160 033010  ADD     #SDRVSN,1$       ;INDEX TO DRIVE SERIAL NUMBER
105 033006 104401          TYPE          ;TYPE THE DRIVE SERIAL NUMBER
106 033010 000000          1$: .WORD  0           ;ADDRESS OF DRIVE SERIAL NUMBER FIELD
107 033012 104401 057711  TYPE    ,PERIOD         ;TYPE '.'
108 033016 000207          RTS      PC           ;RETURN

```

```

109
110 ;ROUTINE TO TYPE ERRORS
111 ;CALL
112     DISPLY  MESSADR        ;MUST DEFINED IN 'TRAP' TABLE
113     RETURN                ;ADDRESS OF MESSAGE
114
115

```

```

116 033020 032777 020000 146126  $DSPLY: BIT   #BIT13,@SWR  ;INHIBIT ERROR TYPEOUT ?
117 033026 001004          BNE      1$          ;BR IF YES
118 033030 005037 177776  CLR     @#PS          ;SET PRIORITY TO ZERO
119 033034 000137 035652  JMP     $TYPE         ;TYPE THE MESSAGE
120 033040 062716 000002  1$: ADD     #2,(SP)     ;INCREMENT THE RETURN
121 033044 000002          RTI                    ;RETURN

```

```

122
123 ;THIS ROUTINE IS USED TO CHECK IF AN
124 ;ASCII CHARACTER IS A DIGIT BETWEEN 0 AND 7.

```



```

125      :CALL
126      :      MOV      #ADR,R1      ;ADDRESS OF ASCII CHARACTER
127      :      JSR      R5,CK.OCT    ;CHECK THE CHARACTER
128      :      RETURN1    ;CHARACTER IS NOT BETWEEN 0-7
129      :      RETURN2    ;CHARACTER IS IN R2 AS A
130      :      :          ;OCTAL DIGIT
131
132 033046 121127 000060 CK.OCT: CMPB      (R1),#'0      ;LESS THAN ZERO?
133 033052 103407      BLO      1$          ;YES -- BRANCH
134 033054 121127 000067      CMPB      (R1),#'7      ;GREATER THAN SEVEN?
135 033060 101004      BHI      1$          ;YES -- BRANCH
136 033062 111102      MOVB      (R1),R2      ;GET THE CHARACTER
137 033064 042702 177770      BIC      #'^C7,R2      ;STRIP AWAY THE ASCII
138 033070 005725      TST      (R5)+      ;ADJUST FOR RETURN
139 033072 000205 1$:      RTS      R5          ;RETURN
140
141      :THIS ROUTINE IS USED TO CHECK AN ASCII CHARACTER
142      :AND DETERMINE IF IT IS A DIGIT BETWEEN 0 AND 9.
143      :CALL
144      :      MOV      #ADR,R1      ;ADDRESS OF ASCII CHARACTER
145      :      JSR      R5,CK.DEC    ;CHECK THE CHARACTER
146      :      RETURN1    ;NOT BETWEEN 0 AND 9
147      :      RETURN2    ;BETWEEN 0 AND 9
148      :      :          ;R = DIGIT
149
150 033074 121127 000060 CK.DEC: CMPB      (R1),# 0      ;LESS THAN ZERO?
151 033100 103407      BLO      1$          ;YES -- BRANCH
152 033102 121127 000071      CMPB      (R1),#'9      ;GREATER THAN NINE?
153 033106 101004      BHI      1$          ;YES -- BRANCH
154 033110 111102      MOVB      (R1),R2      ;GET THE CHARACTER
155 033112 042702 000060      BIC      #'0,R2      ;STRIP AWAY THE ASCII
156 033116 005725      TST      (R5)+      ;ADJUST FOR RETURN
157 033120 000205 1$:      RTS      R5          ;RETURN
158
159      :THIS ROUTINE WILL CHECK AN ASCII CHARACTER TO
160      :DETERMINE WHAT IT IS.
161      :CALL
162      :      MOV      #ADR,R1      ;ADDRESS OF ASCII CHARACTER
163      :      JSR      R5,CK.CHR    ;CHECK CHARACTER
164      :      RETURN  ADR1    ;UNKNOWN CHARACTER
165      :      RETURN  ADR2    ;CARRIAGE RETURN * (R1)=ADR+1
166      :      RETURN  ADR3    ;COMMA * (R1)=ADR+1
167      :      RETURN  ADR4    ;PERIOD * (R1)=ADR+1
168      :      RETURN  ADR5    ;DIGIT BETWEEN 0 AND 7.
169      :      RETURN  ADR6    ;DIGIT BETWEEN 8 AND 9.
170      :      :          ;R2 = DIGIT * (R1)=ADR+1
171
172 033122 105711 CK.CHR: TSTB      (R1)      ;"CARRIAGE RETURN"?
173 033124 001417      BEQ      3$          ;YES -- BRANCH
174 033126 121127 000054      CMPB      (R1),#',      ;"COMMA"?
175 033132 001413      BEQ      2$          ;YES -- BRANCH
176 033134 121127 000056      CMPB      (R1),#'.      ;"PERIOD"?
177 033140 001407      BEQ      1$          ;YES -- BRANCH
178 033142 004537 033074      JSR      R5,CK.DEC    ;"DIGIT"?
179 033146 000410      BR      4$          ;NO -- BRANCH
180 033150 004537 033046      JSR      R5,CK.OCT    ;OCTAL ?
181 033154 005725      TST      (R5)+      ;DIGIT BETWEEN 8-9

```

```

182 033156 005725          TST      (R5)+          ;DIGIT BETWEEN 0-7
183 033160 005725          1$:    TST      (R5)+          ;PERIOD
184 033162 005725          2$:    TST      (R5)+          ;COMMA
185 033164 005725          3$:    TST      (R5)+          ;CARRIAGE RETURN
186 033166 00520          INC      R1              ;MOVE POINTER TO NEXT CHARACTER
187 033170 011505          4$:    MOV      (R5),R5      ;UNKNOWN CHARACTER
188 033172 000205          RTS       R5              ;RETURN
189
190                          ;THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL
191                          ;CHARACTERS AND FORMS A DECIMAL VALUE BINARY NUMBER IN R2.
192                          ;CALL
193                          ;:
194                          ;:    MOV      #ADR,R1          ;ADDRESS OF ASCII STRING
195                          ;:    MOV      #NUM,R2          ;MAX. MAGNITUDE OF INPUT NUMBER
196                          ;:    JSR      R5,CK.DIG        ;CHECK DIGITS
197                          ;:    RETURN  ADR1           ;"CR" ONLY ENTERED -- R2=0
198                          ;:    RETURN  ADR2           ;"PERIOD" ONLY ENTERED -- R2=0
199                          ;:    RETURN  ADR3           ;ILLEGAL CHARACTER OR INPUT TOO LARGE -- R2=?
200                          ;:    RETURN  ADR4           ;"CR" -- R2 = NUMBER
201                          ;:    RETURN  ADR5           ;"COMMA" -- R2 = NUMB·R
202                          ;:    RETURN  ADR6           ;"PERIOD" -- R2 = NUMBER
203 033174 010446          CK.DIG: MOV     R4,-(SP)        ;SAVE R4
204 033176 010346          MOV     R3,-(SP)        ;SAVE R3
205 033200 010246          MOV     R2,-(SP)        ;SAVE THE MAX. SIZE ON THE STACK
206 033202 005002          CLR     R2              ;START WITH 0
207 033204 005003          CLR     R3
208 033206 005004          CLR     R4
209 033210 004537 033122  JSR     R5,CK.CHR        ;CHECK ONE CHARACTER
210 033214 033  0          6$:    ;ILLEGAL CHARACTER
211 033216 033316          9$:    ;CARRIAGE RETURN
212 033220 033310          6$:    ;:
213 033222 033312          7$:    ;:
214 033224 033230          1$:    ;DIGIT 0-7
215 033226 033230          1$:    ;DIGIT 8-9
216 033230 062705 000004  1$:    ADD     #4,R5          ;STEP RETURN POINTER PAST "CR" & "PERIOD" RETURNS
217 033234 006303          2$:    ASL     R3              ;INPUT NUMBER *2
218 033236 010346          MOV     R3,-(SP)        ;SAVE *2
219 033240 006303          ASL     R3              ;*4
220 033242 006303          ASL     R3              ;*8
221 033244 062603          ADD     (SP)+,R3        ;(*2)+(*8) = *10
222 033246 060203          ADD     R2,R3          ;UPDATE THE INPUT NUMBER
223 033250 004537 033122  JSR     R5,CK.CHR        ;CHECK ONE CHARACTER
224 033254 033314          8$:    ;ILLEGAL CHARACTER
225 033256 033300          5$:    ;CARRIAGE RETURN
226 033260 033276          4$:    ;:
227 033262 033270          3$:    ;:
228 033264 033234          2$:    ;DIGIT 0-7
229 033266 033234          2$:    ;DIGIT 8-9
230 033270 105711          3$:    TSTB     (R1)        ;DOES A "CR" FOLLOW THE "PERIOD"
231 033272 001010          BNE     8$              ;BR IF NOT
232 033274 005724          TST     (R4)+          ;INCREMENT THE RETURN
233 033276 005724          4$:    TST     (R4)+          ;INCREMENT THE RETURN
234 033300 005724          5$:    TST     (R4)+          ;INCREMENT THE RETURN
235 033302 020316          CMP     R3,(SP)        ;CHECK THE MAGNITUDE OF THE NUMBER
236 033304 101004          BHI     9$              ;BR IF ENTERED NUMBER TOO LARGE
237 033306 000402          BR      8$              ;BYPASS INCREMENT
238 033310 005725          6$:    TST     (R5)+          ;INCREMENT RETURN PAST INVALID RETURN

```

```

227 033312 005725 7$: TST (R5)+ ;INCREMENT RETURN
228 033314 060405 8$: ADD R4,R5 ;SETUP RETURN POINTER
229 033316 010302 9$: MOV R3,R2 ;ENTERED VALUE
230 033320 005726 TST (SP)+ ;CLEAN MAX. SIZE OFF OF STACK
231 033322 012603 MOV (SP)+,R3 ;RESTORE R3
232 033324 012604 MOV (SP)+,R4 ;RESTORE R4
233 033326 011505 MOV (R5),R5 ;GET RETURN ADDRESS
234 033330 000205 RTS R5 ;RETURN
235
236 ;THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
237 ;UNSIGNED DECIMAL ASCIZ NUMBER.
238 ;CALL
239 ;MOV NUMBER,-(SP) ;PUT THE NUMBER ON THE STACK
240 ;JSR PC,$SB2D ;CALL
241 ;RETURN ;ADDRESS OF THE 1ST ASCIZ CHAR IS ON THE STACK
242
243 ;NOTE: THE PROGRAM REQUIRES THIS FORM OF '$SB2D', NOT THE VERSION ON
244 ;THE SYSMAC LIBRARY, REV C AND LATER
245
246 033332 016637 000002 033356 $SB2D: MOV 2(SP),1$ ;SAVE THE BINARY NUMBER
247 033340 012746 033356 MOV #1$,-(SP) ;SET THE POINTER
248 033344 004737 037324 JSR PC,$DB2D ;CALL THE DOUBLE LENGTH CONVERT
249 033350 012666 000002 MOV (SP)+,2(SP) ;PICKUP THE POINTER
250 033354 000207 RTS PC ;RETURN
251 033356 000000 000000 1$: .WORD 0,0
252
253 ;THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
254 ;UNSIGNED OCTAL ASCIZ NUMBER.
255 ;CALL
256 ;MOV NUMBER,-(SP) ;PUT THE NUMBER ON THE STACK
257 ;JSR PC,$SB20 ;CALL
258 ;RETURN ;ADDRESS OF THE 1ST ASCIZ CHAR IS ON THE STACK
259
260 ;NOTE: THE PROGRAM REQUIRES THIS FORM OF '$SB20', NOT THE VERSION ON
261 ;THE SYSMAC LIBRARY, REV C AND LATER
262
263 033362 016637 000002 033406 $SB20: MOV 2(SP),1$ ;SAVE THE BINARY NUMBER
264 033370 012746 033406 MOV #1$,-(SP) ;SET THE POINTER
265 033374 004737 037520 JSR PC,$DB20 ;CALL THE DOUBLE LENGTH CONVERT
266 033400 012666 000002 MOV (SP)+,2(SP) ;PICKUP THE POINTER
267 033404 000207 RTS PC ;RETURN
268 033406 000000 000000 1$: .WORD 0,0
  
```

1

.SBTTL TTY INPUT ROUTINE

```

*****
ENABL  LSB
033412 000000 $TKCNT: .WORD 0          ;;NUMBER OF ITEMS IN QUEUE
033414 000000 $TKQIN: .WORD 0          ;;INPU  POINTER
033416 CJO000 $TKQOUT: .WORD 0         ;;OUTPUT POINTER
033420 033427 $TKQSR: .BLKB 7         ;;TTY KEYBOARD QUEUE
$TKQEND=.
.EVEN

;*TK INITIALIZE ROUTINE
;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
;CALL:
;*
;*      JSR    PC,$TKINT
;*      RETURN
$TKINT: CLR    $TKCNT          ;;CLEAR COUNT OF ITEMS IN QUEUE
        MOV    # $TKQSR,$TKQIN ;;MOVE THE STARTING ADDRESS OF THE
        MOV    $TKQIN,$TKQOUT  ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
        MOV    # $TKSRV,@TKVEC ;;INITIALIZE THE KEYBOARD VECTOR
        MOV    #200,@TKVEC+2   ;;'BR' LEVEL 4
        TST   @TKKB           ;;CLEAR DONE FLAG
        MOV    #100,@TKS       ;;ENABLE TTY KEYBOARD INTERRUPT
        RTS    PC             ;;RETURN TO CALLER

;*TK SERVICE ROUTINE
;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
;*IT IN THE QUEUE.
;*IF THE CHARACTER IS A 'CONTROL-C' (^C) $TKINT IS CALLED AND
;*UPON RETURN EXIT IS MADE TO THE 'CONTROL-C' RESTART ADDRESS (CTRAP)
$TKSRV: MOVB   @TKKB,-(SP)    ;;PICKUP THE CHARACTER
        BIC   #^C177,(SP)    ;;STRIP THE JUNK
        CMP   (SP),#$XON     ;;IS IT A RANDOM XON?
        BNE   30$           ;;BRANCH IF NO
        TST  (SP)+          ;;CLEAN RANDOM XON OFF STACK
        RTI                    ;;RETURN
30$:    CMP   (SP),#3        ;;IS IT A CONTROL C?
        BNE   1$           ;;BRANCH IF NO
        TYPE ,SCNTLC        ;;TYPE A CONTROL-C (^C)
        JSR   PC,$TKINT     ;;INIT THE KEYBOARD
        TST  (SP)+          ;;CLEAN UP STACK
        JMP   CTRAP         ;;CONTROL C RESTART
1$:    CMP   (SP),#7        ;;IS IT A CONTROL G?
        BNE   2$           ;;BRANCH IF NO
        CMP   #SWREG,SWR    ;;IS SOFT-SWR SELECTED?
        BEQ   6$           ;;GO TO SWR CHANGE
2$:    CMP   #7,$TKCNT      ;;IS THE QUEUE FULL?
        BNE   3$           ;;BRANCH IF NO
        TYPE ,SBELL        ;;RING THE TTY BELL
033430 005037 033412 $TKINT: CLR    $TKCNT          ;;CLEAR COUNT OF ITEMS IN QUEUE
033434 012737 033420 033414 MOV    # $TKQSR,$TKQIN ;;MOVE THE STARTING ADDRESS OF THE
033442 013737 033414 033416 MOV    $TKQIN,$TKQOUT  ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
033450 012737 033500 000060 MOV    # $TKSRV,@TKVEC ;;INITIALIZE THE KEYBOARD VECTOR
033456 012737 000200 000062 MOV    #200,@TKVEC+2   ;;'BR' LEVEL 4
033464 005777 145472 TST   @TKKB           ;;CLEAR DONE FLAG
033470 012777 000100 145462 MOV    #100,@TKS       ;;ENABLE TTY KEYBOARD INTERRUPT
033476 000207 RTS    PC             ;;RETURN TO CALLER

;*TK SERVICE ROUTINE
;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
;*IT IN THE QUEUE.
;*IF THE CHARACTER IS A 'CONTROL-C' (^C) $TKINT IS CALLED AND
;*UPON RETURN EXIT IS MADE TO THE 'CONTROL-C' RESTART ADDRESS (CTRAP)
$TKSRV: MOVB   @TKKB,-(SP)    ;;PICKUP THE CHARACTER
        BIC   #^C177,(SP)    ;;STRIP THE JUNK
        CMP   (SP),#$XON     ;;IS IT A RANDOM XON?
        BNE   30$           ;;BRANCH IF NO
        TST  (SP)+          ;;CLEAN RANDOM XON OFF STACK
        RTI                    ;;RETURN
30$:    CMP   (SP),#3        ;;IS IT A CONTROL C?
        BNE   1$           ;;BRANCH IF NO
        TYPE ,SCNTLC        ;;TYPE A CONTROL-C (^C)
        JSR   PC,$TKINT     ;;INIT THE KEYBOARD
        TST  (SP)+          ;;CLEAN UP STACK
        JMP   CTRAP         ;;CONTROL C RESTART
1$:    CMP   (SP),#7        ;;IS IT A CONTROL G?
        BNE   2$           ;;BRANCH IF NO
        CMP   #SWREG,SWR    ;;IS SOFT-SWR SELECTED?
        BEQ   6$           ;;GO TO SWR CHANGE
2$:    CMP   #7,$TKCNT      ;;IS THE QUEUE FULL?
        BNE   3$           ;;BRANCH IF NO
        TYPE ,SBELL        ;;RING THE TTY BELL
033500 117746 145456 $TKSRV: MOVB   @TKKB,-(SP)    ;;PICKUP THE CHARACTER
033504 042716 177600 BIC   #^C177,(SP)    ;;STRIP THE JUNK
033510 021627 000021 CMP   (SP),#$XON     ;;IS IT A RANDOM XON?
033514 001002 BNE   30$           ;;BRANCH IF NO
033516 005726 TST  (SP)+          ;;CLEAN RANDOM XON OFF STACK
033520 000002 RTI                    ;;RETURN
033522 021627 000003 30$:    CMP   (SP),#3        ;;IS IT A CONTROL C?
033526 001007 BNE   1$           ;;BRANCH IF NO
033530 104401 034635 TYPE ,SCNTLC        ;;TYPE A CONTROL-C (^C)
033534 004737 03343C JSR   PC,$TKINT     ;;INIT THE KEYBOARD
033540 005726 TST  (SP)+          ;;CLEAN UP STACK
033542 000137 034676 JMP   CTRAP         ;;CONTROL C RESTART
033546 021627 000007 1$:    CMP   (SP),#7        ;;IS IT A CONTROL G?
033552 001004 BNE   2$           ;;BRANCH IF NO
033554 022737 000176 001154 CMP   #SWREG,SWR    ;;IS SOFT-SWR SELECTED?
033562 001500 BEQ   6$           ;;GO TO SWR CHANGE
033564 022737 000007 033412 2$:    CMP   #7,$TKCNT      ;;IS THE QUEUE FULL?
033572 001004 BNE   3$           ;;BRANCH IF NO
033574 104401 001176 TYPE ,SBELL        ;;RING THE TTY BELL

```

```

033600 005726          TST      (SP)+          ;;CLEAN CHARACTER OFF OF STACK
033602 000451          BR       5$              ;;EXIT
033604 021627 000023  3$:    CMP      (SP),#23          ;;IS IT A CONTROL-S?
033610 001021          BNE     32$              ;;BRANCH IF NO
033612 005077 145342          CLR     @STKS            ;;DISABLE TTY KEYBOARD INTERRUPTS
033616 005726          TST      (SP)+          ;;CLEAN CHAR OFF STACK
033620 105777 145334  31$:  TSTB   @STKS            ;;WAIT FOR A CHAR
033624 100375          BPL     31$              ;;LOOP UNTIL ITS THERE
033626 117746 145330          MOVB   @STKB,-(SP)       ;;GET THE CHARACTER
033632 042716 177600          BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
033636 022627 000021          CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
033642 001366          BNE     31$              ;;BRANCH IF NO
033644 012777 000100 145306  MOV     #100,@STKS       ;;REENABLE TTY KEYBOARD INTERRUPTS
033652 000002          RTI                    ;;RETURN
033654 005237 033412  32$:  INC     $TKCNT           ;;COUNT THIS CHARACTER
033660 021627 000140          CMP     (SP),#140       ;;IS IT UPPER CASE?
033664 002405          BLT     4$              ;;BRANCH IF YES
033666 021627 000175          CMP     (SP),#175       ;;IS IT A SPECIAL CHAR?
033672 003002          BGT     4$              ;;BRANCH IF YES
033674 042716 000040          BIC     #40,(SP)        ;;MAKE IT UPPER CASE
033700 112677 177510  4$:  MOVB   (SP)+,@STKQIN    ;;AND PUT IT IN QUEUE
033704 005237 033414          INC     $TKQIN          ;;UPDATE THE POINTER
033710 023727 033414 033427  CMP     $TKQIN,$TKQEND  ;;GO OFF THE END?
033716 001003          BNE     5$              ;;BRANCH IF NO
033720 012737 033420 033414  MOV     #$TKQSRT,$TKQIN ;;RESET THE POINTER
033726 000002  5$:    RTI                    ;;RETURN

```

```

*****
;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP
;*CALL WHEN OPERATING IN TTY INTERRUPT MODE.

```

```

033730 022737 000176 001154 $CKSWR: CMP     #SWREG,SWR    ;;IS THE SOFT-SWR SELECTED
033736 001124          BNE     15$              ;;EXIT IF NOT
033740 105777 145214          TSTB   @STKS            ;;IS A CHAR WAITING?
033744 100121          BPL     15$              ;;IF NOT, EXIT
033746 117746 145210          MOVB   @STKB,-(SP)     ;;YES
033752 042716 177600          BIC     #^C177,(SP)    ;;MAKE IT 7-BIT ASCII
033756 021627 000007          CMP     (SP),#7        ;;IS IT A CONTROL-G?
033762 001300          BNE     2$              ;;IF NOT, PUT IT IN THE TTY QUEUE
                                ;;AND EXIT

```

```

*****
;*CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
;*ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
;*CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.

```

```

033764 123727 001150 000001 6$:  CMPB   $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
033772 001674          BEQ     2$              ;;BRANCH IF YES
033774 005726          TST      (SP)+          ;;CLEAR CONTROL-G OFF STACK
033776 004737 033430          JSR     PC,$TKINT       ;;FLUSH THE TTY INPUT QUEUE
034002 005077 145152          CLR     @STKS            ;;DISABLE TTY KEYBOARD INTERRUPTS
034006 112737 000001 001151  MOVB   #1,$INTAG        ;;SET INTERRUPT MODE INDICATOR

034014 104401 034647  $GTSWR: TYPE     ,$CNTLG    ;;ECHO THE CONTROL-G (^G)
034020 104401 034654          TYPE     ,$MSWR         ;;TYPE CURRENT CONTENTS
034024 013746 000176          MOV     SWREG,-(SP)    ;;SAVE SWREG FOR TYPEOUT
034030 104402          TYPOC                ;;GO TYPE--OCTAL ASCII(ALL DIGITS)

```

034032	104401	034665		TYPE	,\$MNEW	:::PROMPT FOR NEW SWR
034036	005046		19\$:	CLR	-(SP)	:::CLEAR COUNTER
034040	005046			CLR	-(SP)	:::THE NEW SWR
034042	105777	145112	7\$:	TSTB	@\$TKS	:::CHAR THERE?
034046	100375			BPL	7\$:::IF NOT TRY AGAIN
034050	117746	145106		MOVB	@\$TKB,-(SP)	:::PICK UP CHAR
034054	042716	177600		BIC	#^C177,(SP)	:::MAKE IT 7-BIT ASCII
034060	021627	000003		CMP	(SP),#3	:::IS IT A CONTROL-C?
034064	001015			BNE	9\$:::BRANCH IF NOT
034066	104401	034635		TYPE	,\$CNTLC	:::YES, ECHO CONTROL-C (^C)
034072	062706	000006		ADD	#6,SP	:::CLEAN UP STACK
034076	123727	001151	000001	CMPB	\$INTAG,#1	:::REENABLE TTY KEYBOARD INTERRUPTS?
034104	001003			BNE	8\$:::BRANCH IF NO
034106	012777	000100	145044	MOV	#100,@\$TKS	:::ALLOW TTY KEYBOARD INTERRUPTS
034114	000137	034676	8\$:	JMP	CTRAP	:::CONTROL-C RESTART
034120	021627	000025	9\$:	CMP	(SP),#25	:::IS IT A CONTROL-U?
034124	001005			BNE	10\$:::BRANCH IF NOT
034126	104401	034642		TYPE	,\$CNTLU	:::YES, ECHO CONTROL-U (^U)
034132	062706	000006	20\$:	ADD	#6,SP	:::IGNORE PREVIOUS INPUT
034136	000737			BR	19\$:::LET'S TRY IT AGAIN
034140	021627	000015	10\$:	CMP	(SP),#15	:::IS IT A <CR>?
034144	001022			BNE	16\$:::BRANCH IF NO
034146	005766	000004		TST	4(SP)	:::YES, IS IT THE FIRST CHAR?
034152	001403			BEQ	11\$:::BRANCH IF YES
034154	016677	000002	144772	MOV	2(SP),@SWR	:::SAVE NEW SWR
034162	062706	000006	11\$:	ADD	#6,SP	:::CLEAR UP STACK
034166	104401	001203	14\$:	TYPE	,\$CRLF	:::ECHO <CR> AND <LF>
034172	123727	001151	000001	CMPB	\$INTAG,#1	:::RE-ENABLE TTY KBD INTERRUPTS?
034200	001003			BNE	15\$:::BRANCH IF NOT
034202	012777	000100	144750	MOV	#100,@\$TKS	:::RE-ENABLE TTY KBD INTERRUPTS
034210	000002		15\$:	RTI		:::RETURN
034212	004737	036064	16\$:	JSR	PC,\$TYPEC	:::ECHO CHAR
034216	021627	000060		CMP	(C'),#60	:::CHAR < 0?
034222	002420			BLT	18\$:::BRANCH IF YES
034224	021627	000067		CMP	(SP),#67	:::CHAR > 7?
034230	003015			BGT	18\$:::BRANCH IF YES
034232	042726	000060		BIC	#60,(SP)+	:::STRIP-OFF ASCII
034236	005766	000002		TST	2(SP)	:::IS THIS THE FIRST CHAR
034242	001403			BEQ	17\$:::BRANCH IF YES
034244	006316			ASL	(SP)	:::NO, SHIFT PRESENT
034246	006316			ASL	(SP)	:::CHAR OVER TO MAKE
034250	006316			ASL	(SP)	:::ROOM FOR NEW ONE.
034252	005266	000002	17\$:	INC	2(SP)	:::KEEP COUNT OF CHAR
034256	056616	177776		BIS	-2(SP),(SP)	:::SET IN NEW CHAR
034262	000667			BR	7\$:::GET THE NEXT ONE
034264	104401	001202	18\$:	TYPE	,\$QUES	:::TYPE ?<CR><LF>
034270	000720			BR	20\$:::SIMULATE CONTROL-U
			.DSABL	LSB		

:::.....

```

: *THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
: *CALL:
: *      RDCHR          :: GET A CHARACTER FROM THE QUEUE
: *      RETURN HERE   :: CHARACTER IS ON THE STACK
: *                   :: WITH PARITY BIT STRIPPED OFF
:
034272 011646          $RDCHR: MOV      (SP), -(SP)      :: PUSH DOWN THE PC AND
034274 016666 000004 000002  MOV      4(SP), 2(SP)    :: THE PS
034302 005066 000004          CLR      4(SP)          :: GET READY FOR A CHARACTER
034306 005046          CLR      -(SP)          :: PUT NEW PS ON STACK
034310 012746 034316  MOV      #64$, -(SP)    :: PUT NEW PC ON STACK
034314 000002          RTI                    :: POP NEW PC AND PS
034316
034316 005737 033412 64$:   TST      $TKCNT          :: WAIT ON A CHARACTER
034322 001775          BEQ      1$
034324 005337 033412          DEC      $TKCNT          :: DECREMENT THE COUNTER
034330 117766 177062 000004  MOVB   @STKQOUT, 4(SP)    :: GET ONE CHARACTER
034336 005237 033416          INC      $TKQOUT          :: UPDATE THE POINTER
034342 023727 033416 033427  CMP     $TKQOUT, #STKQEND :: DID IT GO OFF OF THE END?
034350 001003          BNE     2$              :: BRANCH IF NO
034352 012737 033420 033416  MOV     #STKQSRRT, $TKQOUT :: RESET THE POINTER
034360 000002          RTI                    :: RETURN
: *****
: *THIS ROUTINE WILL INPUT A STRING FROM THE TTY
: *CALL:
: *      RDLIN         :: INPUT A STRING FROM THE TTY
: *      RETURN HERE   :: ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
: *                   :: TERMINATOR WILL BE A BYTE OF ALL 0'S
:
034362 010346          $RDLIN: MOV      R3, -(SP)      :: SAVE R3
034364 005046          CLR      -(SP)          :: CLEAR THE RUBOUT KEY
034366 012703 034616 1$:   MOV     #STTYIN, R3      :: GET ADDRESS
034372 022703 034635 2$:   CMP     #STTYIN+15., R3  :: BUFFER FULL?
034376 101456          BLOS   4$              :: BR IF YES
034400 104410          RDCHR          :: GO READ ONE CHARACTER FROM THE TTY
034402 112613          MOVB   (SP)+, (R3)      :: GET CHARACTER
034404 122713 000177 10$:  CMPB   #177, (R3)        :: IS IT A RUBOUT
034410 001022          BNE   5$              :: BR IF NO
034412 005716          TST   (SP)            :: IS THIS THE FIRST RUBOUT?
034414 001007          BNE   6$              :: BR IF NO
034416 112737 000134 034614  MOVB   #' \, 0$         :: TYPE A BACK SLASH
034424 104401 034614          TYPE   , 9$
034430 012716 177777          MOV   #-1, (SP)        :: SET THE RUBOUT KEY
034434 005303 6$:   DEC     R3              :: BACKUP BY ONE
034436 020327 034616          CMP   R3, #STTYIN      :: STACK EMPTY?
034442 103434          BLO   4$              :: BR IF YES
034444 111337 034614          MOVB   (R3), 9$        :: SETUP TO TYPEOUT THE DELETED CHAR.
034450 104401 034614          TYPE   , 9$          :: GO TYPE
034454 000746          BR    2$              :: GO READ ANOTHER CHAR.
034456 005716          5$:   TST   (SP)            :: RUBOUT KEY SET?
034460 001406          BEQ   7$              :: BR IF NO
034462 112737 000134 034614  MOVB   #' \, 9$         :: TYPE A BACK SLASH
034470 104401 034614          TYPE   , 9$
034474 005016          CLR   (SP)            :: CLEAR THE RUBOUT KEY
034476 122713 000025 7$:   CMPB   #25, (R3)        :: IS CHARACTER A CTRL U?
034502 001003          BNE   8$              :: BR IF NO

```

```

034504 104401 034642          TYPE      $CNTLU          ;;TYPE A CONTROL 'U'
034510 000726          BR          1$          ;;GO START OVER
034512 122713 000022      8$:      CMPB      #22,(R3)      ;;IS CHARACTER A '^R'?
034516 001011          BNE          3$          ;;BRANCH IF NO
034520 105013          CLRB      (R3)          ;;CLEAR THE CHARACTER
034522 104401 001203      TYPE      , $CRLF      ;;TYPE A 'CR' & 'LF'
034526 104401 034616      TYPE      , $TTYIN     ;;TYPE THE INPUT STRING
034532 000717          BR          2$          ;;GO PICKUP ANOTHER CHACTER
034534 104401 001202      4$:      TYPE      , $QUES     ;;TYPE A '?'
034540 000712          BR          1$          ;;CLEAR THE BUFFER AND LOOP
034542 111337 034614      3$:      MOVB      (R3),9$      ;;ECHO THE CHARACTER
034546 104401 034614      TYPE      , 9$
034552 122723 000015      CMPB      #15,(R3)+    ;;CHECK FOR RETURN
034556 001305          BNE          2$          ;;LOOP IF NOT RETURN
034560 105063 177777      CLRB      -1(R3)      ;;CLEAR RETURN (THE 15)
034564 104401 001204      TYPE      , $LF       ;;TYPE A LINE FEED
034570 005726          TST          (SP)+      ;;CLEAN RUBOUT KEY FROM THE STACK
034572 012603          MOV          (SP)+,R3    ;;RESTORE R3
034574 011646          MOV          (SP),-(SP)  ;;ADJUST THE STACK AND PUT ADDRESS OF THE
034576 016666 000004 000002 MOV          4(SP),2(SP)  ;; FIRST ASCII CHARACTER ON IT
034604 012766 034616 000004 MOV          # $TTYIN,4(SP)
034612 000002          RTI          ;;RETURN
034614          000          9$:      .BYTE      0          ;;STORAGE FOR ASCII CHAR. TO TYPE
034615          000          .BYTE      0          ;;TERMINATOR
034616          $TTYIN: .BLKB     15          ;;RESERVE 15 BYTES FOR TTY INPUT
034635          136          103          015 $CNTLC: .ASCIZ  / ^C / <15><12>  ;;CONTROL 'C'
034642          136          125          015 $CNTLU: .ASCIZ  / ^U / <15><12>  ;;CONTROL 'U'
034647          136          107          015 $CNTLG: .ASCIZ  / ^G / <15><12>  ;;CONTROL 'G'
034654          015          012          123 $MSWR: .ASCIZ  <15><12> / SWR = /
034665          040          040          116 $MNEW: .ASCIZ  / NEW = /

2
3
4
5 034676 012737 000001 001334 CTRAP: MOV          #1,CFLAG      ;;SET THE 'CONTROL C' FLAG
6 034704 005237 033412          INC          $TKCNT      ;;COUNT THIS CHARACTER
7 034710 112777 000015 176476          MOVB      #15,@ $TKQIN  ;;PUT 'RETURN' CHARACTER IN QUEUE
8 034716 005237 033414          INC          $TKQIN      ;;UPDATE THE POINTER
9 034722 023727 033414 033427          CMP          $TKQIN,# $TKQEND ;;GO OFF THE END ?
10 034730 001003          BNE          1$          ;;BR IF YES
11 034732 012737 033420 033414          MOV          # $TKQRT,$TKQIN ;;RESET THE POINTER
12 034740 000002          1$:      RTI          ;;RETURN

```

;THIS ROUTINE WILL PROCESS THE (^C) CHARACTER

1

.SBTTL ERROR HANDLER ROUTINE

```

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

```

034742 105037 035330 $ERROR: CLR B IBSAVE          ;;CLEAR THE ITEM BYTE SAVE LOCATION
034746 104407          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
034750 010337 001316          MOV R3,ATTN          ;;SAVE THE ATTENTION REGISTER CONTENTS
034754 010137 001320          MOV R1,DRVNO          ;;DRIVE NUMBER
034760 032777 020000 144166          BIT #SW13,@SWR          ;;INHIBIT PRINTOUTS ?
034766 001004          BNE .+12          ;;BR IF YES
034770 104401 001203          TYPE ,$CRLF          ;;CR-LF
034774 104401 001203          TYPE ,$CRLF          ;;CR-LF
035000 004737 026146          JSR PC,$TIME          ;;TYPE ELAPSED TIME
035004 105237 001117          7$: INCB $ERFLG          ;;SET THE ERROR FLAG
035010 001775          BEQ 7$          ;;DON'T LET THE FLAG GO TO ZERO
035012 013777 001116 144136          MOV $STNM,@DISPLAY          ;;DISPLAY TEST NUMBER AND ERROR FLAG
035020 032777 002000 144126          BIT #BIT10,@SWR          ;;BELL ON ERROR?
035026 001402          BEQ 1$          ;;NO - SKIP
035030 104401 001176          TYPE ,SBELL          ;;RING BELL
035034 005237 001126          1$: INC $ERTTL          ;;COUNT THE NUMBER OF ERRORS
035040 011637 001132          MOV (SP),$ERRPC          ;;GET ADDRESS OF ERR. R INSTRUCTION
035044 162737 000002 001132          SUB #2,$ERRPC
035052 117737 144054 001130          MOV B @ $ERRPC,$ITEMB          ;;STRIP AND SAVE THE ERROR ITEM CODE
035060 032777 001000 144066          BIT #BIT09,@SWR          ;;SEE IF LOOP ON ERROR IS SET
035066 001060          BNE 1004$          ;;BRANCH AROUND ROUTINE IF SO
035070 122737 000177 001130          CMP B #177,$ITEMB          ;;SEE IF THIS IS THE POWER FAIL CALL
035076 001454          BEQ 1004$          ;;BRANCH AROUND ROUTINE IF IT IS
035100 105737 035330          TST B IBSAVE          ;;SEE IF THIS IS THE 2ND ERROR CALL IN THIS ROUTINE
035104 001047          BNE 1003$          ;;BRANCH IF SO
035106 022737 177777 035326          CMP #-1,CPSAVE          ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
035114 001445          BEQ 1004$          ;;BRANCH IF SO
035116 013746 000004          MOV ERRVEC,-(SP)          ;;SAVE CONTENTS OF ERROR VECTOR
035122 012737 035140 000004          MOV #1000$,ERRVEC          ;;SETUP 'TRAP' RETURN ADDRESS
035130 013737 177766 035326          MOV 177766,CPSAVE          ;;MOVE CPU ERROR REGISTER TO CPSAVE FOR TEST
035136 000406          BR 1001$
035140 012737 177777 035326 1000$: MOV #-1,CPSAVE          ;;SET CPU ERROR REGISTER TIMEOUT INDICATOR
035146 012716 035154          MOV #1001$, (SP)          ;;SETUP RETURN ADDRESS
035152 000002          RTI
035154 012637 000004          1001$: MOV (SP)+,ERRVEC          ;;RESTORE CONTENTS OF ERROR VECTOR

035160 022737 177777 035326 1002$: CMP #-1,CPSAVE          ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
035166 001420          BEQ 1004$          ;;BRANCH IF SO
035170 032737 000001 035326          BIT #BIT00,CPSAVE          ;;SEE IF POWER MONITOR BIT IS SET IN CPU ERR REG
035176 001414          BEQ 1004$          ;;BRANCH IF OK
035200 042737 000001 177766          BIC #BIT00,177766          ;;CLEAR THE BIT FOUND SET
035206 113737 001130 035330          MOV B $ITEMB,IBSAVE          ;;MAKE IBSAVE NON-ZERO FOR DUAL ERROR CALL
035214 112737 000177 001130          MOV B #177,$ITEMB          ;;SET $ITEMB TO SPECIAL POWER FAIL POINTER
035222 000402          BR 1004$          ;;BRANCH OVER IBSAVE CLEARING
    
```

```

035224 105037 035330          1003$: CLRB   IBSAVE      ::CLEAR IBSAVE SO 2ND TIME THROUGH EXITS
035230          1004$:
035230 032777 020000 143716    BIT     #BIT13,@SWR  ::SKIP TYPEOUT IF SET
035236 001004          BNE     20$          ::SKIP TYPEOUTS
035240 004737 035332          JSR     PC,$ERRTYP  ::GO TO USER ERROR ROUTINE
035244 104401 001203          TYPE   ,$CRLF
035250
035250 122737 000001 001226    20$:   CMPB   #APTENV,$ENV  ::RUNNING IN APT MODE
035256 001007          BNE     2$          ::NO,SKIP APT ERROR REPORT!
035260 113737 001130 035272    MOVB   $ITEMB,21$  ::SET ITEM NUMBER AS ERROR NUMBER
035266 004737 036676          JSR     PC,$ATY4   ::REPORT FATAL ERROR TO APT
035272          000          21$:   .BYTE  0
035273          000          .BYTE  0
035274 000777          22$:   BR      22$          ::APT ERROR LOOP
035276 105737 035330          2$:   TSTB   IBSAVE      ::SEE IF IBSAVE IS LOADED
035302 001005          BNE     3$          ::BRANCH IF NOT - NO HALT ON PWR MON BIT ERROR
035304 005777 143644          TST     @SWR        ::HALT ON ERROR
035310 100002          BPL     3$          ::SKIP IF CONTINUE
035312 000000          HALT                    ::HALT ON ERROR!
035314 104407          CKSWR                    ::TEST FOR CHANGE IN SOFT-SWR
035316
035316 105737 035330          3$:   TSTB   IBSAVE      ::SEE IF ITEM BYTE SAVE LOCATION HAS AN ERROR CALL
035322 001230          BNE     7$          ::BRANCH BACK TO CALL ORIGINAL ERROR
035324 000002          RTI                    ::RETURN
035326 000000          CPSAVE: .WORD 0      ::LOCATION TO SAVE CPU ERROR REG CONTENTS
035330 000000          IBSAVE: .WORD 0      ::LOCATION TO SAVE ITEM BYTE

```

.SBTTL ERROR MESSAGE TYPEOUT ROUTINE

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

```

035332                                $ERRTYP:
035332 104401 001203                    TYPE      , $SCLF      ;; "CARRIAGE RETURN" & "LINE FEED"
035336 010046                          MOV      RO,-(SP)    ;; SAVE RO
035340 005000                          CLR      RO        ;; PICKUP THE ITEM INDEX
035342 153700 001130                    BISB    @#$ITEMB,RO
035346 001004                          BNE     1$        ;; IF ITEM NUMBER IS ZERO, JUST
                                ;; TYPE THE PC OF THE ERROR
035350 013746 001132                    MOV      $ERRPC,-(SP) ;; SAVE $ERRPC FOR TYPEOUT
                                ;; ERROR ADDRESS
035354 104402                          TYPCC   6$        ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
035356 000437                          BR      6$        ;; GET OUT
035360 122700 000177                    1$:  CMPB   #177,RO   ;; SEE IF THIS ERROR CALL IS SPECIAL POWER FAIL CALL
035364 001006                          BNE     1000$    ;; BRANCH IF NOT
035366 013737 001212 035650            MOV      $TESTN,PFTSTN ;; GET TEST NUMBER
035374 012700 035510                    MOV      #PFECH,RO  ;; MOVE POWER FAIL ERROR CALL TABLE TO RO
035400 000406                          BR      1001$    ;; BRANCH TO CALL ERROR
035402 005300                          1000$: DEC    RO      ;; ADJUST THE INDEX SO THAT IT WILL
035404 006300                          ASL    RO        ;; WORK FOR THE ERROR TABLE
035406 006300                          ASL    RO
035410 006300                          ASL    RO
035412 062700 003360                    ADD     #$ERRTB,RO  ;; FORM TABLE POINTER
035416 012037 035426                    1001$: MOV    (RO)+,2$    ;; PICKUP "ERROR MESSAGE" POINTER
035422 001404                          BEQ    3$        ;; SKIP TYPEOUT IF NO POINTER
035424 104401                          TYPE   3$        ;; TYPE THE "ERROR MESSAGE"
035426 000000                          2$:  .WORD 0      ;; "ERROR MESSAGE" POINTER GOES HERE
035430 104401 001203                    TYPE   , $SCLF    ;; "CARRIAGE RETURN" & "LINE FEED"
035434 012037 035444                    3$:  MOV    (RO)+,4$    ;; PICKUP "DATA HEADER" POINTER
035440 001404                          BEQ    5$        ;; SKIP TYPEOUT IF 0
035442 104401                          TYPE   5$        ;; TYPE THE "DATA HEADER"
035444 000000                          4$:  .WORD 0      ;; "DATA HEADER" POINTER GOES HERE
035446 104401 001203                    TYPE   , $SCLF    ;; "CARRIAGE RETURN" & "LINE FEED"
035452 011000                          5$:  MOV    (RO),RO   ;; PICKUP "DATA TABLE" POINTER
035454 001004                          BNE     7$        ;; GO TYPE THE DATA
035456 012600                          6$:  MOV    (SP)+,RO  ;; RESTORE RO
035460 104401 001203                    TYPE   , $SCLF    ;; "CARRIAGE RETURN" & "LINE FEED"
035464 000207                          RTS     PC        ;; RETURN
035466                                7$:
035466 013046                          MOV     @ (RO)+,-(SP) ;; SAVE @ (RO)+ FOR TYPEOUT
035470 104402                          TYPCC   6$        ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
035472 005710                          TST    (RO)      ;; IS THERE ANOTHER NUMBER?
035474 001770                          BEQ    6$        ;; BR IF NO
035476 104401 035504                    TYPE   8$        ;; TYPE TWO(2) SPACES
035502 G00771                          BR      7$        ;; LOOP
035504 040 040 000 8$: .ASCIZ  / /      ;; TWO(2) SPACES

035510 035520 035602 035634 PFECH: PFECH1,PFECH2,PFECH3,PFECH4 ;; WORDS DEFINING TABLES BELOW
035520 120 117 127 PFECH1: .ASCIZ ?POWER MONITOR BIT IN CPU ERROR REGISTER FOUND SET?
035602 124 105 123 PFECH2: .ASCIZ ?TESTNO ERR PC CPUERREG?
                                .EVEN
035634 035650 001132 035326 PFECH3: .WORD PFTSTN,$ERRPC,CPSAVE,0
    
```

CZRJOAO RP07 PERF EXER MACRO V04.00 1-JAN-83 12:06:52 PAGE 44-1
ERROR MESSAGE TYPEOUT ROUTINE

G 13

SEQ 0162

035644 000 000 000 PFECH4: .BYTE 0,0,0,0
035650 000000 PFTSTN: .WORD 0

::CONTAINS TEST NUMBER FOR PF BIT ERROR

.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

```

```

035652 105737 001173 $TYPE: TSTB $TPFLG      ;; IS THERE A TERMINAL?
035656 100002          BPL 1$          ;; BR IF YES
035660 000000          HALT          ;; HALT HERE IF NO TERMINAL
035662 000430          BR 3$          ;; LEAVE
035664 0'0046          1$: MOV RO,-(SP)  ;; SAVE RO
035666 0'7600 000002  MOV @2(SP),RO  ;; GET ADDRESS OF ASCIZ STRING
035672 122737 000001 001226  CMPB #APTENV,$ENV  ;; RUNNING IN APT MODE
035700 001011          BNE 62$       ;; NO, GO CHECK FOR APT CONSOLE
035702 132737 000100 001227  BITB #APTPOOL,$ENVM ;; SPOOL MESSAGE TO APT
035710 001405          BEQ 62$       ;; NO, GO CHECK FOR CONSOLE
035712 010037 035722  MOV RO,61$    ;; SETUP MESSAGE ADDRESS FOR APT
035716 004737 036666  JSR PC,$ATY3  ;; SPOOL MESSAGE TO APT
035722 000000          61$: .WORD 0    ;; MESSAGE ADDRESS
035724 132737 000040 001227  62$: BITB #APTCSUP,$ENVM ;; APT CONSOLE SUPPRESSED
035732 001003          BNE 60$       ;; YES, SKIP TYPE OUT
035734 112046          2$: MOVB (RO)+,-(SP) ;; PUSH CHARACTER TO BE TYPED ONTO STACK
035736 001005          BNE 4$          ;; BR IF IT ISN'T THE TERMINATOR
035740 005726          TST (SP)+    ;; IF TERMINATOR POP IT OFF THE STACK
035742 012600          60$: MOV (SP)+,RO  ;; RESTORE RO
035744 062716 000002  3$: ADD #2,(SP)  ;; ADJUST RETURN PC
035750 000002          RTI          ;; RETURN
035752 122716 000011  4$: CMPB #HT,(SP)  ;; BRANCH IF <HT>
035756 001430          BEQ 8$          ;;
035760 122716 000200  CMPB #CRLF,(SP)  ;; BRANCH IF NOT <CRLF>
035764 001006          BNE 5$          ;;
035766 005726          TST (SP)+    ;; POP <CR><LF> EQUIV
035770 104401          TYPE          ;; TYPE A CR AND LF
035772 001203          $CRLF
035774 105037 036202  CLRB $CHARCNT  ;; CLEAR CHARACTER COUNT
036000 000755          BR 2$          ;; GET NEXT CHARACTER
036002 004737 0 6064  5$: JSR PC,$TYPEC  ;; GO TYPE THIS CHARACTER
036006 123726 001172  6$: CMPB $FILLC,(SP)+ ;; IS IT TIME FOR FILLER CHARS.?
036012 001350          BNE 2$          ;; IF NO GO GET NEXT CHAR.
036014 013746 001170  MOV $NULL,-(SP)  ;; GET # OF FILLER CHARS. NEEDED
                                ;; AND THE NULL CHAR.
036020 105366 000001  7$: DECB 1(SP)    ;; DOES A NULL NEED TO BE TYPED?
036024 002770          BLT 6$          ;; BR IF NO--GO POP THE NULL OFF OF STACK
036026 004737 036064  JSR PC,$TYPEC  ;; GO TYPE A NULL
036032 105337 036202  DECB $CHARCNT  ;; DO NOT COUNT AS A COUNT
036036 000770          BR 7$          ;; LOOP

```

```

;HORIZONTAL TAB PROCESSOR
036040 112716 000040      8$:   MOVB   #' (SP)      ;;REPLACE TAB WITH SPACE
036044 004737 036064      9$:   JSR    PC,$TYPEC    ;;TYPE A SPACE
036050 132737 000007 036202  BITB   #7,$CHARCNT    ;;BRANCH IF NOT AT
036056 001372          BNE    9$             ;;TAB STOP
036060 005726          TST    (SP)+          ;;POP SPACE OFF STACK
036062 000724          BR     2$             ;;GET NEXT CHARACTER
036064
036064 105777 143070      $TYPEC: TSTB   @STKS          ;;CHAR IN KYBD BUFFER?
036070 100022          BPL    10$           ;;BR IF NOT
036072 017746 143064      MOV    @STKB,-(SP)    ;;GET CHAR
036076 042716 177600      BIC    #177600,(SP)  ;;STRIP EXTRANEIOUS BITS
036102 122716 000023      CMPB   #$XOFF,(SP)  ;;WAS CHAR XOFF
036106 001012          BNE    102$         ;;BR IF NOT
036110
036110 105777 143044      101$: TSTB   @STKS          ;;WAIT FOR CHAR
036114 100375          BPL    101$         ;;BR IF NOT
036116 117716 143040      MOVB   @STKB,(SP)    ;;GET CHAR
036122 042716 177600      BIC    #177600,(SP)  ;;STRIP IT
036126 122716 000021      CMPB   #$XON,(SP)   ;;WAS IT XON?
036132 001366          BNE    101$         ;;BR IF NOT
036134
036134 005726          102$: TST    (SP)+          ;;FIX STACK
036136
036136 105777 143022      10$:   TSTB   @STPS          ;;WAIT UNTIL PRINTER IS READY
036142 100375          BPL    10$           ;;BR IF NOT
036144 116677 000002 143014  MOVB   2(SP),@STPB    ;;LOAD CHAR TO BE TYPED INTO DATA REG.
036152 122766 000015 000002  CMPB   #CR,2(SP)     ;;IS CHARACTER A CARRIAGE RETURN?
036160 001003          BNE    1$           ;;BRANCH IF NO
036162 105037 036202      CLRB   $CHARCNT     ;;YES--CLEAR CHARACTER COUNT
036166 000406          BR     $TYPEX      ;;EXIT
036170 122766 000012 000002  1$:   CMPB   #LF,2(SP)   ;;IS CHARACTER A LINE FEED?
036176 001402          BEQ    $TYPEX      ;;BRANCH IF YES
036200 105227          INCB   (PC)+       ;;COUNT THE CHARACTER
036202 000000      $CHARCNT: .WORD 0   ;;CHARACTER COUNT STORAGE
036204 000207      $TYPEX: RTS      PC
  
```

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
*OCTAL (ASCII) NUMBER AND TYPE IT.
*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
*CALL:
*   MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
*   TYPOS   ;;CALL FOR TYPEOUT
*   .BYTE  N              ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
*   .BYTE  M              ;;M=1 OR 0
*                               ;;1=TYPE LEADING ZEROS
*                               ;;0=SUPPRESS LEADING ZEROS

```

```

*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*$TYPOS OR $TYPOC

```

```

*CALL:
*   MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
*   TYPON   ;;CALL FOR TYPEOUT

```

```

*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER

```

```

*CALL:
*   MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
*   TYPOC   ;;CALL FOR TYPEOUT

```

```

036206 017646 000000
036212 116637 000001 036431
036220 112637 036433
036224 062716 000002
036230 000406
036232 112737 000001 036431
036240 112737 000006 036433
036246 112737 000005 036430
036254 010346
036256 010446
036260 010546
036262 113704 036433
036266 005404
036270 062704 000006
036274 110437 036432
036300 113704 036431
036304 016605 000012
036310 005003
036312 006105 1$:
036314 000404 BR 3$
036316 006105 2$:
036320 006105 ROL R5
036322 006105 ROL R5
036324 010503 MOV R5,R3
036326 006103 3$:
036330 105337 036432 ROL R3
036334 100016 DECB $OMODE
036336 042703 177770 BPL 7$
036342 001002 BIC #177770,R3
036344 005704 BNE 4$
036346 001403 TST R4
036350 005204 BEQ 5$
                                4$: INC R4
                                ;;DON'T SUPPRESS ANYMORE 0'S

```

```

*$TYPOS: MOV @ (SP),-(SP)      ;;PICKUP THE MODE
          MOVB 1(SP), $OFILL   ;;LOAD ZERO FILL SWITCH
          MOVB (SP)+, $OMODE+1 ;;NUMBER OF DIGITS TO TYPE
          ADD #2,(SP)         ;;ADJUST RETURN ADDRESS
          BR $TYPON
*$TYPOC: MOVB #1, $OFILL      ;;SET THE ZERO FILL SWITCH
          MOVB #6, $OMODE+1    ;;SET FOR SIX(6) DIGITS
*$TYPON: MOVB #5, $OCNT       ;;SET THE ITERATION COUNT
          MOV R3,-(SP)         ;;SAVE R3
          MOV R4,-(SP)         ;;SAVE R4
          MOV R5,-(SP)         ;;SAVE R5
          MOVB $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
          NEG R4
          ADD #6,R4            ;;SUBTRACT IT FOR MAX. ALLOWED
          MOVB R4, $OMODE      ;;SAVE IT FOR USE
          MOVB $OFILL,R4      ;;GET THE ZERO FILL SWITCH
          MOV 12(SP),R5        ;;PICKUP THE INPUT NUMBER
          CLR R3               ;;CLEAR THE OUTPUT WORD
          ROL R5               ;;ROTATE MSB INTO 'C'
          BR 3$                ;;GO DO MSB
          ROL R5               ;;FORM THIS DIGIT
          ROL R5
          MOV R5,R3
          ROL R3               ;;GET LSB OF THIS DIGIT
          DECB $OMODE          ;;TYPE THIS DIGIT?
          BPL 7$               ;;BR IF NO
          BIC #177770,R3      ;;GET RID OF JUNK
          BNE 4$               ;;TEST FOR 0
          TST R4               ;;SUPPRESS THIS 0?
          BEQ 5$               ;;BR IF YES
          INC R4               ;;DON'T SUPPRESS ANYMORE 0'S

```

036352	052703	000060		BIS	#'0,R3	::MAKE THIS DIGIT ASCII
036356	052703	000040	5\$:	BIS	#',R3	::MAKE ASCII IF NOT ALREADY
036362	110337	036426		MOVB	R3,8\$::SAVE FOR TYPING
036366	104401	036426		TYPE	8\$::GO TYPE THIS DIGIT
036372	105337	036430	7\$:	DECB	\$OCNT	::COUNT BY 1
036376	003347			BGT	2\$::BR IF MORE TO DO
036400	002402			RLT	6\$::BR IF DONE
036402	005204			INC	R4	::INSURE LAST DIGIT ISN'T A BLANK
036404	000744			BR	2\$::GO DO THE LAST DIGIT
036406	012605		6\$:	MOV	(SP)+,R5	::RESTORE R5
036410	012604			MOV	(SP)+,R4	::RESTORE R4
036412	012603			MOV	(SP)+,R3	::RESTORE R3
036414	016566	000002 000004		MOV	2(SP),4(SP)	::SET THE STACK FOR RETURNING
036422	012616			MOV	(SP)+,(SP)	
036424	000002			RTI		::RETURN
036426	000		8\$:	.BYTE	0	::STORAGE FOR ASCII DIGIT
036427	000			.RYTE	0	::TERMINATOR FOR TYPE ROUTINE
036430	000		\$OCNT:	.BYTE	0	::OCTAL DIGIT COUNTER
036431	000		\$OFILL:	.BYTE	0	::ZERO FILL SWITCH
036432	000000		\$OMODE:	.WORD	0	::NUMBER OF DIGITS TO TYPE

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```

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

```

```

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

```

```

036434          036434 010046          $TYPDS:      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
036436          036436 010146          MOV      R1,-(SP)      ;;PUSH R1 ON STACK
036440          036440 010246          MOV      R2,-(SP)      ;;PUSH R2 ON STACK
036442          036442 010346          MOV      R3,-(SP)      ;;PUSH R3 ON STACK
036444          036444 010546          MOV      R5,-(SP)      ;;PUSH R5 ON STACK
036446          036446 012746 020200    MOV      #20200,-(SP)  ;;SET BLANK SWITCH AND SIGN
036452          036452 016605 000020    MOV      20(SP),R5    ;;GET THE INPUT NUMBER
036456          036456 100004          BPL      R5           ;;BR IF INPUT IS POS.
036460          036460 005405          NEG      R5           ;;MAKE THE BINARY NUMBER POS.
036462          036462 112766 000055 000001  MOVB     #'-,1(SP)    ;;MAKE THE ASCII NUMBER NEG.
036470          036470 005000          1$:      CLR      R0           ;;ZERO THE CONSTANTS INDEX
036472          036472 012703 036650    MOV      #$DBLK,R3    ;;SETUP THE OUTPUT POINTER
036476          036476 112723 000040    MOVB     #' ,(R3)+    ;;SET THE FIRST CHARACTER TO A BLANK
036502          036502 005002          2$:      CLR      R2           ;;CLEAR THE BCD NUMBER
036504          036504 016001 036640    MOV      $DTBL(R0),R1 ;;GET THE CONSTANT
036510          036510 160105          3$:      SUB      R1,R5       ;;FORM THIS BCD DIGIT
036512          036512 002402          BLT      4$          ;;BR IF DONE
036514          036514 005202          INC      R2          ;;INCREASE THE BCD DIGIT BY 1
036516          036516 000774          BR       3$
036520          036520 060105          4$:      ADD      R1,R5       ;;ADD BACK THE CONSTANT
036522          036522 005702          TST      R2          ;;CHECK IF BCD DIGIT=0
036524          036524 001002          BNE      5$          ;;FALL THROUGH IF 0
036526          036526 105716          TSTB     (SP)        ;;STILL DOING LEADING 0'S?
036530          036530 100407          BMI      7$          ;;BR IF YES
036532          036532 106316          5$:      ASLB     (SP)        ;;MSD?
036534          036534 103003          BCC      6$          ;;BR IF NO
036536          036536 116663 000001 177777  MOVB     1(SP),-1(R3)  ;;YES--SET THE SIGN
036544          036544 052702 000060    6$:      BIS      #'0,R2    ;;MAKE THE BCD DIGIT ASCII
036550          036550 052702 000040    7$:      BIS      #' ,R2    ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
036554          036554 110223          MOVB     R2,(R3)+    ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
036556          036556 005720          TST      (R0)+      ;;JUST INCREMENTING
036560          036560 020027 000010    CMP      R0,#10     ;;CHECK THE TABLE INDEX
036564          036564 002746          BLT      2$          ;;GO DO THE NEXT DIGIT
036566          036566 003002          BGT      8$          ;;GO TO EXIT
036570          036570 010502          MOV      R5,R2      ;;GET THE LSD
036572          036572 000764          BR       6$          ;;GO CHANGE TO ASCII
036574          036574 105726          8$:      TSTB     (SP)+    ;;WAS THE LSD THE FIRST NON-ZERO?
036576          036576 100003          BPL      9$          ;;BR IF NO
036600          036600 116663 177777 177776  MOVB     -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
036606          036606 105013          9$:      CLRB     (R3)     ;;SET THE TERMINATOR
036610          036610 012605          MOV      (SP)+,R5   ;;PCP STACK INTO R5
036612          036612 012603          MOV      (SP)+,R3   ;;POP STACK INTO R3
036614          036614 012602          MOV      (SP)+,R2   ;;POP STACK INTO R2
036616          036616 012601          MOV      (SP)+,R1   ;;POP STACK INTO R1

```

036620	012600			MOV	(SP)+,R0	::POP STACK INTO R0
036622	104401	036650		TYPE	\$DBLK	::NOW TYPE THE NUMBER
036626	016666	000002	000004	MOV	2(SP),4(SP)	::ADJUST THE STACK
036634	012616			MOV	(SP)+,(SP)	
036636	000002			RTI		::RETURN TO USER
036640	023420			\$DTBL:	10000.	
036642	001750				1000.	
036644	000144				100.	
036646	000012				10.	
036650				\$DBLK:	.BLKW 4	

.SBTTL APT COMMUNICATIONS ROUTINE

```

*****
036660 112737 000001 037124 $ATY1:  MOV  #1,$FFLG      ;;TO REPORT FATAL ERFOR
036666 112737 000001 037122 $ATY3:  MOV  #1,$MFLG      ;;TO TYPE A MESSAGE
036674 000403                BR          $ATYC
036676 112737 000001 037124 $ATY4:  MOV  #1,$FFLG      ;;TO ONLY REPORT FATAL ERROR
036704                $ATYC:
036704 010046                MOV   RO,-(SP)      ;;PUSH RO ON STACK
036706 010146                MOV   R1,-(SP)      ;;PUSH R1 ON STACK
036710 105737 037122                TSTB  $MFLG        ;;SHOULD TYPE A MESSAGE?
036714 001450                BEQ   5$           ;;IF NOT: BR
036716 122737 000001 001226        CMPB  #APTENV,$ENV   ;;OPERATING UNDER APT?
036724 001031                BNE   3$           ;;IF NOT: BR
036726 132737 000100 001227        BITB  #APTPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
036734 001425                BEQ   3$           ;;IF NOT: BR
036736 017600 000004                MOV   @4(SP),RO    ;;GET MESSAGE ADDR.
036742 062766 000002 000004        ADD   #2,4(SP)    ;;BUMP RETURN ADDR.
036750 005737 001206        1$:  TST   $MSGTYPE   ;;SEE IF DONE W/ LAST XMISSION?
036754 001375                BNE   1$           ;;IF NOT: WAIT
036756 010037 001222        MOV   RO,$MSGAD   ;;PUT ADDR IN MAILBOX
036762 105720        2$:  TSTB  (RO)+      ;;FIND END OF MESSAGE
036764 001376                BNE   2$
036766 163700 001226        SUB   $MSGAD,RO   ;;SUB START OF MESSAGE
036772 006200        ASR   RO          ;;GET MESSAGE LNGTH IN WORDS
036774 010037 001224        MOV   RO,$MSGGLT  ;;PUT LENGTH IN M-ILBOX
037000 012737 000004 00:206        MOV   #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
037006 000413                BR    5$
037010 017637 000004 037034 3$:  MOV   @4(SP),4$   ;;PUT MSG ADDR IN JSR LINKAGE
037016 062766 000002 000004        ADD   #2,4(SP)    ;;BUMP RETJRN ADDRESS
037024 013746 177776        MOV   177776,-(SP) ;;PUSH 177776 ON STACK
037030 004737 035652        JSR   PC,$TYPE   ;;CALL TYPE MACRO
037034 000000        4$:  .WORD  0
037036                5$:
037036 105737 037124        10$: TSTB  $FFLG        ;;SHOULD REPORT FATAL ERROR?
037042 001416                BEQ   12$         ;;IF NOT: BR
037044 005737 001226        TST   $ENV        ;;RUNNING UNDER APT?
037050 001413                BEQ   12$         ;;IF NOT: BR
037052 005737 001206        11$: TST   $MSGTYPE   ;;FINISHED LAST MESSAGE?
037056 001375                BNE   11$         ;;IF NOT: WAIT
037060 017637 000004 001210        MOV   @4(SP),$FATAL ;;GET ERROR #
037066 062766 000002 000004        ADD   #2,4(SP)    ;;BUMP RETURN ADDR.
037074 005237 001206        INC   $MSGTYPE   ;;TELL APT TO TAKE ERROR
037100 105037 037124        12$: CLRB  $FFLG        ;;CLEAR FATAL FLAG
037104 105037 037123        CLRB  $LFLG        ;;CLEAR LOG FLAG
037110 105037 037122        CLRB  $MFLG        ;;CLEAR MESSAGE FLAG
037114 012601        MOV   (SP)+,R1    ;;POP STACK INTO R1
037116 012600        MOV   (SP)+,RO    ;;POP STACK INTO RO
037120 000207        RTS   PC         ;;RETURN
037122 000                $MFLG: .BYTE  0    ;;MESSG. FLAG
037123 000                $LFLG: .BYTE  0    ;;LOG FLAG
037124 000                $FFLG: .BYTE  0    ;;FATAL FLAG
                .EVEN
000200        APTSIZE = 200
000001        APTENV  = J01
000100        APTPOOL = 100
000040        APTCSUP = 040
  
```

.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
    
```

```

037126
037126 010046
037130 010146
037132 010246
037134 013700 037226
037140 013701 037224
037144 012702 177771
037150 006300
037152 006101
037154 005202
037156 001374
037160 063700 037226
037164 005501
037166 063701 037224
037172 062700 001057
037176 005501
037200 062701 047401
037204 010037 037226
037210 010137 037224
037214 012602
037216 012601
037220 012600
037222 000207
037224 176543
037226 123456
    
```

```

$RAND:
      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
      MOV      $LONUM,R0     ;;SET R0 WITH LOW
      MOV      $HINUM,R1     ;;SET R1 WITH HIGH
      MOV      #-7,R2        ;;SET SHIFT COUNT
1$:   ASL      R0            ;;SHIFT R0 LEFT AND
      ROL      R1            ;;ROTATE CARRY INTO R1 AND
      INC      R2            ;;CHECK FOR DONE
      BNE     1$            ;;CONTINUE SHIFT LOOP
      ADD     $LONUM,0       ;;ADD NUMBER TO MAKE X 129
      ADC     R1            ;;PROPOGATE CARRY
      ADD     $HINUM,R1      ;;ADD NUMBER TO MAKE X 129
      ADD     #1057,R0       ;;ADD LOW CONSTANT
      ADC     R1            ;;PROPOGATE CARRY
      ADD     #47401,R1      ;;ADD HIGH CONSTANT
      MOV     R0,$LONUM      ;;SAVE R0
      MOV     R1,$HINUM      ;;SAVE R1
      MOV     (SP)+,R2        ;;POP STACK INTO R2
      MOV     (SP)+,R1        ;;POP STACK INTO R1
      MOV     (SP)+,R0        ;;POP STACK INTO R0
      RTS     PC            ;;RETURN
$HINUM: .WORD 176543
$LONUM: .WORD 123456
    
```

.SBTTL SAVE AND RESTORE R0-R5 ROUTINES

```

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

```

037230
037230 010046
037232 010146
037234 010246
037236 010346
037240 010446
037242 010546
037244 016646 000022
037250 016646 000022
037254 016646 000022
037260 016646 000022
037264 000002
    
```

```

$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 R0-R5
: *CALL:
: *   RESREG
    
```

```

037266
037266 012666 000022
037272 012666 000022
037276 012666 000022
037302 012666 000022
037306 012605
037310 012604
037312 012603
037314 012602
037316 012601
037320 012600
037322 000002
    
```

```

$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      ;; THE FIRST ADDRESS OF ASCII
*      RETURN                                ;; IS ON THE STACK

```

037324	104412		\$DB2D:	SAVREG		;; SAVE REGISTERS
037326	016602	000002		MOV	2(SP), R2	;; PICKUP THE DATA POINTER
037332	012700	037504		MOV	#\$DECVL, R0	;; GET ADDRESS OF '\$DECVL' STRING
037336	010066	000002		MOV	R0, 2(SP)	;; PUT ADDRESS OF ASCII STRING ON STACK
037342	012201			MOV	(R2)+, R1	;; PICKUP THE BINARY NUMBER
037344	012202			MOV	(R2)+, R2	
037346	012737	000012	037422	MOV	#10, 4\$;; SET UP TO DO 10 CONVERSIONS
037354	012704	037434		MOV	#\$TNPWR, R4	;; ADDRESS OF TEN POWER
037360	012705	037436		MOV	#\$TNPWR+2, R5	
037364	005003		1\$:	CLR	R3	;; CLEAR PARTIAL
037366	161401		2\$:	SUB	(R4), R1	;; SUBTRACT TEN POWER
037370	005602			SBC	R2	
037372	161502			SUB	(R5), R2	
037374	002402			BLT	3\$;; BR IF TEN POWER TOO LARGE
037376	005203			INC	R3	;; ADD 1 TO PARTIAL
037400	000772			BR	2\$;; LOOP
037402	062401		3\$:	ADD	(R4)+, R1	;; RESTORE SUBTRACTED VALUE
037404	005502			ADC	R2	
037406	062402			ADD	(R4)+, R2	
037410	022525			CMP	(R5)+, (R5)+	;; MOVE TO NEXT TEN POWER
037412	052703	000060		BIS	#'0, R3	;; CHANGE PARTIAL TO ASCII
037416	110320			MOVB	R3, (R0)+	;; SAVE IT
037420	005327			DEC	(PC)+	;; DONE?
037422	000000		4\$:	.WORD	0	
037424	001357			BNE	1\$;; BR IF NO
037426	105020			CLRB	(R0)+	;; TERMINATOR
037430	104413			RESREG		;; RESTORE REGISTERS
037432	000207			RTS	PC	;; RETURN
037434	145000		\$TNPWR:	145000		;; 1.0E09
037436	035632			35632		
037440	160400			160400		;; 1.0E08
037442	002765			2765		
037444	113200			113200		;; 1.0E07
037446	000230			230		
037450	041100			041100		;; 1.0E06
037452	000017			17		
037454	103240			103240		;; 1.0E05
037456	000001			1		
037460	023420			23420		;; 1.0E04
037462	000000			0		
037464	001750			1750		;; 1.0E03
037466	000000			0		
037470	000144			144		;; 1.0E02
037472	000000			0		

037474 000012
037476 000000
037500 000001
037502 000000
037504

12
0
1
0
\$DECVL: .BLKB 12.

::1.0E01
::1.0E00
;;RESERVE STORAGE FOR ASCII STRING

.SBTTL DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE

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

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

037520	104412		\$DB20:	SAVREG		;; SAVE ALL REGISTERS
037522	016501	000002		MOV	2(SP), R1	;; PICKUP THE POINTER TO LOW WORD
037526	012705	037637		MOV	#\$OCTVL+13., R5	;; POINTER TO DATA TABLE
037532	012704	000014		MOV	#12., R4	;; DO ELEVEN CHARACTERS
037536	012703	177770		MOV	#\$C7, R3	;; MASK
037542	012100			MOV	(R1)+, R0	;; LOWER WORD
037544	012101			MOV	(R1)+, R1	;; HIGH WORD
037546	005002			CLR	R2	;; TERMINATOR
037550	110245		1\$:	MOVB	R2, -(R5)	;; PUT CHARACTER IN DATA TABLE
037552	010002			MOV	R0, R2	;; GET THIS DIGIT
037554	005304			DEC	R4	;; COUNT THIS CHARACTER
037556	003007			BGT	3\$;; BR IF NOT THE LAST DIGIT
037560	001405			BEQ	2\$;; BR IF IT IS THE LAST DIGIT
037562	005205			INC	R5	;; ALL DIGITS DONE-ADJUST POINTER FOR FIRST
037564	010566	000002		MOV	R5, 2(SP)	;; ASCII CHAR. & PUT IT ON THE STACK
037570	104413			RESREG		;; RESTORE ALL REGISTERS
037572	000207			RTS	PC	;; RETURN TO USER
037574	006203		2\$:	ASR	R3	;; POSITION THE MASK FOR THE LAST DIGIT
037576	006001		3\$:	ROR	R1	;; POSITION THE BINARY NUMBER FOR
037600	006000			ROR	R0	;; THE NEXT OCTAL DIGIT
037602	006001			ROR	R1	
037604	006000			ROR	R0	
037606	006001			ROR	R1	
037610	006000			ROR	R0	
037612	040302			BIC	R3, R2	;; MASK OUT ALL JUNK
037614	062702	000060		ADD	#'0, R2	;; MAKE THIS CHAR. ASCII
037620	000753			BR	1\$;; GO PUT IT IN THE DATA TABLE
037622			\$OCTVL:	.BLKB	14.	;; RESERVE DATA TABLE

.SBTTL POWER DOWN AND UP ROUTINES

```

037640 012737 040012 000024 $PWRDN: MOV    $ILLUP,@#PWRVEC  ;;SET FOR FAST UP
037646 012737 000340 000026      MOV    #340,@#PWRVEC+2  ;;PRIO:7
037654 010046      MOV    R0,-(SP)        ;;PUSH R0 ON STACK
037656 010146      MOV    R1,-(SP)        ;;PUSH R1 ON STACK
037660 010246      MOV    R2,-(SP)        ;;PUSH R2 ON STACK
037662 010346      MOV    R3,-(SP)        ;;PUSH R3 ON STACK
037664 010446      MOV    R4,-(SP)        ;;PUSH R4 ON STACK
037666 010546      MOV    R5,-(SP)        ;;PUSH R5 ON STACK
037670 017746      MOV    @SWR,-(SP)      ;;PUSH @SWR ON STACK
037674 010637 040016      MOV    SP,$SAVR6      ;;SAVE SP
037700 012737 037712 000024      MOV    #PWRUP,@#PWRVEC ;;SET UP VECTOR
037706 000000      HALT
037710 000776      BR     -2             ;;HANG UP
    
```

```

037712 012737 040012 000024 $PWRUP: MOV    $ILLUP,@#PWRVEC  ;;SET FOR FAST DOWN
037720 013706 040016      MOV    $SAVR6,SP      ;;GET SP
037724 005037 040016      CLR    $SAVR6         ;;WAIT LOOP FOR THE TTY
037730 005237 040016      1$:   INC    $SAVR6     ;;WAIT FOR THE INC
037734 001375      BNE    1$             ;;OF WORD
037736 005337 040020      2$:   DEC    PWRFLG     ;;TTY WAIT LOOP & SET POWER FAIL FLAG
037742 003775      BLE    2$             ;;WAIT FOR FLAG= 1
037744 012677 141204      MOV    (SP)+,@SWR     ;;POP STACK INTO @SWR
037750 012605      MOV    (SP)+,R5       ;;POP STACK INTO R5
037752 012604      MOV    (SP)+,R4       ;;POP STACK INTO R4
037754 012603      MOV    (SP)+,R3       ;;POP STACK INTO R3
037756 012602      MOV    (SP)+,R2       ;;POP STACK INTO R2
037760 012601      MOV    (SP)+,R1       ;;POP STACK INTO R1
037762 012600      MOV    (SP)+,R0       ;;POP STACK INTO R0
037764 012737 037640 000024      MOV    #PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
037772 012737 000340 000026      MOV    #340,@#PWRVEC+2 ;;PRIO:7
040000 104401      TYPE                                ;;REPORT THE POWER FAILURE
040002 040022      $PWRMG: .WORD  MSGPWR           ;;POWER FAIL MESSAGE POINTER
040004 012716      MOV    (PC)+,(SP)      ;;RESTART AT PWRUP
040006 040044      $PWRAD: .WORD  PWRUP      ;;RESTART ADDRESS
040010 000002      RTI
040012 000000      $ILLUP: HALT           ;;THE POWER UP SEQUENCE WAS STARTED
040014 000776      BR     -2             ;; BEFORE THE POWER DOWN WAS COMPLETE
040016 000000      $SAVR6: 0             ;;PUT THE SP HERE

2
3 040020 000000      PWRFLG: .WORD  0             ;;POWER FAIL FLAG GOES HERE; 'LED= 1
4
5 040022 200 012 042  MSGPWR: .ASCIZ <CRLF><LF>/'POWER UP' AT /
6
7
8
9
10
11 040044 005227 000000      PWRUP:  INC    #0             ;;TTY LOOP, WAIT FOR INCREMENT
12 040050 001375      BNE    -4             ;;OF WORD
13 040052 000005      RESET
14 040054 005037 177776      CLR    PS             ;;CLEAR PSW
    
```

;;THIS ROUTINE HANDLES THE RETURN ON POWER UP. WAIT THREE (3) MINUTES AND
 ;;DO AN AUTO RE-START TO 'SIZMEM'.

15	040060	012706	001100			MOV	#STACK,SP	:SETUP STACK POINTER
16	040064	005037	001344			CLR	SECOND	:RESET SECOND COUNT
17	040070	005037	001472			CLR	INTRVL+2	:RESET THE INTERVAL COUNT
18	040074	004737	033430			JSR	PC,\$TKINT	:MAKE SURE KEYBOARD INTERRUPT AND
19	040100	004737	024514			JSR	PC,CKCLK	:SYSTEM CLOCK ARE STILL ON.
20	040104	004737	026146			JSR	PC,\$TIME	:TYPE ELAPSED TIME
21	040110	104401	001203			TYPE	,\$CRLF	:CR-LF
22	040114	005037	001334			CLR	CFLAG	:CLEAR (^C) FLAG
23	040120	105737	001542			TSTB	ASNLST	:ANY DRIVES ASSIGNED ?
24	040124	001414				BEQ	2\$:BR IF NO
25	040126	104401	040176			TYPE	,\$MSWAIT	:TYPE 'WAITING 3 MINUTES...TO START'
26	040132	005737	001334		1\$:	TST	CFLAG	:WAS (^C) TYPED?
27	040136	001007				BNE	2\$:BR IF YES
28	040140	023727	001472	000003		CMP	INTRVL+2,#3	:THREE MINUTES YET ?
29	040146	002771				BLT	1\$:WAIT IF NOT
30	040150	012737	000400	001332		MOV	#400,CHGADR	:FUDGE 200 START
31	040156	012705	001520		2\$:	MOV	#ORDERQ,R5	:CLEAR UP THE QUE AND BUFFER
32	040162	005025			3\$:	CLR	(R5)+	
33	040164	022705	002056			CMP	#BLKADR,R5	:ALL DONE ?
34	040170	001374				BNE	3\$:BRANCH IF NOT
35	040172	000137	005112			JMP	SIZMEM	:LOOP BACK
36								
37	040176	200	124	131	MSWAIT:	.ASCII	<CRLF>/TYPE ^C TO ABORT/	
38	040217	200	127	101		.ASCIIZ	<CRLF>/WAITING 3 MINUTES...TO START/<CRLF>	
39						.EVEN		

1

.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.

040256	010046			\$TRAP: MOV	R0,-(SP)	::SAVE R0	
040260	016600	000002			MOV	2(SP),R0	::GET TRAP ADDRESS
040264	005740				TST	-(R0)	::BACKUP BY 2
040266	111000				MOVB	(R0),R0	::GET RIGHT BYTE OF TRAP
040270	006300				ASL	R0	::POSITION FOR INDEXING
040272	016000	040312			MOV	\$TRPAD(R0),R0	::INDEX TO TABLE
040276	000200				RTS	R0	::GO TO ROUTINE

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

040300	011646			\$TRAP2: MOV	(SP),-(SP)	::MOVE THE PC DOWN	
040302	016666	000004	000002		MOV	4(SP),2(SP)	::MOVE THE PSW DOWN
040310	000002				RTI		::RESTORE THE PSW

.SBTTL TRAP TABLE

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

					ROUTINE	

040312	040300			\$TRPAD: .WORD	\$TRAP2	
040314	035652				\$TYPE	::CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
040316	036232				\$TYPOC	::CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
040320	036206				\$TYPOS	::CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
040322	036246				\$TYPON	::CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
040324	036434				\$TYPDS	::CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
040326	034020				\$GTSWR	::CALL=GTSWR TRAP+6(104406) GET SOFT-SWR SETTING
040330	033730				\$CKSWR	::CALL=CKSWR TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
040332	034272				\$RDCHR	::CALL=RDCHR TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
040334	034362				\$RDLIN	::CALL=RDLIN TRAP+11(104411) TTY TYPEIN STRING ROUTINE
040336	037230				\$SAVREG	::CALL=SAVREG TRAP+12(104412) SAVE R0-R5 ROUTINE
040340	037266				\$RESREG	::CALL=RESREG TRAP+13(104413) RESTORE R0-R5 ROUTINE
2 040342	033020				\$DSPLY	::CALL=DISPLY TRAP+14(104414) ROUTINE TO TYPE ERROR MESSAGES
3	000032			\$TERM=.	-\$TRPAD	

```

4          .SBTTL  RP07 DRIVER
5
6          ;STORAGE FOR RPDS, RPER1, RPER2, AND RPER3
13
16 040344 000000 000000 000000 RPSTU0: .WORD 0,0,0,0      ;DS, ER1, ER2 & ER3 STORAGE FOR DRIVE 0
    040354 000000 000000 000000 RPSTU1: .WORD 0,0,0,0      ;DS, ER1, ER2 & ER3 STORAGE FOR DRIVE 1
    040364 000000 000000 000000 RPSTU2: .WORD 0,0,0,0      ;DS, ER1, ER2 & ER3 STORAGE FOR DRIVE 2
    040374 000000 000000 000000 RPSTU3: .WORD 0,0,0,0      ;DS, ER1, ER2 & ER3 STORAGE FOR DRIVE 3
    040404 000000 000000 000000 RPSTU4: .WORD 0,0,0,0      ;DS, ER1, ER2 & ER3 STORAGE FOR DRIVE 4
    040414 000000 000000 000000 RPSTU5: .WORD 0,0,0,0      ;DS, ER1, ER2 & ER3 STORAGE FOR DRIVE 5
    040424 000000 000000 000000 RPSTU6: .WORD 0,0,0,0      ;DS, ER1, ER2 & ER3 STORAGE FOR DRIVE 6
    040434 000000 000000 000000 RPSTU7: .WORD 0,0,0,0      ;DS, ER1, ER2 & ER3 STORAGE FOR DRIVE 7
17
18          ;TABLE OF DRIVE ACTIVE INDICATORS (DRVACT=8 BYTES)
19          ;DRVACT=0 IF DRIVE IS IDLE
20          ;DRVACT>0 IF DRIVE IS ACTIVE WITH A COMMAND
21          ;DRVACT<0 IF DRIVE IS ACTIVE WITH AN ERROR RECOVERY OPERATION
22
23 040444      000          DRVACT: .BYTE 0          ;DRIVE 0
24 040445      000          .BYTE 0          ;DRIVE 1
25 040446      000          .BYTE 0          ;DRIVE 2
26 040447      000          .BYTE 0          ;DRIVE 3
27 040450      000          .BYTE 0          ;DRIVE 4
28 040451      000          .BYTE 0          ;DRIVE 5
29 040452      000          .BYTE 0          ;DRIVE 6
30 040453      000          .BYTE 0          ;DRIVE 7
31
32          ;TABLE OF DRIVE STATUS INDICATORS (DRVSTA=8 BYTES)
33          ;DRVSTA=0 IF DRIVE IS OFFLINE OR NONEXISTENT
34          ;DRVSTA>0 IF DRIVE IS ONLINE
35          ;DRVSTA<0 IF DRIVE IS UNSAFE
36
37 040454      000          DRVSTA: .BYTE 0          ;DRIVE 0
38 040455      000          .BYTE 0          ;DRIVE 1
39 040456      000          .BYTE 0          ;DRIVE 2
40 040457      000          .BYTE 0          ;DRIVE 3
41 040460      000          .BYTE 0          ;DRIVE 4
42 040461      000          .BYTE 0          ;DRIVE 5
43 040462      000          .BYTE 0          ;DRIVE 6
44 040463      000          .BYTE 0          ;DRIVE 7
45
46          ;TABLE OF DRIVE TYPES (DRVTYP=8 BYTES)
47          ;DRVTYP= 0 IF DRIVE IS NONEXISTENT (DRVSTA=0, ALSO)
48          ;DRVTYP= 4 IF DRIVE IS RP07+
49          ;DRVTYP= 5 IF DRIVE IS RP07 MOVING HEAD OPTION
50          ;DRVTYP=-1 IF NOT RP07
51
52 040464      000          DRVTYP: .BYTE 0          ;DRIVE 0
53 040465      000          .BYTE 0          ;DRIVE 1
54 040466      000          .BYTE 0          ;DRIVE 2
55 040467      000          .BYTE 0          ;DRIVE 3
56 040470      000          .BYTE 0          ;DRIVE 4
57 040471      000          .BYTE 0          ;DRIVE 5
58 040472      000          .BYTE 0          ;DRIVE 6
59 040473      000          .BYTE 0          ;DRIVE 7
60
61          ;TABLE OF DUAL PORT INITIALIZATION INDICATORS
  
```



```

119 040522 000000      SAVEFG: .WORD  0
120
121                   ;SEEK FLAG (SEEKFG=1 WORD)
122                   ;SEEKFG=0 IF WHEN THE DISK ADDRESS ISN'T IN THE WINDOW
123                   ;FOR A DATA TRANSFER START A SEARCH COMMAND
124                   ;SEEKFG<0 IF DATA TRANSFER WILL DO IMPLIED SEEKS,
125                   ;DISREGARD THE WINDOW
126
127 040524 177777      SEEKFG: .WORD  -1
128
129                   ;TIMEOUT TABLE (TIMER=8 WORDS)
130                   ;THIS TABLE CONTAINS THE TIME ALLOWED FOR AN OPERATION
131
132 040526 177777      TIMER:  .WORD  -1      ;DRIVE 0
133 040530 177777      .WORD  -1      ;DRIVE 1
134 040532 177777      .WORD  -1      ;DRIVE 2
135 040534 177777      .WORD  -1      ;DRIVE 3
136 040536 177777      .WORD  -1      ;DRIVE 4
137 040540 177777      .WORD  -1      ;DRIVE 5
138 040542 177777      .WORD  -1      ;DRIVE 6
139 040544 177777      .WORD  -1      ;DRIVE 7
140
141                   ;DATA TRANSFER UNDERWAY INDICATOR (DTUW=1 WORD)
142                   ;DTUW<0 IF NO DATA TRANSFER UNDERWAY
143                   ;DTUW=+N (WHERE N=0 TO 7) IMPLIES DATA TRANSFER UNDERWAY ON DRIVE N
144
145 040546 177777      DTUW:  .WORD  -1
146
147                   ;ATTENTION BITS TABLE (ATABIT=8 BYTES)
148                   ;THIS TABLE CONTAINS THE CORRESPONDING BIT TO EACH DRIVES
149                   ;ATTENTION BIT
150
151 040550      001      ATABIT: .BYTE  1      ;DRIVE 0
152 040551      002      .BYTE  2      ;DRIVE 1
153 040552      004      .BYTE  4      ;DRIVE 2
154 040553      010      .BYTE 10      ;DRIVE 3
155 040554      020      .BYTE 20      ;DRIVE 4
156 040555      040      .BYTE 40      ;DRIVE 5
157 040556      100      .BYTE 100     ;DRIVE 6
158 040557      200      .BYTE 200     ;DRIVE 7
159
160                   ;STORAGE FOR RPADR (THE FIRST ADDRESS (776700) OF THE RHXX/RP07),
161                   ;RPVEC (THE VECTOR ADDRESS (254)), AND RPVEC+2 (THE BR LEVEL (5)).
162
163 040560 176700      RPADR:  .WORD 176700      ;RHXX/RP07 CONTROL STATUS REGISTER
164 040562 000254 000240 RPVEC:  .WORD 254,5*32.      ;VECTOR ADDRESS & BR LEVEL 5
165 040566 000050      RHEXT:  .WORD 50      ;OFFSET TO RPBAE
166
167                   ;SEARCH DIFFERENCE IS 1 SECTOR
168
169 040570 000001      MXWNDW: .WORD  1
170
171                   ;DEFINITIONS OF THE RHXX/RP07 ADDRESS INDEXES
172
173      000000      RPCS1  = 0      ;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 00)
174      000002      RPWC   = 2      ;WORD COUNT REGISTER (NOT A DRIVE REG)
175      000004      RPBA   = 4      ;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)

```

176	000006	RPDA	= 6	:DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 05)
177	000010	RPCS2	= 10	:CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
178	000012	RPDS	= 12	:DRIVE STATUS REGISTER (DRIVE REG 01)
179	000014	RPER1	= 14	:ERROR REGISTER #1 (DRIVE REG. 02)
180	000016	RPAS	= 16	:ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 04)
181	000020	RPLA	= 20	:LOOK AHEAD REGISTER (DRIVE REG. 07)
182	000022	RPDB	= 22	:DATA BUFFER REGISTER (NOT A DRIVE REG.)
183	000024	RPMR1	= 24	:MAINTAINABILITY REGISTER (DRIVE REG. 03)
184	000026	RPDT	= 26	:DRIVE TYPE REGISTER (DRIVE REG. 06)
185	000030	RPSN	= 30	:SERIAL NUMBER REGISTER (DRIVE REG. 10)
186	000032	RPOF	= 32	:OFFSET REGISTER (DRIVE REG. 11)
187	000034	RPDC	= 34	:DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
188	000036	RPCC	= 36	:DUMMY ADDRESS REGISTER (DRIVE REG. 13)
189	000040	RPER2	= 40	:ERROR REGISTER #2
190	000042	RPER3	= 42	:ERROR REGISTER #2 (DRIVE REG. 15)
191	000044	RPEC1	= 44	:ECC POSITION REGISTER (DRIVE REG. 16)
192	000046	RPEC2	= 46	:ECC PATTERN REGISTER (DRIVE REG. 17)
193	000050	RPBAE	= 50	:BUS ADDRESS EXTENTION REGISTER
194	000052	RPCS3	= 52	:CONTROL AND STATUS REGISTER #3
195				

```

1      ;RHXX/RP07 DRIVER INITIALIZATION CODE
2      ;THIS ROUTINE WILL DETERMINE WHICH RP07 DRIVES ARE
3      ;AVAILABLE FOR TESTING AND SET THE DRVSTA INDICATOR
4      ;TO THE PROPER STATE FOR EACH DRIVE.
5      ;NOTE: THIS ROUTINE CALLS DRVINT
6
7      ;CALL
8
9      JSR    PC,RPINIT
10     RETURN
11
12     ;NOTE: THE 'P' OR 'L' CLOCK MUST BE STARTED
13
14     RPINIT: SAVREG          ;SAVE R0 - R5
15     MOV    @#PS, -(SP)      ;SAVE THE PRESENT PROCESSOR STATUS
16     MOV    #<5*32.>, @#PS   ;CHANGE THE PRIORITY TO 5
17     JSR    PC, CLRQUE      ;CLEAR ALL REQUEST QUEUES
18     MOV    #RPSTUO, R1     ;FIRST ADDRESS TO BE CLEARED
19     MOV    #SEEKFG, R2     ;LAST ADDRESS TO BE CLEARED
20     CLR    (R1)+           ;CLEAR
21     CMP    R1, R1         ;ARE WE DONE?
22     BLO   1$             ;BRANCH IF NO
23
24     MOV    #DTUW, R2       ;LAST ADDRESS
25     MOV    #-1, (R1)+     ;INITIALIZE
26     CMP    R1, R2         ;DONE?
27     BLOS  2$             ;LOOP IF NO
28
29     CLR    DRVSTA          ;SET ALL DRIVES TO OFFLINE
30     CLR    DRVSTA+2
31     CLR    DRVSTA+4
32     CLR    DRVSTA+6
33     MOV    RPVEC, R3       ;SETUP THE RHXX/RP07 VECTOR
34     MOV    #ISR, (R3)+
35     MOV    RPVEC+2, (R3)
36     MOV    RPADR, R4      ;FIRST ADDRESS OF RHXX/RP07
37     MOV    #BIT05, R4     ;MASSBUS INIT
38     CLR    R1             ;START WITH DRIVE 0
39     JSR    R0, DRVINT     ;INIT THE DRIVE
40     BR    4$             ;'DVA' NOT SET OR PARITY ERROR
41     BR    5$             ;NORMAL RETURN
42
43     CLRB   DRVSTA(R1)     ;SET DRIVE STATUS TO OFFLINE
44     INC    R1             ;GO TO NEXT DRIVE
45     BIC    #*(7, R1)      ;MASK OUT UNUSED BITS
46     BNE   3$             ;BR IF MORE DRIVES TO GO
47     MOV    (SP)+, @#PS    ;RESTORE THE PROCESSOR STATUS
48     RESREG          ;RESTORE R0 - R5
49     RTS    PC            ;BYE-BYE
  
```



```

1      ;DRIVE INITIALIZATION ROUTINE
2      ;THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
3      ;AN RP07. IF IT IS, A "READ-IN PRESET" IS ISSUED AND FMT22
4      ;IS SET TO A "1". THEN MOL, DPR, DRY, AND VV ARE CHECKED TO
5      ;INSURE THEY ARE ALL ON A "1". AND DEPENDING ON THEIR STATE,
6      ;DRVSTA IS SET TO THE PROPER CONDITION.
7
8      ;CALL
9      :CALL
10     MOV      RPADR,R4      ;UNIBUS ADDRESS OF RHXX/RP07 (RPCS1)
11     MOV      #DRVNUM,R1   ;DRIVE NUMBER
12     JSR      RO,DRVINT    ;CALLED BY A JSR
13     RETURN1  ;ERROR OCCURRED (PARITY)
14     RETURN2  ;NORPAL RETURN
15
16     040750 010546
17     040752 112761 177777 040474
18     040760 006301
19     040762 012761 023420 040526
20     040770 006201
21
22     040772 105061 040454
23     040776 105061 040464
24     041002 010164 000010
25     041006 112764 000111 000000
26     041014 032764 010000 000010
27     041022 001403
28     041024 004737 045426
29     041030 000513
30
31     041032 105061 040454
32     041036 032764 004000 000000
33     041044 001004
34     041046 105761 040474
35     041052 001347
36     041054 000501
37
38     041056 004037 045036
39     041062 000026
40     041064 041262
41     041066 012605
42     041070 112761 000005 040464
43     041076 022705 020040
44     041102 001420
45     041104 022705 024040
46     041110 001415
47     041112 112761 000004 040464
48     041120 022705 020042
49     041124 001407
50     041126 022705 024042
51     041132 001404
52     041134 112761 177777 040464
53     041142 000446
54
55     041144 012746 000121
56     041150 004037 045130
57     041154 000000
58     041156 041262
59
60     DRVINT: MOV      R5,-(SP)      ;SAVE R5
61             MOVVB   #-1,DPINT(R1) ;SET THE DUAL PORT INITIALIZE FLAG
62             ASL     R1
63             MOV     #10000.,TIMER(R1) ;START 10. SECOND TIMER
64             ASR     R1             ;DRIVE ADDRESS
65
66     1$:     CLR     DRVSTA(R1)      ;START DRIVE STATUS AS OFFLINE
67             CLR     DRVSTYP(R1)   ;CLEAR THE DRIVE TYPE INDICATOR
68             MOV     R1,RPCS2(R4)  ;SELECT A DRIVE
69             MOVVB  #111,RPCS1(R4) ;DO A DRIVE CLEAR COMMAND (& SEIZE DRIVE)
70             BIT     #BIT12,RPCS2(R4) ;NONEXISTENT DRIVE?
71             BEQ     2$             ;NO---BRANCH
72             JSR    PC,SET.IE      ;GO SET "IE" WITHOUT A "TRE"
73             BR     6$             ;LEAVE THIS ROUTINE
74
75     2$:     CLR     DRVSTA(R1)      ;SET DRIVE STATUS TO OFFLINE
76             BIT     #BIT11,RPCS1(R4) ;SEE IF DRIVE AVAILABLE
77             BNE    3$             ;BRANCH IF DVA SET
78             TSTB   DPINT(R1)     ;SOFTWARE TIMEOUT ON DUAL PORT INITIALIZE ?
79             BNE    1$             ;BRANCH IF NOT
80             BR     6$             ;OTHERWISE EXIT
81
82     3$:     JSR     RO,RD.RP        ;READ THE DRIVE TYPE REG.
83             RPDT   8$
84             MOV     (SP)+,R5      ;ERROR RETURN ADDRESS
85             MOVVB  #5,DRVSTYP(R1) ;PUT DRIVE TYPE IN R5
86             CMP    #20040,R5     ;SET RP07 INDICATOR
87             BEQ    4$             ;SINGLE PORT RP07
88             BEQ    4$             ;BR IF YES
89             CMP    #24040,R5     ;DUAL PORT RP07
90             BEQ    4$             ;BR IF YES
91             MOVVB  #4,DRVSTYP(R1) ;SET RP07+ INDICATOR
92             CMP    #20042,R5     ;SINGLE PORT RP07+
93             BEQ    4$             ;BRANCH IF SO
94             CMP    #24042,R5     ;DUAL PORT RP07+
95             BEQ    4$             ;BRANCH IF SO
96             MOVVB  #-1,DRVSTYP(R1) ;SET INDICATOR TO 'OTHER'
97             BR     6$             ;EXIT
98
99     4$:     MOV     #121,-(SP)     ;DO A "READ-IN PRESET"
100            JSR    RO,WRT.RP
101            RPCS1 8$

```

```

RP07 DRIVER
58 041160 012746 010000      MOV      #BIT12,-(SP)      ;SET FMT16=1
59 041164 004037 045130      JSR      RO,WRT.RP
60 041170 000032              RPOF
61 041172 041262              8$
62 041174 004037 045036      JSR      RO,RD.RP        ;READ RPDS
63 041200 000012              RPDS
64 041202 041262              8$
65 041204 012605              MOV      (SP)+,R5        ;AND SAVE IT IN R5
66 041206 100015              BPL      5$              ;BRANCH IF ATA=0
67 041210 116164 040550 000016  MOVVB   ATABIT(R1),RPAS(R4) ;CLEAR ATTENTION BIT
68 041216 004037 045036      JSR      RO,RD.RP        ;FIND OUT WHY ATA=1
69 041222 000014              RPER1
70 041224 041262              8$
71 041226 006126              ROL      (SP)+          ;IS IT UNSAFE?
72 041230 100004              BPL      5$              ;BR IF NOT
73 041232 112761 177777 040454  MOVVB   #-1,DRVSTA(R1)   ;SET UNSAFE INDICATOR
74 041240 000407              BR       6$              ;EXIT
75
76 041242 005105              5$:     COM      R5          ;CHECK MOL, DPR, DRY, AND VV
77 041244 042705 167077      BIC     #^C<BIT12!BIT08!BIT07!BIT06>,R5
78 041250 001003              BNE     6$              ;BRANCH IF MOL, DPR, DRY, OR VV IS CLEAR
79 041252 112761 000001 040454  MOVVB   #1,DRVSTA(R1)   ;SET DRIVE STATUS TO ONLINE
80 041260 005720              6$:     TST      (R0)+      ;STEP OVER THE ERROR RETURN
81 041262              7$:
82 041262 006301              8$:     ASL      R1          ;WORD INDEX
83 041264 012761 177777 040526  MOV     #-1,TIMER(R1)   ;STOP THE CLOCK
84 041272 006201              ASR     R1              ;DRIVE ADDRESS
85 041274 105061 040474      CLRB   DPINT(R1)
86 041300 012605              MOV     (SP)+,R5        ;RESTORE R5
87 041302 000200              RTS     RO              ;EXIT
88
89              ;REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
90
91              ;CALL
92
93              ;
94              ;
95              ;
96              ;
97              ;
98              ;
99 041304 013746 177776      RP07:   MOV     @#PS,-(SP)   ;SAVE THE CALLING STATUS
100 041310 013737 040564 177776  MOV     RPVEC+2,@#PS    ;DON'T ALLOW ANY RP07 INTERRUPTS
101 041316 112737 000001 040520  MOVVB   #1,ACTDRV      ;SET "ACTIVE DRIVER" FLAG
102 041324 104412              SAVREG                ;SAVE R0 - R5
103 041326 011002              MOV     (R0),R2        ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
104 041330 005062 000016      CLR     16(R2)         ;CLEAR THE STATUS/ERROR INDICATOR
105 041334 111201              MOVVB   (R2),R1        ;PICKUP THE DRIVE NUMBER
106 041336 013704 040560      MOV     RPADR,R4       ;UNIBUS ADDRESS OF RPCS1
107 041342 105761 040454      TSTB   DRVSTA(R1)     ;CHECK DRIVES STATUS
108 041346 003006              BGT     1$            ;BRANCH IF ONLINE
109 041350 004037 040750      JSR     RO,DRVINT      ;GO INIT. THE DRIVE
110 041354 000421              BR      3$            ;ERROR RETURN
111
112 041356 105761 040454      TSTB   DRVSTA(R1)     ;IS DRIVE STATUS ONLINE?
113 041362 003435              BLE     5$            ;BR IF NOT
114 041364 105761 040504      1$:    TSTB   DPRQS(R1)     ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?

```

```

115 041370 001016          BNE      4$          ;BR IF YES
116 041372 010164 000010  MOV      R1,RPCS2(R4) ;SELECT THE DRIVE
117 041376 004037 046070  JSR      RO,DRVQUE   ;PUT THIS REQUEST IN QUEUE
118 041402 000450          BR       8$          ;QUEUE IS FULL
119
120 041404 105761 040444  2$:     TSTB     DRVACT(R1) ;IS THIS DRIVE ACTIVE?
121 041410 001042          BNE      7$          ;BR IF YES
122 041412 004737 041542  JSR      PC,OPT      ;CALL THE OPTIMIZER
123 041416 000437          BR       7$
124
125 041420 004737 042670  3$:     JSR      PC,C!7   ;GO HANDLE THE PARITY ERROR
126 041424 000434          BR       7$
127
128 041426 004037 046070  4$:     JSR      RO,DRVQUE ;PUT REQUEST IN QUEUE
129 041432 000434          BR       8$          ;QUEUE IS FULL
130
131 041434 012764 000000 000036  MOV      #0,RPCC(R4)  ;WRITE THE CURRENT CYL REG
132 041442 032714 000100          BIT      #BIT06,(R4) ;IE BIT SET ?
133 041446 001023          BNE      7$          ;YES
134 041450 004737 045426  JSR      PC,SET.IE   ;SET THE INTERRUPT
135 041454 000420          BR       7$          ;RETURN
136
137 041456 105761 040454  5$:     TSTB     DRVSTA(R1) ;SEE IF DRIVE OFFLINE OR UNSAFE
138 041462 002412          BLT      6$          ;BR IF UNSAFE
139 041464 012762 140000 000016  MOV      #BIT15!BIT14,16(R2) ;SET OFFLINE ERROR INDICATOR
143 041472 105761 040464  TSTB     DRVTYP(R1)  ;SEE IF OFFLINE OR NONEXISTENT
144 041476 001007          BNE      7$          ;BR IF OFFLINE
145 041500 012762 100002 000016  MOV      #BIT15!BIT01,16(R2) ;REPORT DRIVE NONEXISTENT
146 041506 000403          BR       7$          ;GO TO EXIT
147
148 041510 012762 110000 000016  6$:     MOV      #BIT15!BIT12,16(R2) ;DRIVE IS UNSAFE
149 041516 104413          7$:     RESREG   ;RESTORE R0 - R5
150 041520 005720          TST      (R0)+       ;SETUP FOR NORMAL RETURN
151 041522 000401          BR       9$          ;FINISH UP, THEN EXIT
152
153 041524 104413          8$:     RESREG   ;RESTORE R0 - R5
154 041526 005720          9$:     TST      (R0)+       ;CORRECT THE RETURN ADDRESS
155 041530 105037 040520  CLRB    ACTDRV      ;CLEAR "ACTIVE DRIVER" FLAG
156 041534 012637 177776  MOV      (SP)+,@#PS  ;RETURN "PS" TO USER LEVEL
157 041540 000200          RTS      RO          ;RETURN TO CALLER
158
159          ;OPTIMIZER-CALLED FOR A PARTICULAR DRIVE
160          ;CALL
161          ;
162          ;     MOV      #DPB,R2          ;ADDRESS OF DRIVE PARAMETER BLOCK
163          ;     MOV      #DRVNUM,R1      ;DRIVE NUMBER
164          ;     JSR      PC,OPT          ;SETUP A COMMAND
165          ;
166 041542 104412          OPT:   SAVREG   ;SAVE R0 - R5
167 041544 013746 177776  MOV      @#PS,-(SP)  ;SAVE PROC. STATUS
168 041550 146137 040550 040516  JICB    ATABIT(R1),SRCHWT ;CLEAR SEARCH FLAG
169 041556 105061 040504  CLRB    DPRQS(R1)   ;RESET THE PORT REQ FLAG
170 041562 004737 046144  JSR      PC,GETREQ  ;GET "DPB" POINTER OF REQUEST
171 041566 005702          TST      R2          ;IS THERE A REQUEST IN QUEUE?
172 041570 001475          BEQ     8$          ;NO--BRANCH TO EXIT
173 041572 010164 000010  MOV      R1,RPCS2(R4) ;LOAD THE DRIVE ADDRESS
174 041576 012764 000111 000000  MOV      #11,RPCS1(R4) ;CLEAR THE DRIVE
  
```

```

175 041604 032764 004000 000000          BIT      #BIT11,RPCS1(R4)          ,DVA SET ?
176 041612 001446                          BEQ      6$                      ;TO PROT REQUEST ,IF NOT
177 041614 105761 040454          1$:    TSTB   DRVSTA(R1)          ;IS DRIVE ONLINE?
178 041620 003014                          BGT      2$                      ;YES--BRANCH
179 041622 004737 046166          JSR      PC,POPQUE              ;NO--REMOVE REQUEST FROM QUEUE
180 041626 012762 140000 000016          MOV      #BIT15!BIT14,16(R2)    ;SET OFFLINE STATUS/ERROR INDICATOR
181 041634 105761 040454          TSTB   DRVSTA(R1)          ;IS DRIVE UNSAFE ?
182 041640 100056                          BPL      9$                      ;BR TO EXIT IF NOT
183 041642 012762 110000 000016          MOV      #BIT15!BIT12,16(R2)    ;SET UNSAFE STATUS/ERROR INDICATOR
184 041650 000452                          BR       9$                      ;BRANCH TO EXIT
185
186 041652 122762 000150 000002 2$:    CMPB   #150,2(R2)          ;IS THE REQUEST FOR I/O?
187 041660 002407                          BLT      3$                      ;YES--BRANCH
188 041662 122762 000135 000002          CMPB   #135,2(R2)          ;IS IT A DIAGNOSTIC COMMAND ?
189 041670 001403                          BEQ      3$                      ;BRANCH IF SO
190 041672 004737 042312          JSR      PC,C14              ;CALL THE COMMAND INITIATOR
191 041676 000437                          BR       9$                      ;BRANCH TO EXIT
192
193 041700 005737 040546          3$:    TST      DTUW              ;DATA TRANSFER UNDERWAY?
194 041704 002006                          BGE      5$                      ;BR IF YES
195 041706 005737 040524          TST      SEEKFG            ;DO IMPLIED SEEKS ?
196 041712 100003                          BPL      5$                      ;NO, DO SEARCH
197 041714 004737 042006          4$:    JSR      PC,C11              ;START A DATA TRANSFER
198 041720 000426                          BR       9$
199
200 041722 004737 042200          5$:    JSR      PC,C13              ;START A SEARCH ON TARGET SECTOR -1
201 041726 000423                          BR       9$                      ;GO TO THE EXIT
202
203 041730 112761 177777 040504 6$:    MOVWB  #-1,DPROS(R1)          ;SET PORT REQUEST INDICATOR
204 041736 010103                          MOV      R1,R3              ;SET UP TO ADDRESS WORDS
205 041740 006303                          ASL      R3                  ;CONVERT TO WORD INDEX
206 041742 012763 035230 040526          MOV      #15000.,TIMER(R3)    ;START 15. SECOND TIMER
207 041750 012764 000000 000036          MOV      #0,RPCC(R4)          ;SET PORT REQUEST
208 041756 000402                          BR       8$                      ;EXIT
209
210 041760 004737 042670          7$:    JSR      PC,C17              ;PROCESS THE PARITY ERROR
211 041764 032714 000100          8$:    BIT      #BIT06,(R4)          ;SEE IF 'IE' ALREADY SET
212 041770 001002                          BNE      9$                      ;BR IF SET
213 041772 004737 045426          JSR      PC,SET.IE          ;SET 'IE' WITHOUT A 'TRE'
214 041776 012637 177776          9$:    MOV      (SP)+,@#PS      ;RESTORE PROC. STATUS
215 042002 104413                          RESREG  ;RESTORE R0 - R5
216 042004 000207                          RTS      PC
217
218          ;COMMAND INITIATOR
219
220          ;CALL
221          ;
222          ;   MOV      #DPB,R2          ;ADDRESS OF DRIVE PARAMETER BLOCK
223          ;   MOV      #DRVNUM,R1      ;DRIVE NUMBER
224          ;   JSR      PC,C1????      ;C1???? = C11, C13, OR C14
225          ;   ;WHERE:
226          ;   ; C11 = DATA TRANSFER
227          ;   ; C13 = SEARCH REQUESTED BY DATA XFER
228          ;   ; C14 = NO DATA TRANSFER
229 042006 004737 046166          C11:  JSR      PC,POPQUE          ;REMOVE REQUEST FROM 'DRIVES WAIT' QUEUE
230 042012 010237 040514          MOV      R2,TRNSWT          ;PUT REQ. IN TRANSFER WAIT QUEUE
231 042016 010203          MOV      R2,R3              ;DPB ADDRESS TO R3

```

232	042020	013704	040560		MOV	RPADR,R4	:RPCS1 ADDRESS	
233	042024	010164	000010		MOV	R1,RP(S2(R4)	:SELECT DRIVE	
234	042030	122762	000135	000002	CMPB	#135,2(R2)	:IS IT A DIAGNOSTIC COMMAND ?	
235	042036	001011			BNE	1\$:BRANCH IF NOT	
236	042040	016246	000004		MOV	4(R2),-(SP)	:GET THE ROUTINE NUMBER, PARAMETERS	
237	042044	052716	100000		BIS	#BIT15,(SP)	:SET THE DIAGNOSTIC MODE BIT	
238	042050	004037	045130		JSR	RO,WRT.RP	:WRITE THE RPMR1 REG	
239	042054	000024			RPMR1			
240	042056	042670			C17			
241	042060	000422			BR	2\$:LOAD THE COMMAND AND EXIT	
242								
243	042062	062703	000004	1\$:	ADD	#4,R3	:DESIRED WORD COUNT	
244	042066	062704	000002		ADD	#2,R4	:RPWC ADDRESS	
245	042072	012324			MOV	(R3)+,(R4)+	:LOAD WORD COUNT	
246	042074	012324			MOV	(R3)+,(R4)+	:LOAD BUFFER ADDRESS	
247	042076	012346			MOV	(R3)+,-(SP)	:LOAD SECTOR AND TRACK	
248	042100	004037	045130		JSR	RO,WRT.RP	:CALL THE LOAD(WRITE) ROUTINE	
249	042104	000006			RPDA		:INDEX OF REGISTER TO LOAD	
250	042106	042670			C17		:ERROR RETURN ADDRESS	
251	042110	012346			MOV	(R3)+,-(SP)	:LOAD CYLINDER ADDRESS	
252	042112	004037	045130		JSR	RO,WRT.RP		
253	042116	000034			RPDC			
254	042120	042670			C17			
261	042122	004737	042152		JSR	PC,C12	:SEE IF BIT15 SHOULD BE SET IN RPMR1	
262								
263	042126	016246	000002	2\$:	MOV	2(R2),-(SP)	:LOAD 'COMMAND+GO', 'A17&A16', AND 'PSEL'	
264	042132	004037	045130		JSR	RO,WRT.RP		
265	042136	000000			RPCS1			
266	042140	042670			C17			
267	042142	010137	040546		MOV	R1,DTUW	:SET 'DATA TRANSFER UNDERWAY'	
268	042146	000137	042632		JMP	C15		
269								
270	042152	026262	000012	000156	C12:	CMP	12(R2),FE1(R2)	:DATA XFER TO FE CYLINDERS ?
271	042160	002406			BLT	1\$:BR IF NO	
272	042162	012746	100000		MOV	#BIT15,-(SP)	:SET THE DIAGNOSTIC MODE BIT	
273	042166	004037	045130		JSR	RO,WRT.RP	:WRITE THE RPMR1 REG	
274	042172	000024			RPMR1			
275	042174	042670			C17			
276	042176	000207		1\$:	RTS	PC		
277								
278	042200	013704	040560	C13	MOV	RPADR,R4	:RPCS1 ADDRESS	
279	042204	010164	000010		MOV	R1,RP(S2(R4)	:SELECT DRIVE	
280	042210	016246	000012		MOV	12(R2),-(SP)	:DESIRED CYLINDER ADDRESS	
281	042214	004037	045130		JSR	RO,WRT.RP		
282	042220	000034			RPDC			
283	042222	042670			C17			
284	042224	004737	042152		JSR	PC,C12	:SEE IF BIT15 SHOULD BE SET IN RPMR1	
285								
286	042230	116203	000010		MOVB	10(R2),R3	:PICKUP SECTOR ADDRESS	
287	042234	163703	040570		SUB	MXWINDW,R3	:BACKUP BY MAX. SEARCH FOR I/O WINDOW	
288	042240	002002			BGE	1\$:BR IF >= 0	
289	042242	062703	000062		ADD	#50.,R3	:ADD MAXIMUM SECTOR COUNT BACK	
290	042246	010346		1\$:	MOV	R3,-(SP)	:COMBINE THE ADJUSTED SECTOR WITH	
291	042250	116266	000011	000001	MOVB	11(R2),1(SP)	:THE DESIRED TRACK	
292	042256	004037	045130		JSR	RO,WRT.RP	:LOAD DESIRED TRACK & SECTOR	
293	042262	000006			RPDA			
294	042264	042670			C17			

```

295
296 042266 012746 000131      MOV      #131,-(SP)      ;START A SEARCH
297 042272 004037 045130      JSR      R0,WRT.RP
298 042276 000000      RPCS1
299 042300 042670      C17
300 042302 156137 040550 040516      BISB     ATABIT(R1),SRCHWT ;SET "SEARCH WAIT" KEY
301 042310 000550      BR      C15
302
303 042312 013704 040560      C14:    MOV      RPADR,R4      ;RPCS1 ADDRESS
304 C'2316 010164 000010      MOV      R1,RPCS2(R4) ;SELECT DRIVE
305 042322 116203 000002      MOV      2(R2),R3     ;PICKUP THE REQUESTED COMMAND
306 042326 122703 000131      CMPB     #131,R3      ;IS IT A SEARCH COMMAND?
307 042332 001007      BNE     1$           ;BRANCH IF NO
308 042334 016246 000010      MOV      10(R2),-(SP) ;LOAD DESIRED TRACK & SECTOR
309 042340 004037 045130      JSR      R0,WRT.RP
310 042344 000006      RPDA
311 042346 042670      C17
312 042350 000403      BR      2$           ;GO LOAD CYLINDER
313
314 042352 122703 000105      1$:     CMPB     #105,R3     ;IS IT A SEEK COMMAND
315 042356 001011      BNE     3$           ;BRANCH IF NO
316 042360 016246 000012      2$:     MOV      12(R2),-(SP) ;LOAD DESIRED CYLINDER
317 042364 004037 045130      JSR      R0,WRT.RP
318 042370 000034      RPDC
319 042372 042670      C17
320 042374 004737 042152      JSR      PC,C12      ;SEE IF BIT15 SHOULD BE SET IN RPMR1
321 042400 000525      BR      C16
322
323 042402 122703 000115      3$:     CMPB     #115,R3     ;IS IT AN "OFFSET" COMMAND ?
324 042406 001013      BNE     4$           ;BR IF NO
325 042410 004037 045036      JSR      R0,RD.RP    ;MERGE THE OFFSET VALUE INTO RPOF
326 042414 000032      RPOF     ;BUT DON'T CHANGE THE UPPER
327 042416 042670      C17
328 042420 116216 000001      MOV      1(R2),(SP)  ;BYTE WHEN LOADING THE
329 042424 004037 045130      JSR      R0,WRT.RP  ;REGISTER (RPOF)
330 042430 000032      RPOF
331 042432 042670      C17
332 042434 000507      BR      C16         ;GO START THE COMMAND
333
334 042436 122703 000107      4$:     CMPB     #107,R3     ;IS IT A "RECALIBRATE" COMMAND?
335 042442 001504      BEQ     C16         ;BRANCH IF YES
336 042444 122703 000117      CMPB     #117,R3     ;IS IT A RETURN TO CENTER?
337 042450 001501      BEQ     C16         ;BRANCH IF YES
338 042452 122703 000143      5$:     CMPB     #143,R3     ;IS IT A "SET FORMAT" COMMAND?
339 042456 001014      BNE     6$           ;BRANCH IF NO
340 042460 004037 045036      JSR      R0,RD.RP    ;READ THE OFFSET REGISTER
341 042464 000032      RPOF
342 042466 042670      C17
343 042470 116266 000001 000001      MOV      1(R2),1(SP) ;COMBINE "FMT16","ECI", AND "HCI"
344 042476 004037 045130      JSR      R0,WRT.RP
345 042502 000032      RPOF
346 042504 042670      C17
347 042506 000436      BR      12$
348
349 042510 122703 000141      6$:     CMPB     #141,R3     ;IS IT A "GET REGISTER" COMMAND?
350 042514 001023      BNE     10$          ;BRANCH IF NO
351 042516 016203 000006      7$:     MOV      6(R2),R3     ;POINTS TO 1ST ADDRESS OF WHERE
  
```

```

352
353 042522 116237 000010 042540      MOVB    10(R2),9$      ;TO PUT THE REGISTER(S)
354 042530 116205 000011      MOVB    11(R2),R5     ;INIT. THE INDEX FOR THE FIRST REG.
355 042534 004037 045036      JSR     R0,RD.RP     ;INDEX OF LAST REG. TO MOVE
356 042540 000000      JSR     RPCS1        ;READ RHXX/RP07 REGISTER
357 042542 042670      C17:    C17          ;INDEX OF REG. TO READ
358 042544 012623      MOV     (SP)+,(R3)+  ;GET THE CONTENTS OF RHXX//RP07 REG.
359 042546 023705 042540      CMP     9$,R5        ;LAST REG. BEEN READ?
360 042552 001414      BEQ     12$          ;GET OUT IF YES
361 042554 062737 000002 042540      ADD     #2,9$        ;INCREASE THE IND'X BY 2
362 042562 000764      BR      8$          ;LOOP--MORE TO READ
363
364 042564 122703 000145      10$:    CMPB    #145,R3  ;IS IT A "SELECT DRIVE" COMMAND?
365 042570 001405      BEQ     12$          ;BRANCH IF YES
366 042572 010346      11$:    MOV     R3,-(SP)   ;LOAD THE COMMAND
367 042574 004037 045130      JSR     R0,WRT.RP
368 042600 000000      JSR     RPCS1
369 042602 042670      C17:    C17
370 042604 004737 046166      12$:    JSR     PC,POPQUE  ;REMOVE REQ. FROM QUEJE
371 042610 052762 000200 000016      BIS    #BIT07,16(R2) ;SET THE "DONE" BIT
372 042616 005737 040522      TST    SAVEFG        ;SAVE RHXX/RP07 REGISTERS ?
373 042622 100002      BPL     13$          ;BR IF NO
374 042624 004737 045262      JSR     PC,SVRHXX    ;YES--GO SAVE THE REGISTERS
375 042630 000207      13$:    RTS     PC      ;RETURN TO USER
376
377 042632 006301      C15:    ASL     R1
378 042634 012761 023420 040526      MOV    #10000.,TIMER(R1) ;START 10. SECOND TIMER
379 042642 006201      ASR    R1
380 042644 112761 000001 040444      MOVB   #1,DRVACT(R1)  ;SET THE DRIVE ACTIVE
381 042652 000207      RTS     PC          ;RETURN TO THE USER
382
383 042654 010346      C16:    MOV     R3,-(SP)   ;LOAD THE COMMAND
384 042656 004037 045130      JSR     R0,WRT.RP
385 042662 000000      JSR     RPCS1
386 042664 042670      C17:    C17
387 042666 000761      BR      C15
388
389 042670 005702      C17:    TST     R2
390 042672 001001      BNE     1$          ;ANYTHING IN QUEUE ?
391 042674 000207      RTS     PC          ;BRANCH IF QUEUE IS THERE
392 042676 012762 104000 000016 1$:     MOV    #BIT15!BIT11,16(R2) ;OTHERWISE EXIT
393
394 042704 012746 000111      C17B:   MOV    #111,-(SP)   ;DO A "DRIVE CLEAR"
395 042710 004037 045130      JSR     R0,WRT.RP
396 042714 000000      JSR     RPCS1
397 042716 042756      C18:    C18
398 042720 004737 046050      JSR     PC,EMPTYQ   ;EMPTY THE QUEUE
399 042724 105061 040504      CLRB   DPRQS(R1)    ;CLEAR THE PORT REQUEST FLAG
400 042730 105061 040444      CLRB   DRVACT(R1)   ;DRIVE IS IDLE
401 042734 020237 040514      CMP    R2,TRNSWT    ;IF THIS DRIVE HAD AN I/O REQUEST
402 042740 001005      BNE     1$          ;IN PROGRESS CLEAR ALL OF THE FLAGS
403 042742 005037 040514      CLR    TRNSWT
404 042746 012737 177777 040546      MOV    #-1,DTUW
405 042754 000207      1$:     RTS     PC
406
407 042756 104412      C18:    SAVREG
408 042760 005001      CLR    R1          ;SAVE R0 - R5

```

```

409 042762 005003
410 042764 105761 040444 1$: CLR R3 ;DRIVE ACTIVE?
411 042770 001003 BNE DRVACT(R1) ;BRANCH IF IN ACTIVE
412 042772 105761 040504 TSTB DPRQS(R1) ;PORT REQUEST
413 042776 001443 BEQ 6$ ;BRANCH IF NOT
414 043000 013702 040514 2$: MOV TRNSWT,R2 ;GET 'DPB' FROM THE 'TRANSFER WAIT' QUEUE
415 043004 020137 040546 CMP R1,DTUW ;DID THIS DRIVE HAVE AN I/O IN PROGRESS?
416 043010 001402 BEQ 3$ ;BRANCH IF YES
417 043012 004737 046144 JSR PC,CETREQ ;GET THE DPB POINTER
418 043016 005702 3$: TST R2 ;QUEUE ENTRY FOR DRIVE ?
419 043020 001413 BEQ 5$ ;BR IF NOT
420 043022 032764 010000 000010 BIT #BIT12,RPCS2(R4) ;'NED' SET ?
421 043030 001404 BEQ 4$ ;BR IF NOT
422 043032 012762 100002 000016 MOV #BIT15!BIT01,16(R2) ;SET 'DRIVE NON-EXISTENT' INDICATOR
423 043040 000403 BR 5$ ;CONTINUE
424
425 043042 012762 102000 000016 4$: MOV #BIT15!BIT10,16(R2) ;SET 'NON-CLEARABLE PARITY' ERROR INDICATOR
426 043050 012763 177777 040526 5$: MOV #-1,TIMER(R3) ;STOP THE TIMER
427
431 043056 105061 040444 CLRB DRVACT(R1) ;SET 'DRIVE ACTIVE' TO IDLE
432 043062 105061 040504 CLRB DPRQS(R1) ;CLEAR PORT REQUEST FLAG
433 043066 020137 040546 CMP R1,DTUW ;IS THIS DRIVE SETUP FOR A TRANSFER
434 043072 001005 BNE 6$ ;BR IF NOT
435 043074 012737 177777 040546 MOV #-1,DTUW ;RESET THE INDICATOR
436 043102 005037 040514 CLR TRNSWT ;CLEAR THE TRANSFER QUEUE
437 043106 005201 6$: INC R1 ;MOVE TO THE NEXT DRIVE
438 043110 062703 000002 ADD #2,R3
439 043114 042701 177770 BIC #^C7,R1
440 043120 001321 BNE 1$ ;BRANCH IF MORE DRIVES
441 043122 012737 177777 040546 MOV #-1,DTUW ;NO DATA TRANSFERS UNDERWAY
442 043130 005037 040514 CLR TRNSWT ;CLEAR THE 'TRANSFER WAIT' QUEUE
443 043134 004737 045772 JSR PC,CLRQUE ;CLEAR ALL OF THE REQUEST QUEUES
444 043140 012764 000040 000010 MOV #BIT05,RPCS2(R4) ;DO A MASSBUS INIT.
445 043146 000406 BR 8$ ;CONTINUE
446
447 043150 004737 046050 7$: JSR PC,EMPTYQ ;CLEAR THE DRIVE'S QUEUE
448 043154 105061 040454 CLRB DRVSTA(R1) ;SET DRIVE TO OFFLINE
449 043160 105061 040464 CLRB DRVTP(R1) ;CLEAR THE DRIVE TYPE INDICATOR
450 043164 004737 045426 8$: JSR PC,SET.IE ;SET 'IE' WITHOUT 'TRE'
451 043170 104413 RESREG ;RESTORE R0 - R5
452 043172 000207 RTS PC ;RETURN
453
454 ;INTERRUPT SERVICE ROUTINE
455
456 043174 112737 000001 040520 ISR: MOVB #1,ACTDRV ;SET 'ACTIVE DRIVER' FLAG
457 043202 104412 SAVREG ;SAVE R0 - R5
458 043204 013704 040560 MOV RPADR,R4 ;ADDRESS OF RPCS1
459 043210 013701 040546 MOV DTUW,R1 ;GET 'DATA TRANSFER UNDERWAY' INDICATOR
460 043214 002402 BLT 1$ ;BRANCH IF NO DATA TRANSFER UNDERWAY
461 043216 004737 043236 JSR PC,TD ;CALL TRANSFER DONE
462 043222 004737 043550 1$: JSR PC,SC ;CALL SPECIAL CONDITIONS
463 043226 104413 RESREG ;RESTORE R0 - R5
464 043230 105037 040520 CLRB ACTDRV ;CLEAR 'ACTIVE DRIVER' FLAG
465 043234 000002 RTI ;RETURN
466
467 ;TRANSFER DONE ROUTINE
468

```



```

469 043236 005037 043460      TD:   CLR      PERM      ;CLR PERMENANT ERROR FLAG
470
471 043242 105061 040444      CLR      DRVACT(R1) ;SET DRIVE ACTIVE INDICATOR TO IDLE
472 043246 012737 177777 040546      MOV      #-1,DTUW   ;NO DATA TRANSFERS UNDERWAY
473 043254 006301                ASL      R1
474 043256 012761 177777 040526      MOV      #-1,TIMER(R1) ;CANCEL TIMEOUT
475 043264 006201                ASR      R1
476 043266 013702 040514      MOV      TRNSWT,R2   ;GET 'DPB' FROM THE 'TRANSFER WAIT' QUEUE
477 043272 005037 040514      CLR      TRNSWT     ;TRANSFER WAIT QUEUE--CLEAR QUEUE
478 043276 052762 000200 000016      BIS      #BIT07,16(R2) ;SET DONE
479 043304 010164 000010      MOV      R1,RPCS2(R4) ;SELECT THE DRIVE
480 043310 004037 045036      JSR      R0,RD.RP   ;TRANSFER ERROR(TRE=1)?
481 043314 000000                RPCS1
482 043316 042670                C17
483 043320 006126                ROL      (SP)+
484 043322 100430                BMI      4$         ;BR IF YES
485 043324 005737 040522      TST      SAVEFG     ;SAVE RHXX/RP07 REGISTERS ?
486 043330 100002                BPL      1$         ;BR IF NO
487 043332 004737 045262      JSR      PC,SVRHXX  ;YES--SAVE THE REGISTERS
488 043336 122762 000135 000002 1$:   CMPB    #135,2(R2)  ;IE FROM DIAGNOSTIC COMMAND ?
489 043344 001003                BNE      2$         ;BRANCH IF NOT
490 043346 116164 040550 000016      MOV      ATABIT(R1),RPAS(R4) ;RESET THE ATA BIT
491
493 043354 004737 043462      2$:   JSR      PC,WC.HK  ;SEE IF WRITE CHECK TO BE PUT IN QUEUE
494 043360 004737 046144      JSR      PC,GETREQ  ;GET DPB POINTER
498 043364 005702                TST      R2         ;ENTRY FOR DRIVE ?
499 043366 001403                BEQ      3$         ;BR IF NOT
500 043370 004737 041542      JSR      PC,JPT     ;CALL OPTIMIZER
501 043374 000207                RTS      PC         ;RETURN
502 043376 012714 000113      3$:   MOV      #113,(R4) ;RELEASE THE DRIVE
503 043402 000207                RTS      PC         ;RETURN
504
505 043404 052762 100100 000016 4$:   BIS      #BIT15!BIT06,16(R2) ;SET DATA ERROR FLAG
506 043412 004737 046050      JSR      PC,EMPTYQ  ;EMPTY THE 'DRIVE'S WAIT' QUEUE
507 043416 004737 045262      JSR      PC,SVRHXX  ;SAVE THE RHXX/RP07 REGISTERS
508 043422 012714 040111      MOV      #40111,(R4) ;ISSUE A 'DRIVE CLEAR'
509
510 043426 032764 040000 000012      BIT      #BIT14,RPDS(R4) ;DID ERROR BIT CLEAR?
511 043434 001406                BEQ      5$         ;BR IF YES
512 043436 005237 043460      INC      PERM       ;SET PERM ERROR FLAG
513 043442 116164 040550 000016      MOV      ATABIT(R1),RPAS(R4) ;CLR ATA BIT
514 043450 000207                RTS      PC         ;RETURN
515
516 043452 012714 000113      5$:   MOV      #113,(R4) ;ISSUE A RELEASE TO THE DRIVE
517 043456 000207                RTS      PC         ;RETURN
518
519 043460 000000      PERM:  .WORD      0      ;PERMENANT ERROR FLAG
520
521      ;FORCE WRITE CHECK ROUTINE
522
523 043462 005737 001506      WC.HK: TST      WRTCHK    ;DO WRITE CHECK ?
524 043466 001427                BEQ      1$         ;BR IF NO
525 043470 122762 000161 000002      CMPB    #WRTDAT,$COMND(R2) ;LAST COMMAND A WRITE COMMAND ?
526 043476 001023                BNE      1$         ;BRANCH IF NOT
527 043500 004037 046070      JSR      R0,DRVQUE  ;PUT INTO THE QUEUE
528 043504 000420                BR       1$         ;BRANCH IF QUEUE IS FULL
529

```

```

530 043506 005062 000016          CLR      $STATUS(R2)      ;CLEAR 'DONE' BIT IN DPB
531 043512 116262 000166 000027    MOVVB   $RPCS1(R2), $PREV0(R2) ;PREVIOUS COMMAND
532 043520 016262 000012 000034    MOV     $CYL(R2), $PREVA+2(R2) ;PREVIOUS CYLINDER ADDRESS
533 043526 016262 000010 000032    MOV     $SEC(R2), $PREVA(R2)   ;PREVIOUS SEC , TRK ADDRESSES
534 043534 105062 000024          CLRB   $CODE(R2)         ;CHANGE WRITE DATA TO WRITE CHECK DATA
535 043540 112762 000151 000002    MOVVB  #WCKD, $COMND(R2)     ;CHANGE FUNCTION CODE TO WRITE CHECK
536 043546 0C0207          1$:    RTS      PC              ;EXIT
537
538          ;SPECIAL CONDITION ROUTINE
539
540 043550 116403 000016          SC:    MOVVB   RPAS(R4), R3    ;READ 'RPAS' REGISTER
541 043554 001013          BNE     2$                    ;BRANCH IF ANY 'ATA' BITS SET
542 043556 004037 045036          JSR     RO, RD.RP             ;READ CONTROL AND STATUS REGISTER
543 043562 000000          RPCS1
544 043564 042756          CIB
545 043566 106126          ROLB   (SP)+                 ;IS 'IE'=1?
546 043570 100404          BMI     1$                    ;YES, NO DRIVES TO CHECK
547 043572 000401          BR     1$-4                  ;BRANCH OVER ERROR, REPLACE BRANCH WITH 'NOP'
548
549 043574 104001          EMT     1                     ;IF ERROR REPORT IS DESIRED
550 043576 004737 045426          JSR     PC, SET.IE           ;REPORT ILLEGAL INTERRUPT
551 043602 000207          1$:    RTS      PC              ;SET INTERRUPT ENABLE
552
553          2$:    CLR      -(SP)          ;PROCESS ALL DRIVES THAT HAVE
554 043606 110316          MOVVB   R3, (SP)            ;STORE ATA BITS ON STACK
555 043610 005001          CLR     R1                  ;SETUP R1 & R3 TO CHECK DRIVE 0
556 043612 012703 000001          MOV     #1, R3
557 043616 030316          SC3:   BIT     R3, (SP)      ;IS THIS ATA BIT SET ?
558 043620 001005          BNE     SC5                  ;BR IF YES
559 043622 005201          SC4:   INC     R1            ;MOVE TO THE NEXT DRIVE
560 043624 106303          ASLB   R3                   ;ANY MORE DRIVES TO CHECK ?
561 043626 001373          BNE     SC3                  ;BR IF YES
562 043630 005726          TST    (SP)+                ;CLEAN OFF THE STACK
563 043632 000207          RTS      PC
564
565 043634 105761 040504          SC5:   TSTB   DPRQS(R1)      ;IS THERE PORT REQUEST OUTSTANDING ?
566 043640 001402          BEQ     1$                    ;BR IF NO
567 043642 000137 044204          JMP     SC13                 ;START THE OUTSTANDING COMMAND
568
569 043646 105761 040454          1$:    TSTB   DRVSTA(R1)     ;IS THIS DRIVE ON-LINE ?
570 043652 003011          BGT     2$                    ;BR IF YES
571 043654 004737 046144          JSR     PC, GETREQ           ;GET DPB POINTER
572 043660 004737 045262          JSR     PC, SVRHXX          ;SAVE THE RHXX/RP07 REGISTERS
573 043664 004737 044120          JSR     PC, SC12            ;SAVE RPDS, RPER1, RPER2 AND RPER3
574
575 043670 105761 040454          TSTB   DRVSTA(R1)          ;DID DRIVE COME ON-LINE ?
576 043674 003405          BLE     3$                    ;BR IF NO
577 043676 105761 040444          2$:    TSTB   DRVACT(R1)     ;DRIVE ACTIVE WITH COMMAND OR ERROR RECOVERY ?
578 043702 001020          BNE     SC6                    ;BR IF EITHER
579 043704 004737 044120          JSR     PC, SC12            ;SAVE RPDS, RPER1, RPER2 AND RPER3
580
581          3$:    TSTB   DRVSTA(R1)          ;CHECK ON DRIVE'S STATUS
582 043714 100412          BMI     4$                    ;BR IF UNSAFE
583 043716 012746 000111          MOV     #111, -(SP)          ;DRIVE CLEAR
584 043722 004037 045130          JSR     RO, WRT.RP          ;WRITE THE COMMAND INTO RPCS1
585 043726 000000          RPCS1
586 043730 043772          SC8

```

```

587 043732 116164 040550 000016      MOVB  ATABIT(R1),RPAS(R4)      ;CLR ATTN BIT
588 043740 000240                      NOP
592 043742 000727                      4$:  BR      SC4              ;CHECK FOR MORE DRIVES
593
594 043744 006301                      SC6:  ASL    R1              ;SETUP TO ADDRESS WORDS
595 043746 012761 177777 040526      MOV    #-1,TIMER(R1)        ;STOP THE TIMER
596 043754 006201                      ASR    R1              ;RESTORE THE DRIVE ADDRESS
597 043756 004737 046144              JSR    PC,GETREQ           ;GET THE DPB POINTER FROM THE QUEUE
598 043762 010164 000010              MOV    R1,RPCS2(R4)        ;SELECT DRIVE
599 043766 000137 044022              JMP    SC11              ;PROCESS THE SEARCH
600
601 043772 105761 040444              SC8:  TSTB  DRVACT(R1)      ;IS DRIVE IDLE?
602 043776 001405                      BEQ    1$                ;YES--BRANCH
603 044000 004737 046144              JSR    PC,GETREQ           ;GET DPB POINTER
604 044004 004737 042670              JSR    PC,C17            ;PROCESS THE PARITY ERROR
605 044010 000402                      BR     2$                ;CONTINUE
606
607 044012 004737 042704              1$:  JSR    PC,C17B        ;PROCESS THE UNCORRECTABLE PARITY ERROR
608 044016 000137 043622              2$:  JMP    SC4              ;CHECK MORE DRIVES
609
610 044022 105061 040444              SC11: CLRB  DRVACT(R1)      ;SET DRIVE IDLE
611 044026 136137 040550 040516      BITB  ATABIT(R1),SRCHWT    ;DOING A SEARCH OPERATION FOR
612                                     ;AN I/O COMMAND?
613 044034 001012                      BNE    1$                ;BRANCH IF YES
614 044036 004737 046166              JSR    PC,POPQUE          ;REMOVE REQUEST FROM QUEUE
615 044042 052762 000200 000016      BIS    #BIT07,16(R2)      ;SET "DONE" BIT
616 044050 005737 040522              TST    SAVEFG            ;SAVE THE REGISTERS?
617 044054 100002                      BPL    1$                ;BR IF NO
618 044056 004737 045262              JSR    PC,SVRHXX          ;YES--SAVE ALL OF THE RHXX/RPO7 REG'S
619 044062 116164 040550 000016      1$:  MOVB  ATABIT(R1),RPAS(R4) ;CLEAR ATTENTION BIT
620 044070 146137 040550 040516      BICB  ATABIT(R1),SRCHWT    ;CLEAR IMPLIED SEEK SET
621 044076 006301                      ASL    R1              ;WORD INDEX
622 044100 012761 177777 040526      MOV    #-1,TIMER(R1)        ;STOP CLOCK
623 044106 006201                      ASR    R1              ;RESTORE R1
624 044110 004737 041542              JSR    PC,OPT            ;START A REQUEST
625 044114 000137 043622              JMP    SC4              ;CHECK FOR MORE DRIVES
626
627 044120 010164 000010              SC12: MOV    R1,RPCS2(R4)      ;SELECT DRIVE
628 044124 006301                      ASL    R1
629 044126 006301                      ASL    R1
630 044130 006301                      ASL    R1
631 044132 016461 000012 040344      MOV    RPDS(R4),RPSTU0(R1)  ;STORE DRIVE STATUS
632 044140 016461 000014 040346      MOV    RPER1(R4),RPSTU0+2(R1) ;STORE ERROR REG #1
633 044146 016461 000040 040350      MOV    RPER2(R4),RPSTU0+4(R1) ;STORE ERROR REG #2
634 044154 016461 000042 040352      MOV    RPER3(R4),RPSTU0+6(R1) ;STORE ERROR REG #3
635 044162 006201                      ASR    R1
636 044164 006201                      ASR    R1
637 044166 006201                      ASR    R1
638 044170 004037 040750              JSR    RO,DRVINT          ;INIT. THE STATE OF THE DRIVE
639 044174 000401                      BR     1$                ;TAKE ERROR EXIT
640 044176 000207                      RTS    PC                ;RETURN
641 044200 005726                      1$:  TST    (SP)+          ;CLEAR THE STACK
642 044202 000673                      BR     SC8              ;PROCESS THE PARITY ERROR
643
649 044204 010164 000010              SC13: MOV    R1,RPCS2(R4)      ;SELECT THE DRIVE
650 044210 116164 040550 000016      MOVB  ATABIT(R1),RPAS(R4)  ;CLEAR THE ATTENTION BIT
651 044216 105761 040474              1$:  TSTB  DPINT(R1)        ;INITIALIZING THE DRIVE ?

```

```

652 044222 001424          BEQ      2$          ;BR IF NOT
653 044224 105061 040474   (LRB    DPINT(R1)     ;CLEAR THE INIT INDICATOR
654 044230 004037 040750   JSR    RO,DRVINT     ;GO INIT THE DRIVE
655 044234 000240          NOP                    ;DUMMY PARITY ERROR RETURN
656 044236 105761 040454   TSTB   DRVSTA(R1)    ;DRIVE ONLINE ?
657 044242 003014          BGT    2$          ;BR IF YES -- START ORDER
658 044244 005702          TST    R2            ;QUEUE ENTRY FOR THE DRIVE
659 044246 001423          BEQ    4$          ;BR IF NOT
660 044250 004737 046 44   JSR    PC,GETREQ     ;GET DPB ADDRESS
661 044254 052762 140000 000016  BIS    #BIT15:BIT14,16(R2) ;INFORM USER THAT DRIVE OFFLINE
662 044262 004737 045262   JSR    PC,SVRHXX     ;SAVE THE REGISTERS
663 044266 004737 046166   JSR    PC,POPQUE     ;REMOVE THE QUEUE
664 044272 000411          BR     4$
665
666 044274 032764 004000 000000 2$:   BIT    #BIT11,RPCS1(R4) ;DVA SET ?
667 044302 001003          BNE    3$          ;SET THEN CALL OPT
673 044304 004737 045426   JSR    PC,SET.IE
674 044310 000402          BR     4$
675
676 044312 004737 041542   3$:   JSR    PC,OPT     ;START THE REQUEST
677 044316 000137 043622   4$:   JMP    SC4         ;PROCESS OTHER DRIVES
678
679          ;/RP07 TIMER ROUTINE
680          ;CALL
681          ;
682          ;   MOV    #TIME,-(SP) ;ELAPSED TIME IN MILLISECONDS ON THE STACK
683          ;   JSR    PC,RPTMR  ;CALL RP07 TIME ROUTINE
684 044322 005737 040520   RPTMR: TST    ACTDRV   ;CHECK 'ACTDRV & ACTSTR'
685 044326 001027          BNE    4$          ;IF NON ZERO EXIT
686 044330 112737 000001 040521   MOVB   #1,ACTSTR    ;SET 'ACTSTR'
687 044336 104412          SAVREG ;SAVE R0 - R5
688 044340 005001          CLR    R1          ;START WITH DRIVE 0
689 044342 005003          CLR    R3
690 044344 005763 040526   1$:   TST    TIMER(R3)   ;IS THE TIMER RUNNING?
691 044350 002406          BLT    2$          ;BRANCH IF NO
692 044352 166663 000002 040526   SUB    2(SP),TIMER(R3) ;COUNT THE INTERVAL
693 044360 003002          BGT    2$          ;BR IF NO SOFTWARE TIMEOUT
694 044362 004737 044412   JSR    PC,STO       ;CALL SOFTWARE TIMEOUT ROUTINE
695 044366 005201          INC    R1          ;MOVE TO NEXT DRIVE
696 044370 005723          TST    (R3)+
697 044372 022701 000010   CMP    #8.,R1      ;OUT OF DRIVES?
698 044376 003362          BGT    1$          ;BRANCH IF NO
699 044400 104413          RESREG ;RESTORE R0 - R5
700 044402 105037 040521   CLRB   ACTSTR       ;ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
701 044406 012616          4$:   MOV    (SP)+,(SP)  ;ADJUST THE STACK
702 044410 000207          RTS    PC          ;RETURN
703
704          ;SOFTWARE TIMEOUT ROUTINE
705          ;
706          ;NOTE: THIS ROUTINE MUST BE ENTERED AT PRIORITY 6
707          ;OR GREATER
708          ;
709          ;CALL
710          ;   MOV    #DRVNUM,R1 ;DRIVE NUMBER
711          ;   JSR    PC,STO     ;SOFTWARE TIME ROUTINE
712          ;
713          ;

```

```

714 044412 010146          STC:  MOV    R1,-(SP)      ;SAVE R1
715 044414 010346          MOV    R3,-(SP)      ;SAVE R3
716 044416 017704 040560    MOV    R^AL,R4       ;GET ADDRESS OF 'RPCS1'
717 044422 010164 000010    MOV    R1,RPCS2(R4)  ;SELECT THE DRIVE
718 044426 004037 045036    JSR    RO,RD.RP      ;READ THE DRIVE STATUS REGISTER
719 044432 000012          RPDS
720 044434 044724          9$
721 044436 105726          TSTB   (SP)+         ;IS 'DRY'=1?
722 044440 100473          BMI    6$           ;BR IF YES
723 044442 105761 040474    1$:  TSTB   DPINT(R1)   ;TRYING TO INITIALIZE THE DRIVE ?
724 044446 001070          BNE    6$           ;BR IF YES
725 044450 105761 040504    TSTB   DPRQS(R1)    ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
726 044454 001065          BNE    6$           ;BR IF YES
727 044456 013702 040514    MOV    TRNSWT,R2     ;GET 'DPB' FROM THE 'TRANSFER WAIT' QUEUE
728 044462 020137 040546    CMP    R1,DTUW       ;TRANSFER UNDERWAY ON THIS DRIVE?
729 044466 001402          BEQ    2$           ;BRANCH IF YES
730 044470 004737 046144    JSR    PC,GETREQ     ;GET DPB ADDRESS
731 044474 052762 101000 000016 2$:  BIS    #BIT15!BIT09,16(R2) ;SET THE ERROR FLAGS
732 044502 004737 045262    JSR    PC,SVRHXX     ;SAVE RHXX/RP07 REGISTERS
733 044506 012764 000040 000010 MOV    #BIT05,RPCS2(R4) ;'INIT' THE MASS BUS
734 044514 105061 040444    CLRB  DRVACT(R1)    ;DRIVE IS IDLE
735 044520 005001          CLR    R1           ;START WITH DRIVE 0
736 044522 005003          CLR    R3
737 044524 004037 040750    3$:  JSR    RO,DRVINT    ;INIT. THIS DRIVE
738 044530 000475          BR     9$           ;PARITY ERROR RETURN
739
740 044532 105761 040444    TSTB  DRVACT(R1)    ;DRIVE IDLE BEFORE THE INIT.?
741 044536 001414          BEQ    5$           ;YES--BRANCH
742 044540 013702 040514    MOV    TRNSWT,R2     ;GET 'DPB' FROM THE 'TRANSFER WAIT' QUEUE
743 044544 023701 040546    CMP    DTUW,R1       ;WAS A DATA TRANSFER UNDERWAY ON THIS DRIVE?
744 044550 001402          BEQ    4$           ;YES--BRANCH
745 044552 004737 046144    JSR    PC,GETREQ     ;GET THE DPB POINTER FROM QUEUE
746 044556 052762 100400 000016 4$:  BIS    #BIT15!BIT08,16(R2) ;INFORM USER OF INIT.
747 044564 105061 040444    CLRB  DRVACT(R1)    ;SET DRIVE ACTIVE TO IDLE
748 044570 012763 177777 040526 5$:  MOV    #-1,TIMER(R3) ;STOP THE TIMER
749 044576 005723          TST   (R3)+         ;UPDATE THE INDEX
750 044600 005201          INC   R1           ;INCREMENT THE DRIVE NUMBER
751 044602 022701 000010    CMP    #8.,R1       ;LAST DRIVE BEEN CHECKED?
752 044606 003346          BGT   3$           ;NO--LOOP
753 044610 012737 177777 040546    MOV    #-1,DTUW     ;NO DATA TRANSFERS UNDERWAY
754 044616 005037 040514    CLR   TRNSWT        ;CLEAR TRANSFER WAIT QUEUE
755 044622 004737 045772    JSR   PC,CLRQUE     ;CLEAR ALL REQUEST QUEUES
756 044626 000500          BR   13$          ;EXIT
757
758 044630 116405 000016    6$:  MOVB   RPAS(R4),R5   ;READ ATTENTION REG
759 044634 136105 040550    BITB  ATABIT(R1),R5 ;IS ATTENTION FOR THIS DRIVE UP ?
760 044640 001017          BNE   7$           ;YES--BRANCH
761 044642 105761 040474    TSTB  DPINT(R1)     ;TRYING TO INITIALIZE THE DRIVE ?
762 044646 001031          BNE   10$          ;BR IF YES - DRIVE NOT ONLINE
763 044650 105761 040504    TSTB  DPRQS(R1)    ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
764 044654 001045          BNE   11$          ;BR IF YES - NO RESPONSE TO REQUEST
765 044656 020137 040546    CMP   R1,DTUW       ;DATA TRANSFER UNDERWAY FOR THIS DRIVE
766 044662 001267          BNE   1$           ;BR IF NO
767 044664 004037 045036    JSR   RO,RD.RP      ;READ CONTROL AND STATUS REGISTER
768 044670 000000          RPCS1
769 044672 044724          9$
770 044674 105726          TSTB  (SP)+         ;CHECK 'RDY'
  
```

```

771 044676 100261          BPL      1$          ;BR IF 'RDY'=0
772 044700 105761 040474 7$:      TSTB    DPINT(R1)      ;INITIALIZING THE DRIVE ?
773 044704 001003          BNE     8$          ;BR IF INIT PENDING
774 044706 105761 040504      TSTB    DPRQS(R1)      ;PORT REQUEST PENDING ?
775 044712 001446          BEQ     13$         ;BR IF NOT
776 044714 012763 177777 040526 8$:      MOV     #-1,TIMER(R3)  ;STOP THE TIMER
777 044722 000442          BR      13$         ;EXIT
778
779 044724 004737 042756 9$:      JSR     PC,C18        ;GO HANDLE THE 'NED'
780 044730 000437          BR      13$
781
782 044732 105061 040474 10$:     CLRB    DPINT(R1)      ;CLEAR THE INITIALIZE INDICATOR
783 044736 105061 040454      CLRB    DRVSTA(R1)     ;SET DRIVE OFFLINE
784 044742 012763 177777 040526      MOV     #-1,TIMER(R3)  ;STOP THE TIMER
785 044750 004737 046144      JSR     PC,GETREQ      ;GET THE DPB ADDRESS
786 044754 005702          TST     R2            ;REQUEST IN QUEUE ?
787 044756 001424          BEQ     13$         ;BR IF NOT
788 044760 052762 140000 000016      BIS    #BIT15!BIT14,16(R2) ;INFORM THE USER DRIVE NOT AVAILABLE
789 044766 000414          BR      12$         ;FINISH
790
791 044770 012763 177777 040526 11$:    MOV     #-1,TIMER(R3)  ;STOP THE TIMER
792 044776 105061 040504      CLRB    DPRQS(R1)      ;CLEAR PORT REQUEST INDICATOR
793 045002 004737 046144      JSR     PC,GETREQ      ;GET DPB ADDRESS
794 045006 005702          TST     R2            ;QUEUE ENTRY FOR DRIVE ?
795 045010 001407          BEQ     13$         ;BR IF NONE
796 045012 012762 100004 000016      MOV     #BIT15!BIT2,16(R2) ;INFORM USER OF PORT REQUEST ERROR
797 045020 004737 046050 12$:      JSR     PC,EMPTYQ      ;CLEAR THE QUEUE FOR THE DRIVE
798 045024 004737 045262      JSR     PC,SVRHXX      ;SAVE RHXX/RP07 REGISTERS
799 045030 012603 13$:      MOV     (SP)+,R3      ;RESTORE R3
800 045032 012601          MOV     (SP)+,R1      ;RESTORE R1
801 045034 000207          RTS     PC            ;RETURN
802
803          ;ROUTINE TO READ A RHXX/RP07 REGISTER
804
805          ;CALL
806          JSR     R0,RD.RP ;GO READ A REGISTER
807          INDEX  ;REG. INDEX FROM BASE
808          ERRADR ;ERROR ADDRESS--PROCESS ERROR STARTING
809          ;AT THIS ADDRESS
810          ;CONTENTS OF REG. IS ON THE STACK
811
812 045036 011646          RD.RP: MOV     (SP),-(SP)    ;SAVE R0
813 045040 013746 040560      MOV     RPADR,-(SP)    ;ADDRESS OF THE
814 045044 062016          ADD     (R0)+,(SP)     ;REG
815 045046 017666 000000 000004      MOV     @ (SP),4(SP)   ;READ THE CONTENTS OF THE REG
816 045054 013716 040560      MOV     RPADR,(SP)     ;CHECK IF NON-EXIST DRIVE
817 045060 062716 000010          ADD     #RPCS2,(SP)   ;
818 045064 032776 010000 000000      BIT     #BIT12,@(SP)   ;NED BIT SET ?
819 045072 001004          BNE     1$           ;ERROR EXIT
820 045074 032777 020000 173456      BIT     #BIT13,@RPADR  ;MCPE SET ?
821 045102 001406          BEQ     2$           ;EXIT
822 045104 016666 000002 000004 1$:      MOV     2(SP),4(SP)   ;MOVE THE R0 TO TOP OF STACK
823 045112 022626          CMP     (SP)+,(SP)+   ;CLEAR OFF THE STACK
824 045114 011000          MOV     (R0),R0      ;ERROR EXIT ADDRESS
825 045116 000403          BR      3$           ;EXIT
826
827 045120 062700 000002 2$:      ADD     #2,R0         ;NORMAL EXIT

```

```

828 045124 005726          TST      (SP)+      ;CLEAR OFF STACK
829 045126 000200      3$:   RTS      R0          ;EXIT
830
831                      ;ROUTINE TO WRITE A REGISTER
832
833                      ;CALL
834                      ;      MOV      DATA,-(SP)      ;DATA TO BE LOADED ON THE STACK
835                      ;      JSR      R0,WRT.RP        ;CALL THE ROUTINE TO LOAD(WRITE) THE REG.
836                      ;      INDEX   ERRADR          ;INDEX OF THE REGISTER TO BE LOADED
837                      ;      ERRADR  RETURN          ;ADDRESS TO RETURN TO ON AN ERROR
838                      ;      RETURN  ;ERROR FREE RETURN
839
840 045130          WRT.RP:
841 045130 012046          MOV      (R0)+,-(SP)      ;FORMING THE REG ADDRESS
842 045132 001017          BNE      2$          ;BRANCH IF NOT RPCS1
843 045134 22766 000150 000004  CMPB    #150,4(SP)      ;DATA XTRNS COMMAND ?
844 045142 002413          BLT      2$          ;BRANCH IF NOT
845 045144 105777 173410 1$:   TSTB    @RPADR        ;SEE IF CONTROLLER READY
846 045150 100375          BPL      1$
847 045152 017746 173402  MOV      @RPADR,-(SP)      ;READ RPCS1
848 045156 000316          SWAB    (SP)          ;MERG THE A17,A18,PSEL BITS
849 045160 042716 177770  BIC      #^C7,(SP)      ;CHOP OFF THE REST BITS FROM RPCS1
850 045164 111666 000007  MOVVB   (SP),7(SP)      ;ATTACH A17,A18,PSEL TO COMMAND
851 045170 005726          TST      (SP)+
852 045172 063716 040560 2$:   ADD      RPADR,(SP)      ;RESTORE STACK LEVEL
853 045176 016676 000004 000000  MOV      4(SP),@ (SP)    ;THE DEST REG ADDRESS
854 045204 013716 040560  MOV      RPADR,(SP)      ;WRITE THE REGISTER
855 045210 062716 000010  ADD      #RPCS2,(SP)     ;CHECK NED,PAR BITS
856 045214 032776 010000 000000  BIT      #BIT12,@(SP)   ;NONE EXIST DRIVE ?
857 045222 001013          BNE      3$          ;BRANCH IF IT IS
858 045224 013716 040560  MOV      RPADR,(SP)      ;ADDRESS RPER1
859 045230 062716 000014  ADD      #RPER1,(SP)
860 045234 032776 000010 000000  BIT      #BIT3,@(SP)   ;PAR SET ?
861 045242 001003          BNE      3$          ;BRANCH IF SO
862 045244 062700 000002  ADD      #2,R0          ;NORMAL RETURN
863 045250 000401          BR      4$          ;EXIT
864
865 045252 011000          3$:   MOV      (R0),R0      ;ERROR EXIT
866 045254 005726          4$:   TST      (SP)+      ;CLEAR OFF THE STACK
867 045256 012616          MOV      (SP)+,(SP)    ;MOVE R0 TO TOP OF STACK
868 045260 000200          RTS      R0          ;EXIT
869
870                      ;ROUTINE TO SAVE THE RHXX/RP07 REGISTERS AS PER DPB+14
871
872                      ;CALL
873                      ;      MOV      #DPB,R2          ;ADDRESS OF DRIVE PARAMETER BLOCK
874                      ;      JSR      PC,SVRHXX        ;SAVE THE DRIVES REG'S
875
876 045262 104412          SVRHXX: SAVREG          ;SAVE R0 - R5
877 045264 005702          TST      R2          ;QUEUE ENTRY FOR THE DRIVE ?
878 045266 001455          BEQ      7$          ;BR IF NONE
879 045270 013704 040560  MOV      RPADR,R4
880 045274 111264 000010  MOVVB   (R2),RPCS2(R4) ;SELECT DRIVE
881 045300 016203 000014  MOV      14(R2),R3      ;GET THE ERROR TABLE POINTER
882 045304 001446          BEQ      7$          ;EXIT IF NO ADDRESS
883 045306 005037 045342  CLR      3$          ;COUNTER & POINTER
884 045312 023727 045342 000022 1$:   CMP      3$,#RPDB      ;REACHED THE BUFFER REGISTER ?

```

```

885 045320 001006          BNE      2$          ;BR IF NOT
886 045322 032764 000200 000010  BIT      #BIT07,RPCS2(R4) ;'OR' SET ?
887 045330 001002          BNE      2$          ;BR IF SET
888 045332 005023          CLR      (R3)+       ;STORE RPDB AS ZEROES
889 045334 000405          BR       4$          ;CONTINUE
890
891 045336 004037 045036    2$:      JSR      R0,RD.RP    ;READ THE SELECTED REGISTER
892 045342 000000          .WORD   0            ;REGISTER INDEX
893 045344 045370          3$:      5$          ;ERROR RETURN ADDRESS
894 045346 012623          MOV      (SP)+,(R3)+ ;STORE THE REGISTER CONTENTS
895 045350 023727 045342 000046  4$:      CMP      3$,#RPEC2  ;REACHED THE END ?
896 045356 001406          BEQ      6$          ;BR IF YES
897 045360 062737 000002 045342  ADD      #2,3$       ;INCREMENT THE REGISTER INDEX
898 045366 000751          BR       1$          ;CONTINUE READING THE REGISTERS
899
900 045370 004737 042670    5$:      JSR      PC,C17      ;PROCESS THE UNCORRECTABLE PARITY ERROR
901 045374 013746 001234    6$:      MOV      $CPUOP,-(SP) ;CHECK THE CPU (RH) TYPE
902 045400 042716 003777    BIC      #^C174000,(SP) ;LEAVE THE CPU TYPE BITS
903 045404 022726 030000    CMP      #30000,(SP)+ ;SEE IF RH70
904 045410 001004          BNE      7$          ;IF NE, NO
905 045412 063704 040566    ADD      RHEXT,R4    ;POINT TO RPBAE
906 045416 012423          MOV      (R4)+,(R3)+ ;STORE THE CONTENTS
907 045420 011413          MOV      (R4),(R3)   ;GET RPCS3
908 045422 104413          7$:      RESREG   ;RESTORE R0 - R5
909 045424 000207          RTS      PC          ;RETURN
910
911          ;ROUTINE TO SET THE INTERRUPT ENABLE (IE) BIT IN RPCS1 WITHOUT GETTING A
912          ;TRANSFER ERROR (TRE).
913          ;CALL
914          ;:
915          ;:      MOV      #DRVNUM,R1    ;DRIVE NUMBER
916          ;:      JSR      PC,SET.IE    ;SET INTERRUPT ENABLE ROUTINE
917          ;:      RETURN
918          SET.IE: MOV      R4,-(SP)    ;SAVE R4
919          045430 013704 040560  MOV      RPADR,R4    ;PICKUP ADDRESS OF RPCS1
920          045434 010164 000010  MOV      R1,RPCS2(R4) ;SELECT DRIVE
921          045440 011446          MOV      (R4)-,(SP)  ;READ RPCS1
922          045442 052716 040000  BIS      #BIT14,(SP) ;SET THE "TRE" BIT OF THE WORD READ
923          045446 000316          SWAB    (SP)        ;ADJUST FOR DATO
924          045450 112714 000100  MOVB    #BIT06,(R4)  ;SET "IE"
925          045454 032764 010000 000010  BIT      #BIT12,RPCS2(R4) ;IS "NED"=1?
926          045462 001002          BNE      1$          ;YES--CLEAR "TRE"
927          045464 005726          TST     (SP)+       ;CLEAN OFF THE STACK
928          045466 000402          BR       2$
929
930 045470 112664 000001    1$:      MOVB    (SP)+,1(R4) ;CLEAR "TRE"
931 045474 012604          2$:      MOV      (SP)+,R4  ;RESTORE R4
932 045476 000207          RTS      PC          ;RETURN TO CALLER
933
934          ;QUEUE COUNT
935          QCNT:  .BYTE   0            ;DRIVE 0
936          045500 000          .BYTE   0            ;DRIVE 1
937          045501 000          .BYTE   0            ;DRIVE 2
938          045502 000          .BYTE   0            ;DRIVE 3
939          045503 000          .BYTE   0            ;DRIVE 4
940          045504 000          .BYTE   0            ;DRIVE 5
941          045505 000          .BYTE   0            ;DRIVE 6
942          045506 000          .BYTE   0

```



```

942 045507      000                .BYTE      0                ;DRIVE 7
943
944                ;QUEUE INPUT POINTERS
945
946 045510      045572      QINPT:  .WORD      QDRV0                ;DRIVE 0
947 045512      045612                .WORD      QDRV1                ;DRIVE 1
948 045514      045632                .WORD      QDRV2                ;DRIVE 2
949 045516      045652                .WORD      QDRV3                ;DRIVE 3
950 045520      045672                .WORD      QDRV4                ;DRIVE 4
951 045522      045712                .WORD      QDRV5                ;DRIVE 5
952 045524      045732                .WORD      QDRV6                ;DRIVE 6
953 045526      045752                .WORD      QDRV7                ;DRIVE 7
954
955                ;QUEUE OUTPUT POINTERS
956
957 045530      045572      QOUTPT: .WORD      QDRV0                ;DRIVE 0
958 045532      045612                .WORD      QDRV1                ;DRIVE 1
959 045534      045632                .WORD      QDRV2                ;DRIVE 2
960 045536      045652                .WORD      QDRV3                ;DRIVE 3
961 045540      045672                .WORD      QDRV4                ;DRIVE 4
962 045542      045712                .WORD      QDRV5                ;DRIVE 5
963 045544      045732                .WORD      QDRV6                ;DRIVE 6
964 045546      045752                .WORD      QDRV7                ;DRIVE 7
965
966 045550      045572      QSTART: .WORD      QDRV0                ;DRIVE 0 START ADDRESS
967 045552      045612      QSTOP:  .WORD      QDRV1                ;DRIVE 0 STOP ADDRESS & DRIVE 1 START ADDRESS
968 045554      045632                .WORD      QDRV2                ;STOP DRIVE 1--START DRIVE 2
969 045556      045652                .WORD      QDRV3                ;STOP DRIVE 2--START DRIVE 3
970 045560      045672                .WORD      QDRV4                ;STOP DRIVE 3--START DRIVE 4
971 045562      045712                .WORD      QDRV5                ;STOP DRIVE 4--START DRIVE 5
972 045564      045732                .WORD      QDRV6                ;STOP DRIVE 5--START DRIVE 6
973 045566      045752                .WORD      QDRV7                ;STOP DRIVE 6--START DRIVE 7
974 045570      045772                .WORD      QTERP                ;STOP DRIVE 7
975
976                ;DRIVE REQUEST QUEUES
977
978 045572      QDRV0:  .BLKW      10
979 045612      QDRV1:  .BLKW      10
980 045632      QDRV2:  .BLKW      10
981 045652      QDRV3:  .BLKW      10
982 045672      QDRV4:  .BLKW      10
983 045712      QDRV5:  .BLKW      10
984 045732      QDRV6:  .BLKW      10
985 045752      QDRV7:  .BLKW      10
986                QTERP=.
987
988                ;ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
989
990                ;CALL
991                ;      JSR      PC,CLRQUE
992
993 045772      104412      CLRQUE: SAVREG                ;SAVE R0 - R5
994 045774      012702      MOV      #QCNT,R2                ;ZERO THE QUEUE COUNTS
995 046000      005022      CLR      (R2)+                ;DRIVES 0 & 1
996 046002      005022      CLR      (R2)+                ;DRIVES 2 & 3
997 046004      005022      CLR      (R2)+                ;DRIVES 4 & 5
998 046006      005022      CLR      (R2)+                ;DRIVES 6 & 7
  
```

```

999 046010 012703 000010      MOV      #8,R3          ;MOVE THE STARTING
1000 046014 012701 045550      MOV      #QSTART,R1     ;ADDRESS OF THE QUEUE INTO
1001 046020 012122             1$:      MOV      (R1)+,(R2)+  ;THE QUEUE INPUT POINTER
1002 046022 005303             DEC      R3
1003 046024 001375             BNE     1$
1004 046026 012703 000010      MOV      #8,R3          ;MOVE THE STARTING ADDRESS
1005 046032 012701 045550      MOV      #QSTART,R1     ;OF THE QUEUE INTO THE
1006 046036 012122             2$:      MOV      (R1)+,(R2)+  ;QUEUE OUTPUT POINTER
1007 046040 005303             DEC      R3
1008 046042 001375             BNE     2$
1009 046044 104413             RESREG
1010 046046 000207             RTS      PC              ;RESTORE R0 - R5
1011
1012             ;EMPTY THE QUEUE SPECIFIED BY R1
1013             ;CALL
1014             ;
1015             ;      MOV      #DRVNUM,R1          ;DRIVE NUMBER
1016             ;      JSR      PC,EMPTYQ
1017
1018 046050 105061 045500      EMPTYQ: CLR      QCNT(R1) ;CLEAR NUMBER OF ITEMS IN QUEUE
1019 046054 006301             ASL      R1
1020 046055 016161 045510 045530      MOV      QINPT(R1),QOUTPT(R1) ;SET OUTPUT QUEUE POINTER=INPUT POINTER
1021 046064 006201             ASR      R1
1022 046066 000207             RTS      PC
1023
1024             ;ROUTINE TO PUT A REQUEST IN QUEUE
1025             ;CALL
1026             ;
1027             ;      MOV      #DPB,R2          ;ADDRESS OF DRIVE PARAMETER BLOCK
1028             ;      MOV      #DRVNUM,R1     ;DRIVE NUMBER
1029             ;      JSR      R0,DRVQUE     ;GO PUT REQUEST IN QUEUE
1030             ;      -----
1031             ;      -----
1032             ;      ;RETURN HERE IF QUEUE IS FULL
1033             ;      ;RETURN HERE IF REQUEST IS IN QUEUE
1033 046070 122761 000010 045500      DRVQUE: CMP      #10,QCNT(R1) ;IS QUEUE FULL?
1034 046076 001421             BEQ     2$              ;BR IF YES-TAKE RETURN1
1035 046100 105261 045500             INCB    QCNT(R1)       ;INCREMENT QUEUE COUNT
1036 046104 006301             ASL      R1
1037 046106 010271 045510             MOV      R2,@QINPT(R1) ;PUT THIS REQUEST IN QUEUE
1038 046112 062761 000002 045510             ADD     #2,QINPT(R1)   ;UPDATE THE QUEUE POINTER
1039 046120 026161 045510 045552             CMP     QINPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER
1040 046126 001003             BNE     1$              ;BRANCH IF NO
1041 046130 016161 045550 045510             MOV     QSTART(R1),QINPT(R1) ;YES--RESET POINTER
1042 046136 006201             1$:      ASR      R1
1043 046140 005720             TST     (R0)+          ;TAKE RETURN 2
1044 046142 000200             2$:      RTS      R0          ;RETURN TO USER
1045
1046             ;ROUTINE TO GET THE "DPB" ADDRESS OF NEXT REQUEST IN QUEUE
1047             ;
1048             ;ON RETURN, R2 WILL CONTAIN POINTER ADDRESS OF "DPB" REQUESTED OR
1049             ;ZERO IF NO REQUEST IN QUEUE.
1050             ;CALL
1051             ;
1052             ;      MOV      #DRVNUM,R1     ;DRIVE NUMBER
1053             ;      JSR      PC,GETREQ     ;GO GET THE REQUEST
1054
1055 046144 005002      GETREQ: CLR      R2

```

```

1056 046146 105761 045500          TSTB   QCNT(R1)          ;IS THERE ANY REQUEST IN QUEUE?
1057 046152 001404          BEQ    2$              ;NO---BRANCH
1058 046154 006301          1$:   ASL   R1
1059 046156 017102 045530          MOV   @QOUTPT(R1),R2 ;PICKUP 'DPB' POINTER FOR THIS DRIVE
1060 046162 006201          ASR   R1
1061 046164 000207          2$:   RTS   PC          ;RETURN TO USER
1062
1063          ;ROUTINE TO 'POP' THE REQUEST FROM QUEUE
1064          ;ON RETURN, R2 WILL CONTAIN POINTER ADDRESS OF 'DPB' REMOVED
1065          ;CALL
1066          ;
1067          ;
1068          ;   MOV   #DRVNUM,R1          ;DRIVE NUMBER
1069          ;   JSR   PC,POPQUE         ;CALL TO REMOVE REQUEST
1070
1071 046166 105361 045500          POPQUE: DECB  QCNT(R1)          ;DECREMENT QUEUE COUNT
1072 046172 006301          ASL   P1
1073 046174 017102 045530          MOV   @QOUTPT(R1),R2 ;GET THE 'DPB' POINTER
1074 046200 005071 045530          CLR   @QOUTPT(R1)    ;REMOVE DPB ADDRESS FROM THE QUEUE
1075 046204 062761 000002 045530          ADD   #2,QOUTPT(R1)  ;UPDATE THE QUEUE POINTER
1076 046212 026161 045530 045552          CMP   QOUTPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER?
1077 046220 001003          BNE   1$              ;NO--BRANCH TO EXIT
1078 046222 016161 045550 045530          MOV   QSTART(R1),QOUTPT(R1) ;YES--RESET THE POINTER
1079 046230 006201          1$:   ASR   R1
1080 046232 000207          RTS   PC          ;RETURN TO USER
  
```

```

1      .SBTTL  DEVICE PARAMETER BLOCKS
2
3      ;BLOCK LOCATION EQUATE STATEMENTS
4
5      :
6      000001      $FMT      = 0      ;DRIVE NUMBER (BYTE)
7      000002      $COMND    = 1      ;FMT,HCI,ECI OR OFFSET CODE (BYTE)
8      000003      $PSEL     = $FMT+1 ;OPERATION CODE (BYTE)
9      000004      $WCNT     = $FMT+2 ;PORT SELECT & BITS A16, A17 (BYTE)
10     000006      $SBUF     = $FMT+3 ;WORD COUNT (2'S COMP)
11     000010      $SEC      = $FMT+5 ;BUFFER ADDR OR REGISTER TABLE POINTER
12     000011      $STRK     = $FMT+7 ;SECTOR ADDRESS OR 1ST REG ADDR
13     000012      $CYL     = $FMT+10 ;TRACK ADDRESS OF LAST REG ADDR
14     000014      $REG      = $FMT+11 ;CYLINDER ADDR
15     000016      $STATUS   = $FMT+13 ;REGISTER STORAGE (IF ERROR)
16
17     ;DRIVE'S HISTORY AND CURRENT INDICATOR STORAGE EQUATES
18
19     000020      $WRDL     = $FMT+17 ;WORD COUNT (NOT 2'S COMP)
20     000022      $SSEC     = $WRDL+2 ;SECTOR SIZE FOR CURRENT OPERATION (256. OR 258.)
21     000024      $CODE     = $WRDL+4 ;PRESENT COMMAND SELECTION CODE
22     000026      $PACK     = $WRDL+6 ;READ/WRITE COMMAND INDICATOR (BYTE)
23     000027      $PREVO    = $WRDL+7 ;PREVIOUS COMMAND SELECTION CODE (BYTE)
24     000030      $PATT     = $WRDL+10 ;PATTERN CODE
25     000032      $PREVA    = $WRDL+12 ;PREVIOUS ADDRESS- TRK, SEC, CYL (DOUBLE WORD)
26     000036      $RDPAS    = $WRDL+16 ;WORDS READ PER PASS (DOUBLE WORD)
27     000042      $WTPAS    = $WRDL+22 ;WORDS WRITTEN PER PASS (DOUBLE WORD)
28     000046      $SEKES    = $WRDL+26 ;NUMBER OF SEKES PER PASS (DOUBLE WORD)
29     000052      $OPERC    = $WRDL+32 ;OPERATION COUNT (DOUBLE WORD)
30     000056      $WTOTL    = $WRDL+36 ;TOTAL WORDS WRITTEN X10^6 (DOUBLE WORD) &
31                                     ;1 X10^6 REPETITION COUNTER (DOUBLE WORD)
32     000066      $RTOTL    = $WRDL+46 ;TOTAL WORDS READ X10^6 (DOUBLE WORD) &
33                                     ;1 X10^6 REPETITION COUNTER (DOUBLE WORD)
34     000076      $STOTL    = $WRDL+56 ;TOTAL SEEK COUNT (DOUBLE WORD)
35     000102      $TOTAL    = $WRDL+62 ;TOTAL ERRORS COUNT (ALL TYPES)
36     000104      $SOFT     = $WRDL+64 ;'SOFT' ERROR COUNT
37     000106      $HARD     = $WRDL+66 ;'HARD' ERROR COUNT
38     000110      $SKI      = $WRDL+70 ;'SKI' ERROR COUNT
39     000112      $MISPO    = $WRDL+72 ;PROG DETECTED MIS-POSITIONING ERROR COUNT
40     000114      $PASSC    = $WRDL+74 ;PASS COUNTER
41     000116      $FAIR     = $WRDL+76 ;OPERATION QUEUE 'FAIRNESS' COUNT
42     000120      $HLDWC    = $WRDL+100 ;HOLD WORD FOR 'RELBUF' ROUTINE
43
44     ;INDEX EQUATES TO THE NEXT OPERATION PARAMETERS
45
46     000122      $NCODE    = $WRDL+102 ;NEXT OPERATION CODE
47     000123      $NPATC    = $NCODE+1 ;NEXT PATTERN
48     000124      $NSEC     = $NCODE+2 ;NEXT SECTOR
49     000125      $NTRK     = $NCODE+3 ;NEXT TRACK
50     000126      $NCYL     = $NCODE+4 ;NEXT CYLINDER
51     000130      $NEXT     = $NCODE+6 ;PARAMETER SELECTION INDICATOR
52     000132      $FIRST    = $NCODE+10 ;FIRST OPERATION INDICATOR
53
54     ;INDEX EQUATES FOR MAXIMUM/MINIMUM ADDRESSES
55
56     000134      MAXCYL    = $NCODE+12 ;MAXIMUM CYLINDER ADDRESS
57     000136      MINCYL    = MAXCYL+2 ;MINIMUM CYLINDER ADDRESS
    
```

```

58          000140          MAXTRK = MAXCYL+4          :MAXIMUM TRACK ADDRESS
59          .              MINTRK = MAXCYL+6          :MINIMUM TRACK ADDRESS
60          000144          MAXSEC = MAXCYL+10         :MAXIMUM SECTOR ADDRESS
61          000146          MINSEC = MAXCYL+12         :MINIMUM SECTOR ADDRESS
62
63          ;INDEX EQUATES FOR CYLLMT, TRKLMT, SECLMT ADDRESSES LIMITS AND 1ST FE CYLINDER
64
65          000150          CYLLMT = MAXCYL+14         :CYLINDER ADDRESS LIMIT
66          000152          SECLMT = CYLLMT+2          :SECTOR ADDRESS LIMIT
67          000154          TRKLMT = CYLLMT+4          :TRACK ADDRESS LIMIT
68          000156          FE1   = CYLLMT+6           :1ST FE CYLINDER
69
70          ;DRIVE SERIAL NUMBER AREA INDEX EQUATE
71
72          000160          $DRVSN = CYLLMT+10         :DRIVE SERIAL NUMBER (6 BYTES)
73
74          ;RP/RH REGISTER EQUATES
75
76          000166          $RPCS1 = $DRVSN+6          :RP REGISTER STORAGE
77          000170          $RPWC  = $RPCS1+2
78          000172          $RPBA  = $RPCS1+4
79          000174          $RPDA  = $RPCS1+6
80          000176          $RPCS2  = $RPCS1+10
81          000200          $RPDS  = $RPCS1+12
82          000202          $RPER1  = $RPCS1+14
83          000204          $RPAS  = $RPCS1+16
84          000206          $RPPLA = $RPCS1+20
85          000210          $RPDB  = $RPCS1+22
86          000212          $RPMR1  = $RPCS1+24
87          000214          $RPDT  = $RPCS1+26
88          000216          $RPSN  = $RPCS1+30
89          000220          $RPOF  = $RPCS1+32
90          000222          $RPDC  = $RPCS1+34
91          000224          $RPCC  = $RPCS1+36
92          000226          $RPER2  = $RPCS1+40
93          000230          $RPER3  = $RPCS1+42
94          000232          $RPEC1  = $RPCS1+44
95          000234          $RPEC2  = $RPCS1+46
96          000236          $RPBAE  = $RPCS1+50
97          000240          $RPCS3  = $RPCS1+52
98
106         046234          000          000          ;DPB FOR DRIVE 0
          046236          .BYTE          0,0          :DRIVE NUMBER 0
          046250          046422          .BLKW          5
          046252          .WORD          .+$RPCS1-$REG
          .BLKB          $RPCS3-$REG
          ;DPB FOR DRIVE 1
          046476          001          000          ;DRIVE NUMBER 1
          046500          .BYTE          1,0
          046512          046664          .BLKW          5
          046514          .WORD          .+$RPCS1-$REG
          .BLKB          $RPCS3-$REG
          ;DPB FOR DRIVE 2
          046740          002          000          ;DRIVE NUMBER 2
          046742          .BYTE          2,0
          046754          047126          .BLKW          5
          .WORD          .+$RPCS1-$REG
    
```

```

046756                .BLKB  $RPCS3-$REG

047202      003      000      ;DPB FOR DRIVE 3
047204      047370      DRIVE3: .BYTE  3,0                ;DRIVE NUMBER 3
                                .BLKW  5
                                .WORD  .+$RPCS1-$REG
                                .BLKB  $RPCS3-$REG

047444      004      000      ;DPB FOR DRIVE 4
047446      047632      DRIVE4: .BYTE  4,0                ;DRIVE NUMBER 4
                                .BLKW  5
                                .WORD  .+$RPCS1-$REG
                                .BLKB  $RPCS3-$REG

047706      005      000      ;DPB FOR DRIVE 5
047710      050074      DRIVE5: .BYTE  5,0                ;DRIVE NUMBER 5
                                .BLKW  5
                                .WORD  .+$RPCS1-$REG
                                .BLKB  $RPCS3-$REG

050150      006      000      ;DPB FOR DRIVE 6
050152      050336      DRIVE6: .BYTE  6,0                ;DRIVE NUMBER 6
                                .BLKW  5
                                .WORD  .+$RPCS1-$REG
                                .BLKB  $RPCS3-$REG

050412      007      000      ;DPB FOR DRIVE 7
050414      050600      DRIVE7: .BYTE  7,0                ;DRIVE NUMBER 7
                                .BLKW  5
                                .WORD  .+$RPCS1-$REG
                                .BLKB  $RPCS3-$REG

107
108
109      ;GENERAL PURPOSE DEVICE PARAMETER BLOCK
110 050654      000      GENDPB: .BYTE  0                ;DRIVER PARAMETER BLOCK, DRIVE #
111 050655      000      .BYTE  0                ;OFFSET VALUE OR FMT16, HCI OR ECI
112 050656      000      .BYTE  0                ;COMMAND CODE
113 050657      000      .BYTE  0                ;PSEL, A16 AND A17
114 050660      177776 .WORD  -2                ;WORD COUNT (NEG)
115 050662      063324 .WORD  CYLNDR            ;BUFFER ADDRESS
116 050664      000      .BYTE  0                ;SECTOR ADDRESS
117 050665      000      .BYTE  0                ;TRACK ADDRESS
118 050666      000000 .WORD  0                ;CYLINDER ADDRESS
119 050670      050674 .WORD  GENREG            ;ADDRESS TO SAVE ALL RHXX/RP07 REG'S
120 050672      000000 .WORD  0                ;STATUS WORD
121
122 050674      GENREG: .BLKW  24                ;REGISTER STORAGE
    
```

```

1
2          .SBTTL  ERROR MESSAGES
3 050744    122    110    040  EM1:  .ASCIZ  /RH CONTROLLER INTERRUPT OCCURRED (RPAS= 0)/
4 051017    125    116    105  EM2:  .ASCIZ  /UNEXPECTED ATTENTION OCCURRED/
5 051055    115    101    123  EM3:  .ASCIZ  /MASSBUS PARITY ERROR (MCPE=1)/
6 051113    115    101    123  EM4:  .ASCIZ  /MASSBUS PARITY ERROR (PAR=1)/
7 051150    101    104    104  EM5:  .ASCIZ  /ADDRESS PLUG CHANGE BIT SET/
8 051204    122    110    040  EM6:  .ASCIZ  /RH CONTROLLER DIDN'T RESPOND TO ADDRESSING/
9 051257    125    116    103  EM10: .ASCIZ  /UNCORRECTABLE MASSBUS PARITY ERROR/
10 051322   106    101    124  EM11: .ASCIZ  /FATAL MASSBUS PARITY ERROR/
11 051355   120    105    122  EM12: .ASCIZ  /PERSISTENT DEVICE UNSAFE/
12 051406   117    120    105  EM13: .ASCIZ  /OPERATION NOT COMPLETED WITHIN TIME LIMIT/
13 051460   104    122    111  EM14: .ASCIZ  /DRIVE WENT OFFLINE/
14 051503   116    117    040  EM15: .ASCIZ  /NO RESPONSE TO PORT REQUEST/
15 051537   110    105    101  EM20: .ASCIZ  /HEADER CRC ERROR/
16 051560   104    101    124  EM21: .ASCIZ  /DATA CHECK ('DCK') ERROR/
17 051611   127    122    111  EM22: .ASCIZ  /WRITE CHECK ERROR - DATA CHECK ('DCK') SET/
18 051664   127    122    111  EM23: .ASCIZ  /WRITE CHECK ERROR - DATA CHECK ('DCK') NOT SET/
19 051743   110    105    101  EM24: .ASCIZ  /HEADER READ ERROR - 'FMT' BIT DROPPED/
20 052011   110    105    101  EM25: .ASCIZ  /HEADER READ ERROR - HEADER COMPARE ('HCE') ERROR/
21 052072   106    117    122  EM26: .ASCIZ  /FORMAT ERROR ('FER')/
22 052117   110    105    101  EM27: .ASCIZ  /HEADER COMPARE ('HCE') ERROR/
23 052154   115    111    123  EM30: .ASCIZ  /MISCELLANEOUS DRIVE ERROR/
24 052206   117    120    105  EM31: .ASCIZ  /OPERATION INCOMPLETE ('OPI') ERROR/
25 052251   104    122    111  EM32: .ASCIZ  /DRIVE TIMING ('DTE') ERROR/
26 052304   120    101    122  EM33: .ASCIZ  /PARITY ('PAR') ERROR AFTER OPERATION STARTED/
27 052361   127    122    111  EM34: .ASCIZ  /WRITE CLOCK FAILURE ('WCF') ERROR/
28 052423   111    116    126  EM35: .ASCIZ  /INVALID ADDRESS ('IAE') ERROR/
29 052461   127    122    111  EM36: .ASCIZ  /WRITE LOCK ('WLE') ERROR/
30 052512   104    101    124  EM37: .ASCIZ  /DATA CHECK ('DCK') SET DURING WRITE CHECK/
31 052564   122    110    130  EM40: .ASCIZ  /RHXX OR UNIBUS TRANSFER ERROR/
32 052622   102    125    123  EM41: .ASCIZ  /BUS ADDRESS OR WORD COUNT INCORRECT/
33 052666   104    101    124  EM42: .ASCIZ  /DATA COMPARE ERRORS - NO OTHER ERROR(S) DETECTED/
34 052747   103    101    116  EM43: .ASCIZ  /CAN'T MATCH DATA READ WITH A PATTERN - UNKNOWN DATA PATTERN/
35 053043   105    122    122  EM44: .ASCIZ  /ERROR BIT(S) SET, BUT NO ERROR SIGNALLED BY THE RH CONTROLLER/
36 053140   105    103    103  EM45: .ASCIZ  /ECC LOGIC FAILURE - POSITION REGISTER VALUE NOT VALID/
37 053226   102    125    123  EM46: .ASCIZ  /BUS ADDRESS AND WORD COUNT NOT CONSISTENT/
38 053300   105    103    103  EM47: .ASCIZ  /ECC LOGIC FAILURE - PATTERN REGISTER IS ZERO/
39 053355   123    105    105  EM50: .ASCIZ  /SEEK INCOMPLETE ('SKI') ERROR/
40 053413   120    122    117  EM51: .ASCIZ  /PROGRAM DETECTED POSITIONING ERROR/
41 053456   105    103    110  EM52: .ASCIZ  /ECC ERROR - UNCORRECTABLE ECC ERROR/
42 053522   104    122    111  EM60: .ASCIZ  /DRIVE UNSAFE ERROR/
43 053545   105    101    122  EM70: .ASCIZ  /EARLY WARNING, TEMPERATURE WARNING (TPE) ERROR/
44 053624   105    101    122  EM71: .ASCIZ  /EARLY WARNING, AIR SYSTEM WARNING (AIR) ERROR/
45 053702   105    101    122  EM72: .ASCIZ  /EARLY WARNING ERROR, AIR, TPE, NOT SET/
    
```


				.SBTTL	ASCII MESSAGES
1					
2					
4	054536	120	122	123	LIN2C: .ASCIZ /PRNT COMMAND= /
5	054556	040	040	120	LIN2P: .ASCIZ / PREVS COMAND= /
6	054577	052	040	105	LIN2S: .ASCIZ @* ERROR AT BAD TRACK/SECTOR@
7	054633	105	122	122	LINM3: .ASCIZ /ERROR AT C/
8	054646	040	124	000	T: .ASCIZ / T/
9	054651	120	122	123	LINN3: .ASCIZ /PRNT ADDR= C/
10	054667	040	123	000	S: .ASCIZ / S/
11	054672	040	040	120	LINP3: .ASCIZ / PREVS ,DDR= C/
12	054712	123	124	101	LINS3: .ASCIZ /START CYL= /
13	054726	040	040	105	LINEN3: .ASCIZ / END CYL= /
14	054742	040	040	101	LINA3: .ASCIZ / ACTUAL CYL= /
15	054761	040	040	124	LINT3: .ASCIZ / TRK= /
16	054771	040	122	120	LINCA3: .ASCIZ / RPDC= /
17	055001	122	120	104	LINDA3: .ASCIZ /RPDA= /
18	055010	122	120	102	LINB3: .ASCIZ /RPBA= /
19	055017	040	040	122	LINW3: .ASCIZ / RPWC= /
20	055030	123	124	101	LINST3: .ASCIZ /START TRK= /
21	055044	123	124	101	LINSS3: .ASCIZ /START SEC= /
22	055060	102	125	106	LINM4: .ASCIZ /BUFFER ADDR= /
23	055076	040	040	127	LINS4: .ASCIZ / WRD CRT= /
24	055112	040	040	116	LINX4: .ASCIZ / NMBR WRDS XFRD= /
25	055135	105	130	120	LIND5: .ASCIZ /EXPCTD DATA= /
26	055153	040	040	122	LINB5: .ASCIZ / RECEVD DATA= /
27	055173	040	040	127	LINP5: .ASCIZ / WORD POS= /
28	055210	110	105	101	LINS5: .ASCIZ /HEADER FROM ERROR SECTOR= /
29	055243	122	120	105	LINP5: .ASCIZ /RPEC1= /
30	055253	040	040	122	LINE05: .ASCIZ / RPEC2= /
31	055265	123	105	103	LINB6: .ASCIZ /SECTOR IS ECC CORRECTABLE /
32	055320	123	105	103	LINC6: .ASCIZ /SECTOR READ CORRECTLY AFTER /
33	055355	103	117	122	LING6: .ASCIZ /CORRECTED ON /
34	055373	040	122	105	LINR6: .ASCIZ / RETRY(S)/
35	055405	125	116	103	LINU06: .ASCIZ /UNCORRECTABLE AFTER /
36	055432	040	040	115	LIN7M: .ASCIZ / MIS POS ERRORS= /
40	055455	124	117	124	LIN7P: .ASCIZ /TOTALS; SEEKS= /
41	055475	040	040	123	LIN7S: .ASCIZ / SKI ERRORS= /
42	055514	124	117	124	LIN7T: .ASCIZ /TOTALS; ERRORS= /
43	055535	040	127	122	LIN7X: .ASCIZ / WRDS WRITN= /
44	055553	040	127	122	LIN7R: .ASCIZ / WRDS READ= /
45					
46	055570	105	122	122	LIN8M: .ASCIZ /ERROR DURING RETRY/
47	055613	104	101	124	LIN9B: .ASCIZ /DATA COMPARISON ERRORS/
48	055642	040	040	040	LIN9H: .ASCII / WORD EXPCTD RECEVD/<CRLF>
49	055700	101	104	104	.ASCIZ /ADDR POS DATA DATA/<CRLF>
50	055735	040	040	040	LIN9I: .ASCII / WORD RECEVD/<CRLF>
51	055763	101	104	104	.ASCIZ /ADDR POS DATA/<CRLF>
52	056010	124	117	124	LIN9E: .ASCIZ /TOTAL COMPARE ERRORS= /
53	056037	124	110	105	LIN9G: .ASCIZ /THE DATA COMPARED OK/<CRLF>
54	056065	105	122	122	LIN10A: .ASCIZ /ERROR BURST BEGINS AT WORD /
55	056121	040	111	116	LIN10B: .ASCIZ / IN DATA FIELD OF ERROR SECTOR/<CRLF>
56	056161	105	122	122	LIN10C: .ASCII /ERROR WAS NOT IN THE DATA READ - /<CRLF>
57	056223	105	103	103	.ASCIZ /ECC CORRECTION CAN'T BE PERFORMED/
58	056265	105	103	103	LIN10H: .ASCII /ECC CORRECTION RESULTS/<CRLF>
59	056314	040	040	040	.ASCII / WORD BAD CORRECTED /<CRLF>
60	056356	101	104	104	.ASCIZ /ADDR POS DATA DATA/<CRLF>
61	056413	103	117	116	LIN11H: .ASCIZ /CONTENTS OF ERROR SECTOR (REPORTED ABOVE)/<CRLF>

62	056466	101	104	104	LIN11: .ASCIZ	/ADDR	HEADER/<CRLF>
63	056507	101	104	104	LIN11A: .ASCIZ	/ADDR	DATA/<CRLF>
64	056526	040			BLNKS4: .ASCII	/ /	
65	056527	040			BLNKS3: .ASCII	/ /	
66	056530	040			BLNKS2: .ASCII	/ /	
67	056531	040	000		BLNKS1: .ASCIZ	/ /	
68	056533	122	105	124	LINX5: .ASCIZ	/RETRIEVED BY A RDHD COMMAND/	

62	060164	200	103	110	MSPRM:	.ASCIZ	<CRLF>/CHANGE DRIVE PARAMETERS (L) N ? /
83	060226	200	104	117	MESFE:	.ASCIZ	<CRLF>/DO YOU WANT TO WRITE ANYWHERE ON MEDIA (L) N ? /
84	060307	200			OVRWRT:	.ASCII	<CRLF>
85	060310	007	011	041		.ASCII	<BELL>/ ! CUSTOMER DATA WILL BE OVERWRITTEN !/<CRLF>
86	060360	007	011	055		.ASCII	<BELL>/ -----/<CRLF>
87	060430	103	117	116		.ASCIZ	/CONTINUE (L) ? /
88	060451	200	052	040	FEONLY:	.ASCIZ	<CRLF>/* TESTING WILL OCCUR ON FE CYLINDER ONLY */<CRLF>
89	060526	200	052	040	MREAD:	.ASCIZ	<CRLF>/* PROGRAM IS LOCKED IN 'READ ONLY' MODE */<CRLF>
91					.EVEN		

```

1
2
3 060604 060742 031000 001466 PARLST: .WORD PAR1,12800.,WPCNT
4 060612 061001 077777 001470 .WORD PAR2,32767.,INTRVL
5 060620 061271 077777 001462 .WORD PAR12,32767.,CMPTIM
6 060626 061515 077777 001474 .WORD PAR19,32767.,PASSES
7 060634 061061 000017 001476 .WORD PAR3,15.,PATTERN
8 060642 061224 000001 001500 .WORD PAR11,1,RANDWC
9 060650 061354 000007 001502 .WORD PAR14,7,RATIO
10 060656 061451 000001 001504 .WORD PAR16,1,ENDING
11 060664 061407 000001 001506 .WORD PAR15,1,WRTCHK
15 060672 061537 000001 001510 .WORD PAR21,1,RANDOM
22 060700 000000 .WORD 0 ;TABLE TERMINATOR
23
24 060702 040 057 040 SLASH: .ASCIZ @ / @
25 060706 200 103 110 ASKPAR: .ASCIZ <CRLF>/CHANGE PARAMETERS (L) N ? /
26 060742 115 101 130 PAR1: .ASCIZ /MAXIMUM WORD COUNT (6-12800.) /
27 061001 124 111 115 PAR2: .ASCIZ /TIME BETWEEN REPORTS (IN MINUTES, 0=NO REPORT) /
28 061061 104 101 124 PAR3: .ASCIZ /DATA PATTERN NUMBER (0=RANDOM, 1-15.=FIXED) /
29 061136 115 101 130 PAR4: .ASCIZ /MAX CYL /
30 061147 115 111 116 PAR5: .ASCIZ /MIN CYL /
31 061160 115 101 130 PAR6: .ASCIZ /MAX TRK /
32 061171 115 111 116 PAR7: .ASCIZ /MIN TRK /
33 061202 115 101 130 PAR8: .ASCIZ /MAX SEC /
34 061213 115 111 116 PAR9: .ASCIZ /MIN SEC /
38 061224 127 117 122 PAR11: .ASCIZ /WORD COUNT MODE (0=RANDOM, 1=FIXED) /
39 061271 124 111 115 PAR12: .ASCIZ /TIME BETWEEN DATA COMPARES (IN MINUTES, 0=ALWAYS) /
40 061354 122 105 101 PAR14: .ASCIZ /READ TO WRITE RATIO (0-7) /
41 061407 105 116 101 PAR15: .ASCIZ /ENABLE WRITE CHECK (0=NO, 1=YES) /
42 061451 105 116 104 PAR16: .ASCIZ /END OF PASS MODE (0=SEEKS, 1=DATA) /
43 061515 116 125 115 PAR19: .ASCIZ /NUMBER OF PASSES /
47 061537 123 105 105 PAR21: .ASCIZ /SEEK MODE (0=RANDOM, 1=SEQUENTIAL) /
51
52 .EVEN

```

1
2
3
4
5
8
9
10
11
20

.SBTTL DRIVE PARAMETER TABLES (DPB)

:PARAMETER TABLE POINTERS FOR ADDRESS LIMITS

TABLE:	.WORD	TABLE0	:PARAMETER TABLE FOR DRIVE 0
	.WORD	TABLE1	:PARAMETER TABLE FOR DRIVE 1
	.WORD	TABLE2	:PARAMETER TABLE FOR DRIVE 2
	.WORD	TABLE3	:PARAMETER TABLE FOR DRIVE 3
	.WORD	TABLE4	:PARAMETER TABLE FOR DRIVE 4
	.WORD	TABLE5	:PARAMETER TABLE FOR DRIVE 5
	.WORD	TABLE6	:PARAMETER TABLE FOR DRIVE 6
	.WORD	TABLE7	:PARAMETER TABLE FOR DRIVE 7

:PARAMETER TABLE FOR ADDRESS LIMITS

TABLE0:	.WORD	PAR5,0,MINCYL+DRIVE0
	.WORD	PAR4,630,MAXCYL+DRIVE0
	.WORD	PAR7,29,MINTRK+DRIVE0
	.WORD	PAR6,29,MAXTRK+DRIVE0
	.WORD	PAR9,33,MINSEC+DRIVE0
	.WORD	PAR8,33,MAXSEC+DRIVE0,0

TABLE1:	.WORD	PAR5,0,MINCYL+DRIVE1
	.WORD	PAR4,630,MAXCYL+DRIVE1
	.WORD	PAR7,29,MINTRK+DRIVE1
	.WORD	PAR6,29,MAXTRK+DRIVE1
	.WORD	PAR9,33,MINSEC+DRIVE1
	.WORD	PAR8,33,MAXSEC+DRIVE1,0

TABLE2:	.WORD	PAR5,0,MINCYL+DRIVE2
	.WORD	PAR4,630,MAXCYL+DRIVE2
	.WORD	PAR7,29,MINTRK+DRIVE2
	.WORD	PAR6,29,MAXTRK+DRIVE2
	.WORD	PAR9,33,MINSEC+DRIVE2
	.WORD	PAR8,33,MAXSEC+DRIVE2,0

TABLE3:	.WORD	PAR5,0,MINCYL+DRIVE3
	.WORD	PAR4,630,MAXCYL+DRIVE3
	.WORD	PAR7,29,MINTRK+DRIVE3
	.WORD	PAR6,29,MAXTRK+DRIVE3
	.WORD	PAR9,33,MINSEC+DRIVE3
	.WORD	PAR8,33,MAXSEC+DRIVE3,0

TABLE4:	.WORD	PAR5,0,MINCYL+DRIVE4
	.WORD	PAR4,630,MAXCYL+DRIVE4
	.WORD	PAR7,29,MINTRK+DRIVE4
	.WORD	PAR6,29,MAXTRK+DRIVE4
	.WORD	PAR9,33,MINSEC+DRIVE4
	.WORD	PAR8,33,MAXSEC+DRIVE4,0

TABLE5:	.WORD	PAR5,0,MINCYL+DRIVE5
	.WORD	PAR4,630,MAXCYL+DRIVE5
	.WORD	PAR7,29,MINTRK+DRIVE5
	.WORD	PAR6,29,MAXTRK+DRIVE5
	.WORD	PAR9,33,MINSEC+DRIVE5
	.WORD	PAR8,33,MAXSEC+DRIVE5,0

061604	061624		
061606	061672		
061610	061740		
061612	062006		
061614	062054		
061616	062122		
061620	062170		
061622	062236		
061624	061147	000000	046372
061632	061136	001166	046370
061640	061171	000035	046376
061646	061160	000035	046374
061654	061213	000041	046402
061662	061202	000041	046400
061672	061147	000000	046634
061700	061136	001166	046632
061706	061171	000035	046640
061714	061160	000035	046636
061722	061213	000041	046644
061730	061202	000041	046642
061740	061147	000000	047076
061746	061136	001166	047074
061754	061171	000035	047102
061762	061160	000035	047100
061770	061213	000041	047106
061776	061202	000041	047104
062006	061147	000000	047340
062014	061136	001166	047336
062022	061171	000035	047344
062030	061160	000035	047342
062036	061213	000041	047350
062044	061202	000041	047346
062054	061147	000000	047602
062062	061136	001166	047600
062070	061171	000035	047606
062076	061160	000035	047604
062104	061213	000041	047612
062112	061202	000041	047610
062122	061147	000000	050044
062130	061136	001166	050042
062136	061171	000035	050050
062144	061160	000035	050046
062152	061213	000041	050054
062160	061202	000041	050052

062170	061147	000000	050306	TABLE6:	.WORD	PAR5,0,MINCYL+DRIVE6
062176	061136	001166	050304		.WORD	PAR4,630,MAXCYL+DRIVE6
062204	061171	000035	050312		.WORD	PAR7,29,MINTRK+DRIVE6
062212	061160	000035	050310		.WORD	PAR6,29,MAXTRK+DRIVE6
062220	061213	000041	050316		.WORD	PAR9,33,MINSEC+DRIVE6
062226	061202	000041	050314		.WORD	PAR8,33,MAXSEC+DRIVE6,0
062236	061147	000000	050550	TABLE7:	.WORD	PAR5,0,MINCYL+DRIVE7
062244	061136	001166	050546		.WORD	PAR4,630,MAXCYL+DRIVE7
062252	061171	000035	050554		.WORD	PAR7,29,MINTRK+DRIVE7
062260	061160	000035	050552		.WORD	PAR6,29,MAXTRK+DRIVE7
062266	061213	000041	050560		.WORD	PAR9,33,MINSEC+DRIVE7
062274	061202	000041	050556		.WORD	PAR8,33,MAXSEC+DRIVE7,0

.SBTTL ROUTINE TO SIZE MEMORY

```

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

```

062304 010046
062306 010146
062310 013746 000114
062314 013746 000116
062320 012737 000116 000114
062326 012737 000002 000116
062334 013746 000004
062340 013746 000006
062344 010600

062346 104400
062350 012637 000006
062354 012737 062374 000004
062362 012701 020000
062366 005711
062370 005721
062372 000775
062374 162701 000002
062400 010006
062402 012637 000006
062406 012637 000004
062412 012637 000116
062416 012637 000114
062422 010137 062434
062426 012601
062430 012600
062432 000207
062434 000000
    
```

```

$SIZE:  MOV   RO,-(SP)      ;;SAVE RO ON THE STACK
        MOV   R1,-(SP)      ;;SAVE R1 ON THE STACK
        MOV   @#114,-(SP)   ;;SAVE MEMORY ERROR VECTOR PS & PC
        MOV   @#116,-(SP)
        MOV   #116,@#114   ;;IGNORE PARITY ERRORS WHILE SIZING
        MOV   #RTI,@#116
        MOV   @#ERRVEC,-(SP) ;;SAVE PRESENT ERROR VECTOR PS & PC
        MOV   @#ERRVEC+2,-(SP)
        MOV   SP,RO        ;;SAVE THE STACK POINTER
        ;;SET THE ERRVEC PS TO THE PRESENT PS
        TRAP
        MOV   (SP)+,@#ERRVEC+2 ;;PUSH OLD PSW AND PC ON STACK
        MOV   #2,@#ERRVEC    ;;SAVE THE PSW IN @#ERRVEC+2
        MOV   #20000,R1     ;;SET FOR TIME=OUT
        MOV   #20000,R1     ;;FIRST ADDRESS
i$:     TST   (R1)          ;;TEST THIS ADDRESS
        TST   (R1)+        ;;STEP TO NEXT ADDRESS
        BR    1$           ;;TRY ANOTHER
2$:     SUB   #2,R1         ;;DROP BACK
        MOV   RO,SP        ;;RESTORE THE STACK
        MOV   (SP)+,@#ERRVEC+2 ;;RESTORE ERROR VECTOR
        MOV   (SP)+,@#ERRVEC
        MOV   (SP)+,@#116   ;;RESTORE MEMORY ERROR VECTOR
        MOV   (SP)+,@#114
        MOV   R1,$LSTAD    ;;LAST ADDRESS
        MOV   (SP)+,R1     ;;RESTORE R1
        MOV   (SP)+,RO     ;;RESTORE RO
        RTS   PC
$LSTAD: .WORD 0           ;;CONTAINS THE LAST ADDRESS
    
```

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

.SBTTL BUSADR - GET BUS ADDRESS AND VECTOR ADDRESS

:THIS ROUTINE IS USED TO INSURE THE BUS ADDRESS OF THE RMXX/RP07
 :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
    :
    BUSADR: TST      CHGADR          ;INPUT FROM TTY REQUESTED?
            BGE      3$             ;NO--BRANCH
            CLR      CHGADR          ;YES--CLEAR THE REQUEST FLAG
            TYPE     ,BCRLF         ;CR-LF
    1$:     CLR      CFLAG          ;CLEAR CONTROL C FLAG
            MOV      #SRPADR,R0     ;FIRST ADDRESS
            TYPE     ,MRPCS1        ;"RPCS1="
            MOV      (R0),-(SP)     ;PRESENT RPCS1 ADDRESS
            TYPOC    ,BLNKS2        ;TYPE IT
            TYPE     ,BLNKS2        ;TYPE 2 BLANKS
            RDLIN   (SP)+,R1        ;GET THE ENTRY
            TST      CFLAG          ;ADDRESS OF ASCII TEXT
            BNE      1$             ;WAS (^C) TYPED?
            JSR      R5,CK.NUM      ;BR IF YES
            BR       1$             ;ENTER AND STORE THE NEW ADDRESS
            BR       1$             ;ERROR EXIT

    2$:     MOV      #SRPVEC,R0     ;VECTOR ADDRESS
            TYPE     ,MRPVEC        ;"RPVEC="
            MOV      (R0),-(SP)     ;PRESENT RMXX/RP07 VECTOR ADDRESS ON THE STACK
            TYPOC    ,BLNKS2        ;TYPE IT
            TYPE     ,BLNKS2        ;TYPE 2 BLANKS
            RDLIN   (SP)+,R1        ;READ THE ENTRY
            TST      CFLAG          ;ASCII TEXT ADDRESS
            BNE      1$             ;WAS (^C) TYPED?
            JSR      R5,CK.NUM      ;BR IF YES
            BR       2$             ;ENTER AND STORE NEW ADDRESS
            BR       2$             ;ERROR EXIT

    3$:     MOV      #SRPADR,R0     ;FIRST ADDRESS OF NEW PARAMETERS
            MOV      #RPADR,R1      ;FIRST ADDRESS OF WHERE TO PUT THEM
            MOV      (R0)+,(R1)+    ;BUS ADDRESS
            MOV      (R0)+,(R1)+    ;VECTOR ADDRESS
            RTS      PC             ;RETURN
    
```

103 MRPCS1: .ASCIZ @RPCS1=@
 126 MRPVEC: .ASCIZ @RPVEC=@

1			.SBTTL CK.NUM - (CHECK NUMBER (OCTAL))	
2			:THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL CHARACTERS	
3			:AND FORMS AN OCTAL NUMBER IN R2	
4			:CALL:	
5			:MOV #ADR,R1 :ADRS OF ASCII STRING	
6			:JSR R5,CK.NUM :R5 CHANGED	
7			:RET :ERROR EXIT	
8			:RET :NORMAL EXIT	
9				
10	062614	010246	CK.NUM: MOV R2,-(SP)	:SAVE R2
11	062616	010346	MOV R3,-(SP)	:SAVE R3
12	062620	010446	MOV R4,-(SP)	:SAVE R4
13	062622	012703	MOV #6,R3	:MAX OCTAL DIGITS IN THE NUMBER
14	062626	005002	CLR R2	:FINAL OCTAL VALUE
15	062630	112104	1\$: MOVB (R1)+,R4	:GET CURRENT POINTED BYTE
16	062632	001424	BEQ 3\$:BRANCH,IF TERMINATOR DETECTED
17	062634	120427	000060 CMPB R4,#'0	:SMALLER THAN ASCII-C
18	062640	103425	BLO 5\$:YES,ERROR EXIT
19	062642	120427	000067 CMPB R4,#'7	:LARGER THAN ASCII-7 ?
20	062646	101022	BHI 5\$:YES,ERROR EXIT
21	062650	006302	ASL R2	:SHIFT LEFT
22	062652	103420	BCS 5\$	
23	062654	006302	ASL R2	:ONE
24	062656	103416	BCS 5\$	
25	062660	006302	ASL R2	:OCTAL DIGIT
26	062662	103414	BCS 5\$:ERROR IF CARRY BIT SET
27	062664	042704	177770,R4	:CHOP OFF HIGHER BITS
28	062670	060402	ADD R4,R2	:APPENDING CURRENT DIGIT TO NUMBER
29	062672	005303	DEC R3	:DECREMENT BYTE COUNT
30	062674	001401	BEQ 2\$:BRANCH,IF LAST BYTE
31	062676	000754	BR 1\$:LOOPING BACK
32				
33	062700	112104	2\$: MOVB (R1)+,R4	:CHECK TERMINATOR
34	062702	001004	BNE 5\$:ERROR EXIT
35	062704	005702	3\$: TST R2	:FINAL VALUE= 0
36	062706	001401	BEQ 4\$:YES,THEN NOT REPLACE THE ORIGINAL VALUE
37	062710	010210	MOV R2,(R0)	:REPLACE THE ORIGINAL VALUE
38	062712	005725	4\$: TST (R5)+	:ADJUST FOR NORMAL RETURN
39	062714	012604	5\$: MOV (SP)+,R4	:RESTORE R4
40	062716	012603	MOV (SP)+,R3	:RESTORE R3
41	062720	012602	MOV (SP)+,R2	:RESTORE R2
42	062722	000205	RTS R5	:EXIT
71			.EVEN	
72	062724	000000	PATCH: .WORD 0	:ALLOCATE 200 OCTAL LOCATIONS FOR PATCHING
78				
79	063324		CYLNR: .BLKW 258.	:ONE SECTOR OF WORDS
80		064330	ENDPGM=.	

4				.SBTTL	HELP TEXT MESSAGE
5					
6	064330	200		MSHELP:	.ASCII <CRLF>
7	064331	007	124		.ASCII <BELL>/THE HELP MESSAGE CAN ONLY BE TYPED ONCE./
8	064402	007			.ASCII <BELL>
9	064403	200	104		.ASCIIZ <CRLF>/DO YOU WANT IT TYPED (L) ? /
10	064440	200		HELPTX:	.ASCII <CRLF>
11	064441	200	104		.ASCII <CRLF>/DRIVE PARAMETERS:/
12	064463	200	115		.ASCII <CRLF>/MAXCYL :max cylinder adrs/
13	064517	200	115		.ASCII <CRLF>/MINCYL :min cylinder adrs/
14	064553	200	115		.ASCII <CRLF>/MAXTRK :max track adrs/
15	064604	200	115		.ASCII <CRLF>/MINTRK :min track adrs/
16	064635	200	115		.ASCII <CRLF>/MAXSEC :max sector adrs/
17	064667	200	115		.ASCII <CRLF>/MINSEC :min sector adrs/
18	064721	200			.ASCII <CRLF>
19	064722	200	124		.ASCII <CRLF>/THE FOLLOWING ARE VALID COMMANDS:/
20	064764	200	040		.ASCII <CRLF>/ W<CR> assign drive and do SEQUENTIAL WRITE data/
21	065046	200	040		.ASCII <CRLF>/ R<CR> assign drive and do SEQUENTIAL READ data/
22	065127	200	040		.ASCII <CRLF>/ T<CR> assign drive to do (random or sequential) TEST/
23	065217	200	127		.ASCII <CRLF>/WTF<CR> assign drive, to do SEQUENTIAL WRITE data and/
24	065305	200	040		.ASCII <CRLF>/ to do (random or sequential) TEST/
25	065357	200	040		.ASCII <CRLF>/ D<CR> DROP a drive from test/
26	065416	200	040		.ASCII <CRLF>/ S<CR> type the performance SUMMARY report/
27	065472	200			.ASCII <CRLF>
28	065473	200	040		.ASCII <CRLF>/ # is the drive number (0-7 or 'A' for all drives)/
29	065557	200			.ASCII <CRLF>
30	065560	200	123		.ASCII <CRLF>/SWITCH REGISTER SETTINGS:/
31	065612	200	123		.ASCII <CRLF>/SW15= (100000) halt on error/
32	065650	200	123		.ASCII <CRLF>/SW14= (040000) /
33	065671	200	123		.ASCII <CRLF>/SW13= (020000) inhibit error timeout/
34	065737	200	123		.ASCII <CRLF>/SW12= (010000) /
35	065760	200	123		.ASCII <CRLF>/SW11= (004000) /
36	066001	200	123		.ASCII <CRLF>/SW10= (002000) ring tty bell on error/
37	066050	200	123		.ASCII <CRLF>/SW09= (001000) change end of pass to 1/4 of normal@
38	066134	200	123		.ASCII <CRLF>/SW08= (000400) inhibit end of pass messages/
39	066211	200	123		.ASCII <CRLF>/SW07= (000200) display all data compare errors/
40	066271	200	123		.ASCII <CRLF>/SW06= (000100) don't change parameters (loop on present values)/
41	066372	200	123		.ASCII <CRLF>/SW05= (000040) A. partial register display if error/
42	066457	200	040		.ASCII <CRLF>/ B. no ECC correction results displayed if error/
43	066557	200	123		.ASCII <CRLF>/SW04= (000020) A. do not check for maximum error count/
44	066647	200	040		.ASCII <CRLF>/ B. do not drop drive at end of test/
45	066733	200	123		.ASCII <CRLF>/SW03= (000010) A. display error sector if DCK, DTE or WCF error/
46	067034	200	040		.ASCII <CRLF>/ B. display sector if DCK uncorrectable after 28th ret
47	067144	200	040		.ASCII <CRLF>/ C. if data compare error & SW07 set, display remainde
48	067265	200	123		.ASCII <CRLF>/SW02= (000004) A. do not type drive status at program start/
49	067362	200	040		.ASCII <CRLF>/ B. do not type performance report after specified tim
51	067471	200	040		.ASCII <CRLF>/ C. prompt 'write anywhere' question during auto test
53	067603	200	123		.ASCII <CRLF>/SW01= (000002) inhibit data compare after read without DCK error@
54	067705	200	123		.ASCII <CRLF>/SW00= (000001) read only mode/
55	067744	200			.ASCII <CRLF>
56	067745	200	116		.ASCII <CRLF>/NOTE: If a DCK error occurs, the program will execute/
57	070035	200	040		.ASCII <CRLF>/ a data compare on the data in memory, regardless/
58	070125	200	040		.ASCII <CRLF>/ of the setting in SW01./
59	070163	200			.ASCII <CRLF>
60	070164	200	124		.ASCII <CRLF>/Type control- (^C) to HALT the program/
61	070234	200			.ASCII <CRLF>
62	070235	200	124		.ASCII <CRLF>/To change RHXX addresses, start program at address 204/

63	070324	200			.ASCII	<CRLF>
64	070325	200	056	105	.ASCII	<CRLF>/.END OF HELP/
65	070342	200			.ASCII	<CRLF>
66	070343	200	000		.ASCII	<CRLF>
67						
69	000200			.END	200	



ABASE = 176700	ASGN7 027544	BITS = 000040	CMPTIM 001462	DPR = 000400
ABNRML 031372	ASGN8 027606	BIT6 = 000100	CMSEC 001372	DPRQS 040504
ACDW1 = 000000	ASKPAR 060706	BIT7 = 000200	CMSTR 014234	DRIVE = 001220
ACDW2 = 000000	ASNERR 031252	BIT8 = 000400	CMSTR2 014374	DRIVE0 046234
ACPUOP= 000000	ASNLST 001542	BIT9 = 001000	CMTRK 001373	DRIVE1 046476
ACTDRV 040520	ASNMSG 031276	BLKADR 002056	COLON 060074	DRIVE2 046740
ACTSTR 040521	ASSIGN 027122	BLNKS1 056531	COMMA 056573	DRIVE3 047202
ADDW0 = 000000	ASWREG= 000000	BLNKS2 056530	COMTBL 002076	DRIVE4 047444
ADDW1 = 000000	ATA = 100000	BLNKS3 056527	CPSAVE 035326	DRIVE5 047706
ADDW10= 000000	ATABIT 040550	BLNKS4 056526	CR = 000015	DRIVE6 050150
ADDW11= 000000	AATESTN= 000000	BPE = 000020	CRLF = 000200	DRIVE7 050412
ADDW12= 000000	ATTN 001316	BPTVEC= 000014	CTRAP 034676	DROP 031302
ADDW13= 000000	ATO = 000001	BSE = 100000	CYLLMT= 000150	DROPD 027620
ADDW14= 000000	AT1 = 000002	BUFTBL 001654	CYLNRD 063324	DROPNG 057561
ADDW15= 000000	AT2 = 000004	BUSADR 062436	DASH 056575	DRQ = 004000
ADDW2 = 000000	AT3 = 000010	CFLAG 001334	DASH13 057327	DRSTAT 057033
ADDW3 = 000000	AT4 = 000020	CHGADR 001332	DASH5 057321	DRVACT 040444
ADDW4 = 000000	AT5 = 000040	CHKWC 021512	DATAPK 030052	DRVCLR= 000111
ADDW5 = 000000	AT6 = 000100	CI1 042006	DATA0 002360	DRVER 011756
ADDW6 = 000000	AT7 = 000200	CI2 042152	DATA1 002420	DRVINT 040750
ADDW7 = 000000	AUNIT = 000000	CI3 042200	DATA10 003060	DRVMSG 056601
ADDW8 = 000000	AUSWR = 000000	CI4 042312	DATA11 003120	DRVNO 001320
ADDW9 = 000000	AUTLST 032020	CI5 042632	DATA12 003160	DRVPAR 001426
ADEVCT= 000000	AVAIL 001610	CI6 042654	DATA13 003220	DRVPRM 030322
ADEVM = 000000	AVECT1= 000254	CI7 042670	DATA14 003260	DRVQUE 046070
AENV = 000000	AVECT2= 000000	CI7B 042704	DATA15 003320	DRVSN 060152
AENVM = 000000	A16 = 000400	CI8 042756	DATA2 002460	DRVSTA 040454
AFATAL= 000000	A17 = 001000	CKBUS 014046	DATA3 002520	DRVTYP 040464
AIR = 000004	BADENT 060103	CKCLK 024514	DATA4 002560	DRY = 000200
AMADR1= 000000	BADSEC 001336	CKCLK1 024576	DATA5 002620	DSWR = 177570
AMADR2= 000000	BADIMO 003440	CKCLK2 024650	DATA6 002660	DTE = 010000
AMADR3= 000000	BA1 = 000010	CKCLK3 024702	DATA7 002720	DTEER 012562
AMADR4= 000000	BEGCOD 001514	CKERR 013746	DATA8 002760	DTUW 040546
AMAMS1= 000000	BEGPAT 001512	CKMT 012010	DATA9 003020	DT00 = 000001
AMAMS2= 000000	BEGWC 001516	CKHCE 012172	DCK = 100000	DT01 = 000002
AMAMS3= 000000	BELL = 000007	CKLMTS 021332	DCKER 010546	DT02 = 000004
AMAMS4= 000000	BIT0 = 000001	CKSWR = 104407	DCKER1 010722	DT03 = 000010
AMSGAD= 000000	BIT00 = 000001	CKCHR 033122	DCU = 000040	DT04 = 000020
AMSGLG= 000000	BIT01 = 000002	CKDEC 033074	DDISP = 177570	DT05 = 000040
AMSGTY= 000000	BIT02 = 000004	CKDIG 033174	DDRVS = 001544	DT06 = 000100
AMTYP1= 000000	BIT03 = 000010	CKNUM 062614	DGE = 000001	DT07 = 000200
AMTYP2= 000000	BIT04 = 000020	CKOCT 033046	DH1 053751	DT08 = 000400
AMTYP3= 000000	BIT05 = 000040	CKFLG 001310	DH14 054126	DT1 054424
AMTYP4= 000000	BIT06 = 000100	CLKOFF 024710	DH15 054225	DT14 054450
AOE = 001000	BIT07 = 000200	CLR = 000040	DH16 054324	DT15 054470
APASS = 000000	BIT08 = 000400	CLRDPB 030104	DH17 054404	DT16 054512
APRIOR= 000000	BIT09 = 001000	CLRQUE 045772	DH2 053756	DT17 054530
APTCSU= 000040	BIT1 = 000002	CMCNT 001366	DH3 054033	DT2 054430
APTENV= 000001	BIT10 = 002000	CMCYL 001370	DH4 054061	DT3 054434
APTSIZ= 000200	BIT11 = 004000	CMDAT 014406	DH6 054120	DT4 054440
APTSPO= 000100	BIT12 = 010000	CMHED 014316	DISPLA 001156	DT6 054444
ASGND 057623	BIT13 = 020000	CMPAR 014132	DISPLY= 104414	DVA = 004000
ASGN1 027220	BIT14 = 040000	CMPARD 014136	DISPRE 000174	DVC = 000200
ASGN2 027274	BIT15 = 100000	CMPLMT 001460	DLT = 100000	ECBADO 001412
ASGN3 027364	BIT2 = 000004	CMRES 022200	DONE 010206	ECBAD1 001420
ASGN4 027530	BIT3 = 000010	CMPT 015016	DPE = 000010	ECBIT 001376
ASGN6 027532	BIT4 = 000020	CMPRX 015010	DPINT 040474	FCC 015534

ECCX	016350	ENDCMP	015376	HOUR	001340	LINE6	023776	MAIN1	006506
ECC1	016146	ENDCON	001446	HT	= 000011	LINE6A	024010	MAIN2	006644
ECC2	016344	ENDING	001504	IAE	= 002000	LINE6C	024016	MASK	001322
ECGD	001410	ENDPGM=	064330	IAEER	012670	LINE6D	024024	MATCH	015444
ECGD1	001416	ENDSEK	001452	IBSAVE	035330	LINE7	024056	MAXCYL=	000134
ECH	= 000100	ENTADR	060044	IDLE	007234	LINE7A	024176	MAXER	001456
ECI	= 004000	ENTCOM	060001	IE	= 000100	LINE8	024274	MAXSEC=	000144
ECMSK0	001402	ENTLMT	060022	ILF	= 000001	LING6	055355	MAXTRK=	000140
ECMSK1	001404	ENTPR	005556	ILR	= 000002	LINM3	054633	MCPE	= 020000
ECSEC	001400	EQUAL	056571	ILV	= 000004	LINM4	055060	MDPE	= 000400
ECWRD	001406	ERCTR	001362	INCHRD	026026	LINN3	054651	MESFE	060226
ECWRD1	001414	ERPRC1	007636	INCMIS	026076	LINOCT	024306	MINCYL=	000136
EMPTYQ	046050	ERPROC	007622	INCSK	026052	LINP3	054672	MINSEC=	000146
EMTVEC=	000030	ERR	= 040000	INCSOF	026002	LINP5	055173	MINTRK=	000142
EM1	050744	ERROR	= 104000	INCTOT	026122	LINR6	055373	MINUTE	001342
EM10	051257	ERRVEC=	000004	INTRVL	001470	LINSS3	055044	MNTBL	002124
EM11	051322	EWN	= 000002	INVLD	057756	LINST3	055030	MOH	= 020000
EM12	051355	EWNERR	013432	IOTVEC=	000020	LINS3	054712	MOL	= 010000
EM13	051406	FACTOR	017510	IR	= 000100	LINS4	055076	MREAD	060526
EM14	051460	FAIRNS	001326	ISR	043174	LINS5	055210	MRPCS1	062576
EM15	051503	FALPAR	007776	ITCNT	022174	LINT3	054761	MRPVEC	062605
EM2	051017	FALPR1	010012	IXU	= 000100	LINU06	055405	MSGCON	057513
EM20	051537	FEONLY	060451	KSR	026454	LINW3	055017	MSGDRP	057357
EM21	051560	FER	= 000020	KSR1	026462	LINX4	055112	MSGGEP	057371
EM22	051611	FE1	= 000156	KWSVR	026272	LINX5	056533	MSGEOT	057417
EM23	051664	FILBUF	020102	LBC	= 002000	LIN10A	056065	MSGON	057616
EM24	051743	FILLZ	032546	LBT	= 002000	LIN10B	056121	MSGPG	057052
EM25	052011	FILLO	032654	LCE	= 001000	LIN10C	056161	MSGPWR	040022
EM26	052072	FMTER	012750	LF	= 000012	LIN10H	056265	MSGREP	057107
EM27	052117	FMTRK	= 000163	LIMIT	001364	LIN11	056466	MSGSUM	057345
EM3	051055	FMT16	= 010000	LINA3	054742	LIN11A	056507	MSGX10	057552
EM30	052154	FRSTER	001352	LINB3	055010	LIN11H	056413	MSHELP	064330
EM31	052206	F0	= 000002	LINB5	055153	LIN2C	054536	MSHW1	057055
EM32	052251	F1	= 000004	LINB6	055265	LIN2P	054556	MSHW2	057065
EM33	052304	F2	= 000010	LINCA3	054771	LIN2S	054577	MSPASS	057532
EM34	052361	F3	= 000020	LINC6	055320	LIN3.1	023162	MSPRM	060164
EM35	052423	F4	= 000040	LINDA3	055001	LIN3.3	023270	MSTOTL	057542
EM36	052461	GENDPB	050654	LINDEC	024340	LIN3.4	023322	MSWAIT	040176
EM37	052512	GENPAR	020612	LIND5	055135	LIN6.2	024032	MSWRO	057713
EM4	051113	GENREG	050674	LINEN3	054726	LIN7M	055432	MXF	= 001000
EM40	052564	GETBUF	017512	LINE05	055253	LIN7P	055455	MXWNDW	040570
EM41	052622	GETID	031010	LINEP5	055243	LIN7R	055553	M.DPID	032170
EM42	052666	GETLMT	030706	LINE1	022214	LIN7S	055475	M.DP40	032226
EM43	052747	GETPAT	021272	LINE2	022262	LIN7T	055514	M.DP41	032262
EM44	053043	GFTREG=	000141	LINE2A	022432	LIN7X	055535	M.DP42	032272
EM45	053140	GETREM	032022	LINE2B	022450	LIN8M	055570	M.DP44	032324
EM46	053226	GETREQ	046144	LINE3	022716	LIN9B	055613	M.DP50	032336
EM47	053300	GO	= 000001	LINE3A	022724	LIN9E	056010	N	057705
EM5	051150	GODRIV	020160	LINE3B	022732	LIN9G	056037	NED	= 010000
EM50	053355	GTSWR	= 104406	LINE3C	022744	LIN9H	055642	NEDCLK	057634
EM51	053413	HCE	= 000200	LINE3D	022754	LIN9I	055735	NEM	= 004000
EM52	053456	HCEER	013026	LINE3E	023022	LKPAR	005246	NEWASN	030030
EM6	051204	HCI	= 002000	LINE3F	023110	LODEV	056761	NEWUNT	001566
EM60	053522	HCRC	= 000400	LINE4	023366	LODPAR	021662	NINLEV	056776
EM70	053545	HCR CER	011624	LINES	023456	LSTAD	001330	NODFLT	057017
EM71	053624	HELPTX	064440	LINE5A	023666	MAIN	006340	NODRVS	060124
EM72	053702	HERTZ	001312	LINE5B	023734	MAINDA	006364	NOMTCH	013554

NONE 060076
 NOOP = 000101
 NUTAVL 056732
 NOTPRS 056715
 NOTRP 056700
 NOTSAF 056751
 NSA = 100000
 OFFDIR= 000200
 OFFON = 000001
 OFLIN 010110
 ONES 003262
 ONESEC 001346
 ONESUM 025042
 OPI = 020000
 OPIER 012460
 OPIER1 012522
 OPT 041542
 OPTBL 002104
 OR = 000200
 ORDERQ 001520
 U,RWRT 060307
 PACK 030102
 PAR = 000010
 PARENT 031110
 PARER 012576
 PARLST 060604
 PAR1 060742
 PAR11 061224
 PAR12 061271
 PAR14 061354
 PAR15 061407
 PAR16 061451
 PAR19 061515
 PAR2 061001
 PAR21 061537
 PAR3 061061
 PAR4 061136
 PAR5 061147
 PAR6 061160
 PAR7 061171
 PAR8 061202
 PAR9 061213
 PASSES 001474
 PAT = 000020
 PATCH 062724
 PATER 001476
 PERIOD 057711
 PERM 043460
 PFECM 035510
 PFECH1 035520
 PFECH2 035602
 PFECH3 035634
 PFECH4 035644
 PFTSTN 035650
 PGE = 100000
 PGM = 001000
 PIP = 020000

PIRQ = 177772
 PIRQVE = 000240
 POPQUE 046166
 POSER 012406
 PROCES 007472
 PRTBAD 016356
 PRTIM 010150
 PRO = 000000
 PR1 = 000040
 PR2 = 000100
 PR3 = 000140
 PR4 = 000200
 PR5 = 000240
 PR6 = 000300
 PR7 = 000340
 PS = 177776
 PSEL = 002000
 PSW = 177776
 PUNSAF 007720
 PWRFLG 040020
 PWRUP 040044
 PWRVEC = 000024
 QCNT 045500
 QDRV0 045572
 QDRV1 045612
 QDRV2 045632
 QDRV3 045652
 QDRV4 045672
 QDRV5 045712
 QDRV6 045732
 QDRV7 045752
 QINPT 045510
 QOUTPT 045530
 QSTART 045550
 QSTOP 045552
 QTERP = 045772
 QUES 056567
 RANCYL 021002
 RANDOM 001510
 RANDWC 001500
 RANPAT 021242
 RANSEC 021072
 RANSIZ 021150
 RANTRK 021036
 RANXIT 021262
 RATL 001502
 RDCHR = 104410
 RDDAT = 000171
 RDHD = 000173
 RDLIN = 104411
 RDONLY 001424
 RDY = 000200
 RD.RP 045036
 READDR 024364
 READHD 016746
 READIN = 000121
 RECAL = 000107

RECALT 016630
 RECALO 016720
 REDAPK 030040
 RELBUF 017646
 RELSE = 000113
 REPLZ 032552
 RESREG = 104413
 RESVEC = 000010
 RETRY 001324
 REV 001432
 RHEXT 040566
 RMR = 000004
 RPADR 040560
 RPAS = 000016
 RPBA = 000004
 RPBAE = 000050
 RPCC = 000036
 RPCS1 = 000000
 RPCS2 = 000010
 RPCS3 = 000052
 RPDA = 000006
 RPDB = 000022
 RPDC = 000034
 RPDS = 000012
 RPDT = 000026
 RPEC1 = 000044
 RPEC2 = 000046
 RPER1 = 000014
 RPER2 = 000040
 RPER3 = 000042
 RPINIT 040572
 RPLA = 000020
 RPMR1 = 000024
 RPOF = 000032
 RPSN = 000030
 RPSTU0 040344
 RPSTU1 040354
 RPSTU2 040364
 RPSTU3 040374
 RPSTU4 040404
 RPSTU5 040414
 RPSTU6 040424
 RPSTU7 040434
 RPTMR 044322
 RPVEC 040562
 RPWC = 000002
 RP07 041304
 RTC = 000117
 RTURN 031770
 RWU1 = 002000
 RWU2 = 004000
 RWU3 = 010000
 R6 = %000006
 R7 = %000007
 S 054667
 SAVEFG 040522
 SAVER1 001356

SAVERS 001360
 SAVPOS 001354
 SAVREG = 104412
 SC 043550
 SCMND 027726
 SCOPE = 000004
 SC04 = 000400
 SC1 = 000100
 SC10 = 001000
 SC100 = 010000
 SC11 044022
 SC12 044120
 SC13 044204
 SC2 = 000200
 SC20 = 002000
 SC3 043616
 SC4 043622
 SC40 = 004000
 SC5 043634
 SC6 043744
 SC8 043772
 SEARCH = 000131
 SECLMT = 000152
 SECOND 001344
 SEEK = 000105
 SEEKFG 040524
 SELDRV = 000145
 SEQPAR 020250
 SETFMT = 000143
 SETVEC 005604
 SET.IE 045426
 SIZE70 004742
 SIZMEM 005112
 SKI = 040000
 SKIER 013204
 SLASH 060702
 SRCHWT 040516
 STA 006074
 STACK = 001100
 START 003522
 START1 003532
 START2 003550
 STATIN 001314
 STATIS 017210
 STATPR 024740
 STKLMT = 177774
 STNDAT 002314
 STO 044412
 SUPRS 032362
 SUPRSL 032346
 SUPR1 032374
 SUPR2 032436
 SVRHXX 045262
 SWR 001154
 SWREG 000176
 SWTIM 010052
 SWO = 000001

SW00 = 000001
 SW01 = 000002
 SW02 = 000004
 SW03 = 000010
 SW04 = 000020
 SW05 = 000040
 SW06 = 000100
 SW07 = 000200
 SW08 = 000400
 SW09 = 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
 T 054646
 TAB 056577
 TABLE 061604
 TABLE0 061624
 TABLE1 061672
 TABLE2 061740
 TABLE3 062006
 TABLE4 062054
 TABLE5 062122
 TABLE6 062170
 TABLE7 062236
 TAB.XY = 001114
 TAP = 040000
 TBITVE = 000014
 TCF = 000400
 TD 043236
 THEAD 020716
 TIMER 040526
 TKVEC = 000060
 TPE = 000002
 TPVEC = 000064
 TRAPVE = 000034
 TRE = 040000
 TRFER 013104
 TRKLMT = 000154
 TRMREP 057214
 TRNSWT 040514
 TRTVEC = 000014
 TSTANY 001422
 TST1 003540
 TYDRV 032764
 TYPDRV 032770

TYPDS = 104405	\$CKSWR 033730	\$ICNT 001120	\$PWAD 040006	\$SUPRS 032462
TYPE = 104401	\$CMTAG 001114	\$ILLUP 040012	\$PWADN 037640	\$SUPR1 032474
TYPOC = 104402	\$CM3 = 000000	\$INTAG 001151	\$PWARMG 040002	\$SUPR2 032536
TYPON = 104404	\$CM4 = 000001	\$ITEMB 001130	\$PWURUP 037712	\$SVPC = 000210
TYPOS = 104403	\$CNTLC 034635	\$LF 001204	\$QUES 001202	\$SWR = 122000
TYP SUM 025070	\$CNTLG 034647	\$LFLG 037123	\$RAND 037126	\$SWREG 001230
UCPAR 007760	\$CNTLU 034642	\$LKCSB 001300	\$RDCHR 034272	\$STATUS= 000016
UNS = 040000	\$CODE = 000024	\$LKCSR 001276	\$RDLIN 034362	\$TERM = 000032
UNSAF 013340	\$COMND= 000002	\$LKS 001304	\$RD PAS= 000036	\$TESTM 001212
UNTASN 056652	\$CPUOP 001234	\$LLVEC 001306	\$RDSZ = 000017	\$TIME 026146
UNTNOT 056630	\$CRLF 001203	\$LONUM 037226	\$REG = 000014	\$TKB 001162
UNTOFF 056607	\$CYL = 000012	\$LPADR 001122	\$REPLZ 032662	\$TKCNT 033412
UNTON 056620	\$DBDIV 032114	\$LPERR 001124	\$RESRE 037266	\$TKINT 033430
UPE = 020000	\$DBLK 036650	\$LPVEC 001302	\$RETRY 017040	\$TKQEN= 033427
US1 = 000001	\$DB2D 037324	\$LSTAD 062434	\$RPADR 001272	\$TKQIN 033414
US2 = 000002	\$DB2O 037520	\$MADR1 001240	\$RPAS = 000204	\$TKQOU 033416
US4 = 000004	\$DECVL 037504	\$MADR2 001244	\$RPBA = 000172	\$TKQSR 033420
VV = 000100	\$DEVCT 001216	\$MADR3 001250	\$RPBAE= 000236	\$TKS 001160
WAIT 001632	\$DEV M 001264	\$MADR4 001254	\$RPCC = 000224	\$TKSRV 033500
WATPAK 030064	\$DIV 032046	\$MAIL 001206	\$RPCS1= 000166	\$TMP0 001174
WCE = 040000	\$DOAGN 031764	\$MAMS1 001236	\$RPCS2= 000176	\$TN = 000002
WCF = 000040	\$DRVSN= 000160	\$MAMS2 001242	\$RPCS3= 000240	\$TNPWR 037434
WCFER 013242	\$DSPLY 033020	\$MAMS3 001246	\$RPDA = 000174	\$TOTAL= 000102
WCHKX = 040344	\$DTBL 036640	\$MAMS4 001252	\$RPDB = 000210	\$TPB 001166
WCKD = 000151	\$ENDAD 031754	\$MBADH 001102	\$RPDC = 000222	\$TPFLG 001173
WCKER 011234	\$ENDCT 031740	\$MFLG 037122	\$RPDS = 000200	\$TPS 001164
WCKHD = 000153	\$ENV 001226	\$MISPO= 000112	\$RPDT = 000214	\$TRAP 040256
WC.HK 043462	\$ENV M 001227	\$MNEW 034645	\$RPEC1= 000232	\$TRAP2 040300
WLE = 004000	\$EOP 031420	\$MSGAD 001222	\$RPEC2= 000234	\$TRK = 000011
WLEER 012722	\$EOPCT 031732	\$MSGLG 001224	\$RPER1= 000202	\$TRP = 000015
WOR = 001000	\$ERFLG 001117	\$MSGTY 001206	\$RPER2= 000226	\$TRPAD 040312
WRDCNT 001466	\$ERMAX 001131	\$MSWR 034654	\$RPER3= 000230	\$TSTM 001104
WRDPOS 001374	\$ERROR 034742	\$MTYP1 001237	\$RPLA = 000206	\$TSTM M 001116
WRL = 004000	\$ERRPC 001132	\$MTYP2 001243	\$RPMR1= 000212	\$TTYIN 034616
WRTCHK 001506	\$ERRTB 003360	\$MTYP3 001247	\$RPOF = 000220	\$TYPDS 036434
WRTDAT= 000161	\$ERRTY 035332	\$MTYP4 001253	\$RPSN = 000216	\$TYPE 035652
WRT.RP 045130	\$ERTTL 001126	\$NCODE= 000122	\$RPVEC 001274	\$TYPEC 036064
WRYUNS= 000400	\$ETABL 001226	\$NCYL = 000126	\$RPWC = 000170	\$TYPEX 036204
XXDP 001430	\$ETEND 001272	\$NEXT = 000130	\$RP07 057075	\$TYPOC 036232
Y 057707	\$FAIR = 000116	\$NPATC= 000123	\$RP07P 057102	\$TYPON 036246
ZEROS 002360	\$FATAL 001210	\$NSEC = 000124	\$RTNAD 031766	\$TYPOS 036206
ZROIND 001350	\$FFLG 037124	\$NTRK = 000125	\$RTOTL= 000066	\$UNIT 001220
\$APTHD 001100	\$FILLC 001172	\$NULL 001170	\$SAVRE 037230	\$UNITM 001110
\$ATYC 036704	\$FILLS 001171	\$NWTST= 000000	\$SAVR6 040016	\$USWR 001232
\$ATY1 036660	\$FILLZ 032656	\$OCNT 036430	\$SB2D 033332	\$VECT1 001256
\$ATY3 036666	\$FIRJT= 000132	\$OCTVL 037622	\$SB2O 033362	\$VECT2 001260
\$ATY4 036676	\$FMT = 000001	\$OMODE 036432	\$SEC = 000010	\$WCNT = 000004
\$AUTOB 001150	\$GDADR 001134	\$OPERC= 000052	\$SEESK= 000046	\$WRDL = 000020
\$BASE 001262	\$GDDAT 001140	\$PACK = 000026	\$SETUP= 000156	\$WTOTL= 000056
\$BDADR 001136	\$GET42 031744	\$PASS = 000114	\$SIZE 062304	\$WTPAS= 000042
\$BD DAT 001142	\$GTSWR 034020	\$PASSC= 000114	\$SKI = 000110	\$XOFF = 000023
\$BELL 001176	\$HARD = 000106	\$PASTM 001106	\$SOFT = 000104	\$XON = 000021
\$BUF = 000006	\$HD = 000000	\$PATTC= 000030	\$SSEC = 000022	\$SGET4= 000000
\$CDW1 001266	\$HIBTS 001100	\$PREVA= 000032	\$STCTL= 000076	\$OFILL 036431
\$CDW2 001270	\$HINUM 037224	\$PREVO= 000027	\$STUP = 177777	\$.SX = 001100
\$CHARC 036202	\$HLDWC= 000120	\$PSEL = 000003	\$SUPRL 032446	

BYON	42-1	45-1	45-1			
.SASTA	48-1	48-1				
.SX	6-11	6-11#				
A16	5-112#	11-162	11-167			
A17	5-113#	11-162	11-167			
ABASE	5-307#	7-0	7-0			
ABNRML	13-183	37-41#				
ACDW1	7-0	7-0				
ACDW2	7-0	7-0				
ACPUOP	7-0	7-0				
ACTDRV	55-105#	57-101#	57-155#	57-456#	57-464#	57-684
ACTSTR	55-111#	57-686#	57-700#			
ADDW0	7-0					
ADDW1	7-0					
ADDW10	7-0					
ADDW11	7-0					
ADDW12	7-0					
ADDW13	7-0					
ADDW14	7-0					
ADDW15	7-0					
ADDW2	7-0					
ADDW3	7-0					
ADDW4	7-0					
ADDW5	7-0					
ADDW6	7-0					
ADDW7	7-0					
ADDW8	7-0					
ADDW9	7-0					
ADEVCT	7-0	7-0				
ADEVN	7-0	7-0				
AENV	7-0	7-0				
AENVN	7-0	7-0				
AFATAL	7-0	7-0				
AIR	5-264#					
AMADR1	7-0	7-0				
AMADR2	7-0	7-0				
AMADR3	7-0	7-0				
AMADR4	7-0	7-0				
AMAMS1	7-0	7-0				
AMAMS2	7-0	7-0				
AMAMS3	7-0	7-0				
AMAMS4	7-0	7-0				
AMSGAD	7-0	7-0				
AMSLG	7-0	7-0				
AMSGTY	7-0	7-0				
AMTYP1	7-0	7-0				
AMTYP2	7-0	7-0				
AMTYP3	7-0	7-0				
AMTYP4	7-0	7-0				
AQE	5-186#					
APASS	7-0	7-0				
APRIOR	7-0					
APTCSU	45-1	48-1#				
APTENV	43-1	45-1	48-1	48-1#		
APTSIZ	11-25	48-1#				
APTSPO	45-1	48-1	48-1#			

MTOTL	29-100	29-116	29-132	63-390					
MUAI1	53-23	53-37							
MUHO	53-93	63-52							
MIF	5-136								
MUNDM	55-169	57-287							
M	63-49								
MFD	5-139								
MFDCLK	28-30	63-48							
MER	5-138								
MFWASH	32-51	33-52							
MFWUNT	8-0	13-53	13-54	13-67	13-74	13-75	32-181		
MINLEV	11-372	11-199	63-180						
MODFLT	11-132	11-250	63-180						
MODRVS	13-96	63-60							
NOTMCH	15-555	16-59							
NINE	63-58								
NOOP	5-284								
NOTAVL	63-15								
NOTPRS	11-339	32-196	63-160						
NOTRP	11-336	32-192	63-150						
NOTSAF	11-345	32-186	63-160						
NSA	5-223								
OFFDIR	5-240								
OFFON	5-161								
OFFSET	8-0	8-0	8-0	15-60	15-81	18-92	27-344		
OFLIN	14-60	14-111							
ONES	9-0	9-0							
ONESEC	8-0	11-89	13-90	31-31	31-33				
ONESUM	13-43	29-45	38-1						
OPI	5-190								
OPIER	14-175	15-357							
OPIER1	15-367								
OPT	57-122	57-166	57-500	57-624	57-676				
OPTBL	8-0	27-44	27-46						
OR	5-134								
ORDERO	8-0	11-83	13-5	13-111	13-122	13-149	13-173	53-31	
OVRWRT	11-233	63-84							
PACK	32-182	33-52	33-57	33-62	33-68	33-71			
PAR	5-180								
PAR1	64-3	64-26							
PAR11	64-8	64-38							
PAR12	64-5	64-39							
PAR14	64-9	64-40							
PAR15	64-11	64-41							
PAR16	64-10	64-42							
PAR19	64-6	64-43							
PAR2	64-4	64-27							
PAR21	64-15	64-47							
PAR3	64-7	64-28							
PAR4	64-29	65-20	65-20	65-20	65-20	65-20	65-20	65-20	65-20
PAR5	64-30	65-20	65-20	65-20	65-20	65-20	65-20	65-20	65-20
PAR6	64-31	65-20	65-20	65-20	65-20	65-20	65-20	65-20	65-20
PAR7	64-32	65-20	65-20	65-20	65-20	65-20	65-20	65-20	65-20
PAR8	64-33	65-20	65-20	65-20	65-20	65-20	65-20	65-20	65-20
PAR9	64-34	65-20	65-20	65-20	65-20	65-20	65-20	65-20	65-20
PARENT	11-304	34-92	36-8						

