RM02/03/05 RM05/3/2 PERF EXER AH-S071B-MC AUG 1981
CZRMUBO FICHE 1 OF 2
MADE IN USA

RM02/03/05 RM05/3/2 PERF EXER AH-S071B-MC
CZRMUBO FICHE 2 OF 2

```
                        .REM    @
```

IDENTIFICATION

PRODUCT CODE: AC-S069B-MC

PRODUCT NAME: CZRMUBO RM05/3/2 PERFORMANCE EXERCISER

PRODUCT DATE: APRIL 1981

MAINTAINER: CX DIAGNOSTIC GROUP

AUTHOR: MIKE LEAVITT

CONTENTS

1. ABSTRACT

THE RM05/3/2 PERFORMANCE EXERCISER PROGRAM IS DESIGNED TO PERFORM AN INTERACTIVE TEST ON RM 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 GENERAL DOCUMENT NOTES

A. IN REFERENCE TO ALL NUMBERS IN THIS DOCUMENTION. 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 (ANY COMBINATION OF RM05'S, RM03'S OR RM02'S)

2.2 MEDIA

THE PERFORMANCE EXERCISER PROGRAM REQUIRES FORMATTED DISK PACKS GENERATED BY THE RM05/3/2 FORMATTER PROGRAM (CZRML). THE PACKS MUST BE FORMATTED IN 32. SECTOR (16 BIT) MODE; THE ALTERNATE (30. SECTOR - 18 BIT) MODE IS NOT SUPPORTTED.

2.3 PRELIMINARY PROGRAMS

RM05/3/2 DISKLESS TESTS, PART 1 & 2

RM05/3/2 FUNCTIONAL TESTS, PART 1, 2 & 3

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 RM/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	INHIBIT ERROR TYPEOUT
SW<12>	NOT USED
SW<11>	NOT USED
SW<10>=1	BELL ON ERROR
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 INHIBIT PERFORMANCE REPORT AFTER SPECIFIED TIME

```
                SW<01>=1        INHIBIT DATA COMPARE AFTER READ COMMAND, W/O ERROR
                SW<00>=1        'READ ONLY'' MODE


                SW<00>

                WHEN THE SWITCH IS SET(1), THE PROGRAM WILL OPERATE IN 'READ
                ONLY'' MODE. IF THE SWITCH IS CLEARED(0), THE PROGRAM WILL RETURN
                TO READ/WRITE MODE DURING TESTING. THIS SWITCH ONLY EFFECTS THE
                TESTING PORTION OF THE PROGRAM.

                FOR EXAMPLE, IF THE PROGRAM IS STARTED AT ADDRESS 200 . 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. HOWEVER, IF THE OPERATOR SWITCHES TO 'READ ONLY''
                MODE (SW0=1) JUST PRIOR TO OR DURING THE SEQUENTIAL WRITTING
                OF THE DISK, THE PROGRAM WILL CONTINUE WRITTING UNTIL THE
                SEQUENTIAL WRITE IS COMPLETED. UPON COMPLETION OF THE SEQUENTIAL
                WRITE, THE PROGRAM WILL SWITCH TO A 'READ ONLY'' TESTING MODE
                UNTIL SW0 IS RESET TO ZERO BY THE OPERATOR.


        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 RM DRIVE IS NO MORE THAN 1 SOFT
                ERROR (NON-DISK RELATED) IN 1 X 10^10 BITS READ. (SEE SECTION 3.5.1
                FOR THE 90% CONFIDENCE LEVEL)

                THE SEEK ERROR SPECIFICATION FOR THE RM DRIVE IS NO MORE THAN 1 SEEK
                ERROR IN 1 X 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 3 X 10^9 BITS (1.875 X 10^8 WORDS). IT WILL TAKE
                    APPROXIMATELY 3.33 PASSES TO REACH THE SOFT ERROR RATE OF
                    1 X 10^10 BITS (6.25 X 10^8 WORDS) READ. HOWEVER, IT WILL TAKE
                    10. PASSES TO REACH THE 90% CONFIDENCE LEVEL OF 3 X 10^10 BITS
                    (1.875 X 10^9 WORDS) READ.

                B.  IF PARAMETER 'ENDING' IS 0, END OF PASS OCCURS WHEN THE DRIVE
                    HAS PERFORMED 1 X 10^6 SEEKS. IT WILL TAKE 1 PASS TO REACH
                    THE SEEK ERROR RATE OF 1 X 10^6 SEEKS. HOWEVER, IT WILL TAKE
                    3 PASSES TO REACH THE 90% CONFIDENCE LEVEL OF 3 X 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, 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)

1 DRIVE - APPROX. 1 HR. 45 MIN. (TO REACH 3 X 10^9 BITS OR 1.875 X 10^8 WORDS)

WITH SW<00> =1 AND SW<01> =1, THE PROGRAM WILL RUN APPROX. 20% FASTER

3.6.2 SEEK VERIFICATION MODE

PARAMETER 'WRDCNT' = 256. (1 SECTOR)
PARAMETER 'MAXTRK' = 'MINTRK' (SAME VALUES)
PARAMETER 'MAXSEC' = 'MINSEC' (SAME VALUES)
SW<01> =1 (NO DATA COMPARE)
SW<00> =1 (READ ONLY MODE)

1 DRIVE - APPROX. 4.0 HRS (TO REACH 1 X 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 RM05/3/2 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 SUCESSIVE CHARACTERS FROM THE INPUT.

B. TO DELETE AN ENTIRE LINE, TYPE A 'CONTROL U' (^U).

C. AN ENTRY MUST BE TERMINATED BY EITHER A 'CARRIAGE RETURN' OR A 'PERIOD'. THE 'PERIOD' TERMINATION IS RECOGNIZED BY THE PROGRAM AS A DEFAULT ENTRY REQUEST. WHEN A LINE IS TERMINATED BY A 'PERIOD' INSTEAD OF A 'CARRIAGE RETURN', THE PROGRAM WILL ACCEPT THE ENTERED VALUE AND WILL DEFAULT TO THE PRELOADED VALUES FOR ANY REMAINING ENTRIES.

D. IF A 'CONTROL C' IS TYPED DURING 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

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 UNIT STATUS PRINTOUT:

```
'UNIT STATUS:
  0    ONLINE   RM05
  1    LOAD DEVICE
  2    OFFLINE  RM03
  3    NOT PRESENT
  4    NOT PRESENT
  5    NOT AN RM05/3/2
  6    NOT PRESENT
  7    NOT PRESENT'
```

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

4.1.1 KEYBOARD ENTRY PARAMETERS

NAME	BASE	DEFAULT VALUE	VALUE RANGE	FUNCTION
WRDCNT	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 SPECIFIY 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)
INTRVL	10.	0	0 - 32767.	DETERMINES THE INTERVAL (IN

					MINUTES) BETWEEN AUTOMATIC PERFORMANCE SUMMARY TYPEOUTS; NO TYPEOUT IF THIS PARAMETER IS 0 OR IF SW<02> =1
	PASSES	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)
	PATTERN	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)
	RANDWC	8	000000	0 OR 1	IF PARAMETER = 0, THE WORD COUNT IS RANDOMLY SELECTED BETWEEN 6 AND THE VALUE 'WRDCNT'. IF PARAMETER 1, THE WORD COUNT WILL BE THE VALUE 'WRDCNT'.
	RATIO	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
	ENDING	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.
	WRTCHK	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.
	MESSAGE	8	000001	0 OR 1	IF PARAMETER =1, DO NOT PRINT ERROR MESSAGES FOR DATA ERRORS OCCURING AT LOCATIONS DEFINED BY THE OPERATOR AS BAD PACK LOCATION.

				IF PARAMETER = 0, PRINT ERROR MESSAGES ASSOCIATED WITH BAD PACK LOCATIONS.
RANDOM	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.
BADBLK	8	000000	0 OR 1	IF EQ TO 1, THE BAD SECTOR ENTRY TABLE WILL ALWAYS BE INITIALIZED WHEN ASSIGNING A DRIVE; IF EQ TO 0, THE BAD SECTOR ENTRY TABLE WILL ONLY BE INITIALIZED IF THE PACK SERIAL NUMBER HAS CHANGED SINCE THE LAST TIME IT WAS READ. NOTE: IF THE SERIAL NUMBER HAS CHANGED, THIS MOST LIKELY MEANS THAT THE PACK OR DRIVE HAD BEEN REPLACED WHILE THE DRIVE WAS DEASSIGNED.

4.1.2 CHANGE DEVICE ADDRESS

THE RM/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 RM/RH ADDRESS.

(DEFAULT ADDRESS = 176700, VECTOR = 254)

ADDRESS SELECTION EXAMPLES

EXAMPLE 1

```
RMCS1=176700 <CR>          ;NO CHANGE IN ADDRESS
RMVEC=000254 <CR>          ;NO CHANGE IN ADDRESS
```

EXAMPLE 2

```
RMCS1=176700 177200<CR> ;CHANGE BASE ADDRESS TO 177200
RMVEC=000254 260<CR>    ;CHANGE VECTOR ADDRESS TO 260
```

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 RM05/3/2	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 OR 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.

'********** DRIVE # N'

THE PROGRAM WILL THEN INFORM THE OPERATOR WHAT THE HARD WIRED MBA SERIAL NUMBER IS AND THE PACK SERIAL NUMBER IS, IN THE FOLLOWING FORMAT:

'MBA S/N: X PACK S/N: Y'

WHERE 'X' IS THE HARD WIRED DECIMAL SERIAL NUMBER CONTAINED IN THE RMSN REGISTER OF THE MBA. IF THE MBA SERIAL NUMBER IS NOT JUMPERED IN THE RMSN REGISTER, 'X' WILL APPEAR AS '????'.

WHERE 'Y' IS THE OCTAL SERIAL NUMBER READ FROM THE DEC144 BAD SECTOR FILE ON THE PACK. IF THE DEC144 FILE WAS UNABLE TO BE READ SUCCESSFULLY, 'Y' WILL APPEAR AS 'NONE'.

THE PROGRAM WILL THEN ASK FOR ADDRESS LIMIT CHANGES WITH THE FOLLOWING TYPEOUT:

'ENTER ADDRESS LIMITS:'

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

NAME	DEFAULT VALUE	VALUE RANGE	FUNCTION
MINCYL	0	0 - 821.	THE MINIMUM CYLINDER ADDRESS
MAXCYL	821.	0 - 821.	THE MAXIMUM CYLINDER ADDRESS
MINTRK	0	0 - NN	THE MINIMUM TRACK ADDRESS
MAXTRK	NN	0 - NN	THE MAXIMUM TRACK ADDRESS
MINSEC	0	0 - 31.	THE MINIMUM SECTOR ADDRESS
MAXSEC	31.	0 - 31.	THE MAXIMUM SECTOR ADDRESS

WHERE 'NN' IS 4. FOR AN RM03/02 AND 18. FOR AN RM05.

THE PROGRAM WILL THEN ASK FOR BAD SECTOR ADDRESSES WITH THE FOLLOWING TYPEOUT:

'ENTER BAD SECTR ADRS:'

THE FORMATS USED TO ENTER BAD SECTOR ADDRESS LOCATIONS ARE AS FOLLOWS:

EXAMPLE 1: CYL,TRK,SEC= C,T,S<CR>

A. THE PROGRAM WILL INHIBIT DATA ERROR MESSAGES FOR ERRORS OCCURRING AT THE SPECIFIED ADDRESS.

B. LEADING ZEROS ARE NOT REQUIRED.

EXAMPLE 2: CYL,TRK,SEC= C,T<CR>

A. WHEN THIS FORMAT IS USED, THE ENTIRE TRACK WILL BE CONSIDERED BAD.

B. DATA ERRORS WILL BE HANDLED AS IN 'EXAMPLE 1'.

```
685                                   EXAMPLE 3:    CYL,TRK,SEC- C<CR>
686
687                                         A. WHEN THIS FORMAT IS USED, THE ENTIRE CYLINDER WILL BE
688                                            CONSIDERED BAD
689
690                                         B. DATA ERRORS WILL BE HANDLED AS IN 'EXAMPLE 1'.
691
692                                   IF CONTROL C (^C) IS TYPED AS AN ENTRY, ALL CURRENT BAD SECTOR ENTRIES
693                                   WILL BE LOST AND THE FOLLOWING MESSAGE WILL BE TYPED.
694
695                                     '* ALL CURRENT ENTRIES LOST *'
696
697                                   AFTER TYPING THE PREVIOUS MESSAGE, THE PROGRAM WILL WAIT FOR THE
698                                   OPERATOR TO ENTER ANOTHER BAD SECTOR AS IN THE PREVIOUS EXAMPLES.
699
700                                   IF 'L' IS TYPED FOR AN INPUT CHARACTER, THE PROGRAM WILL TYPE A LIST
701                                   OF DEC144 BAD SECTORS, AND THE MANUALLY ENTERED BAD SECTORS, WHICH ARE
702                                   STORED IN THE DRIVE PARAMETER TABLE(DPB) FOR THAT PARTICULAR DRIVE.
703
704                                   IF THERE ARE NO BAD SECTORS IN THE DPB TABLE, THE FOLLOWING MESSAGE
705                                   WILL BE TYPED:
706
707                                     'DEC144 AND MANUAL BAD SECTOR LIST
708                                      * NO ENTRIES *'
709
710                                   HOWEVER, IF THERE ARE ENTRIES IN DPB TABLE, THE LIST WILL BE TYPED
711                                   IN THE FOLLOWING FORMAT:
712
713                                     'DEC144 AND MANUAL BAD SECTOR LIST
714                                      8,8,3
715                                      16,13
716                                      256
717                                      500,1,29'
718
719                                   THE ABOVE LIST  OF BAD SECTORS,  ENTRY 1 INDICATES THAT CYLINDER 8.,
720                                   TRACK 8., SECTOR 3 IS  THE  BAD SECTOR.  ENTRY 2 INDICATES  THAT ON
721                                   CYLINDER 16., TRACK 13., ALL THE  SECTORS  ARE BAD  (ENTIRE TRACK IS
722                                   BAD). ENTRY 3 INDICATES THAT ON CYLINDER 256., ALL TRACKS AND SECTORS
723                                   ARE  BAD (ENTIRE CYLINDER IS BAD).  ENTRY 4 INDICATES  THAT CYLINDER
724                                   500., TRACK 1, SECTOR 29. IS THE  BAD SECTOR.
725
726                                   AFTER TYPING EITHER OF THE TWO PREVIOUS MESSAGES, THE PROGRAM WILL
727                                   RETURN TO WAIT FOR MORE ENTRIES TO BE MADE INTO THE BAD SECTOR TABLE,
728                                   AS IN EXAMPLES 1, 2 AND 3.
729
730                                   TO TERMINATE THE BAD SECTOR ADDRESS ENTRY, TYPE A 'CARRIAGE RETURN'
731                                   IN RESPONSE TO THE ENTRY REQUEST OR TERMINATE THE ENTRY WITH A 'PERIOD'
732                                   FOLLOWED BY A 'CARRIAGE RETURN'.
733
734                             4.2.1 'T' COMMAND
735
736                                   USED TO ASSIGN A DRIVE(S) FOR A TEST. THIS COMMAND IS REQUIRED
737                                   TO PERFORM THE TEST OF  THE DRIVE(S).
738
739                                   FORMAT: TN<CR>
740
741                                   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 TERMINIATED 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 TERMINIATED 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.

```
799                         4.2.6   'WT' COMMAND
800
801                                 USED TO PERFORM A SEQUENTIAL WRITE DATA, FOLLOWED BY A 'T' COMMAND.
802
803                                 FORMAT: WTN<CR>
804
805                                 N = DRIVE NUMBER 0 TO 7 OR "A". ENTRY MUST BE TERMINATED BY A
806                                     CARRIAGE RETURN <CR>.
807
808                                 EXAMPLE: WT0<CR> - WRITE A DATA PATTERN AND TEST DRIVE 0
809                                          WTA<CR> - WRITE A DATA PATTERN AND TEST ALL DRIVES
810
811
812
813                         5.      PERFORMANCE SUMMARY TYPEOUT
814                                 ---------------------------
815
816                         5.1     THE PROGRAM WILL DISPLAY A PERFORMANCE SUMMARY FOR THE DRIVES
817                                 BEING EXERCISED.  THIS SUMMARY WILL BE DISPLAYED AUTOMATICALLY
818                                 IF THE PARAMETER 'INTRVL' IS NOT ZERO AND SW<02>=0, OR IF THE
819                                 DRIVE HAS REACHED THE DEFINED NUMBER OF PASSES AND SW<08>=0,
820                                 OR IF THE OPERATOR REQUESTS TO DO SO BY USE OF THE 'S' COMMAND.
821
822                                 THE SUMMARY TYPEOUT CONTAINS THE FOLLOWING FIELDS:
823
824                                 'TIME'          ELAPSED TIME OF PROGRAM
825                                 'DRIVE'         DRIVE NUMBER - DRIVE TYPE
826                                 'PASS'          PRESENT PASS COUNT FOR THE DRIVE
827                                 'MBA S/N'       HARD WIRED MASSBUS ADAPTER SERIAL NUMBER(RMSN)
828                                 'PACK S/N'      SERIAL NUMBER READ FROM THE DEC144 FILE ON THE PACK
829                                 'WT OFLOW'      NUMBER OF TIMES 'WRDS WRITN' HAS OVERFLOWED
830                                 'WRDS WRITN'    TOTAL NUMBER OF WORDS WRITTEN BY THE DRIVE
831                                 'RD OFLOW'      NUMBER OF TIMES 'WRDS READ' HAS OVERFLOWED
832                                 'WRDS READ'     TOTAL NUMBER OF WORDS READ BY THE DRIVE
833                                 'SEEKS'         NUMBER OF SEEK OPERATIONS THE DRIVE PERFORMED
834                                 'SOFT'          NUMBER OF SOFT DATA ERRORS
835                                 'HARD'          NUMBER OF HARD DATA ERRORS
836                                 'SKI'           NUMBER OF 'SKI' ERRORS
837                                 'MISP'          NUMBER OF PROGRAM DETECTED POSITIONING ERRORS
838                                 'OTHER'         TOTAL ERRORS OF OTHER TYPES
839
840                                 ALL DATA TRANSFER COUNTS, SEEK COUNTS AND ERROR COUNTS ARE ACCUMULATIVE
841                                 AND WILL NOT BE CLEARED AFTER EACH PASS.
842
843                                 TO CALCULATE THE TOTAL NUMBER WORDS READ OR WRITTEN, TAKE THE OVERFLOW
844                                 COUNT (RD OFLOW OR WT OFLOW), MULTIPLY IT BY 2,147,483,647., THEN ADD
845                                 THAT NUMBER TO THE WORDS READ OR WRITTEN (WRDS READ OR WRDS WRITN).
846
847                                 NOTE:  ERRORS EM1, EM2, EM3, EM4, EM5, & EM10 ARE NOT INCLUDED IN THE
848                                        'OTHER' ERROR TOTAL.
849
850                         5.2     SOFT/HARD ERROR DEFINITIONS
851
852                         5.2.1   HARD ERRORS
853
854                                 A.  A 'DTE' (DRIVE TIMING ERROR) OR A 'DCK' (DATA CHECK ERROR)
855                                     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 ATTEMP'S 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

WHEN A DRIVE IS ASSIGNED TO BE TESTED, THE PROGRAM READS THE BAD SECTOR FILE (DEC144) FROM THE DISK AND THEN ALLOWS ADDITIONAL BAD SECTORS TO BE ENTERED MANUALLY.

A MAXIMUM OF 252. BAD SECTORS ARE ALLOWED FOR EACH DRIVE. BOTH READING FROM THE DEC144 FILE AND ENTERING FROM KEYBOARD.

THE MANUALLY ENTERED BAD SECTORS ARE NOT RECORDED TO THE BAD SECTOR FILE OF THE DISK CURRENT UNDER TESTING.

IF ONE OF THE FOLLOWING ERRORS OCCURS AT A LOCATION IDENTIFIED IN THE BAD SECTOR TABLE, THE PROGRAM WILL INHIBIT THE ERROR REPORT FOR THAT ERROR.

DATA CHECK ERRORS ('DCK')

```
                        WRITE CHECK ERRORS ('WCE')
                        OPERATION INCOMPLETE ERRORS ('OPI')
                        DRIVE TIMING ERRORS ('DTE')
                        HEADER READ ERRORS ('FER W/ HCRC', 'HCE W/ HCRC' OR 'HCRC')


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 RM 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
```

```
1027                                    A WRITE CHECK ERROR OCCURRED AND 'DCK' WAS NOT SET.  THE
1028                                    WORDS WHICH CAUSED THE ERROR ARE DISPLAYED IN THE ERROR
1029                                    MESSAGE.  THE OPERATION WILL BE RETRIED 3 TIMES.
1030
1031                            EM24    HEADER READ ERROR - 'FMT' BIT DROPPED
1032
1033                                    A WRITE DATA, WRITE CHECK DATA, OR A READ DATA WAS BEING
1034                                    PERFORMED AND A 'FMT' ERROR OCCURRED.  THE PROGRAM RE-READ THE
1035                                    HEADER OF THE ERROR SECTOR AND THE 'HCRC' BIT WAS SET.  THE
1036                                    CONTENTS OF THE HEADER ARE DISPLAYED.  THE OPERATION WILL
1037                                    BE RETRIED 3 TIMES.
1038
1039                            EM25    HEADER READ ERROR - HEADER COMPARE ('HCE') ERROR
1040
1041                                    SIMILAR TO EM24, EXCEPT THAT THE 'HCE' ERROR BIT WAS
1042                                    SET INITIALLY.  THE OPERATION WILL BE RETRIED 3 TIMES.
1043
1044                            EM26    FORMAT ERROR ('FER')
1045
1046                                    FORMAT ERROR OCCURRED.  WHEN THE HEADER WAS RE-READ, THE
1047                                    'HCRC' BIT WAS NOT SET.  THE CONTENTS OF THE HEADER ARE
1048                                    DISPLAYED.  THE OPERATION WILL BE RETRIED 3 TIMES.
1049
1050                            EM27    HEADER COMPARE ('HCE') ERROR
1051
1052                                    SIMILAR TO EM26 EXCEPT THAT THE 'HCE' BIT WAS SET INITIALLY.
1053                                    THE OPERATION WILL BE RETRIED 3 TIMES.
1054
1055                            EM30    MISCELLANEOUS DRIVE ERROR
1056
1057                                    THIS MESSAGE IS GIVEN FOR THE FOLLOWING ERROR BITS:
1058                                     'AOE', 'RMR', 'ILF', OR 'ILR'
1059
1060                            EM31    OPERATION INCOMPLETE ('OPI') ERROR
1061
1062                                    AN OPERATION INCOMPLETE ERROR OCCURRED AT THE INDICATED
1063                                    SECTOR.
1064
1065                            EM32    DRIVE TIMING ('DTE') ERROR
1066
1067                                    DRIVE TIMING ERROR OCCURRED ON THE INDICATED SECTOR.  THE
1068                                    OPERATION WILL BE RETRIED 3 TIMES.
1069
1070                            EM33    PARITY ('PAR') ERROR AFTER OPERATION STARTED
1071
1072                                    THE 'PAR' BIT WAS SET WHEN THE OPERATION WAS COMPLETED.  THE
1073                                    OPERATION WILL BE RETRIED 3 TIMES.
1074
1075                            EM34    WRITE CLOCK FAILURE ('WCF')
1076
1077                                    A WRITE CLOCK FAILURE OCCURRED DURING THE OPERATION.  THE
1078                                    OPERATION WILL BE RETRIED 3 TIMES.
1079
1080                            EM35    INVALID ADDRESS ('IAE') ERROR
1081
1082                                    AN INVALID ADDRESS ERROR OCCURRED DURING THE OPERATION.
1083
```

```
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
```

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 WILI 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 SIGNALED; 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 SIGNALED BY THE RH CONTROLLER

THE OPERATION COMPLETED NORMALLY; HOWEVER, THE PROGRAM FOUND EITHER ERROR BITS IN THE RM 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 SIGNALED EITHER 'SKI' ERROR.

EM51 NOT USED

FM60 DEVICE UNSAFE

```
1141
1142                                    THE INDICATED DRIVE UNSAFE ERROR OCCURRED; THE ERROR WAS
1143                                    CLEARED BY A 'DRIVE CLEAR' INSTRUCTION.
1144
1145
1146                            7.2     DETAIL ERROR LINES
1147
1148                                    THE LINE NUMBERS GIVEN BELOW ARE FOR REFERENCE ONLY.
1149
1150                                    LINE 1
1151                                    ------
1152
1153                                    HH:MM:SS
1154
1155                                    'HH:MM:SS' IS THE TIME SINCE THE PROGRAM WAS STARTED.
1156                                    (HOURS, MINUTES, SECONDS)
1157
1158                                    LINE 2
1159                                    ------
1160
1161                                    'PRSNT COMMAND= XXXX  PREV COMMAND= YYYY'
1162
1163                                    MNEMONICS USED FOR THE COMMANDS ARE DEFINED BELOW:
1164
1165                                            SEEK    - SEEK (OCTAL 5)
1166                                            RECAL   - RECALIBRATE (OCTAL 7)
1167                                            DRVCLR  - DRIVE CLEAR (OCTAL 11)
1168                                            RELSE   - RELEASE (OCTAL 13)
1169                                            OFFSET  - OFFSET (OCTAL 15)
1170                                            RTC     - RETURN TO CENTERLINE (OCTAL 17)
1171                                            READIN  - READIN PRESET (OCTAL 21)
1172                                            PACK    - PACK ACKNOWLEDGE (OCTAL 23)
1173                                            SEARCH  - SEARCH (OCTAL 31)
1174                                            *GETREG - GET REGISTERS (OCTAL 41)
1175                                            *SETFMT - SET FORMAT (ECI OR HCI) (OCTAL 43)
1176                                            *SELDRV - SELECT DRIVE (OCTAL 45)
1177                                            WCKD    - WRITE CHECK DATA (OCTAL 51)
1178                                            WCKHD   - WRITE CHECK HEADER & DATA (OCTAL 53)
1179                                            WRTDAT  - WRITE DATA (OCTAL 61)
1180                                            WRTHD   - WRITE CHECK HEADER & DATA (OCTAL 63)
1181                                            RDDAT   - READ DATA (OCTAL 71)
1182                                            RDHD    - READ HEADER & DATA (OCTAL 73)
1183
1184                                    * SPECIAL RM DRIVER COMMAND (NOT A CONTROLLER COMMAND)
1185
1186                                    (DISPLAY OF THE RH/RM REGISTERS IN TWO GROUPS:
1187                                    RMCS1, RMCS2, RMDS, RMER1, RMER2, RMEC1 AND RMEC2 FORM THE FIRST
1188                                    GROUP;  ALL THE OTHER REGISTERS ARE IN THE SECOND GROUP.
1189                                    IF SW<05> IS SET, ONLY THE REGISTERS IN THE FIRST GROUP WILL BE
1190                                    DISPLAYED.)
1191
1192
1193                                    THE ABOVE LINE WILL BE TYPED IF THE ERROR OCCURRED DURING
1194                                    THE NON-DATA TRANSFER PART OF THE OPERATION.
1195
1196                                    '* ERROR AT BAD TRACK/SECTOR'
1197
```

```
1198                    THE ABOVE LINE WILL BE PRINTED IF A DATA ERROR OCCURES AT AN ADDRESS
1199                    ON THE DISK WHICH THE OPERATOR HAS IDENTIFIED AS BEING BAD.  PARAMETER
1200                    'NOTPRT' MUST BE 0 FOR THE ERROR TO BE REPORTED.
1201
1202                    A WORD CALLED 'STATUS' IS DISPLAYED WITH THE RM REGISTERS.  THE
1203                    CONTENTS OF THIS WORD IDENTIFY HOW THE ERROR WAS PROCESSED BY THE
1204                    RM DRIVE HANDLER ROUTINE.  (SEE SECTION 9.7)
1205
1206                    LINE 3
1207                    ------
1208
1209                    ERROR AT CXXX TYY SZZ  PREV ADDR= CUUU TVV SWW
1210
1211                    THE ACTUAL ADDRESS OF THE ERROR SECTOR AND THE PREVIOUS
1212                    DISK ADDRESS ARE GIVEN IN THIS LINE.  CYLINDER, TRACK, &
1213                    SECTOR ADDRESSES ARE IN DECIMAL.
1214
1215                    LINE 4
1216                    -------
1217
1218                    PRSNT ADDR= CXXX TYY SZZ  PREV ADDR= CUUU TVV SWW
1219
1220                    THIS LINE IDENTIFIES THE ADDRESS WHEN THE ERROR WAS DETECTED;
1221                    THE PREVIOUS ADDRESS IS ALSO GIVEN.  CYLINDER, TRACK, & SECTOR
1222                    ADDRESSES ARE GIVEN IN DECIMAL.
1223
1224                    LINE 5
1225                    -------
1226
1227                    START CYL= XXX  END CYL= YYY
1228
1229                    THIS LINE IDENTIFIES THE STARTING CYLINDER OR A SEEK (IMPLIED)
1230                    AND THE DESTINATION CYLINDER.  CYLINDER ADDRESSES ARE IN
1231                    DECIMAL.
1232
1233                    LINE 6
1234                    -------
1235
1236                    START CYL= XXX  END CYL= YYY    ACTUAL CYL= ZZZ
1237
1238                    THIS LINE IDENTIFIES THE STARTING CYLINDER OF AN IMPLIED SEEK,
1239                    THE DESTINATION CYLINDER, AND THE CYLINDER THE DISK ACTUALLY
1240                    STOPPED AT.  CYLINDER ADDRESSES ARE IN DECIMAL.
1241
1242                    LINE 7
1243                    -------
1244
1245                    RMBA= XXXX  RMWC= YYYY
1246
1247                    THIS LINE GIVES THE CONTENTS OF THE RH CONTROLLER BUFFER ADDRESS
1248                    REGISTER AND THE RH CONTROLLER WORD COUNT REGISTER.  THIS LINE IS
1249                    NOT PRINTED IF SW<05> IS NOT SET.
1250
1251                    LINE 8
1252                    -------
1253
1254                    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

RMDA= XXXX RMCA= YYYY

THIS LINE GIVES THE CONTENTS OF THE RM 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

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

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

LINE 12

HEADER CONTENTS OF ERROR SECTOR= XXXX XXXX XXXX XXXX

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

LINE 13

RMEC1= XXXX RMEC2= YYYY

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

LINE 14

ECC CORRECTABLE WITHOUT OFFSET

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

```
1312
1313                                    LINE 15
1314                                    -------
1315
1316                                    READ CORRECTLY AT (NEG OR POS) OFFSET
1317
1318                                    THE SECTOR IN ERROR WAS READ WITHOUT ERROR AT THE INDICATED
1319                                    OFFSET VALUE.
1320
1321                                    LINE 16
1322                                    -------
1323
1324                                    ECC CORRECTABLE AT (NEG OR POS) OFFSET
1325
1326                                    THE SECTOR IN ERROR BECAME ECC CORRECTABLE AT THE INDICATED
1327                                    OFFSET.
1328
1329                                    LINE 17
1330                                    -------
1331
1332                                    CORRECTED ON X RETRY
1333
1334                                    THE OPERATION WAS PERFORMED ERROR FREE ON THE INDICATED RETRY
1335                                    ATTEMPT.
1336
1337                                    LINE 18
1338                                    -------
1339
1340                                    UNCORRECTABLE AFTER X RETRIES
1341
1342                                    THE OPERATION COULD NOT BE PERFORMED CORRECTLY AFTER THE
1343                                    INDICATED NUMBER OF RETRY ATTEMPTS.
1344
1345                                    LINE 19
1346                                    ------
1347
1348                                    DIFFERENT ERROR DURING RETRY
1349
1350                                    WHILE THE PROGRAM WAS RETRYING THE ERROR, A DIFFERENT OCCURRED.
1351                                    IF THIS LINE IS PRINTED, THE RH/RM REGISTERS WILL ALSO BE
1352                                    PRINTED (SEE LINE 2).
1353
1354                                    LINE 20
1355                                    -------
1356
1357                                    DATA COMPARISON ERRORS
1358
1359                                    A PRINTOUT OF THE DATA COMPARISON ERRORS FOLLOW THIS LINE.
1360
1361                                    LINE 21
1362                                    -------
1363
1364                                    TOTAL COMPARE ERRORS= XXXX
1365
1366                                    THIS LINE GIVES THE TOTAL DATA COMPARISON ERROR COUNT.  THE
1367                                    VALUE GIVEN IS IN DECIMAL.
1368
```

```
1369                                    LINE 22
1370                                    -------
1371
1372                                    THE DATA COMPARED OK
1373
1374                                    THIS LINE INDICATES THE RESULTS OF THE DATA COMPARISON FOLLOWING
1375                                    ECC CORRECTION.
1376
1377                                    LINE 23
1378                                    --------
1379
1380                                    ECC CORRECTION RESULTS
1381
1382                                    THE PROGRAM PERFORMED ECC CORRECTION AND THE RESULTS ARE REPORTED.
1383                                    THE ADDRESS IN MEMORY OF THE WORD(S) IN ERROR ARE GIVEN, THE WORD(S)
1384                                    BEFORE CORRECTION AND THE WORD(S) AFTER CORRECTION ARE PRINTED.
1385
1386                                    LINE 24
1387                                    -------
1388
1389                                    ERROR BURST BEGINS AT WORD XXX IN DATA FIELD OF ERROR SECTOR
1390
1391                                    THIS IS AN INFORMATIONAL LINE WHICH WILL BE PRINTED FOR 'DCK' ERRORS
1392                                    WHICH ARE ECC CORRECTABLE OR WHICH BECOME ECC CORRECTABLE DURING
1393                                    RETRY.  'XXX' IS THE WORD OFFSET VALUE FROM 'RMEC1' AND IS IN
1394                                    DECIMAL.
1395
1396                                    LINE 25
1397                                    --------
1398
1399                                    ERROR WAS NOT IN THE DATA READ -
1400                                    ECC CORRECTION CAN'T BE PERFORMED
1401
1402                                    THE DATA ERROR WAS NOT IN DATA TRANSFERED TO MEMORY.
1403
1404                                    LINE 26
1405                                    -------
1406
1407                                    CONTENTS OF THE ERROR SECTOR (REPORTED ABOVE)
1408
1409                                    IF SW<03> IS SET, THE SECTOR WHICH GAVE THE 'DCK', 'DTE' OR,
1410                                    'WCF' ERROR OR 'HARD' DATA CHECK ERROR IS PRINTED.  THE
1411                                    CONTENTS OF THE SECTOR FOLLOW THIS LINE.
1412
1413                                    LINE 27
1414                                    ------
1415
1416                                    TOTAL ERRORS:X  WOFL:N WRDS WRITN: YYYY  ROFL:N WRDS READ: ZZZZ
1417
1418                                    THIS IS THE LAST LINE PRINTED FOR ALL NON-POSITIONING
1419                                    TYPE ERRORS.
1420
1421                                    'ERRORS IS THE TOTAL ERROR COUNT FOR THE DRIVE AND INCLUDES
1422                                    EVERY ERROR DETECTED, REGARDLESS OF TYPE.
1423
1424                                    'WOFL' NUMBER OF TIMES 'WRDS WRITN' HAS OVERFLOWED
1425                                    'WRDS WRITN' IS THE TOTAL NUMBER OF WORDS WRITTEN THE DRIVE.
```

'ROFL' NUMBER OF TIMES 'WRDS READ' HAS OVERFLOWED
'WRDS READ' IS THE TOTAL NUMBER OF WORD READ BY THE DRIVE.

LINE 28

TOTAL 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 SIGNALED 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 SUMMARY TYPEOUT REQUESTS
3) DRIVES REQUIRING COMMAND INITIATION, BUFFER ASSIGNEMENT, 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 RM05/3/2, 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.

```
1483
1484                    THE COMMAND INITIATION ROUTINE FIRST LOOKS AT THE CYLINDER ADDRESS OF
1485                    THE REQUESTED OPERATION. IF THE DRIVE MUST SEEK TO ANOTHER CYLINDER TO
1486                    PERFORM THE OPERATION, THE PROGRAM ISSUES A SEARCH INSTRUCTION
1487                    TO THE DRIVE WITH A 'TARGET' SECTOR WHICH IS 5 SECTORS EARLIER
1488                    THAN THE 'TRANSFER' SECTOR.  (THIS ALLOWS THE PROGRAM TO INITIATE
1489                    OPERATIONS ON ANOTHER DRIVE WHILE THE PRESENT DRIVE, OR OTHER
1490                    DRIVES, ARE SEARCHING FOR 'TARGET' SECTORS.  ALL SEEKS ISSUED BY
1491                    THE PROGRAM ARE IMPLIED SEEK SEARCH OPERATIONS.)  WHEN A SEARCHING
1492                    DRIVE FINDS THE 'TARGET' SECTOR AND INTERRUPTS, THE PROGRAM THEN
1493                    ISSUES THE REQUESTED COMMAND TO THE DRIVE THAT INTERRUPTED.
1494
1495                    WHEN THE DATA TRANSFER OPERATION IS COMPLETE, THE DRIVE REGISTERS
1496                    ARE STORED AND A DATA TRANSFER IS INITIATED FOR A WAITING DRIVE.
1497
1498                    IF THE OPERATION HAS BEEN COMPLETED NORMALLY, THE SAVED DRIVE REGISTERS
1499                    ARE CHECKED TO VERIFY THAT NO ERROR BITS ARE SET; THE RH CONTROLLER
1500                    BUS ADDRESS AND WORD COUNT ADDRESS REGISTERS ARE CHECKED TO VERIFY
1501                    THAT THE CORRECT NUMBER OF WORDS HAVE BEEN TRANSFERED AND THAT THE
1502                    TWO REGISTERS ARE CONSISTENT WITH EACH OTHER; AND IF THE COMMAND WAS
1503                    A READ COMMAND, THE DATA BUFFER IS COMPARED.  WHEN THIS SEQUENCE IS
1504                    COMPLETED, THE DRIVE IS RETURNED TO THE ASSIGNED, INACTIVE LIST. THE
1505                    PROGRAM THEN INITIATES A DATA TRANSFER ON A WAITING DRIVE AND
1506                    RESELECTS AND REINITIATES ANOTHER OPERATION ON THE RELEASED DRIVE.
1507
1508                    ERRORS WHICH OCCUR ARE PROCESSED IN THE FOLLOWING ORDER.  MULTIPLE
1509                    ERRORS WILL BE REPORTED AS THE FIRST ERROR TYPE CHECKED.
1510
1511                    A.  ERRORS REPORTED FOR OPERATIONS WHICH HAVE NOT COMPLETED NORMALLY.
1512
1513                        OPERATION NOT COMPLETED WITHIN TIME LIMIT - EM13
1514                        UNIT WENT OFFLINE - EM14
1515
1516                    B.  ERRORS REPORTED FOR OPERATIONS WHICH COMPLETE NORMALLY.
1517
1518                        CORRECTABLE UNSAFE - EM60
1519                        DRIVE TIMING ERROR - EM32
1520                        DATA CHECK ERROR - EM21
1521                        WRITE CHECK WITH DCK SET - EM22
1522                        HEADER CRC ERRORS - EM20
1523                        FORMAT ERRORS - EM24, EM26
1524                        HEADER COMPARE ERRORS - EM25, EM27
1525                        PROGRAM DETECTED POSITIONING ERROR - EM51
1526                        SEEK INCOMPLETE ERROR - EM50
1527                        WRITE CHECK WITHOUT 'DCK' SET - EM23
1528                        RH CONTROLLER OR UNIBUS TRANSFER ERROR - EM40
1529                        'OPI' ERROR - EM31
1530                        'PAR' ERROR - EM33
1531                        'WCF' ERROR - EM34
1532                        'IAE' ERROR - EM35
1533                        'WLE' ERROR - EM36
1534                        MISCELLANEOUS DRIVE ERROR - EM30
1535
1536                    C.  ERRORS NOT FLAGGED BY THE HARDWARE ERROR DETECTION LOGIC.
1537
1538                        BUS ADDRESS OR WORD COUNT INCORRECT - EM41
1539                        DATA COMPARE ERRORS - NO DRIVE ERROR DETECTED - EM42
```

```
1540                        CAN'T MATCH DATA READ WITH A PATTERN - EM43
1541                        ERROR BIT(S) SET, BUT NO ERROR SIGNALED BY THE RH CONTROLLER - EM44
1542                        ECC LOGIC FAILURE - EM45
1543                        BUS ADDRESS OR WORD COUNT NOT CONSISTENT - EM46
1544
1545            8.2     DUAL PORT OPERATION
1546
1547                    DUAL PORT OPERATION IS NEARLY IDENTICAL TO THE OPERATION DESCRIBED
1548                    IN SECTION 8.1.  THE DIFFERENCES ARE IN COMMAND SEQUENCE INITIATION
1549                    AND COMMAND TERMINATION.
1550
1551                    WHEN THE DUAL PORT HANDLER ROUTINE IN THE EXERCISER PROGRAM RECEIVES
1552                    A REQUEST FOR A DRIVE, THE PROGRAM VERIFIES THAT THE DRIVE IS
1553                    ONLINE.  THE DRIVE IS SELECTED AND THE RMCS1 REGISTER IS READ TO
1554                    TEST THE "DVA" BIT. IF THE DRIVE IS IN NEUTRAL, THIS WILL SEIZE THE
1555                    DRIVE. IF THE DRIVE IS SEIZED BY THE OTHER PORT, A DRIVE CLEAR COMMAND
1556                    IS ISSUED TO THE DRIVE TO SET 'PORT REQUEST'. THE PROGRAM THEN CHECKS
1557                    'DVA' IN 'RMCS1'.  IF THE DRIVE IS AVAILABLE AS INDICATED BY THE 'DVA'
1558                    BIT, THE COMMAND SEQUENCE WILL BE INITIATED IN THE NORMAL MANNER (SEE
1559                    SECTION 8.1 ABOVE).  IF 'DVA' WAS NOT SET, THE PROGRAM MAKES AN ENTRY
1560                    FOR THE DRIVE IN AN INTERNAL 'PORT REQUEST PENDING' TABLE AND
1561                    STARTS A 15. SECOND TIMER FOR THE DRIVE.  IF THE DRIVE HAS
1562                    NOT SWITCHED TO THE REQUESTING SYSTEM WITHIN THE 15. SECOND INTERVAL,
1563                    THE PROGRAM REPORTS A 'NO RESPONSE TO PORT REQUEST' ERROR.  NORMALLY
1564                    THIS ERROR MESSAGE INDICATES A FAILURE IN THE DUAL PORT CONTROL
1565                    LOGIC IN THE DRIVE BEING TESTED; HOWEVER, UNDER CERTAIN CONDITIONS
1566                    (E.G. MASSBUS PARITY ERRORS BEING REPORTED ON THE OTHER SYSTEM
1567                    ON A TTY), THE OTHER PROCESSOR WAS UNABLE TO PROCESS THE DRIVE
1568                    AFTER IT HAD REQUESTED THE DRIVE.  THE OPERATOR MUST BE AWARE OF
1569                    WHAT THE OTHER SYSTEM IS DOING AT ALL TIMES TO INTERPRET THE PORT
1570                    RELATED ERROR MESSAGES PROPERLY.
1571
1572                    AFTER A DRIVE HAS COMPLETED AN OPERATION, THE PROGRAM WILL STORE
1573                    THE REGISTERS AND ISSUE A 'RELEASE' TO THE DRIVE; IF THE OPERATION
1574                    TERMINATED WITH AN ERROR, THE DRIVE WILL NOT BE RELEASED UNTIL
1575                    ERROR PROCESSING HAS BEEN COMPLETED.
1576
1577                    SINGLE PORT DRIVES, DRIVES WHICH ARE IN NEUTRAL BUT NOT BEING
1578                    EXERCISED BY THE OPPOSITE PORT ARE STILL TREATED AS DUAL PORT
1579                    DRIVES IN THAT A RELEASE COMMAND IS ISSUED AT THE END OF NORMAL
1580                    COMMAND PROCESSING OR AT THE END OF ERROR PROCESSING. A RELEASE
1581                    COMMAND ISSUED UNDER THESE CONDITIONS HAS NO FUNCTIONAL EFFECT
1582                    ON THE OPERATION OF THE DRIVE.
1583
1584            8.3     SELECTION OF OPERATION VARIABLES
1585
1586                    A.  SECTOR ADDRESS SELECTION IS RANDOM BETWEEN THE VALUES IN
1587                        'MINSEC' AND 'MAXSEC'.  TRACK ADDRESS SELECTION IS RANDOM
1588                        BETWEEN THE VALUES IN 'MINTRK' AND 'MAXTRK'.  CYLINDER ADDRESS
1589                        SELECTION IS RANDOM BETWEEN 'MINCYL' AND 'MAXCYL'.  IF A MINIMUM
1590                        ADDRESS IS GREATER THAN THE CORRESPONDING MAXIMUM ADDRESS, THE
1591                        PROGRAM WILL SWAP 'MAX' AND 'MIN' ADDRESSES AND CONTINUE.
1592
1593                    B.  THE WORD COUNT IS RANDOMLY SELECTED BETWEEN 6 AND THE VALUE
1594                        'WRDCNT'. THIS IS NECESSARY AS THE PROGRAM REQUIRES 4 LOCATIONS
1595                        IN THE DATA PORTION OF THE SECTOR TO BE ABLE TO MATCH THE DATA
1596                        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	155555	026455	155555
000003	177774	000000	133331	052525	155555	026455	133333
000007	177770	000000	133331	052525	155555	026455	155555
000017	177760	177777	133331	125252	155555	151322	133333
000037	177740	177777	133331	125252	155555	151322	155555
000077	177700	177777	133331	125252	155555	151322	133333
000177	177600	000000	133331	052525	155555	026455	155555
000377	177400	000000	133331	052525	155555	026455	133333
000777	177000	177777	133331	125252	155555	151322	155555
001777	176000	177777	133331	125252	155555	151322	133333
003777	174000	000000	133331	052525	155555	026455	155555
007777	170000	177777	133331	125252	155555	151322	133333
017777	160000	000000	133331	052525	155555	026455	155555
037777	140000	177777	133331	125252	155555	151322	133333
077777	100000	000000	133331	052525	155555	026455	155555
177777	000000	177777	133331	125252	155555	151322	133333

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

```
                        004000  173777  155555  177757  066667  177777  000000
                        010000  167777  172666  177767  153333  177777  000000
                        020000  157777  155555  177773  066667  177777  000000
                        040000  137777  172666  177775  153333  177777  000000
                        100000  077777  155555  177776  066667  177777  000000

                        * WORST CASE PATTERN

9.1                     RH/RM DRIVER
                        ------------

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

9.2                     TO INITIALIZE THE DRIVER:

                                JSR     PC,RMINIT
                                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 RM05/3/2, 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             RM03
                                  5             RM02
                                  7             RM05
                                 -1             NOT AN RM05/3/2

                        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,RM05         ;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
```

```
1711
1712                        *DPB (DATA PARAMETER BLOCK)
1713
1714                        PNTDPB: .BYTE   0               ;(0) DRIVE NUMBER
1715                                .BYTE   0               ;(1) OFFSET VALUE OR FMT16, ECT, AND HCI
1716                                .BYTE   0               ;(2) COMMAND
1717                                .BYTE   0               ;(3) PSEL AND A17 AND A16
1718                                .WORD   0               ;(4) WORD COUNT (MUST BE NEG.)
1719                                .WORD   0               ;(6) BUFFER ADDRESS OR
1720                                                        ;REGISTER TABLE POINTER
1721                                .BYTE   0               ;(10) SECTOR ADDRESS OR
1722                                                        ;FIRST REG. INDEX
1723                                .BYTE   0               ;(11) TRACK ADDRESS OR
1724                                                        ;LAST REG. INDEX
1725                                .WORD   0               ;(12) CYLINDER ADDRESS
1726                                .WORD   0               ;(14) ERROR TABLE POINTER
1727                                                        ;POINTS TO THE FIRST OF TWENTY
1728                                                        ;LOCATIONS OF WHERE THE DRIVER
1729                                                        ;IS TO STORE THE RH/RM
1730                                                        ;REGISTERS ON AN ERROR. IF LEFT
1731                                                        ;ZERO REGISTERS ARE NOT SAVED.
1732                                .WORD   0               ;(16) STATUS/ERROR INDICATOR
1733                                                        ;BIT15=1=ERROR OCCURRED
1734                                                        ;BIT07=1=DONE
1735                                                        ;BIT14-BIT09 AND BIT06-BIT03
1736                                                        ;INDICATE TYPE OF ERROR
1737
1738                    9.4     THE DRIVER PROVIDES A SOFTWARE TIMEOUT CAPABILITY.
1739                            TO UTILIZE THIS CAPABILITY YOU MUST SUPPLY THE 'RM TIMER'' ROUTINE
1740                            WITH THE ELAPSED TIME IN THE FOLLOWING MANNER:
1741
1742                                    MOV     #16.,-(SP)      ;16. MILLISECONDS BETWEEN
1743                                                            ;CLOCK TICKS
1744                                    JSR     PC,RMTMR        ;CALL THE TIMER ROUTINE
1745
1746                            IT SHOULD BE NOTED THAT YOU MUST PROVIDE THE CODE TO DRIVE THE
1747                            CLOCK AND THE ELAPSED TIME MUST BE IN MILLISECONDS.
1748
1749
1750                    9.4.1   EXAMPLE - WRITE 1000. WORDS
1751
1752                            1$:     JSR     R0,RM05         ;CALL THE DRIVER
1753                                    WRTDPB                  ;DPB ADDRESS
1754                                    BR      1$              ;WAIT FOR QUEUE IF FULL
1755                            2$:     TST     WRTDPB+16       ;WAIT FOR COMMAND TO COMPLETE
1756                                    BEQ     2$
1757                                    BMI     ERROR1          ;ERROR OCCURRED
1758                                    .
1759                                    .
1760                                    .
1761
1762                            WRTDPB: .BYTE   5               ;DRIVE #5
1763                                    .BYTE   0               ;
1764                                    .BYTE   161             ;WRITE COMMAND
1765                                    .BYTE   0               ;
1766                                    .WORD   -1000.          ;WORD COUNT
1767                                    .WORD   WRTBUF          ;BUFFER ADDRESS
```

```
                        .BYTE   3               ;SECTOR
                        .BYTE   5               ;TRACK
                        .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/RM REGISTERS

MNEMONIC	INDEX
RMCS1	0
RMWC	2
RMBA	4
RMDA	6
RMCS2	10
RMDS	12
RMER1	14
RMAS	16
RMLA	20
RMDB	22
RMMR1	24
RMDT	26
RMSN	30
RMOF	32
RMDC	34
RMHR	36
RMMR2	40
RMER2	42
RMEC1	44
RMEC2	46
RMBAE	50*
RMCS3	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 VARIES 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/
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/RM REGISTERS WERE NOT SAVED)
(2) =	REQUEST QUEUE HAS BEEN EMPTIED. THE DRIVER ISSUED A 'DRIVE CLEAR' TO THE DRIVE. NOTE: ALL RH/RM 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/RM 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/RM INTERRUPT OCCURRED (RHAS=0)	*R4= RMCS1'S ADDRESS
2	UNEXPECTED ATTENTION OCCURRED	R1= DRIVE NUMBER R3= ATA BIT *R4= RMCS1'S ADDRESS R5= (RMAS) RMERRS =RMDS RMERRS+2=RMER1 RMERRS+4=RMER2 RMERRS+6=RMMR2
3	MASSBUS PARITY ERROR (MCPE=1)	RD.ADR= ADDRESS OF REG. READ RD.WRD= WORD READ

```
                        4       MASSBUS PARITY          WRT.AD= ADDRESS OF REG. WRITTEN
                                ERROR (PAR=1)           WRT.WD= WORD WRITTEN
                                                        RD.WRD= WORD READ BACK

                        5       ADDRESS PLUG CHANGE     R1= DRIVE NUMBER
                                BIT SET ('OPE' ERROR)   R3= ATA BIT
                                                       *R4= RMCS1'S ADDRESS
                                                        R5= (RMAS)
                                                        RMERRS  =RMDS
                                                        RMERRS+2=RMER1
                                                        RMERRS+4=RMER2
                                                        RMERRS+6=RMWR2

                        * THIS IS THE ACTUAL UNIBUS ADDRESS (176700)

                    @
```

```
                        ;*LAST REVISION 04-APR-81

 55
 56                     .TITLE  CZRMUB0 RM05/3/2 PERF EXER
                        ;*COPYRIGHT (C) 1981
                        ;*DIGITAL EQUIPMENT CORPORATION
                        ;*COLORADO 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
                        ;*
 57                     .SBTTL  OPERATIONAL SWITCH SETTINGS
                        ;*
                        ;*      SWITCH                  USE
                        ;*      ------          --------------------
                        ;*        15            HALT ON ERROR
                        ;*        13            INHIBIT ERROR TYPEOUTS
                        ;*        10            BELL ON ERROR
                        ;*         8            INHIBIT END OF PASS MESSAGES
 58                     ;*         7            DISPLAY ALL DATA COMPARE ERRORS
 59                     ;*         6            DON'T CHANGE PARAMETERS (LOOP ON PRESENT VALUES)
 60                     ;*         5            A. PARTIAL REGISTER DISPLAY IF ERROR
 61                     ;*                      B. NO ECC CORRECTION RESULTS DISPLAYED IF ERROR
 62                     ;*         4            A. DO NOT CHECK FOR MAXIMUM ERROR COUNTS
 63                     ;*                      B. DO NOT DROP DRIVE AT END OF TEST
 64                     ;*         3            A. DISPLAY ERROR SECTOR IF 'DCK', 'DTE', OR 'WCF' ERROR
 65                     ;*                      B. DISPLAY SECTOR IF 'DCK' ERR UNCORRECTABLE AFTER
 66                     ;*                         28TH RETRY
 67                     ;*                      C. IF DATA COMPARE ERROR & SW07 SET, DISPLAY
 68                     ;*                         REMAINDER OF BUFFER
 69                     ;*         2            A. DO NOT TYPE UNIT STATUS AT PROGRAM START
 70                     ;*                      B. DO NOT TYPE PERFORMANCE REPORT AFTER SPECIFIED TIME
 71                     ;*         1            INHIBIT DATA COMPARE AFTER READ COMMAND, W/O ERROR
 72                     ;*         0            READ ONLY MODE
 73
 74                     .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
 75
 76                              .SBTTL  RH CONTROLLER REGISTERS
 77
 78                              ;CONTROL AND STATUS REGISTER 1 (RMCS1)
 79
 80          000100              IE      = 100                   ;INTERRUPT ENABLE (BIT #6)
 81          000200              RDY     = 200                   ;READY (BIT #7)
 82          000400              A16     = 400                   ;HIGH ORDER BUS ADDRESS BIT (BIT #8)
 83          001000              A17     = 1000                  ;HIGH ORDER BUS ADDRESS BIT (BIT #9)
 84          002000              PSEL    = 2000                  ;PORT SELECT (BIT #10)
 85          020000              MCPE    = 20000                 ;MASSBUSS PARITY ERROR (BIT #13)
 86          040000              TRE     = 40000                 ;TRANSFER ERROR (BIT #14)
 87                              ;SC     = 100000                ;SPECIAL CONDITION (BIT #15)
 88
 89                              ;WORD COUNT REGISTER (RMWC)
 90                              ;(EACH BIT IS CALLED BY BIT NUMBER)
 91
 92                              ;BUS ADDRESS REGISTER (RMBA)
 93                              ;(EACH BIT IS CALLED BY BIT NUMBER)
 94
 95                              ;CONTROL AND STATUS REGISTER 2 (RMCS2)
 96
 97          000001              US1     = 1                     ;UNIT SELECT (BIT #0)
 98          000002              US2     = 2                     ;UNIT SELECT (BIT #1)
 99          000004              US4     = 4                     ;UNIT SELECT (BIT #2)
```

```
                        ;BUS ADDRESS INCREMENT INHIBIT (BIT #3)
                        ;MASSBUS PARITY TEST (BIT #4)
                        ;CLEAR (BIT #5)
                        ;INPUT READY (BIT #6)
                        ;OUTPUT READY (BIT #7)
                        ;MASS BUS PARITY ERROR (BIT #8)
                        ;MISSED TRANSFER ERROR (BIT #9)
                        ;PROGRAM ERROR (BIT #10)
                        ;NON EXISTENT MEMORY (BIT #11)
                        ;NON EXISTENT DRIVE (BIT #12)
                        ;UNIBUS PARITY ERROR (BIT #13)
                        ;WRITE CHECK ERROR (BIT #14)
                        ;DATA LATE (BIT #15)

                        ;DATA BUFFER REGISTER (RMDB)
                        ;(EACH BIT IS CALLED BY BIT NUMBER)

                        .SBTTL  RM REGISTERS

                        ;CONTROL AND STATUS 1 REGISTER. (#00)

                        ;GO BIT (BIT #0)
                        ;FUNCTION CODE BIT #1
                        ;FUNCTION CODE BIT #2
                        ;FUNCTION CODE BIT #3
                        ;FUNCTION CODE BIT #4
                        ;FUNCTION CODE BIT #5
                        ;DEVICE AVAILABLE (BIT #11)

                        ;DRIVE STATUS REGISTER (RMDS1) (#01)

                        ;OFFSET ON (BIT #0)
                        ;VOLUME VALID (BIT #6)
                        ;DRIVE READY (BIT #7)
                        ;DRIVE PRESENT (BIT #8)
                        ;PROGRAMABLE (BIT #9)
                        ;LAST BLOCK TRANSFERRED (BIT #10)
                        ;WRITE LOCK (BIT #11)
                        ;MEDIUM ON-LINE (BIT #12)
                        ;POSITIONING OPERATION IN PROGRESS (BIT #13)
                        ;COMPOSITE ERROR (BIT #14)
                        ;ATTENTION ACTIVE (BIT #15)

                        ;ERROR REGISTER #01 (RMER1) (#02)

                        ;ILLEGAL FUNCTION (BIT #0)
                        ;ILLEGAL REGISTER (BIT #1)
                        ;REGISTER MODIFICATION REFUSED (BIT #2)
                        ;PARITY ERROR (BIT #3)
                        ;FORMAT ERROR (BIT #4)
                        ;WRITE CLOCK FAIL (BIT #5)
                        ;ECC HARD ERROR (BIT #6)
                        ;HEADER COMPARE ERROR (BIT #7)
                        ;HEADER CRC ERROR (BIT #8)
                        ;ADDRESS OVERFLOW ERROR (BIT #9)
                        ;INVALID ADDRESS ERROR (BIT #10)
                        ;WRITE LOCK ERROR (BIT #11)
```

```
                                ;DRIVE TIMING ERROR (BIT #12)
                                ;OPERATION INCOMPLETE (BIT #13)
                                ;DRIVE UNSAFE (BIT #14)
                                ;DATA CHECK ERROR (BIT 15)

                                ;MAINTAINABILITY REGISTER (RMMR1)(#03)


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

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

                                ;DESIRED SECTOR/TRACK ADDRESS REGISTER (RMDA) (#05)


                                ;DRIVE TYPE REGISTER (RMDT) (#06)

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

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

                                ;SECTOR COUNT FIELD 0 (BIT #6)
                                ;SECTOR COUNT FIELD 1 (BIT #7)
                                ;SECTOR COUNT FIELD 2 (BIT #8)
                                ;SECTOR COUNT FIELD 3 (BIT #9)
                                ;SECTOR COUNT FIELD 4 (BIT #10)

                                ;SERIAL NUMBER REGISTER (RMSN) (#10)
                                ;(EACH IS CALLED BY BIT NUMBER)

                                ;OFFSET REGISTER (PMOF) (#11)

                                ;OFFSET DIRECTION
                                ;HEADER COMPARE INHIBIT (BIT #10)
                                ;ERROR CORRECTION CODE INHIBIT (BIT #11)
                                ;FORMAT BIT (BIT #12)

                                ;DESIRED CYLINDER ADDRESS (RMDC) (#12)
```

```
                        ;(EACH BIT IS CALLED BY BIT NUMBER)

                        ;CURRENT CYLINDER ADDRESS (RMCC) (#13)
                        ;(REGISTER CURRENTLY NOT USED)

                        ;RM ERROR REGISTER #02 (RMER2) (#15)

000010                  DPE     = 10                    ;DATA PARITY ERROR (BIT #3)
000200                  DVC     = 200                   ;DEVICE CHECK (BIT #7)
002000                  LBC     = 2000                  ;LOSS OF BIT CLOCK (BIT #10)
004000                  LSC     = 4000                  ;LOSS OF SYSTEM CLOCK (BIT #11)
010000                  IVC     = 10000                 ;INVLAID COMMAND ERROR (BIT #12)
020000                  OPE     = 20000                 ;OPERATOR PLUG ERROR (BIT #13)
040000                  SKI     = 40000                 ;SEEK INCOMPLETE (BIT #14)
100000                  BSE     = 100000                ;BAD SECTOR ERROR (BIT #15)

                        ;ECC POSITION REGISTER (RMEC1) (#16)
                        ;(EACH BIT IS CALLED BY BIT NUMBER)

                        ;ECC PATTERN REGISTER (RMEC2) (#17)
                        ;(EACH BIT IS CALLED BY BIT NUMBER)

                        .SBTTL  RM DRIVER COMMANDS

000101                  RNOP    = 101                   ;NO OPERATION
000105                  SEEK    = 105                   ;SEEK
000107                  RECAL   = 107                   ;RECALIBRATE
000111                  DRVCLR  = 111                   ;DRIVE CLEAR
000113                  RELSE   = 113                   ;RELEASE
000115                  OFFSET  = 115                   ;OFFSET
000117                  RTC     = 117                   ;RETURN TO CENTER LINE
000121                  READIN  = 121                   ;READ IN PRESET
000123                  ACK     = 123                   ;PACK ACKNOWLEDGE
000131                  SEARCH  = 131                   ;SEARCH
000141                  GETREG  = 141                   ;GET REGISTERS
000143                  SETFMT  = 143                   ;SET FORMAT (& ECI OR HCI)
000145                  SELDRV  = 145                   ;SELECT DRIVE
000151                  WCKD    = 151                   ;WRITE CHECK DATA
000153                  WCKHD   = 153                   ;WRITE CHECK HEADER & DATA
000161                  WRTDAT  = 161                   ;WRITE DATA
000163                  WRTHD   = 163                   ;WRITE HEADER & DATA
000171                  RDDAT   = 171                   ;READ DATA
000173                  RDHD    = 173                   ;READ HEADER & DATA

176700                  ABASE   = 176700
000254                  AVECT1  = 254
```

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

                                .SBTTL  STARTING ADDRESS(ES)

  000200        000137  003644        JMP     @#START1          ;;JUMP TO STARTING ADDRESS OF PROGRAM
2
3 000204        000137  003634        JMP     @#START           ;CHANGE THE RH/RM ADDRESS
4
5
                                .SBTTL  ACT11 HOOKS

                                ;;*****************************************************************
                                ;HOOKS REQUIRED BY ACT11
                000210          $SVPC=.                   ;SAVE PC
                000046          .=46
  000046        031622          $ENDAD                    ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
                000052          .=52
  000052        040000          .WORD   40000             ;;2)SET LOC.52 TO 40000
                000210          .=$SVPC                   ;; RESTORE PC
6
7               001100  .=1100
8                               .SBTTL  APT PARAMETER BLOCK
                                ;;*****************************************************************
                                ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                                ;;*****************************************************************
                001100          .$X=.   ;;SAVE CURRENT LOCATION
                000024          .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
  000024        000200          200     ;;FOR APT START UP
                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)
9               001114  TAB.XY .        ;CMTAGSTARING ADDRESS
10
```

```
                        .SBTTL  COMMON TAGS

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

001114                          .=TAB.XY
001114                  $CMTAG:                         ;;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         SWR:     .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            $TPFLG:  .BYTE   0               ;;''TERMINAL AVAILABLE'' FLAG (BIT<07>=0 YES)
001174  000000         $TMP0:   .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

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

4P' MAILBOX-ETABLE

```
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'' ABO,E
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              $DEVM:  .WORD   ADEVM   ;;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          $RMADR: .WORD   176700  ;FIRST ADDRESS OF RH/RM REGISTERS
001274  000254          $RMVEC: .WORD   254     ;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  177777          PCLOCK: .WORD   -1      ;'0' IF KW11-P IS ON SYSTEM
001312  177777          CLKFLG: .WORD   -1      ;'0' IF A CLOCK IS AVAILABLE
001314  000074          HZ:     .WORD   60.     ;60. IF 60HZ SYSTEM, 50. IF 50HZ SYSTEM
001316  000000          STATIN: .WORD   0       ;'TYPE STATISTICS' INDICATOR
001320  000000          PACK:   .WORD   0       ;'W ' COMMAND INDICATOR

        001220          DRIVE   =$UNIT          ;DRIVE # STORAGE: ERRORS 1-5 & 10
                                                ;SAME AS USED IN APT
001322  000000          ATTN:   .WORD   0       ;ATTN REG STORAGE: ERRORS 1-5 & 10
001324  000000          UNIT:   .WORD   0       ;DRIVE # STORAGE FOR PRINTOUT
001326  000000          MASK:   .WORD   0       ;ERROR RETRY REGISTER MASK
001330     000      000 RETRY:  .BYTE   0,0     ;ERROR RETRY LIMIT IN THE LOWER BYTE
                                                ;RETRY COUNT IN THE UPPER BYTE
001332  000003          FAIRNS: .WORD   3       ;MAXIMUM TIME IN QUEUE VALUE
001334  000000          LSTAD:  .WORD   0       ;STORE LAST MEMORY ADDRESS HERE
001336  000000          CHGADR: .WORD   0       ;CHANGE RH/RM UNIBUS ADDRESS FLAG
001340  000000          CFLAG:  .WORD   0       ;IF = -1, 'CONTROL C' WAS FLAGGED
001342  000000          BADSEC: .WORD   0       ;BAD TRACK/SECTOR FLAG
001344  000000          HOUR:   .WORD   0       ;HOUR COUNT STORED HERE
001346  000000          MINUTE: .WORD   0       ;MINUTE'S COUNT STORED HERE
001350  000000          SECOND: .WORD   0       ;SECOND'S COUNT STORED HERE
001352  00000C          ONESEC: .WORD   0       ;TIMER ROUTINE COUNTER (FOR ONE SECOND)
001354  177777          ZROIND: .WORD   -1      ;ZERO INDICATOR FOR THE DATA COMPARE ROUTINE
001356     000          FRSTER: .BYTE   0       ;DATA COMPARE ERROR FLAG
                                                ;IF > 0, PROCESSING 'DCKER' OR CAN'T MATCH PATTERN
                                                ;IF < 0, MISCOMPARSION FOUND
001357     000                  .BYTE   0       ;MISCOMPARSION OR CAN'T MATCH PATTERN FLAG
                                                ;IF < 0, ERROR IN BUFFER
001360  000000          SAVER1: .WORD   0       ;SAVE R1 HERE
001362  000000          SAVER5: .WORD   0       ;SAVE R5 HERE
001364  000000          ERCTR:  .WORD   0       ;NUMBER OF ERRORS
001366  000000          LIMIT:  .WORD   0       ;DISPLAY LIMIT
001370  000000          CMCNT:  .WORD   0       ;WORD COUNT
001372  000000          CMCYL:  .WORD   0       ;CYLINDER ADDRESS
001374     000          CMSEC:  .BYTE   0       ;SECTOR ADDRESS
001375     000          CMTRK:  .BYTE   0       ;TRACK ADDRESS
001376  000000          ECBIT:  .WORD   0       ;ERROR BURST BIT OFFSET
001400  000000          ECSEC:  .WORD   0       ;ERROR BURST WORD OFFSET (RELATIVE TO SECTOR)
001402  000000          ECMSK0: .WORD   0       ;CORRECTION MASK FOR FIRST ERROR WORD
001404  000000          ECMSK1: .WORD   0       ;CORRECTION MASK FOR SECOND ERROR WORD
001406  000000          ECWRD:  .WORD   0       ;LOCATION OF FIRST ERROP WORD
001410  000000          ECGD:   .WORD   0       ;GOOD DATA, FIRST WORD
001412  000000          ECBAD0: .WORD   0       ;BAD DATA, FIRST WORD
001414  000000          ECWRD1: .WORD   0       ;LOCATION OF SECOND ERROR WORD
001416  000000          ECGD1:  .WORD   0       ;GOOD DATA, SECOND WORD
001420  000000          ECBAD1: .WORD   0       ;BAD DATA, SECOND WORD
001422  001055          CYLIMT: .WORD   557.    ;CYLINDER ADDRESS LIMIT
001424  000036          SECLMT: .WORD   30.     ;SECTOR ADDRESS LIMIT
```

```
001426  000015                  TRKLMT: .WORD   13.     ;TRACK ADDRESS LIMIT
001430  000000                  DRVPAR: .WORD   0       ;WHEN DRIVES ARE BEING ASSIGNED,
                                                        ;0=CHANGE DRIVE PARAMETERS
                                                        ;1=DO NOT CHANGE DRIVE PARAMETERS
001432  000000                  XXDP:   .WORD   0       ;THE LOW BYTE CONTAINS THE DRIVE NUMBER FROM WHICH
                                                        ;THE PROGRAM WAS LOADED. THE HIGH BYTE CONTAINS THE
                                                        ;'XXDP' DEVICE CODE FOR THE RM05/3/2.

                                .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 3.33 PASSES TO REACH THE SOFT ERROR RATE OF
                                ;1 X 10^10 BITS (6.25 X 10^8 WORDS) READ OR 10. PASSES TO REACH THE 90%
                                ;CONFIDENCE LEVEL OF 3 X 10^10 BITS (1.875 X 10^9 WORDS) READ.
                                ;ENDCON= LSB AND ENDCON+2= MSB

001434  002740                  ENDCON: .WORD   002740  ;(1.875 X 10^8 WORDS) OR (3 X 10^9 BITS) READ
001436  005455                          .WORD   005455

                                ;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 X 10^6 SEEKS OR 3
                                ;PASSES TO REACH A 90% CONFIDENCE LEVEL OF 3 X 10^6 SEEKS.
                                ;ENDSEK= LSB AND ENDSEK+2= MSB

001440  015200                  ENDSEK: .WORD   015200  ;(1 X 10^6 SEEKS)
001442  000006                          .WORD   000006

001444  000031                  MAXER:  .WORD   25.     ;MAXIMUM ERRORS ALLOWED PER DRIVE
001446  000004                  CMPLMT: .WORD   4       ;NUMBER OF COMPARE ERRORS TYPED OUT
001450  020000                  WRDCNT: .WORD   8192.   ;MAXIMUM WORD COUNT (32. SECTORS)
001452  000000  000000          INTRVL: .WORD   0,0     ;FIRST WORD IS THE PERFORMANCE TYPEOUT INTERVAL
                                                        ;(IN MINUTES). SECOND WORD IS THE INTERVAL COUNTER.
001456  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].
001460  000000                  PATTERN:.WORD   0       ;IF EQ 0,RANDOMLY SELECT DATA PATTERN
                                                        ;IF NOT EQ 0,SELECT ONE SET OF PATTERN
                                                        ;POINTED BY THE 'PATTERN'.
001462  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
                                                        ;THE WORD COUNT
001464  000003                  RATIO:  .WORD   3       ;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
001466  000001                  ENDING: .WORD   1       ;IF NOT EQ 0, END OF PASS DETERMINED
                                                        ;BY THE 'WORDS READ' COUNT. (2.5 X 10^8 WORDS)
                                                        ;IF EQ 0, END OF PASS DETERMINED
                                                        ;BY THE SEEK COUNT. (4 X 10^5 SEEKS)
```

```
001470  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.
001472  000001                  MESSAGE:.WORD   1       ;IF EQ 1, DO NOT PRINT DATA ERROR MESSAGES
                                                        ;ASSOCIATED WITH OPERATOR SPECIFIED
                                                        ;BAD SECTOR AREAS.
                                                        ;IF NOT EQ 0, PRINT ERROR MESSAGES RELATING TO
                                                        ;THESE AREAS.
001474  000000                  RANDOM: .WORD   0       ;IF EQ TO 0, RANDOMLY SELECT DATA BLOCK
                                                        ;ADDRESS. IF NOT EQU 0,SEQUENTIALLY
                                                        ;SELECT DATA BLOCK ADDRESS
001476  000000                  BADBLK: .WORD   0       ;IF EQ TO 1, THE BAD SECTOR ENTRY TABLE WILL ALWAYS
                                                        ;BE INITIALIZED WHEN ASSIGNING A DRIVE; IF EQ TO 0,
                                                        ;THE BAD SECTOR ENTRY TABLE WILL ONLY BE INITIALIZED
                                                        ;IF THE PACK SERIAL NUMBER HAS CHANGED SINCE THE
                                                        ;LAST TIME IT WAS READ. (NOTE: IF THE SERIAL NO. HAS
                                                        ;CHANGED, THIS MOST LIKELY MEANS THAT THE PACK OR DRIVE
                                                        ;HAD BEEN REPLACED WHILE THE DRIVE WAS DEASSIGNED)

                                .SBTTL  VALUES FOR FIRST OPERATION

001500  000010                  BEGPAT: .WORD   8.      ;STARTING PATTERN CODE [RANGE 1 - 15.]
001502  000004                  BEGCOD: .WORD   4       ;STARTING COMMAND CODE [RANGE 0 - 5]
                                                        ;0 = WRITE CHECK DATA ('WCKD')
                                                        ;1 = WRITE CHECK HEADER & DATA ('WCHKHD' - NOT USED)
                                                        ;2 = WRITE DATA ('WRTDAT')
                                                        ;3 = WRITE HEADER & DATA ('WRTHD' - NOT USED)
                                                        ;4 = READ DATA ('RDDAT')
                                                        ;5 = READ HEADER & DATA ('RDHD')
001504  000400                  BEGWC:  .WORD   256.    ;STARTING  WRD CNT [RANGE 6 - WRDCNT]

                                .SBTTL  TABLES, CONSTANTS, AND VARIABLE LOCATIONS

                                ;LIST OF DRIVES PERFORMING COMMANDS
001506  000000                  ORDERQ: .WORD   0
001510  000000                          .WORD   0
001512  000000                          .WORD   0
001514  000000                          .WORD   0
001516  000000                          .WORD   0
001520  000000                          .WORD   0
001522  000000                          .WORD   0
001524  000000                          .WORD   0
001526  000000                          .WORD   0

001530  000000                  ASNLST: .WORD   0               ;A BIT SET IS AN ASSIGNED DRIVE

                                ;ADDRESSES OF DRIVES TO BE DEASSIGNED
001532  000000                  DUNIT:  .WORD   0
001534  000000                          .WORD   0
001536  000000                          .WORD   0
001540  000000                          .WORD   0
001542  000000                          .WORD   0
001544  000000                          .WORD   0
001546  000000                          .WORD   0
001550  000000                          .WORD   0
001552  000000                          .WORD   0
```

```
                                        ;ADDRESSES OF NEWLY ASSIGNED DRIVES
001554  000000                  NEWUNT: .WORD   0
001556  000000                          .WORD   0
001560  000000                          .WORD   0
001562  000000                          .WORD   0
001564  000000                          .WORD   0
001566  000000                          .WORD   0
001570  000000                          .WORD   0
001572  000000                          .WORD   0
001574  000000                          .WORD   0

                                        ;LIST OF DRIVES WAITING FOR BUFFERS/PARAMETERS
001576  000000                  AVAIL:  .WORD   0
001600  000000                          .WORD   0
001602  000000                          .WORD   0
001604  000000                          .WORD   0
001606  000000                          .WORD   0
001610  000000                          .WORD   0
001612  000000                          .WORD   0
001614  000000                          .WORD   0
001616  000000                          .WORD   0

                                        ;LIST OF DRIVES WAITING FOR BUFFERS
001620  000000                  WAIT:   .WORD   0
001622  000000                          .WORD   0
001624  000000                          .WORD   0
001626  000000                          .WORD   0
001630  000000                          .WORD   0
001632  000000                          .WORD   0
001634  000000                          .WORD   0
001636  000000                          .WORD   0
001640  000000                          .WORD   0


                                        ;BUFFER ALLOCATION TABLE ENTRY COUNT
001642  000000                  BUFTBL: .WORD   0
001644  000000  000000                  .WORD   0,0
001650  000000  000000                  .WORD   0,0
001654  000000  000000                  .WORD   0,0
001660  000000  000000                  .WORD   0,0
001664  000000  000000                  .WORD   0,0
001670  000000  000000                  .WORD   0,0
001674  000000  000000                  .WORD   0,0
001700  000000  000000                  .WORD   0,0
001704  000000  000000                  .WORD   0,0
001710  000000  000000                  .WORD   0,0
001714  000000  000000                  .WORD   0,0
001720  000000  000000                  .WORD   0,0
001724  000000  000000                  .WORD   0,0
001730  000000  000000                  .WORD   0,0
001734  000000  000000                  .WORD   0,0
001740  000000  000000                  .WORD   0,0
001744  000000  000000                  .WORD   0,0
001750  000000  000000                  .WORD   0,0
001754  000000  000000                  .WORD   0,0
001760  000000  000000                  .WORD   0,0
```

```
001764  000000  000000                  .WORD   0,0
001770  000000  000000                  .WORD   0,0
001774  000000  000000                  .WORD   0,0
002000  000000  000000                  .WORD   0,0
002004  000000  000000                  .WORD   0,0
002010  000000  000000                  .WORD   0,0
002014  000000  000000                  .WORD   0,0
002020  000000  000000                  .WORD   0,0
002024  000000  000000                  .WORD   0,0
002030  000000  000000                  .WORD   0,0
002034  000000  000000                  .WORD   0,0
002040  000000  000000                  .WORD   0,0

                                ;DRIVE PARAMETER BLOCK(DPB) POINTER TABLE
002044  046202                  BLKADR: .WORD   DRIVE0  ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 0
002046  050414                          .WORD   DRIVE1  ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 1
002050  052626                          .WORD   DRIVE2  ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 2
002052  055040                          .WORD   DRIVE3  ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 3
002054  057252                          .WORD   DRIVE4  ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 4
002056  061464                          .WORD   DRIVE5  ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 5
002060  063676                          .WORD   DRIVE6  ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 6
002062  066110                          .WORD   DRIVE7  ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 7

                                ;DRIVER COMMAND CONTROL TABLE (USED IN RM DRIVER)
002064      151                 COMTBL: .BYTE   WCKD    ;WRITE CHECK DATA
002065      377                         .BYTE   -1      ;WRITE CHECK HEADER AND DATA (NOT USED)
002066      161                         .BYTE   WRTDAT  ;WRITE DATA
002067      377                         .BYTE   -1      ;WRITE HEADER AND DATA (NOT USED)
002070      171                         .BYTE   RDDAT   ;READ DATA
002071      173                         .BYTE   RDHD    ;READ HEADER AND DATA

                                ;FUNCTION(COMMAND) CODE CONTROL TABLE
002072      004                 OPTBL:  .BYTE   4       ;SEEK
002073      006                         .BYTE   6       ;RECAL
002074      010                         .BYTE   10      ;DRIVE CLEAR
002075      012                         .BYTE   12      ;RELEASE
002076      014                         .BYTE   14      ;OFFSET
002077      016                         .BYTE   16      ;RETURN TO CENTERLINE
002100      020                         .BYTE   20      ;READIN PRESET
002101      022                         .BYTE   22      ;PACK ACKNOWLEDGE
002102      030                         .BYTE   30      ;SEARCH
002103      050                         .BYTE   50      ;WRITE CHECK DATA
002104      052                         .BYTE   52      ;WRITE CHECK HEADER AND DATA
002105      060                         .BYTE   60      ;WRITE DATA
002106      062                         .BYTE   62      ;WRITE HEADER AND DATA
002107      070                         .BYTE   70      ;READ DATA
002110      072                         .BYTE   72      ;READ HEADER AND DATA
002111      377                         .BYTE   -1      ;TERMINATOR
                                .EVEN

                                ;MESSAGE CONTROL TABLE FOR 'OPTBL' TABLE
002112  123     105     105     MNTBL:  .ASCIZ  /SEEK   /
002122  122     105     103             .ASCIZ  /RECAL  /
002132  104     122     126             .ASCIZ  /DRVCLR /
002142  122     105     114             .ASCIZ  /RELSE  /
002152  117     106     106             .ASCIZ  /OFFSET /
002162  122     124     103             .ASCIZ  /RTC    /
```

```
002172  122  105  101          .ASCIZ  /READIN /
002202  120  101  103          .ASCIZ  /PACK   /
002212  123  105  101          .ASCIZ  /SEARCH /
002222  127  103  113          .ASCIZ  /WCKD   /
002232  127  103  113          .ASCIZ  /WCKHD  /
002242  127  122  124          .ASCIZ  /WRTDAT /
002252  127  122  124          .ASCIZ  /WRTHD  /
002262  122  104  104          .ASCIZ  /RDDAT  /
002272  122  104  110          .ASCIZ  /RDHD   /
002302  116  117  116          .ASCIZ  /NONE   /

002312  101  106  124  OFMSG0: .ASCIZ  /AFTER RETRY WITHOUT OFFSET/
002345  101  124  040  OFMSG1: .ASCIZ  /AT NEGATIVE OFFSET/
002370  101  124  040  OFMSG2: .ASCIZ  /AT POSITIVE OFFSET/
                      .EVEN

002414  000            OFFCOD: .BYTE   0       ;OFFSET CODE TABLE
002415  001                    .BYTE   1       ;NUMBER FOR NEGATIVE OFFSET (DIR = OUT)
002416  000                    .BYTE   0       ;NUMBER FOR POSITIVE OFFSET (DIR - IN)
                      .EVEN

002420  002312         OFMTBL: .WORD   OFMSG0  ;1ST OFFSET MESSAGE
002422  002345                 .WORD   OFMSG1  ;2ND OFFSET MESSAGE
002424  002370                 .WORD   OFMSG2  ;3RD OFFSET MESSAGE

                      .SBTTL  DATA PATTERNS

                      ;STANDARD DATA PATTERN POINTER TABLE
002426  002472         STNDAT: .WORD   DATA0   ;STANDARD DATA PATTERN 0
002430  002532                 .WORD   DATA1   ;STANDARD DATA PATTERN 1
002432  002572                 .WORD   DATA2   ;STANDARD DATA PATTERN 2
002434  002632                 .WORD   DATA3   ;STANDARD DATA PATTERN 3
002436  002672                 .WORD   DATA4   ;STANDARD DATA PATTERN 4
002440  002732                 .WORD   DATA5   ;STANDARD DATA PATTERN 5
002442  002772                 .WORD   DATA6   ;STANDARD DATA PATTERN 6
002444  003032                 .WORD   DATA7   ;STANDARD DATA PATTERN 7
002446  003072                 .WORD   DATA8   ;STANDARD DATA PATTERN 8
002450  003132                 .WORD   DATA9   ;STANDARD DATA PATTERN 9
002452  003172                 .WORD   DATA10  ;STANDARD DATA PATTERN 10
002454  003232                 .WORD   DATA11  ;STANDARD DATA PATTERN 11
002456  003272                 .WORD   DATA12  ;STANDARD DATA PATTERN 12
002460  003332                 .WORD   DATA13  ;STANDARD DATA PATTERN 13
002462  003372                 .WORD   DATA14  ;STANDARD DATA PATTERN 14
002464  003432                 .WORD   DATA15  ;STANDARD DATA PATTERN 15
002466  002472                 .WORD   ZEROS   ;ALL 0'S PATTERN
002470  003374                 .WORD   ONES    ;ALL 1'S PATTERN

002472         ZEROS:
002472  000000 DATA0:  .WORD   0       ;ALL 0'S DATA PATTERN
002474  000000         .WORD   0
002476  000000         .WORD   0
002500  000000         .WORD   0
002502  000000         .WORD   0
002504  000000         .WORD   0
002506  000000         .WORD   0
002510  000000         .WORD   0
002512  000000         .WORD   0
```

```
002514  000000                          .WORD   0
002516  000000                          .WORD   0
002520  000000                          .WORD   0
002522  000000                          .WORD   0
002524  000000                          .WORD   0
002526  000000                          .WORD   0
002530  000000                          .WORD   0

002532  000001          DATA1:          .WORD   000001  ;STANDARD PATTERN 1
002534  000003                          .WORD   000003
002536  000007                          .WORD   000007
002540  000017                          .WORD   000017
002542  000037                          .WORD   000037
002544  000077                          .WORD   000077
002546  000177                          .WORD   000177
002550  000377                          .WORD   000377
002552  000777                          .WORD   000777
002554  001777                          .WORD   001777
002556  003777                          .WORD   003777
002560  007777                          .WORD   007777
002562  017777                          .WORD   017777
002564  037777                          .WORD   037777
002566  077777                          .WORD   077777
002570  177777                          .WORD   177777

002572  177776          DATA2:          .WORD   177776  ;STANDARD PATTERN 2
002574  177774                          .WORD   177774
002576  177770                          .WORD   177770
002600  177760                          .WORD   177760
002602  177740                          .WORD   177740
002604  177700                          .WORD   177700
002606  177600                          .WORD   177600
002610  177400                          .WORD   177400
002612  177000                          .WORD   177000
002614  176000                          .WORD   176000
002616  174000                          .WORD   174000
002620  170000                          .WORD   170000
002622  160000                          .WORD   160000
002624  140000                          .WORD   140000
002626  100000                          .WORD   100000
002630  000000                          .WORD   000000

002632  000000          DATA3:          .WORD   000000  ;STANDARD PATTERN 3
002634  000000                          .WORD   000000
002636  000000                          .WORD   000000
002640  177777                          .WORD   177777
002642  177777                          .WORD   177777
002644  177777                          .WORD   177777
002646  000000                          .WORD   000000
002650  000000                          .WORD   000000
002652  177777                          .WORD   177777
002654  177777                          .WORD   177777
002656  000000                          .WORD   000000
002660  177777                          .WORD   177777
002662  000000                          .WORD   000000
002664  177777                          .WORD   177777
002666  000000                          .WORD   000000
```

```
002670  177777                          .WORD   177777

002672  133331          DATA4:  .WORD   133331  ;STANDARD PATTERN 4
002674  133331                  .WORD   133331
002676  133331                  .WORD   133331
002700  133331                  .WORD   133331
002702  133331                  .WORD   133331
002704  133331                  .WORD   133331
002706  133331                  .WORD   133331
002710  133331                  .WORD   133331
002712  133331                  .WORD   133331
002714  133331                  .WORD   133331
002716  133331                  .WORD   133331
002720  133331                  .WORD   133331
002722  133331                  .WORD   133331
002724  133331                  .WORD   133331
002726  133331                  .WORD   133331
002730  133331                  .WORD   133331

002732  052525          DATA5:  .WORD   052525  ;STANDARD PATTERN 5
002734  052525                  .WORD   052525
002736  052525                  .WORD   052525
002740  125252                  .WORD   125252
002742  125252                  .WORD   125252
002744  125252                  .WORD   125252
002746  052525                  .WORD   052525
002750  052525                  .WORD   052525
002752  125252                  .WORD   125252
002754  125252                  .WORD   125252
002756  052525                  .WORD   052525
002760  125252                  .WORD   125252
002762  052525                  .WORD   052525
002764  125252                  .WORD   125252
002766  052525                  .WORD   052525
002770  125252                  .WORD   125252

002772  155555          DATA6:  .WORD   155555  ;STANDARD PATTERN 6
002774  155555                  .WORD   155555
002776  155555                  .WORD   155555
003000  155555                  .WORD   155555
003002  155555                  .WORD   155555
003004  155555                  .WORD   155555
003006  155555                  .WORD   155555
003010  155555                  .WORD   155555
003012  155555                  .WORD   155555
003014  155555                  .WORD   155555
003016  155555                  .WORD   155555
003020  155555                  .WORD   155555
003022  155555                  .WORD   155555
003024  155555                  .WORD   155555
003026  155555                  .WORD   155555
003030  155555                  .WORD   155555

003032  026455          DATA7:  .WORD   026455  ;STANDARD PATTERN 7
003034  026455                  .WORD   026455
003036  026455                  .WORD   026455
003040  151322                  .WORD   151322
```

```
003042  151322                          .WORD   151322
003044  151322                          .WORD   151322
003046  026455                          .WORD   026455
003050  026455                          .WORD   026455
003052  151322                          .WORD   151322
003054  151322                          .WORD   151322
003056  026455                          .WORD   026455
003060  151322                          .WORD   151322
003062  026455                          .WORD   026455
003064  151322                          .WORD   151322
003066  026455                          .WORD   026455
003070  151322                          .WORD   151322

003072  155555                  DATA8:  .WORD   155555  ;STANDARD PATTERN 8 (WORST CASE)
003074  133333                          .WORD   133333
003076  155555                          .WORD   155555
003100  133333                          .WORD   133333
003102  155555                          .WORD   155555
003104  133333                          .WORD   133333
003106  155555                          .WORD   155555
003110  133333                          .WORD   133333
003112  155555                          .WORD   155555
003114  133333                          .WORD   133333
003116  155555                          .WORD   155555
003120  133333                          .WORD   133333
003122  155555                          .WORD   155555
003124  133333                          .WORD   133333
003126  155555                          .WORD   155555
003130  133333                          .WORD   133333

003132  000001                  DATA9:  .WORD   000001  ;STANDARD PATTERN 9
003134  000002                          .WORD   000002
003136  000004                          .WORD   000004
003140  000010                          .WORD   000010
003142  000020                          .WORD   000020
003144  000040                          .WORD   000040
003146  000100                          .WORD   000100
003150  000200                          .WORD   000200
003152  000400                          .WORD   000400
003154  001000                          .WORD   001000
003156  002000                          .WORD   002000
003160  004000                          .WORD   004000
003162  010000                          .WORD   010000
003164  020000                          .WORD   020000
003166  040000                          .WORD   040000
003170  100000                          .WORD   100000

003172  177776                  DATA10: .WORD   177776  ;STANDARD PATTERN 10
003174  177775                          .WORD   177775
003176  177773                          .WORD   177773
003200  177767                          .WORD   177767
003202  177757                          .WORD   177757
003204  177737                          .WORD   177737
003206  177677                          .WORD   177677
003210  177577                          .WORD   177577
003212  177377                          .WORD   177377
003214  176777                          .WORD   176777
```

```
003216  175777                          .WORD   175777
003220  173777                          .WORD   173777
003222  167777                          .WORD   167777
003224  157777                          .WORD   157777
003226  137777                          .WORD   137777
003230  077777                          .WORD   077777

003232  172666                  DATA11: .WORD   172666  ;STANDARD PATTERN 11
003234  155555                          .WORD   155555
003236  172666                          .WORD   172666
003240  155555                          .WORD   155555
003242  172666                          .WORD   172666
003244  155555                          .WORD   155555
003246  172666                          .WORD   172666
003250  155555                          .WORD   155555
003252  172666                          .WORD   172666
003254  155555                          .WORD   155555
003256  172666                          .WORD   172666
003260  155555                          .WORD   155555
003262  172666                          .WORD   172666
003264  155555                          .WORD   155555
003266  172666                          .WORD   172666
003270  155555                          .WORD   155555

003272  077777                  DATA12: .WORD   077777  ;STANDARD PATTERN 12
003274  137777                          .WORD   137777
003276  157777                          .WORD   157777
003300  167777                          .WORD   167777
003302  173777                          .WORD   173777
003304  175777                          .WORD   175777
003306  176777                          .WORD   176777
003310  177377                          .WORD   177377
003312  177577                          .WORD   177577
003314  177677                          .WORD   177677
003316  177737                          .WORD   177737
003320  177757                          .WORD   177757
003322  177767                          .WORD   177767
003324  177773                          .WORD   177773
003326  177775                          .WORD   177775
003330  177776                          .WORD   177776

003332  153333                  DATA13: .WORD   153333  ;STANDARD PATTERN 13
003334  066667                          .WORD   066667
003336  153333                          .WORD   153333
003340  066667                          .WORD   066667
003342  153333                          .WORD   153333
003344  066667                          .WORD   066667
003346  153333                          .WORD   153333
003350  066667                          .WORD   066667
003352  153333                          .WORD   153333
003354  066667                          .WORD   066667
003356  153333                          .WORD   153333
003360  066667                          .WORD   066667
003362  153333                          .WORD   153333
003364  066667                          .WORD   066667
003366  153333                          .WORD   153333
003370  066667                          .WORD   066667
```

```
003372  000000          DATA14: .WORD   000000  ;STANDARD PATTERN 14
003374  177777          ONES:   .WORD   177777  ;ALL 1'S DATA PATTERN
003376  177777                  .WORD   177777
003400  177777                  .WORD   177777
003402  177777                  .WORD   177777
003404  177777                  .WORD   177777
003406  177777                  .WORD   177777
003410  177777                  .WORD   177777
003412  177777                  .WORD   177777
003414  177777                  .WORD   177777
003416  177777                  .WORD   177777
003420  177777                  .WORD   177777
003422  177777                  .WORD   177777
003424  177777                  .WORD   177777
003426  177777                  .WORD   177777
003430  177777                  .WORD   177777

003432  177777          DATA15: .WORD   177777  ;STANDARD PATTERN 15
003434  000000                  .WORD   000000
003436  000000                  .WORD   000000
003440  000000                  .WORD   000000
003442  000000                  .WORD   000000
003444  000000                  .WORD   000000
003446  000000                  .WORD   000000
003450  000000                  .WORD   000000
003452  000000                  .WORD   000000
003454  000000                  .WORD   000000
003456  000000                  .WORD   000000
003460  000000                  .WORD   000000
003462  000000                  .WORD   000000
003464  000000                  .WORD   000000
003466  000000                  .WORD   000000
003470  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


  003472                        $ERRTB:
    1
    2                           ;ERROR 1
    3
    4 003472  070412                    EM1             ;RH CONTROLLER INTERRUPT OCCURRED (RMAS 0)
    5 003474  073063                    DH1
    6 003476  073530                    DT1
    7 003500  000000                    0
    8
    9                           ;ERROR 2
   10
   11 003502  070464                    EM2             ;UNEXPECTED ATTENTION OCCURRED
   12 003504  073070                    DH2
   13 003506  073534                    DT2
   14 003510  000000                    0
   15
   16                           ;ERROR 3
   17
   18 003512  070522                    EM3             ;MASSBUS PARITY ERROR (MCPE 1)
   19 003514  073144                    DH3
   20 003516  073552                    DT3
   21 003520  000000                    0
   22
   23                           ;ERROR 4
   24
   25 003522  070560                    EM4             ;MASSBUS PARITY ERROR (PAR=1)
   26 003524  073172                    DH4
   27 003526  073562                    DT4
   28 003530  000000                    0
   29
   30                           ;ERROR 5
   31
   32 003532  070615                    EM5             ;ADDRESS PLUG BIT CHANGED
   33 003534  073070                    DH2
   34 003536  073534                    DT2
   35 003540  000000                    0
   36
   37                           ;ERROR 6
   38
   39 003542  070651                    EM6             ;RH CON'ROLLER DIDN'T RESPOND TO ADDRESSING
   40 003544  073231                    DH6
   41 003546  073574                    DT6
   42 003550  000000                    0
```

```
                                        ;THIS ROUTINE HANDLES UNEXPECTED TIMEOUTS
                                        ;
003552  011600                  BADTMO: MOV     (SP),R0         ;SAVE PC WHERE THE TIMF OUT OCCURED
003554  005740                          TST     -(R0)           ;ADJUST PC -2
003556  022626                          CMP     (SP)+,(SP)+     ;RESTORE STACK POINTER
003560  104401  003566                  TYPE    ,65$            ;;TYPE ASCIZ STRING
003564  000417                          BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <CRLF>/UNEXPECTED BUS TIMEOUT, PC-/
003624                          64$:
003624  010046                          MOV     R0,-(SP)        ;SETUP FOR TYPING OUT PC
003626  104402                          TYPOC
003630  000240                          NOP                     ;PUT 'HALT(0)' INSTRUCTION HERE IF YOU WISH
                                                                ;TO STOP ON UNEXPECTED TIMEOUT.
003632  000404                          BR      START1          ;BRANCH TO START1
                                        .SBTTL  START OF PROGRAM
003634  012737  177777  001336  START:  MOV     #-1,CHGADR      ;SET RH/RM ADDRESS CHANGE FLAG
003642  000407                          BR      START2          ;START THE PROGRAM
003644  012737  000400  001336  START1: MOV     #400,CHGADR     ;CLEAR THE RH/RM ADDRESS CHANGE FLAG
                                ;;**************************************************************
003652  000240                  TST1:   NOP
003654  012737  000001  001212          MOV     #1,$TESTN       ;;SET TEST NUMBER IN APT MAIL BOX
003662  005227  000000          START2: INC     #0              ;TTY LOOP, WAIT FOR INCREMENT
003666  001375                          BNE     .-4             ;OF WORD
003670  000005                          RESET                   ;CLEAR THE WORLD
                                        .SBTTL  INITIALIZE THE COMMON TAGS
                                ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
003672  012706  001114                  MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
003676  005026                          CLR     (R6)+           ;;CLEAR MEMORY LOCATION
003700  022706  001154                  CMP     #SWR,R6         ;;DONE?
003704  001374                          BNE     .-6             ;;LOOP BACK IF NO
003706  012706  001100                  MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                ;;INITIALIZE A FEW VECTORS
003712  012737  034540  000030          MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
003720  012737  000340  000032          MOV     #340,@#EMTVEC+2 ;;LEVEL 7
003726  012737  037730  000034          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
003734  012737  000340  000036          MOV     #340,@#TRAPVEC+2;LEVEL 7
003742  012737  037436  000024          MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
003750  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;LEVEL 7
003756  012737  176543  037022          MOV     #176543,$HINUM  ;;PRIME THE RANDOM NUMBER GENERATOR
003764  012737  123456  037024          MOV     #123456,$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.
003772  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
003776  012737  004032  000004          MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
004004  012737  177570  001154          MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
004012  012737  177570  001156          MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
004020  022777  177777  175126          CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
004026  001012                          BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                ;;AND  THE HARDWARE SWR IS NOT = -1
004030  000403                          BR      65$             ;;BRANCH IF NO TIMEOUT
004032  012716  004040          64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
004036  000002                          RTI
```

```
004040  012737  000176  001154  65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
004046  012737  000174  001156          MOV     #DISPREG,DISPLAY
004054  012637  000004          66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR

004060  005037  001214                  CLR     $PASS           ;;CLEAR PASS COUNT
004064  132737  000200  001227          BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
004072  001403                          BEQ     67$             ;;YES,USE NON-APT SWITCH
004074  012737  001230  001154          MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
004102                          67$:
                ;SETUP ''TIMEOUT'' TRAP VECTOR FOR UNEXPECTED BUS TIMEOUTS
004102  012737  003552  000004          MOV     #BADTMO,ERRVEC  ;SETUP FOR UNEXPECTED TIMEOUT
004110  012737  000300  000006          MOV     #PR6,ERRVEC+2   ;LEVEL 6

                .SBTTL  TYPE PROGRAM NAME
                ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
004116  005227  177777                  INC     #-1             ;;FIRST TIME?
004122  001031                          BNE     68$             ;;BRANCH IF NO
004124  104401  004132                  TYPE    ,69$            ;;TYPE ASCIZ STRING
004130  000426                          BR      68$             ;;GET OVER THE ASCIZ
                        ;;69$:  .ASCIZ  <CRLF>@CZRMUB0 - RM05/3/2 PERFORMANCE EXERCISER@<CRLF>
004206                          68$:
                .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
004206  005737  000042                  TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
004212  001012                          BNE     70$             ;;BRANCH IF YES
004214  123727  001226  000001          CMPB    $ENV,#1         ;;ARE WE RUNNING UNDER APT?
004222  001406                          BEQ     70$             ;;BRANCH IF YES
004224  023727  001154  000176          CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
004232  001005                          BNE     71$             ;;BRANCH IF NO
004234  104406                          GTSWR                   ;;GET SOFT-SWR SETTINGS
004236  000403                          BR      71$
004240  112737  000001  001150  70$:    MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
004246                          71$:

                ;THE FOLLOWING FINDS OUT THE PROGRAM CONTROL MODE:
                ;PAPER TAPE (MANUAL), ACT11, XXDP CHAIN OR DUMP

004246  005037  001432                  CLR     XXDP            ;CLEAR 'XXDP' LOAD DEVICE STORAGE
004252  122737  000016  000041          CMPB    #16,@#41        ;LOADED FROM AN RM05/3/2 ?
004260  001121                          BNE     3$              ;BR IF NOT
004262  013737  000040  001432          MOV     @#40,XXDP       ;GET DEVICE INDICATOR AND NUMBER
004270  122737  000007  001432          CMPB    #7,XXDP         ;IS IT A VALID NUMBER ?
004276  103002                          BHIS    1$              ;YES
004300  105037  001432                  CLRB    XXDP            ;NO, DEFAULT TO DRIVE 0
004304  005737  000042          1$:     TST     @#42            ;CHAIN MODE OR ACT11 AUTO ACCEPT ?
004310  001425                          BEQ     2$              ;BR IF NEITHER
004312  104401  004320                  TYPE    ,73$            ;;TYPE ASCIZ STRING
004316  000412                          BR      72$             ;;GET OVER THE ASCIZ
                        ;;73$:  .ASCIZ  <CRLF>/NOT TESTING DRIVE /
004344                          72$:
004344  005046                          CLR     -(SP)           ;CLEAR WORD ON STACK
004346  113716  001432                  MOVB    XXDP,(SP)       ;GET DRIVE ADDRESS
004352  104403                          TYPOS                   ;TYPE THE ADDRESS
004354     001                          .BYTE   1               ;ONLY 1 CHARACTER
004355     000                          .BYTE   0               ;SUPRESS LEADING ZEROS
004356  104401  001203                  TYPE    ,$CRLF          ;CR-LF
004362  000460                          BR      3$              ;GET NUMBER OF DRIVES
```

```
                                2$:     INC     #-1             ;FIRST TIME THRU HERE ?
                                        BNE     3$              ;NO
                                        TYPE    ,75$            ;;TYPE ASCIZ STRING
                                        BR      74$             ;;GET OVER THE ASCIZ
                                ;;75$:  .ASCIZ  <CRLF>/TO TEST DRIVE /
                                74$:
                                        CLR     -(SP)           ;CLEAR WORD ON STACK
                                        MOVB    XXDP,(SP)       ;GET DRIVE ADDRESS
                                        TYPOS                   ;TYPE DRIVE ADDRESS
                                        .BYTE   1               ;ONLY 1 CHARACTER
                                        .BYTE   0               ;SUPRESS LEADING ZEROS
                                        TYPE    ,76$            ;;TYPE ASCIZ STRING
                                        BR      3$              ;;GET OVER THE ASCIZ
                                ;;76$:  .ASCIZ  /, HALT PROGRAM, CLEAR LOC. 40 AND RESTART PROGRAM./<CRLF>
                                3$:
                                        JSR     PC,$TKINT       ;TURN ON THE KEYBOARD INTERRUPT
                                        TSTB    $ENV            ;RUN UNDER APT MODE
                                        BEQ     5$              ;NO,DO NOT BOTHER
                                        TSTB    $VECT1          ;NEW VECTOR ?
                                        BEQ     4$              ;NOT LOAD IF = 0
                                        MOVB    $VECT1,$RMVEC   ;NEW VECTOR
                                4$:     TST     $BASE           ;NEW BASE ADDRESS ?
                                        BEQ     6$              ;NO
                                        MOV     $BASE,$RMADR    ;NEW BASE ADDRESS
                                        BR      6$

                                5$:     TSTB    $AUTOB          ;RUNNING IN AUTO MODE ?
                                        BNE     6$              ;YES
                                        JSR     PC,BUSADR       ;CHECK RH/RM BUS ADDRESS

                                6$:     MOV     $RMADR,RMADR    ;LOAD ADDRESS INTO DRIVER
                                        MOV     $RMVEC,RMVEC    ;LOAD VECTOR INTO DRIVER
                                        CLR     STATIN          ;CLEAR PERFORMANCE SUMMARY TYPEOUT FLAG
                                        MOV     #ORDERQ,R5      ;START OF AREA TO CLEAR
                                7$:     CLR     (R5)+
                                        CMP     #BLKADR,R5      ;LOOK FOR END OF CLEAR AREA
                                        BNE     7$              ;BR IF NOT FINISHED
                                        MOV     #STACK,SP       ;SETUP THE STACK POINTER
                                        CLR     PS              ;CLEAR THE PROCESSOR STATUS WORD
                                        MOV     HZ,ONESEC       ;RESTORE ONE SECOND COUNTER VALUE
                                        CLR     HOUR            ;CLEAR THE HOUR'S COUNTER
                                        CLR     MINUTE          ;CLEAR THE MINUTE'S COUNTER
                                        CLR     SECOND          ;CLEAR THE SECOND'S COUNTER
                                        CLR     INTRVL+2        ;CLEAR INTERVAL COUNTER
                                        CLR     PACK            ;SET 'T' & CLEAR 'R' OR 'W' COMMAND FLAG
                                        CLR     CFLAG           ;CLEAR THE 'CONTROL C' FLAG
                                        CLR     DRIVE0+$FIRST   ;RESET $FIRST FLAG FOR DRIVE 0
                                        CLR     DRIVE1+$FIRST   ;RESET $FIRST FLAG FOR DRIVE 1
                                        CLR     DRIVE2+$FIRST   ;RESET $FIRST FLAG FOR DRIVE 2
                                        CLR     DRIVE3+$FIRST   ;RESET $FIRST FLAG FOR DRIVE 3
                                        CLR     DRIVE4+$FIRST   ;RESET $FIRST FLAG FOR DRIVE 4
                                        CLR     DRIVE5+$FIRST   ;RESET $FIRST FLAG FOR DRIVE 5
                                        CLR     DRIVE6+$FIRST   ;RESET $FIRST FLAG FOR DRIVE 6
                                        CLR     DRIVE7+$FIRST   ;RESET $FIRST FLAG FOR DRIVE 7

                                ;AUTO SIZE FOR RH70 CONTROLLER AND DETERMINE IF IT IS
                                ;JUMPERED FOR 22 OR 32 REGISTERS
```

```
                                                ;CLEAR RMBAE OFFSET
004744  005037  040162          SIZE70: CLR     RHEXT
004750  042737  174000  001234          BIC     #174000,$CPUOP  ;CLEAR CPU TYPE REGISTER
004756  013746  000004                  MOV     ERRVEC,-(SP)    ;SAVE CONTENTS OF ERROR VECTOR
004762  012737  005034  000004          MOV     #2$,ERRVEC      ;SETUP 'TRAP' RETURN ADDRESS
004770  013700  001272                  MOV     $RMADR,R0       ;GET RMCS1 ADDRESS
004774  062700  000050                  ADD     #50,R0          ;GET REGISTER OFFSET FOR RH70
005000  012701  000012                  MOV     #10.,R1         ;GET NUMBER OF REGISTERS TO CHECK
005004  005720                  1$:     TST     (R0)+           ;TRAP IF NOT A VALID RMBAE
005006  005720                          TST     (R0)+           ;TRAP IF NOT A VALID RMCS3
005010  012737  000050  040162          MOV     #50,RHEXT       ;LOAD OFFSET FOR RMBAE (22 REGISTER RH)
005016  005720                          TST     (R0)+           ;TRAP IF NOT A VALID REGISTER
005020  005301                          DEC     R1              ;DONE WITH ALL 32 REGISTERS ?
005022  001375                          BNE     1$              ;BR IF NO
005024  012737  000074  040162          MOV     #74,RHEXT       ;LOAD OFFSET FOR RMBAE (32 REGISTER RH)
005032  000403                          BR      3$
005034  012716  005042          2$:     MOV     #3$,(SP)        ;SETUP RETURN ADDRESS
005040  000002                          RTI

005042  013700  001272          3$:     MOV     $RMADR,R0       ;GET RMCS1 REGISTER
005046  013701  040162                  MOV     RHEXT,R1        ;GET RMBAE REGISTER OFFSET
005052  001415                          BEQ     4$              ;BR IF NONE
005054  060001                          ADD     R0,R1           ;GET RMBAE REGISTER
005056  052710  001400                  BIS     #A17!A16,(R0)   ;SET EXTENDED ADDRESS BITS IN RMCS1
005062  022711  000003                  CMP     #3,(R1)         ;ARE THE EXTENDED BITS SET IN RMBAE ?
005066  001007                          BNE     4$              ;BR IF NO
005070  005011                          CLR     (R1)            ;CLEAR EXTENDED ADDRESS BITS IN RMBAE
005072  033710  001400                  BIT     A17!A16,(R0)    ;ARE THE EXTEND BITS CLEAR IN RMCS1 ?
005076  001003                          BNE     4$              ;BR IF NO
005100  052737  030000  001234          BIS     #BIT13!BIT12,$CPUOP     ;SET THE 11/70 CPU TYPE CODE
005106  012637  000004          4$:     MOV     (SP)+,ERRVEC    ;RESTORE CONTENTS OF ERROR VECTOR

                        ;ROUTINE TO DETERMINE BUFFER MAX WORD COUNT AND FUDGE PACK SERIAL NUMBER
                        ;TO ALLOW BAD SECTOR FILE(DEC144) TO BE READ FROM EACH DRIVE AT
                        ;LEAST ONE TIME.

005112  005227  177777  SIZMEM: INC     #-1             ;FIRST TIME THRU HERE ?
005116  001027                  BNE     1$              ;BR IF NO
005120  012700  177777          MOV     #-1,R0          ;FUDGE MSB'S FOR INITIALIZING PACK S/N
005124  010037  046344          MOV     R0,DRIVE0+$PSNM ;INIT. S/N FOR DRIVE 0
005130  010037  050556          MOV     R0,DRIVE1+$PSNM ;INIT. S/N FOR DRIVE 1
005134  010037  052770          MOV     R0,DRIVE2+$PSNM ;INIT. S/N FOR DRIVE 2
005140  010037  055202          MOV     R0,DRIVE3+$PSNM ;INIT. S/N FOR DRIVE 3
005144  010037  057414          MOV     R0,DRIVE4+$PSNM ;INIT. S/N FOR DRIVE 4
005150  010037  061626          MOV     R0,DRIVE5+$PSNM ;INIT. S/N FOR DRIVE 5
005154  010037  064040          MOV     R0,DRIVE6+$PSNM ;INIT. S/N FOR DRIVE 6
005160  010037  066252          MOV     R0,DRIVE7+$PSNM ;INIT. S/N FOR DRIVE 7
005164  004737  100434          JSR     PC,$SIZE        ;SEE HOW MUCH MEMORY ON SYSTEM
005170  013737  100564  001334  1$:     MOV     $LSTAD,LSTAD    ;SAVE THE LAST ADDRESS
005176  012737  000001  001642          MOV     #1,BUFTBL       ;LOAD NUMBER OF BUFFERS
005204  012737  102072  001644          MOV     #ENDPGM,BUFTBL+2        ;STARTING ADDRESS OF BUFFER
005212  013737  001334  001646          MOV     LSTAD,BUFTBL+4  ;LAST ADDR TO BUFFER ALLOCATION TABLE
005220  023727  001334  160000          CMP     LSTAD,#160000   ;OVER 28K ?
005226  101403                          BLOS    2$              ;NO
005230  012737  160000  001646          MOV     #160000,BUFTBL+4        ;XXDP MAX MEMORY 28K
005236  162737  102072  001646  2$:     SUB     #ENDPGM,BUFTBL+4        ;SUBTRACT PROGRAM SPACE
005244  000241                          CLC                     ;CLEAR THE 'C' BIT
```

```
                                ROR     BUFTBL+4            ;CONVERT TO WORD COUNT
                                SUB     #100.,BUFTBL+4      ;SAVE ROOM FOR THE 'ABS' LOADER
                                TSTB    $AUTOB              ;RUNNING IN AUTO MODE ?
                                BEQ     3$                  ;BR IF NO
                                SUB     #1536.,BUFTBL+4     ;SUBTRACT 'XXDP' LOADER SIZE
                        3$:     CMP     WRDCNT,BUFTBL+4     ;IS MAX WORD COUNT TOO LARGE ?
                                BLE     4$                  ;BR IF NO
                                MOV     BUFTBL+4,WRDCNT     ;USE MAX AVAIL MEMORY AS MAX WRD CNT
                                MOV     WRDCNT,PARLST+2     ;VALUE FOR THE PARAMETER TABLE
                        4$:

                        ;SEE IF THE OPERATOR WANTS TO CHANGE ANY PARAMETERS

                        LKPAR:  TST     PWRFLG              ;RETURNING FROM POWER FAIL ?
                                BNE     SETVEC              ;BRANCH IF YES

                        1$:     CLR     CFLAG               ;CLEAR CONTROL C FLAG
                                TSTB    $AUTOB              ;RUNNING IN AUTO MODE ?
                                BNE     SETVEC              ;BR IF YES
                                TYPE    ,ASKPAR             ;TYPE 'CHANGE PARAMETERS ?'
                                RDLIN                       ;READ THE ENTRY
                                MOV     (SP)+,R0            ;SAVE ADDRESS OF RESPONSE
                                TST     CFLAG               ;CONTROL C TYPED ?
                                BNE     1$                  ;BR IF YES
                                TSTB    (R0)                ;WAS RESPONSE A CARRIAGE RETURN (DEFAULT 'N')?
                                BEQ     SETVEC              ;BR IF YES
                                TSTB    1(R0)               ;WAS IT TERMINATED WITH CARRIAGE RETURN ?
                                BNE     2$                  ;BR IF NO
                                CMPB    #'Y,(R0)            ;WAS IT A 'Y' RESPONSE ?
                                BEQ     ENTPR               ;BR IF YES
                                CMPB    #'N,(R0)            ;WAS IT A 'N' RESPONSE ?
                                BEQ     SETVEC              ;BR IF YES
                        2$:     TYPE    ,BADENT             ;TYPE BAD ENTRY MESSAGE
                                BR      1$                  ;TRY AGAIN

                        ENTPR:  MOV     #PARLST,R3          ;PARAMETER TABLE ADDRESS
                                JSR     PC,PARENT           ;GET THE PARAMETER ENTRY
                                CMP     WRDCNT,#6           ;IS THE 'WRDCNT' VALUE OK ?
                                BHIS    SETVEC              ;BR IF IT IS
                                MOV     #6,WRDCNT           ;SET 'WRDCNT' TO THE MINIMUM VALUE

                        ;DISPLAY DRIVE STATUS AND SET UP THE OTHER SYSTEM DEVICES THAT
                        ;THE PROGRAM WILL USE. PROGRAM RETURN HERE ON POWER FAIL

                        SETVEC: JSR     PC,CKCLK            ;START THE CLOCK
                                JSR     PC,RMINIT           ;INITIALIZE THE RM DRIVER
                                MOV     #-1,SAVEFG          ;SET THE SAVE REGISTERS FLAG
                                MOV     RMADR,R4            ;GET RM/RH BASE ADDRESS
                                MOV     #CLR,RMCS2(R4)      ;CLEAR MASSBUS CONTROLLER
                                INC     #-1                 ;FIRST TIME THRU ?
                                BEQ     1$                  ;BR IF YES
                                TST     PWRFLG              ;RETURNING FROM POWER FAIL ?
                                BNE     1$                  ;BRANCH IF YES
                                BIT     #SW02,@SWR          ;TYPEOUT THE DRIVE STATUS TABLE ?
                                BNE     12$                 ;BR IF NOT
                        1$:     CLR     PWRFLG              ;CLEAR POWER FAIL FLAG
                                CLR     R4                  ;DRIVE TABLE POINTER
```

```
                                        TYPE    ,SYSTAT         ;TYPE STATUS HEADING
                                        TYPE    ,$CRLF          ;CR-LF
                                        MOV     R4,-(SP)        ;;SAVE R4 FOR TYPEOUT
                                                                ;;TYPE DRIVE NUMBER
                                        TYPOS                   ;;GO TYPE--OCTAL ASCII
                                        .BYTE   2               ;;TYPE 2 DIGIT(S)
                                        .BYTE   0               ;;SUPPRESS LEADING ZEROS
                                        TYPE    ,BLNKS4         ;TYPE 4 BLANKS
                                        TSTB    DRVSTA(R4)      ;CHECK DRIVE'S STATUS
                                        BMI     5$              ;BR IF UNSAFE
                                        BNE     6$              ;BR IF ONLINE

                                        TSTB    DRVTYP(R4)      ;SEE IF OFFLINE OR NONEXISTENT
                                        BEQ     3$              ;BR IF NONEXISTENT
                                        BPL     4$              ;BR IF OFFLINE
                                        TYPE    ,NOTRM          ;DRIVE NOT AN RM05/3/2
                                        BR      11$             ;CHECK NEXT DRIVE

                                3$:     TYPE    ,NOTPRS         ;DRIVE NOT PRESENT
                                        BR      11$             ;CHECK NEXT DRIVE

                                4$:     TYPE    ,UNTOFF         ;DRIVE OFFLINE
                                        BR      8$              ;PRINT DRIVE TYPE

                                5$:     TYPE    ,NOTSAF         ;DRIVE UNSAFE
                                        BR      8$              ;PRINT DRIVE TYPE

                                6$:     TST     XXDP            ;LOADED FROM THIS DEVICE ?
                                        BEQ     7$              ;BR IF NO
                                        CMPB    XXDP,R4         ;LOADED FROM THIS DRIVE ?
                                        BNE     7$              ;BR IF NO
                                        TYPE    ,LODEV          ;TYPE 'LOAD DEVICE'
                                        BR      11$

                                7$:     TYPE    ,UNTON          ;DRIVE ONLINE
                                8$:     TYPE    ,BLNKS2         ;TYPE 2 BLANKS
                                        MOV     #$RM03,10$      ;ADDRESS OF RM03 MESSAGE
                                        CMPB    #4,DRVTYP(R4)   ;IS DEVICE AN RM03 ?
                                        BEQ     9$              ;BR IF YES
                                        MOV     #$RM02,10$      ;ADDRESS OF RM02 MESSAGE
                                        CMPB    #5,DRVTYP(R4)   ;IS DEVICE AN RM02 ?
                                        BEQ     9$              ;BR IF YES
                                        MOV     #$RM05,10$      ;ADDRESS OF RM05 MESSAGE

                                9$:     TYPE                    ;TYPE THE DRIVE TYPE MESSAGE
                                10$:    .WORD   0               ;MESSAGE ADDRESS HERE

                                11$:    INC     R4              ;INCREMENT DRIVE NUMBER/TABLE POINTER
                                        CMP     R4,#8.          ;FINISHED ?
                                        BNE     2$              ;BR IF NOT
                                        TYPE    ,$CRLF          ;CR-LF
                                12$:
```

```
                                        ;INITIALIZE PROGRAM PARAMETERS FOR STARTUP
                                        
005730  004737  033226          STA:    JSR     PC,$TKINT       ;INITIALIZE THE KEYBOARD INTERRUPT HANDLER
005734  012737  002740  001434          MOV     #002740,ENDCON  ;INITIALIZE XFER COUNT(LSB)
005742  012737  005455  001436          MOV     #005455,ENDCON+2
005750  105737  001226                  TSTB    $ENV            ;APT SCRIPT MODE, THEN MAKE IT RUN 2 MIN.
005754  001411                          BEQ     1$              ;NO
005756  012737  000001  00146-          MOV     #1,RATIO        ;SPEED UP TEST
005764  012737  077777  001434          MOV     #77777,ENDCON   ;INITIALIZE QUICK XFER COUNT(LSB)
005772  012737  000027  001436          MOV     #27,ENDCON+2
                                        
006000  105737  001150          1$:     TSTB    $AUTOB          ;RUNNING IN AUTO MODE ?
006004  001003                          BNE     2$              ;BR IF YES
006006  005737  001336                  TST     CHGADR          ;START AT 200 ?
006012  00345C                          BLE     8$              ;NO
                                        
006014  005001                  2$:     CLR     R1              ;DRIVE #
006016  005002                          CLR     R2              ;AVAIL TABLE INDEX
006020  005003                          CLR     R3              ;DRIVE# * 2
006022  005737  001432          3$:     TST     XXDP            ;LOADED FROM THIS DEVICE ?
006026  001403                          BEQ     4$              ;BR IF NO
006030  123701  001432                  CMPB    XXDP,R1         ;LOADED FROM THIS DRIVE ?
006034  001426                          BEQ     7$              ;BR IF YES
006036  105761  040036          4$:     TSTB    DRVSTA(R1)      ;DRIVE ON LINE ?
006042  003423                          BLE     7$              ;NO
006044  016300  002044                  MOV     BLKADR(R3),R0   ;LOAD DPB ADDRESS
006050  004737  026602                  JSR     PC,CLRDPB       ;CLEAR DPB BLOCK
006054  004737  027420                  JSR     PC,GETID        ;GET DRIVE (MBA) SERIAL NUMBER
006060  004537  027520                  JSR     R5,GETADR       ;RETRIEVE BAD SECTOR FILE
006064  010062  001554                  MOV     R0,NEWUNT(R2)   ;LOAD DPB ADDRESS TO ABAIL QUEUE
006070  004737  027020                  JSR     PC,DRVPRM       ;SETUP DRIVE PARAMETER LIMITS
006074  005060  000122                  CLR     $FIRST(R0)      ;RESET $FIRST FLAG FOR FIRST 204 START
006100  112760  177776  000026          MOVB    #-2,$PACK(R0)   ;SETUP COMMAND 'WT' (WRITE DATA AND TEST)
006106  004737  017140                  JSR     PC,WRTPK        ;SETUP INITIAL PARAMETERS
                                        
006112  022322                  7$:     CMP     (R3)+,(R2)+     ;INCREMENT INDEX
006114  005201                          INC     R1              ;NEXT DRIVE
006116  020127  000007                  CMP     R1,#7           ;ALL DRIVES ASSIGNNED ?
006122  003737                          BLE     3$              ;NO
006124  005037  001336                  CLR     CHGADR          ;CLEAR START FLAG
006130  000137  006306                  JMP     MAIN1           ;JUMP TO WAIT PARAMETER AND BUFFER LOOP
                                        
006134  012737  000001  001340  8$:     MOV     #1,CFLAG        ;DUMMY 'CONTROL C' FLAG
```

```
                                .SBTTL  MAIN PROGRAM
                        ;
006142  005737  001340  MAIN:   TST     CFLAG           ;KEYBOARD INTERRUPTED ?
006146  001407                  BEQ     3$              ;BR IF NOT
006150  005737  001506  1$:     TST     ORDERQ          ;ANY DRIVES IN ORDER QUE ?
006154  001402                  BEQ     2$              ;BR IF NO, ELSE
006156  000137  007006          JMP     IDLE            ;LET ALL DRIVES FINISH ORDER
006162  004737  025240  2$:     JSR     PC,KSR          ;SERVICE THE KEYBCARD
006166  000240          3$:     NOP                     ; .! FOR DEBUGGING !!
                        ;
                        ;CHECK FOR DRIVES TO BE DROPPED
                        ;
006170  012703  000010  MAINDA: MOV     #8.,R3          ;DRIVE COUNTER
006174  012705  001532          MOV     #DUNIT,R5       ;ADDRESS OF 'DROP DRIVE' TABLE
006200  005715          1$:     TST     (R5)            ;SEE IF ENTRY AT PRESENT POSITION
006202  001004                  BNE     3$              ;BR IF THERE IS ONE
006204  005725          2$:     TST     (R5)+           ;INCREMENT TO NEXT TABLE POSITION
006206  005303                  DEC     R3              ;DECREMENT DRIVE COUNTER
006210  001373                  BNE     1$              ;BR IF MORE TO CHECK
006212  000435                  BR      MAIN1           ;GO CHECK FOR NEW ASSIGNED DRIVES
                        ;
006214  012701  001576  3$:     MOV     #AVAIL,R1       ;ADDRESS OF 'AVAILABLE DRIVES' TABLE
006220  005711          4$:     TST     (R1)            ;IF AT END OF 'AVAIL' TABLE ?
006222  001404                  BEQ     5$              ;BR IF YES
006224  021115                  CMP     (R1),(R5)       ;IS DRIVE IN 'AVAIL' THE TABLE ?
006226  001412                  BEQ     7$              ;BR IF YES
006230  005721                  TST     (R1)+           ;NO, INCREMENT 'AVAIL' TABLE ADDRESS
006232  000772                  BR      4$              ;AND CONTINUE LOOKING
                        ;
006234  012701  001620  5$:     MOV     #WAIT,R1        ;ADDRESS OF THE 'WAIT' BUFFER TABLE
006240  005711          6$:     TST     (R1)            ;AT THE END OF 'WAIT' TABLE ?
006242  001760                  BEQ     2$              ;BR IF YES
006244  021115                  CMP     (R1),(R5)       ;IS DRIVE IN THE 'WAIT' TABLE ?
006246  001402                  BEQ     7$              ;BR IF YES
006250  005721                  TST     (R1)+           ;NO, INCREMENT 'WAIT' TABLE ADDRESS
006252  000772                  BR      6$              ;AND CONTINUE LOOKING
                        ;
006254  011100          7$:     MOV     (R1),R0         ;PUT THE DRIVE'S BLOCK ADDRESS IN R0
006256  104401  001203          TYPE    ,$CRLF          ;CR-LF
006262  104401  076341          TYPE    ,DEASSG         ;TYPE 'DRIVE DEASSIGNED'
006266  004737  023636          JSR     PC,SUMARY       ;TYPE THE DRIVE'S PERFORMANCE SUMMARY
006272  104401  076155          TYPE    ,STAR5          ;TYPE '****...ETC'
006276  005015                  CLR     (R5)            ;CLEAR THE 'DROP DRIVE' TABLE ENTRY
006300  004737  020700          JSR     PC,CMPRES       ;COMPRESS THE RESPECTIVE TABLE
006304  000737                  BR      2$              ;SEE IF ANY MORE DRIVES
                        ;
                        ;LOOK FOR DRIVES TO BE ASSIGNED
                        ;
006306  012703  000010  MAIN1:  MOV     #8.,R3          ;DRIVE COUNT
006312  005001                  CLR     R1              ;ASSIGN LIST INDEX
006314  005002                  CLR     R2              ;'AVAIL' INDEX
006316  005005                  CLR     R5              ;NEW DRIVE INDEX
006320  005765  001554  1$:     TST     NEWUNT(R5)      ;NEW DRIVE IN THIS POSITION
006324  001005                  BNE     3$              ;BR IF THERE IS
006326  005725          2$:     TST     (R5)+           ;INCREMENT R5
006330  005201                  INC     R1              ;INCREMENT ASSIGN INDEX
006332  005303                  DEC     R3              ;DECREMENT DRIVE COUNT
```

```
                                        BNE     1$              ;BR IF MORE DRIVES
                                        BR      MAIN2           ;START OPERATIONS FOR THE AVAILABLE DRIVES

006340  104401  001203        3$:     TYPE    ,$CRLF          ;CR-LF
006344  104401  075625                TYPE    ,UNTMSG         ;'DRIVE'
006350  010146                        MOV     R1,-(SP)        ;;SAVE R1 FOR TYPEOUT
                                                                ;;TYPE DRIVE NUMBER
006352  104403                        TYPOS                   ;;GO TYPE--OCTAL ASCII
006354     002                        .BYTE   2               ;;TYPE 2 DIGIT(S)
006355     000                        .BYTE   0               ;;SUPPRESS LEADING ZEROS
006356  104401  076411                TYPE    ,ASGND          ;'STARTED'
006362  005762  001576        4$:     TST     AVAIL(R2)       ;AT END OF AVAILABLE TABLE
006366  001402                        BEQ     5$              ;BR IF YES
006370  005722                        TST     (R2)+           ;INCREMENT AVAILABLE TABLE INDEX
006372  000773                        BR      4$              ;CONTINUE LOOKING FOR END OF TABLE
006374  016562  001554  001576  5$:   MOV     NEWUNT(R5),AVAIL(R2)    ;MOVE ADDR OF DRIVE INTO AVAIL LST
006402  005065  001554                CLR     NEWUNT(R5)      ;TAKE DRIVE OUT OF NEW DRIVE TABLE
006406  156137  040142  001530        BISB    ATABIT(R1),ASNLST       ;SET DRIVE ASSIGNED INDICATOR
006414  005037  031662                CLR     AUTLST          ;CLEAR AUTO ASSIGN
006420  005722                        TST     (R2)+           ;INCREMENT AVAILABLE TABLE POINTER
006422  000741                        BR      2$              ;LOOK FOR MORE DRIVES

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

006424  005002                MAIN2:  CLR     R2              ;START FROM THE FIRST LOCATION
006426  105737  001530                TSTB    ASNLST          ;ANY DRIVES ACTIVE ?
006432  001025                        BNE     2$              ;BR IF YES
006434  105737  001150                TSTB    $AUTOB          ;RUNNING IN AUTO MODE ?
006440  001020                        BNE     1$              ;BR IF YES
006442  012737  000001  001340        MOV     #1,CFLAG        ;DUMMY 'CONTROL C' FLAG
006450  013737  001314  001352        MOV     HZ,ONESEC       ;RESTORE ONE SECOND COUNTER VALUE
006456  005037  001344                CLR     HOUR            ;CLEAR THE HOUR'S COUNTER
006462  005037  001346                CLR     MINUTE          ;CLEAR THE MINUTE'S COUNTER
006466  005037  001350                CLR     SECOND          ;CLEAR THE SECOND'S COUNTER
006472  005037  001454                CLR     INTRVL+2        ;CLEAR INTERVAL COUNTER
006476  104401  077120                TYPE    ,NODRVS         ;TYPE 'NO DRIVES ASSIGNED'
006502  000137  031612        1$:     JMP     $GET42          ;GIVE CONTROL TO MONITOR

006506  005762  001620        2$:     TST     WAIT(R2)        ;ANY DRIVES WAITING FOR THE BUFFER ?
006512  001435                        BEQ     5$              ;BR IF NO
006514  016200  001620                MOV     WAIT(R2),R0     ;LOAD R0 WITH THE DPB ADDRESS
006520  005046                        CLR     -(SP)           ;CLEAR THE STACK FOR BUFFER REQ
006522  004737  016402                JSR     PC,GETBUF       ;CALL TO GET THE BUFFER RT.
006526  012660  000006                MOV     (SP)+,$BUF(R0)  ;IF 0,BUFFER IS STILL NOT AVAILABLE
006532  001423                        BEQ     4$              ;BRANCH IF NO BUFFER AVAILABLE
006534  005060  000120                CLR     $NEXT(R0)       ;CLEAR PARAMETER SELECT FLAG
006540  005060  000106                CLR     $FAIR(R0)       ;CLEAR THE FAIR FLAG
006544  004737  016772                JSR     PC,FILBUF       ;FILL THE BUFFER
006550  004737  017050                JSR     PC,GODRIV       ;SET COMMAND AND GO
006554  012705  001506                MOV     #ORDERQ,R5      ;PUT THE WAIT QUE INTO ORDER QUE
006560  005725                3$:     TST     (R5)+           ;QUE AVAILABLE ?
006562  001376                        BNE     3$              ;BR IF NO
006564  010045                        MOV     R0,-(R5)        ;LOAD THE DPB ADDRESS INTO THE ORDER QUE
006566  012701  001620                MOV     #WAIT,R1        ;REMOVE THE DRIVE FROM THE 'WAIT' QUE
006572  060201                        ADD     R2,R1           ;OFFSET THE QUE POSITION
006574  004737  020700                JSR     PC,CMPRES       ;COMPRESS THE QUE
```

```
                                                BR      2$              ;BRANCH IF DONE
                                        4$:     TST     (R2)+           ;CHECK THE NEXT QUE
                                                BR      2$              ;LOOPING BACK

                                        5$:     TST     ORDERQ          ;ANY OUTSTANDING ORDERS ?
                                                BNE     IDLE            ;BR IF YES

                                                CLR     R2              ;CLEAR DRIVE TABLE POINTER
                                        6$:     TST     AVAIL(R2)       ;ANY DRIVES WAITING FOR PARAMETERS
                                                BNE     7$              ;BRANCH IF ANY
                                                JMP     IDLE            ;BRANCH IF NONE
                                        7$:     MOV     AVAIL(R2),R0    ;CONTROL BLOCK ADDR IN R0
                                                TST     $NEXT(R0)       ;PARAMETERS BEEN SELECTED ?
                                                BNE     9$              ;BR IF THEY HAVE
                                                TSTB    $PACK(R0)       ;'R' OR 'W' COMMAND FOR THIS DRIVE ?
                                                BEQ     8$              ;BR IF NO
                                                JSR     PC,WRTPK        ;GET DATA PARAMETERS
                                                BR      10$             ;GET THE BUFFER

                                        8$:     JSR     PC,GENPAR       ;GO GENERATE THE PARAMETERS
                                        9$:     JSR     PC,LODPAR       ;LOAD THE PARAMETERS JUST GENERATED
                                        10$:    CLR     -(SP)           ;MAKE ROOM ON THE STACK FOR THE BUFFER ADDR
                                                JSR     PC,GETBUF       ;GET BUFFER
                                                MOV     (SP)+,$BUF(R0)  ;MOVE BUFFER ADDR TO DPB
                                                BEQ     12$             ;BR IF '0' ADDR (NO BUFFER)
                                                JSR     PC,FILBUF       ;FILL THE BUFFER
                                                CLR     $NEXT(R0)       ;CLEAR PARAMETER SELECT FLAG
                                                CLR     $FAIR(R0)       ;CLEAR THE 'FAIRNESS' COUNT
                                                JSR     PC,GODRIV       ;PUT CURRENT DPB IN DRIVER
                                                MOV     #ORDERQ,R5      ;ADDRESS OF ORDER QUE IN R5
                                        11$:    TST     (R5)+           ;END OF QUE ?
                                                BNE     11$             ;BR IF NOT
                                                MOV     R0,-(R5)        ;PUT BLOCK ADDRESS INTO QUE
                                                BR      15$             ;CONTINUE LOOKING

                                        12$:    INC     $FAIR(R0)       ;INCREMENT THE FAIR COUNT
                                                CMP     #3,$FAIR(R0)    ;THREE TIMES,BUFFER IS NOT AVAILABLE?
                                                BHI     14$             ;BRANCH IF NOT OVER THREE TIMES
                                                MOV     #WAIT,R5        ;LOAD INTO THE WAIT QUE
                                        13$:    TST     (R5)+           ;AN AVAILABLE LOCATION ?
                                                BNE     13$             ;BRANCH IF NOT
                                                MOV     R0,-(R5)        ;LOAD INTO WAIT QUE
                                                BR      15$             ;REMOVE THE DPB FROM AVAILABLE QUE

                                        14$:    TST     (R2)+           ;INCREMENT INDEX
                                                BR      6$              ;BRANCH BACK TO FIRE NEXT DRIVE
                                        15$:    MOV     #AVAIL,R1       ;'AVAILABLE' TABLE ADDRESS
                                                ADD     R2,R1           ;FORM ADDRESS OF LAST ENTRY
                                                JSR     PC,CMPRES       ;COMPRESS THE TABLE
                                                BR      6$              ;CONTINUE LOOKING

                                        ;WAIT FOR A COMMAND TO FINISH

                                        IDLE:   MOV     #ORDERQ,R1      ;ADDRESS OF THE ORDER QUE IN R1
                                        1$:     MOV     (R1)+,R0        ;PUT BLOCK ADDRESS INTO R0
                                                BEQ     4$              ;BR IF END OF QUE
                                        2$:     TST     $STATUS(R0)     ;SEE IF DRIVE FINISHED
```

```
168 007022  001775                        BEQ     2$              ;BR IF DRIVE NOT FINISHED
169 007024  005741                        TST     -(R1)           ;BACKUP THE QUE POINTER
170 007026  010146                        MOV     R1,-(SP)        ;SAVE THE QUE ADDRESS
171 007030  004737  016232                JSR     PC,STATIS       ;ACCUMULATE STATISTICS FOR DRIVE IN R0
172 007034  004737  007150                JSR     PC,PROCES       ;PROCESS END OF COMMAND
173 007040  005037  001342                CLR     BADSEC          ;CLEAR THE BAD TRK/SEC ERROR INDICATOR
174 007044  004737  031134                JSR     PC,ABNRML       ;SEE IF ANY DRIVES HAVE TOO MANY ERRORS
175 007050  004737  031162                JSR     PC,$EOP         ;IS IT END OF PASS ?
176 007054  012601                        MOV     (SP)+,R1        ;RESTORE THE ORDER TABLE INDEX
177 007056  012705  001576                MOV     #AVAIL,R5       ;ADDRESS OF THE 'AVAIL' TABLE
178 007062  005725              3$:       TST     (R5)+           ;IS IT THE END OF THE 'AVAIL' TABLE ?
179 007064  001376                        BNE     3$              ;BR IF NO
180 007066  011145                        MOV     (R1),-(R5)      ;MOVE THE DPB INTO THE 'AVAIL' TABLE
181 007070  004737  020700                JSR     PC,CMPRES       ;COMPRESS THE ORDER QUE
182 007074  004737  016536                JSR     PC,RELBUF       ;RESTORE BUFFER
183
184 007100  032777  000004  172046  4$:   BIT     #SW02,@SWR      ;TYPE PERFORMANCE SUMMARY
185 007106  001016                        BNE     5$              ;BR IF NOT
186 007110  005737  001316                TST     STATIN          ;TIME TO TYPE THE PERFORMANCE SUMMARY ?
187 007114  001413                        BEQ     5$              ;BR IF NOT
188 007116  005037  001316                CLR     STATIN          ;CLEAR THE INDICATOR
189 007122  005737  001530                TST     ASNLST          ;ANY DRIVES ASSIGNED ?
190 007126  001406                        BEQ     5$              ;BR IF NO
191 007130  104401  076063                TYPE    ,REPHD          ;TYPE PERFORMANCE REPORT HEADING
192 007134  004737  023550                JSR     PC,STATPR       ;TYPE THE SUMMARY
193 007140  104401  076124                TYPE    ,STAR30         ;TYPE "****...ETC"
194 007144  000137  006142          5$:   JMP     MAIN            ;CONTINUE THE LOOP
```

```
                                        ;PROCESS THE COMMAND TERMINATION
                                        
007150  111037  001324          PROCES: MOVB    (R0),UNIT       ;DRIVE NUMBER FOR ANY ERROR MESSAGES
007154  005760  000016                  TST     $TATUS(R0)      ;SEE IF DRIVER SIGNALED AN ERROR
007160  100427                          BMI     ERPROC          ;BR IF ERROR
007162  032760  100000  002136          BIT     #BIT15,$RMCS1(R0)       ;SEE IF 'SC' SET
007170  001410                          BEQ     1$              ;BR IF NOT SET
007172  032760  040000  002136          BIT     #BIT14,$RMCS1(R0)       ;SEE IF 'TRE' SFT
007200  001017                          BNE     ERPROC          ;BR IF SET
007202  032760  040000  002150          BIT     #BIT14,$RMDS(R0)        ;SEE IF 'ERR' SET
007210  001013                          BNE     ERPROC          ;BR IF SET
                                        
                                        ;NO ERRORS DETECTED IN REGISTERS, DO SOME CHECKING ANYWAY
                                        
007212  004737  013176          1$:     JSR     PC,CKERR        ;CHECK ERROR BITS
007216  004737  013270                  JSR     PC,CKBUS        ;CHECK BUS ADDRESS & WORD COUNT
007222  032777  000002  171724          BIT     #SW01,@SWR      ;DO DATA COMPARE ?
007230  001002                          BNE     2$              ;BR IF NO
007232  004737  013354                  JSR     PC,CMPAR        ;COMPARE DATA W/O ERROR
007236  000207                  2$:     RTS     PC              ;RETURN
                                        
                                        ;COMMAND TERMINATED WITH AN ERROR - PROCESS THE ERROR
                                        
007240  032760  000200  000016  ERPROC: BIT     #BIT07,$TATUS(R0)       ;DONE BIT SET ?
007246  001402                          BEQ     ERPRC1          ;BR IF COMMAND DIDN'T COMPLETE NORMALLY
007250  000137  007634                  JMP     DONE            ;PROCESS ERROR WITH 'DONE' BIT SET
                                        
                                        ;PROCESS COMMAND COMPLETION WITH 'ERROR' & 'DONE NOT' BITS
                                        
007254                          ERPRC1:
007254  032760  010000  000016          BIT     #BIT12,$TATUS(R0)       ;SEE IF DRIVE WAS UNSAFE
007262  001025                          BNE     PUNSAF          ;BR IF YES
007264  032760  004000  000016          BIT     #BIT11,$TATUS(R0)       ;PARITY ERROR OCCURRED
007272  001043                          BNE     UCPAR           ;BR IF IT DID
007274  032760  002000  000016          BIT     #BIT10,$TATUS(R0)       ;FATAL PARITY ERROR?
007302  001046                          BNE     FALPAR          ;BR IF THERE IS ONE
007304  032760  001000  000016          BIT     #BIT09,$TATUS(R0)       ;TIMEOUT?
007312  001070                          BNE     SWTIM           ;BR IF YES
007314  032760  040002  000016          BIT     #BIT14!BIT01,$TATUS(R0) ;DRIVE WENT OFFLINE ?
007322  001103                          BNE     OFLIN           ;BR IF IT DID
007324  032760  000004  000016          BIT     #BIT2,$TATUS(R0)        ;PORT REQUEST TIME OUT ?
007332  001121                          BNE     PRTIM           ;BR IF IT DID
007334  000207                          RTS     PC              ;ERROR. RETURN
                                        
                                        ;DRIVE IS PERSISTENTLY UNSAFE
                                        
007336  104401  001203          PUNSAF: TYPE    ,$CRLF          ;CR-LF
007342  004737  021046                  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
007346  104414  021022                  DISPLY  ,EM12           ;PERSISTENT DEVICE UNSAFE MESSAGE
007352  004737  021126                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
007356  004737  021566                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
007362  004737  022236                  JSR     PC,LINE4        ;PRINT LINE 4 OF THE ERROR MESSAGE
007366  004737  024724                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
007372  004737  022710                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
007376  000137  031054                  JMP     DROP            ;DROP THE DRIVE
                                        
                                        ;UNCORRECTABLE MASSBUS PARITY ERROR OCCURRED
```

```
                                ;'UNCORRECTABLE' MASSBUS PARITY ERROR OCCURRED
007402  104401  001203  UCPAR:  TYPE    ,$CRLF          ;CR-LF
007406  104401  001203          TYPE    ,$CRLF          ;CR-LF
007412  104401  070724          TYPE    ,EM10           ;'UNCORRECTABLE PARITY ERROR' MESSAGE
007416  000406                  BR      FALPR1          ;FINISH PROCESSING THE ERROR

                                ;'FATAL' MASSBUS PARITY ERROR OCCURRED

007420  104401  001203  FALPAR: TYPE    ,$CRLF          ;CR-LF
007424  104401  001203          TYPE    ,$CRLF          ;CR-LF
007430  104401  070767          TYPE    ,EM11           ;'FATAL PARITY ERROR' MESSAGE

007434  104401  076311  FALPR1: TYPE    ,MSGON          ;TYPE 'ON'
007440  104401  075625          TYPE    ,UNTMSG         ;TYPE 'DRIVE'
007444  013746  001324          MOV     UNIT,-(SP)      ;;SAVE UNIT FOR TYPEOUT
                                                        ;;TYPE DRIVE NUMBER
007450  104403                  TYPOS                   ;;GO TYPE--OCTAL ASCII
007452     002                  .BYTE   2               ;;TYPE 2 DIGIT(S)
007453     000                  .BYTE   0               ;;SUPPRESS LEADING ZEROS
007454  004737  024724          JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
007460  032777  100000  171466  BIT     #SW15,@SWR      ;HALT ON ERROR ?
007466  001401                  BEQ     1$              ;BR IF NOT
007470  000000                  HALT                    ;ERROR HALT
007472  000207          1$:     RTS     PC

                                ;SOFTWARE TIMEOUT OCCURRED

007474  004737  021046  SWTIM:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
007500  104414  071053          DISPLY  ,EM13           ;PRINT THE TIME OUT MESSAGE
007504  004737  021126          JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
007510  004737  021566          JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
007514  004737  022236          JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
007520  004737  024724          JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
007524  004737  022710          JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
007530  000207                  RTS     PC              ;RETURN

                                ;DRIVE WENT OFFLINE

007532  104401  001203  OFLIN:  TYPE    ,$CRLF          ;CR-LF
007536  004737  021046          JSR     PC,LINE1        ;PRINT LINE 1 OF THE ERROR MESSAGE
007542  104414  071125          DISPLY  ,EM14           ;PRINT OFFLINE MESSAGE
007546  004737  021126          JSR     PC,LINE2        ;PRINT LINE 2 OF THE ERROR MESSAGE
007552  004737  021566          JSR     PC,LINE3        ;PRINT LINE 3 OF THE ERROR MESSAGE
007556  004737  022236          JSR     PC,LINE4        ;PRINT LINE 4 OF THE ERROR MESSAGE
007562  004737  024724          JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
007566  004737  022710          JSR     PC,LINE7        ;PRINT LINE 7 OF THE ERROR MESSAGE
007572  000137  031054          JMP     DROP            ;DROP THE DRIVE

                                ;PORT REQUEST TIMEOUT ERROR

007576  004737  021046  PRTIM:  JSR     PC,LINE1        ;TYPE LINE 1 OF THE ERROR MESSAGE
007602  104414  071150          DISPLY  ,EM15           ;PRINT PORT TIME OUT MESSAGE
007606  004737  021126          JSR     PC,LINE2        ;TYPE LINE 2 OF THE ERROR MESSAGE
007612  004737  021566          JSR     PC,LINE3        ;TYPE LINE 3 OF THE ERROR MESSAGE
007616  004737  022236          JSR     PC,LINE4        ;TYPE LINE 4 OF THE ERROR MESSAGE
007622  004737  024724          JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
007626  004737  022710          JSR     PC,LINE7        ;TYPE LINE 7 OF THE ERROR MESSAGE
```

```
 11 007632  000207                          RTS     PC              ;RETURN
112
113                                 ;PROCESS COMMAND COMPLETION WITH 'ERROR' & 'DONE' BITS SET
114
119 007634  032760  000030  000016  DONE:   BIT     #BIT04.BIT03,$TATUS(R0) ;UNSAFE OCCURRED
120 007642  001402                          BEQ     1$              ;BR IF NOT
121 007644  000137  012610                  JMP     UNSAF           ;REPORT UNSAFE
122
123 007650  032760  040000  002152  1$:     BIT     #BIT14,$RMER1(R0)       ;UNSAFE OCCURRED
124 007656  001402                          BEQ     2$              ;BR IF NOT
125 007660  000137  012610                  JMP     UNSAF           ;REPORT UNSAFE
126
127 007664  032760  040000  002146  2$:     BIT     #BIT14,$RMCS2(R0)       ;IS 'WCE' SET ?
128 007672  001402                          BEQ     3$              ;BRANCH IF NOT SET
129 007674  000137  010624                  JMP     WCKER           ;WRITE CHECK ERROR
130
131 007700  032760  040000  002150  3$:     BIT     #BIT14,$RMDS(R0)        ;CHECK 'ERR'
132 007706  001002                          BNE     4$              ;BR IF SET
133 007710  000137  012350                  JMP     TRFER           ;PROCESS 'TRE'
134
135 007714  032760  000400  002152  4$:     BIT     #BIT08,$RMER1(R0)       ;'HCRC' SET?
136 007722  001402                          BEQ     5$              ;BR IF NOT
137 007724  000137  011132                  JMP     HCRCER          ;PROCESS 'HCRC'
138
139 007730  032760  000020  002152  5$:     BIT     #BIT04,$RMER1(R0)       ;'FMT' SET?
140 007736  001402                          BEQ     6$              ;BR IF NOT SET
141 007740  000137  011310                  JMP     CKFMT           ;CHECK FORMAT ERROR
142
143 007744  032760  000200  002152  6$:     BIT     #BIT07,$RMER1(R0)       ;'HCE' SET?
144 007752  001402                          BEQ     7$              ;BR IF NOT SET
145 007754  000137  011464                  JMP     CKHCE           ;CHECK 'HCE' ERROR
146
147 007760  032760  020000  002152  7$:     BIT     #BIT13,$RMER1(R0)       ;'OPI' SET?
148 007766  001402                          BEQ     8$              ;BR IF NOT SET
149 007770  000137  011744                  JMP     OPIER           ;REPORT 'OPI'
150
151 007774  032760  000010  002152  8$:     BIT     #BIT3,$RMER1(R0)        ;'PAR' SET?
152 010002  001402                          BEQ     9$              ;BR IF NOT SET.
153 010004  000137  012072                  JMP     PARER           ;REPORT 'PAR'
154
155 010010  032760  000040  002152  9$:     BIT     #BIT5,$RMER1(R0)        ;'WCF' SET?
156 010016  001402                          BEQ     10$             ;BR IF NOT SET
157 010020  000137  012512                  JMP     WCFER           ;REPORT 'WCF'
158
159 010024  032760  002000  002152  10$:    BIT     #BIT10,$RMER1(R0)       ;'IAE' SET?
160 010032  001402                          BEQ     11$             ;BR IF NOT SET
161 010034  000137  012164                  JMP     IAEER           ;REPORT 'IAE'
162
163 010040  032760  004000  002152  11$:    BIT     #BIT11,$RMER1(R0)       ;'WLE' SET?
164 010046  001402                          BEQ     12$             ;BR IF NOT SET
165 010050  000137  012216                  JMP     WLEER           ;REPORT 'WLE'
166
167 010054  032760  001000  002152  12$:    BIT     #BIT9,$RMER1(R0)        ;'AOE' SET?
168 010062  001405                          BEQ     13$             ;BR IF NOT SET
169 010064  032760  002000  002150          BIT     #BIT10,$RMDS(R0)        ;'LBT' SET?
170 010072  001401                          BEQ     13$             ;BR IF NOT SET
171 010074  000207                          RTS     PC              ;'AOE' & 'LBT' SET, EXIT
```

```
                                                ;SEE IF 'DTE' SET
010076  032760  010000  002152  13$:    BIT     #BIT12,$RMER1(R0)
010102  001402                         BEQ     14$             ;BR IF NOT
010104  000137  012050                 JMP     DTEER           ;REPORT 'DTE' ERROR

010112  005760  002152          14$:    TST     $RMER1(R0)      ;SEE IF 'DCK' SET
010116  100002                         BPL     15$             ;BR IF NOT
010120  000137  010156                 JMP     DCKER           ;PROCESS 'DCK'

010124  032760  040000  002200  15$:    BIT     #BIT14,$RMER2(R0)       ;'SKI' SET
010132  001006                         BNE     16$             ;BRANCH IF SKI SET
010134  032760  100000  002200         BIT     #BIT15,$RMER2(R0)       ;'BSE' SET ?
010142  001004                         BNE     17$             ;BRANCH IF SO (NO, OTHER ERROR)
010144  000137  011256                 JMP     DRVER           ;REPORT ERROR

010150  000137  012450          16$:    JMP     SKIER           ;REPORT SKI ERROR
010154  000207                 17$:    RTS     PC              ;EXIT FROM ERROR ANALYSIS  ROUT.

                                ;PROCESS DATA ('DCK') CHECK ERROR

010156  004737  020714          DCKER:  JSR     PC,SPOTCK       ;SEE IF ERROR AT A BAD SECTOR ON THE DISK
010162  000207                         RTS     PC              ;IT IS, DON'T REPORT IT
010164  026027  002202  010040         CMP     $RMEC1(R0),#10040       ;IS POSITION COUNT OVER MAXIMUM ?
010172  101006                         BHI     1$              ;BR IF YES
010174  005760  002202                 TST     $RMEC1(R0)      ;POSITION COUNT 0 ?
010200  001403                         BEQ     1$              ;BR IF YES
010202  005760  002204                 TST     $RMEC2(R0)      ;VALUE IN PATTERN REGISTER ?
010206  001026                         BNE     4$              ;BR IF YES
010210  004737  021046          1$:     JSR     PC,LINE1        ;TYPE FIRST LINE OF ERROR MESSAGE
010214  104414  072577                 DISPLY  ,EM45           ;TYPE 'ECC LOGIC ERROR'
010220  004737  021126                 JSR     PC,LINE2        ;TYPE LINE 2 OF ERROR MESSAGE
010224  004737  024724                 JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
010230  012737  000003  001330         MOV     #3,RETRY        ;RETRY COUNT
010236  004737  016104                 JSR     PC,$RETRY       ;RETRY THE COMMAND
010242  000403                         BR      2$              ;RETRY WAS NOT SUCCESSFUL
010244  004737  022626                 JSR     PC,LINE6C       ;TYPE LINE 6C OF ERROR MESSAGE
010250  000402                         BR      3$              ;FINISH THE ERROR REPORT
010252  004737  022634          2$:     JSR     PC,LINE6D       ;TYPE LINE 6D OF ERROR MESSAGE
010256  004737  022710          3$:     JSR     PC,LINE7        ;TYPE LINE 7 OF ERROR MESSAGE
010262  000207                         RTS     PC              ;RETURN

                                ;THE VALUES IN THE ECC REGISTERS ARE CORRECT, REPORT 'DCK' ERROR

010264  004737  021046          4$:     JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
010270  104414  071225                 DISPLY  ,EM21           ;DATA CHECK ERROR

010274  004737  021126          DCKER1: JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
010300  004737  021566                 JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
010304  004737  022236                 JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
010310  004737  015352                 JSR     PC,PRTBAD       ;SEE IF BAD SECTOR TO BE PRINTED
010314  012737  110100  001326         MOV     #BIT15!BIT12!BIT06,MASK ;LOAD ERROR MASK
010322  032760  010100  002152         BIT     #BIT12!BIT06,$RMER1(R0) ;CHECK 'DTE' & 'ECH'
010330  001003                         BNE     1$              ;BR IF SET
010332  004737  022600                 JSR     PC,LINE6        ;PRINT LINE 6 OF ERROR MESSAGE
010336  000477                         BR      8$              ;FINISH THE ERROR REPORT
010340  012737  000020  001330  1$:     MOV     #16.,RETRY      ;RETRY COUNT
010346  005001                         CLR     R1              ;R1 IS OFFSET CODE POINTER
```

```
                                                ;RETRY
010350  004737  017050          2$:   JSR   PC,GODRIV       ;TEST FOR DONE
010354  005760  000016          3$:   TST   $TATUS(R0)      ;BR IF NOT DONE
010360  001775                        BEQ   3$              ;BR IF NOT ERROR
010362  100075                        BPL   10$             ;SEE IF COMMAND TERMINIATED NORMALLY
010364  032760  000200  000016        BIT   #BIT7,$TATUS(R0)
010372  001006                        BNE   14$             ;INCREMENT TOTAL ERROR COUNT
010374  004737  024724                JSR   PC,INCTOT       ;'DIFFERENT ERROR DURING RETRY'
010400  104414  074732                DISPLY ,LIN8M         ;SEE WHICH ERROR
010404  000137  007254                JMP   ERPRC1          ;LOOK AT CURRENT ERROR
010410  033760  001326  002152  14$:  BIT   MASK,$RMER1(R0) ;BR IF DIFFERENT ERROR
010416  001430                        BEQ   5$              ;'ECH' OR 'DTE' STILL SET ?
010420  032760  010100  002152        BIT   #BIT12!BIT6,$RMER1(R0)
010426  001437                        BEQ   7$              ;BR IF NEITHER SET
010430  105237  001331                INCB  RETRY+1         ;INCREMENT RETRY COUNT
010434  123737  001330  001331        CMPB  RETRY,RETRY+1   ;DONE ?
010442  001342                        BNE   2$              ;BR IF NOT
010444  005201                        INC   R1              ;INCREMENT TABLE INDEX
010446  116137  002414  070323        MOVB  OFFCOD(R1),GENDPB+$FMT  ;OFFSET CODE
010454  001435                        BEQ   9$              ;BR IF END OF OFFSET TABLE
010456  062737  000002  001330        ADD   #2,RETRY        ;NEW RETRY LIMIT
010464  004737  015766                JSR   PC,OFFST        ;OFFSET
010470  005737  070340          4$:   TST   GENDPB+$STATUS  ;SEE IF FINISHED WITH OFFSET
010474  001775                        BEQ   4$              ;BR IF NOT
010476  100324                        BPL   2$              ;BR IF NO ERROR PERFORMING OFFSET
010500  004737  023132          5$:   JSR   PC,LINE8        ;PRINT LINE 8 OF ERROR MESSAGE
010504  004737  024630          6$:   JSR   PC,INCHRD       ;INCREMENT 'HARD' ERROR COUNT
010510  004737  024724                JSR   PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
010514  004737  022710                JSR   PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
010520  004737  015352                JSR   PC,PRTBAD       ;PRINT THE BAD SECTOR
010524  000436                        BR    13$             ;CLEAN UP AND RETURN
010526  004737  022620          7$:   JSR   PC,LINE6B       ;PRINT LINE 6B OF ERROR MESSAGE
010532  004737  022536                JSR   PC,LINE5B       ;PRINT LINE 5B OF THE ERROR MESSAGE
010536  004737  024604          8$:   JSR   PC,INCSOF       ;INCREMENT 'SOFT' ERROR COUNT
010542  004737  014006                JSR   PC,ECC          ;CORRECT THE ERROR USING ECC AND CHECK IT
010546  000407                        BR    11$             ;COMPARE THE BUFFER

010550  004737  022634          9$:   JSR   PC,LINE6D       ;PRINT LINE 6D OF ERROR MESSAGE
010554  000753                        BR    6$              ;INCREMENT ERROR COUNT
010556  004737  022612          10$:  JSR   PC,LINE6A       ;PRINT LINE 6A OF ERROR MESSAGE
010562  004737  024604                JSR   PC,INCSOF       ;INCREMENT 'SOFT' ERROR COUNT
010566  012737  000001  001356  11$:  MOV   #1,FRSTER       ;SET PROCESSING 'DCKER' INDICATOR
010574  004737  013372                JSR   PC,CMPARD       ;COMPARE THE BUFFER
010600  105737  001357                TSTB  FRSTER+1        ;ERROR IN COMPARE ?
010604  100406                        BMI   13$             ;BRANCH IF ERROR
010606  004737  024724                JSR   PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
010612  104414  075126                DISPLY ,LIN9G         ;'DATA COMPARE OK' MESSAGE
010616  004737  022710          12$:  JSR   PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
010622  000207                  13$:  RTS   PC              ;RETURN

                                ;WRITE CHECK ERROR PROCESSING

010624  032760  100000  002152  WCKER: BIT  #BIT15,$RMER1(R0)  ;SEE IF 'DCK' SET ALSO
010632  001034                        BNE   2$              ;BR IF IT IS
010634  004737  021046                JSR   PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
010640  104414  071331                DISPLY ,EM23          ;PRINT WCE & DCK NOT
010644  005037  001326                CLR   MASK            ;CLEAR ERROR MASK
```

```
288 010650  004737  021126                      JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
    010654  004737  021566                      JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
    010660  004737  022236                      JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
    010664  004737  022326                      JSR     PC,LINE5        ;PRINT LINE 5 OF ERROR MESSAGE
289 010670  004737  024724                      JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
290 010674  012737  000003  001330              MOV     #3,RETRY        ;RETRY LIMIT
291 010702  004737  016104                      JSR     PC,$RETRY       ;RETRY THE OPERATION
292 010706  000403                              BR      1$              ;RETRY UNSUCESSFUL
293 010710  004737  022626                      JSR     PC,LINE6C       ;PRINT LINE 6C OF ERROR MESSAGE
294 010714  000501                              BR      10$             ;FINISH PROCESSING THE ERROR
295 010716  004737  022634          1$:         JSR     PC,LINE6D       ;PRINT LINE 6D OF ERROR MESSAGE
296 010722  000476                              BR      10$             ;FINISH PROCESSING THE ERROR
297 010724  004737  02071           2$:         JSR     PC,SPOTCK       ;SEE IF ERROR AT BAD SECTOR ON THE DISK
298 010730  000477                              BR      11$             ;EXIT IF AT BAD SECTOR ON DISK
299 010732  004737  021046                      JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
300 010736  012737  071256  010764              MOV     #EM22,4$        ;ASSUME THAT EM22 WILL BE PRINTED
301 010744  032760  040000  002146              BIT     #BIT14,$RMCS2(R0)       ;DID 'WCK' ALSO SET ?
302 010752  001003                              BNE     3$              ;BR IF IT DID
303 010754  012737  072157  010764              MOV     #EM37,4$        ;MESSAGE FOR 'DCK' AND 'WCK' NOT DURING
304                                                                     ;WRITE CHECK
305 010762  104414                  3$:         DISPLY                  ;TYPE THE ERROR MESSAGE
306 010764  000000                  4$:         .WORD   0               ;MESSAGE ADDRESS GOES HERE
309 010766  004737  021126                      JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
    010772  004737  021566                      JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
    010776  004737  022236                      JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
    011002  004737  022326                      JSR     PC,LINE5        ;PRINT LINE 5 OF ERROR MESSAGE
310 011006  032760  000100  002152              BIT     #BIT06,$RMER1(R0)       ;ECH SET ALSO ?
311 011014  001441                              BEQ     10$             ;FINISH PROCESSING THE ERROR
312 011016  012737  000020  001330  5$:         MOV     #16.,RETRY      ;RETRY LIMIT - 16 (10)
313 011024  004737  017050          6$:         JSR     PC,GODRIV       ;RETRY THE COMMAND
314 011030  005760  000016          7$:         TST     $TATUS(R0)      ;COMMAND FINISHED ?
315 011034  001775                              BEQ     7$              ;BR IF NOT
316 011036  100405                              BMI     8$              ;BR IF ERROR ON COMMAND
317 011040  105237  001331                      INCB    RETRY+1         ;INCREMENT RETRY COUNT
318 011044  004737  022626                      JSR     PC,LINE6C       ;PRINT LINE 6C OF ERROR MESSAGE
319 011050  000423                              BR      10$             ;FINISH ERROR PROCESSING
320 011052  105237  001331          8$:         INCB    RETRY+1         ;INCREMENT RETRY COUNT
321 011056  123737  001330  001331              CMPB    RETRY,RETRY+1   ;DONE ?
322 011064  001714                              BEQ     1$              ;BR IF AT RETRY LIMIT
323 011066  032760  100000  002152              BIT     #BIT15,$RMER1(R0)       ;'DCK' SET
324 011074  001407                              BEQ     9$              ;BR IF NOT - DIFFERENT ERROR
325 011076  032760  000100  002152              BIT     #BIT06,$RMER1(R0)       ;'ECH' ALSO SET ?
326 011104  001347                              BNE     6$              ;BR IF IT IS, RETRY COMMAND
327 011106  004737  022626                      JSR     PC,LINE6C       ;PRINT LINE 6C OF ERROR MESSAGE
328 011112  000402                              BR      10$             ;FINISH PROCESSING ERROR
329 011114  004737  023132          9$:         JSR     PC,LINE8        ;PRINT LINE 8 - 'DIFFERENT ERROR '
330 011120  004737  024724          10$:        JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
331 011124  004737  022710                      JSR     PC,LINE7        ;FINISH THE ERROR MESSAGE
332 011130  000207                  11$:        RTS     PC              ;RETURN
333
334                                 ;REPORT 'HCRC' ERROR
335
336 011132  004737  020714          HCRCER:     JSR     PC,SPOTCK       ;SEE IF ERROR AT BAD SECTOR
337 011136  000446                              BR      3$              ;EXIT IF IT IS
338 011140  004737  023216                      JSR     PC,READDR       ;READ ERROR SECTOR HEADER
339 011144  004737  016012                      JSR     PC,READHD       ;GET THE HEAD INFORMATION
340 011150  004737  021046                      JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
```

```
                                        DISPLY  ,EM20           ;REPORT 'HCRC'
011154  104414  071204                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
011160  004737  021126                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
011164  004737  021566                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
011170  004737  022236                  JSR     PC,LINE5A       ;PRINT THE HEADER INFORMATION
011174  004737  022470          1$:     JSR     PC,INCSOF       ;INCREMENT 'SOFT' ERROR COUNT
011200  004737  024604                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
011204  004737  024724          001326  MOV     #BIT8,MASK      ;SET ERROR MASK
011210  012737  000400  001326          MOV     #3,RETRY        ;RETRY LIMIT
011216  012737  000003  001330          JSR     PC,$RETRY       ;RETRY COMMAND
011224  004737  016104                  BR      2$              ;RETRY NOT SUCESSFUL
011230  000405                          JSR     PC,LINE6C       ;PRINT LINE 6C OF ERROR MESSAGE
011232  004737  022626                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
011236  004737  022710                  BR      3$              ;EXIT
011242  000404                  2$:     JSR     PC,LINE6D       ;PRINT LINE 6D OF ERROR MESSAGE
011244  004737  022634                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
011250  004737  022710          3$:     RTS     PC              ;RETURN
011254  000207

                                ;REPORT DRIVE ERROR

                                DRVER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
011256  004737  021046                  DISPLY  ,EM30           ;REPORT DRIVE ERROR
011262  104414  071621                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
011266  004737  021126                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
011272  004737  021566                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
011276  004737  024724                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
011302  004737  022710                  RTS     PC              ;RETURN
011306  000207

                                ;PROCESS FORMAT ('FER') ERROR

011310  032760  000400  002152  CKFMT:  BIT     #BIT8,$RMER1(R0)        ;'HCRC' SET ON ORIGINAL ERROR ?
011316  001402                          BEQ     1$              ;BR IF NOT SET
011320  000137  011132                  JMP     HCRCER          ;REPORT HCRC ERROR

011324  004737  023216          1$:     JSR     PC,READDR       ;GET CORRECTED TRACK & SECTOR ADDRSSES
011330  004737  016012                  JSR     PC,READHD       ;READ HEADER
011334  032737  000400  070356          BIT     #BIT8,GENREG+RMER1      ;'HCRC' SET WHEN HEADER READ?
011342  001002                          BNE     2$              ;BR IF 'HCRC' SET
011344  000137  012244                  JMP     FMTER           ;NO, ERROR IS 'FMT' ONLY

011350  004737  020714          2$:     JSR     PC,SPOTCK       ;SEE IF ERROR AT BAD SECTOR ON THE DISK
011354  000442                          BR      5$              ;EXIT IF IT IS
011356  004737  021046                  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
011362  104414  071410                  DISPLY  ,EM24           ;HEADER READ ERROR - FMT BIT DROPPED UP
011366  004737  021126                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
011372  004737  021566                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
011376  004737  022236                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
011402  004737  022470          3$:     JSR     PC,LINE5A       ;DISPLAY HEADER
011406  004737  024604                  JSR     PC,INCSOF       ;INCREMENT SOFT ERROR COUNT
011412  004737  024724                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
011416  012737  000020  001326          MOV     #BIT4,MASK      ;SET ERROR MASK
011424  012737  000003  001330          MOV     #3,RETRY        ;RETRY LIMIT
011432  004737  016104                  JSR     PC,$RETRY       ;RETRY THE COMMAND
011436  000405                          BR      4$              ;RETRY NOT SUCESSFUL
011440  004737  022626                  JSR     PC,LINE6C       ;PRINT LINE 6C OF ERROR MESSAGE
011444  004737  022710                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
011450  000404                          BR      5$              ;EXIT
```

```
                                        4$:     JSR     PC,LINE6D       ;PRINT LINE 6D OF ERROR MESSAGE
                                                JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
                                        5$:     RTS     PC              ;RETURN

                                        ;PROCESS HEADER COMPARE ('HCE') ERROR

                                        CKHCF:  BIT     #BIT8,$RMER1(R0)        ;HCRC SET ON ORIGINAL ERROR ?
                                                BEQ     1$              ;BR IF NOT SET
                                                JMP     HCRCER          ;REPORT HEADER CRC ERROR

                                        1$:     JSR     PC,READDR       ;GET CURRENT SECTOR & TRACK ADDRS
                                                JSR     PC,READHD       ;READ HEADER OF CURRENT SECTOR
                                                BIT     #BIT8,GENREG+RMER1      ;'HCRC' SET ?
                                                BNE     3$              ;BR IF SET
                                                MOV     CYLNDR,-(SP)    ;;PUSH CYLNDR ON STACK
                                                BIC     #150000,CYLNDR  ;CLEAR FORMAT, MFG AND USER BITS FROM HEADER
                                                CMP     $RMDC(R0),CYLNDR        ;CORRECT CYLINDER ?
                                                BEQ     2$              ;BR IF IT IS
                                                JMP     POSER           ;REPORT POSITIONING ERROR
                                        2$:
                                                MOV     (SP)+,CYLNDR    ;;POP STACK INTO CYLNDR
                                                JMP     HCEER           ;REPORT 'HCE' ERROR

                                        3$:     JSR     PC,SPOTCK       ;SEE IF ERROR AT BAD SECTOR
                                                BR      6$              ;EXIT IF IT IS
                                                JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
                                                DISPLY  ,EM25           ;HEADER READ ERROR - 'HCE' SET
                                                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
                                        4$:     JSR     PC,LINE5A       ;PRINT LINE 5 OF ERROR MESSAGE
                                                JSR     PC,INCSOF       ;INCREMENT SOFT ERROR COUNT
                                                JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
                                                MOV     #BIT7,MASK      ;SET ERROR MASK
                                                MOV     #3,RETRY        ;RETRY LIMIT
                                                JSR     PC,$RETRY       ;RETRY THE COMMAND
                                                BR      5$              ;RETRY NOT SUCESSFUL
                                                JSR     PC,LINE6C       ;PRINT LINE 6C OF ERROR MESSAGE
                                                JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
                                                BR      6$              ;EXIT
                                        5$:     JSR     PC,LINE6D       ;PRINT LINE 6D OF ERROR MESSAGE
                                                JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
                                        6$:     RTS     PC              ;RETURN

                                        ;REPORT POSSIBLE POSITIONING ERROR

                                        POSER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
                                                DISPLY  ,EM51           ;PROGRAM DETECTED POSITIONING ERROR
                                                JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
                                                JSR     PC,LINE3C       ;PRINT LINE 3C OF ERROR MESSAGE
                                                MOV     (SP)+,CYLNDR    ;;POP STACK INTO CYLNDR
                                                JSR     PC,LINE5A       ;PRINT LINE 5A OF THE ERROR MESSAGE
                                                JSR     PC,INCMIS       ;INCREMENT MISPOSITIONING COUNT
                                                JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
                                                JSR     PC,LINE7A       ;PRINT LINE 7A OF ERROR MESSAGE
                                                JSR     PC,RECALT       ;RECALIBRATE
                                                RTS     PC              ;EXIT
```

```
                                        ;REPORT 'OPI' ERROR

011744  004737  020714          OPIER:  JSR     PC,SPOTCK       ;SEE IF ERROR AT BAD SECTOR
011750  000207                          RTS     PC              ;RETURN IF IT IS
011752  004737  021046                  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
011756  104414  071653                  DISPLY  ,EM31           ;'OPI' ERROR
011762  004737  021126                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
011766  004737  021566                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
011772  004737  022236                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
011776  004737  024724                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
012002  012737  020000  001326          MOV     #BIT13,MASK     ;ERROR MASK
012010  012737  000003  001330  OPIER1: MOV     #3,RETRY        ;RETRY LIMIT
012016  004737  016104                  JSR     PC,$RETRY       ;RETRY THE COMMAND
012022  000405                          BR      1$              ;RETRY UNSUCESSFUL
012024  004737  022626                  JSR     PC,LINE6C       ;PRINT LINE 6C OF ERROR MESSAGE
012030  004737  022710                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
012034  000207                          RTS     PC              ;EXIT
012036  004737  022634          1$:     JSR     PC,LINE6D       ;PRINT LINE 6D OF ERROR MESSAGE
012042  004737  022710                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
012046  000207                          RTS     PC              ;RETURN

                                        ;REPORT 'DTE' ERROR

012050  004737  020714          DTEER:  JSR     PC,SPOTCK       ;SEE IF ERROR AT BAD SECTOR
012054  000207                          RTS     PC              ;RETURN IF IT IS
012056  004737  021046                  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
012062  104414  071716                  DISPLY  ,EM32           ;'DTE' ERROR
012066  000137  010274                  JMP     DCKER1          ;FINISH PROCESSING THE 'DTE' ERROR

                                        ;REPORT 'PAR' ERROR

012072  004737  021046          PARER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
012076  104414  071751                  DISPLY  ,EM33           ;REPORT 'PAR'
012102  004737  021126                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
012106  004737  021672                  JSR     PC,LINE3E       ;PRINT LINE 3E OF ERROR MESSAGE
012112  004737  022236                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
012116  004737  024724                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
012122  012737  000010  001326          MOV     #BIT03,MASK     ;ERROR MASK
012130  012737  000003  001330          MOV     #3,RETRY        ;RETRY LIMIT
012136  004737  016104                  JSR     PC,$RETRY       ;RETRY COMMAND
012142  000405                          BR      2$              ;RETRY UNSUCESSFUL
012144  004737  022626                  JSR     PC,LINE6C       ;RETRY SUCESSFUL
012150  004737  022710          1$:     JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
012154  000207                          RTS     PC              ;EXIT
012156  004737  022634          2$:     JSR     PC,LINE6D       ;PRINT LINE 6D OF ERROR MESSAGE
012162  000772                          BR      1$              ;FINISH ERROR MESSAGE

                                        ;REPORT 'IAE' ERROR

012164  004737  021046          IAEER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
012170  104414  072070                  DISPLY  ,EM35           ;REPORT 'IAE'
012174  004737  021126                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
012200  004737  021760                  JSR     PC,LINE3F       ;PRINT LINE 3F OF ERROR MESSAGE
012204  004737  024724                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
012210  004737  022710                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
012214  000207                          RTS     PC              ;RETURN
```

```
                                        ;REPORT 'WLE' ERROR

012216  004737  021046        WLEER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
012222  104414  072126                DISPLY  ,EM36           ;REPORT 'WLE'
012226  004737  021126                JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
012232  004737  024724                JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
012236  004737  022710                JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
012242  000207                        RTS     PC              ;RETURN

                                        ;REPORT FORMAT ERROR

012244  004737  021046        FMTER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
012250  104414  071537                DISPLY  ,EM26           ;FORMAT ERROR
012254  004737  021126                JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
012260  004737  021566                JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
012264  004737  022236                JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
012270  004737  022470                JSR     PC,LINE5A       ;PRINT LINE 5A OF ERROR MESSAGE
012274  004737  024724                JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
012300  004737  022710                JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
012304  000207                        RTS     PC

                                        ;REPORT HEADER COMPARE ERROR

012306  004737  021046        HCEER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
012312  104414  071564                DISPLY  ,EM27           ;HEADER COMPARE ERROR
012316  004737  021126                JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
012322  004737  021566                JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
012326  004737  022236                JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
012332  004737  022470                JSR     PC,LINE5A       ;PRINT LINE 5A OF ERROR MESSAGE
012336  004737  024724                JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
012342  004737  022710                JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
012346  000207                        RTS     PC              ;RETURN

                                    ;PROCESS CONTROL/INTERFACE TRANSFER ERROR

012350  004737  021046        TRFER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
012354  104414  072241                DISPLY  ,EM 0           ;RH/RM CONTROLLER OR UNIBUS TRANSFER ERROR
012360  004737  021126                JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
012364  004737  021566                JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
012370  004737  022236                JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
012374  004737  024724                JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
012400  032760  121400  002146        BIT     #BIT15!BIT13!BIT9!BIT8,$RMCS2(R0)       ;'DLT','UPE','MXF', 'MDPE' SET ?
012406  001415                        BEQ     2$              ;BR IF NONE SET
012410  012737  000003  001330        MOV     #3,RETRY        ;RETRY LIMIT
012416  005037  001326                CLR     MASK            ;CLEAR ERROR MASK
012422  004737  016104                JSR     PC,$RETRY       ;RETRY THE OPERATION
012426  000403                        BR      1$              ;RETURN HERE IF RETRY UNSUCESSFUL
012430  004737  022626                JSR     PC,LINE6C       ;PRINT LINE 6C OF ERROR MESSAGE
012434  000402                        BR      2$              ;FINISH THE ERROR REPORT
012436  004737  022634        1$:     JSR     PC,LINE6D       ;PRINT LINE 6D OF ERROR MESSAGE
012442  004737  022710        2$:     JSR     PC,LINE7        ;PRINT LINE 7 OF ERROP MESSAGE
012446  000207                        RTS     PC

                                        ;PROCESS 'SKI' ERRORS

012450  004737  021046        SKIER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
```

```
595 012454  104414  072737                      DISPLY  ,EM50           ;'SKI' ERROR
596 012460  004737  021126                      JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
597 012464  004737  021602                      JSR     PC,LINE3B       ;PRINT LINE 3B OF ERROR MESSAGE
598 012470  004737  024724                      JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
599 012474  004737  024654                      JSR     PC,INCSKI       ;INCREMENT 'SKI' ERROR COUNT
600 012500  004737  023040                      JSR     PC,LINE7A       ;PRINT LINE 7A OF ERROR MESSAGE
601 012504  004737  015650                      JSR     PC,RECALT       ;RECALIBRATE
602 012510  000207                              RTS     PC
603
604                                     ;REPORT WRITE CLOCK FAILURE ('WCF')
605
606 012512  004737  021046              WCFER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
607 012516  104414  072026                      DISPLY  ,EM34           ;REPORT WRITE CLOCK FAILURE
608 012522  004737  021126                      JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
609 012526  004737  021574                      JSR     PC,LINE3A       ;PRINT LINE 3A OF ERROR MESSAGE
610 012532  004737  022236                      JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
611 012536  004737  024724                      JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
612 012542  004737  015352                      JSR     PC,PRTBAD       ;SEE IF BAD SECTOR TO BE PRINTED
613 012546  012737  000003  001330              MOV     #3,RETRY        ;RETRY COUNT
614 012554  012737  000040  001326              MOV     #BIT05,MASK     ;ERROR MASK
615 012562  004737  016104                      JSR     PC,$RETRY       ;RETRY THE COMMAND
616 012566  000405                              BR      2$              ;RETURN HERE IF RETRY UNSUCESSFUL
617 012570  004737  022626                      JSR     PC,LINE6C       ;PRINT LINE 6C OF ERROR MESSAGE
618 012574  004737  022710              1$:     JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
619 012600  000207                              RTS     PC
620 012602  004737  022634              2$:     JSR     PC,LINE6D       ;PRINT LINE 6D OF ERROR MESSAGE
621 012606  000772                              BR      1$
622
623                                     ;PROCESS DRIVE UNSAFE ERROR
624
625 01261C  004737  021046              UNSAF:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
626 012614  104414  073040                      DISPLY  ,EM60           ;REPORT DRIVE UNSAFE
627 012620  004737  021126                      JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
628 012624  004737  021566                      JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
629 012630  004737  024724                      JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
630 012634  032760  040000  002200              BIT     #BIT14,$RMER2(R0)       ;IS 'SKI' ALSO SET ?
631 012642  001016                              BNE     2$              ;BR IF YES
632 012644  012737  040000  001326              MOV     #BIT14,MASK     ;LOAD THE ERROR MASK
633 012652  012737  000003  001330              MOV     #3,RETRY        ;RETRY COUNT
634 012660  004737  016104                      JSR     PC,$RETRY       ;RETRY THE COMMAND
635 012664  000403                              BR      1$              ;RETRY WAS UNSUCESSFUL
636 012666  004737  022626                      JSR     PC,LINE6C       ;PRINT LINE 6C OF ERROR MESSAGE
637 012672  000402                              BR      2$              ;CONTINUE WITH ERROR REPORT
638 012674  004737  022634              1$:     JSR     PC,LINE6D       ;PRINT LINE 6D OF ERROR MESSAGE
639 012700  004737  022710              2$:     JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
640 012704  032760  040000  002200              BIT     #BIT14,$RMER2(R0)       ;CHECK 'SKI' AGAIN
641 012712  001001                              BNE     3$              ;BR IF SET
642 012714  000207                              RTS     PC              ;RETURN
643 012716  004737  015650              3$:     JSR     PC,RECALT       ;RECALIBRATE
644 012722  000207                              RTS     PC              ;RETURN
645
646                                     ;REPORT AN 'UNKNOWN' DATA PATTERN
647
648 012724  105737  001356              NOMTCH: TSTB    FRSTER          ;FIRST ERROR IN THE SECTOR ?
649 012730  001013                              BNE     1$              ;BR IF NOT OR IF PROCESSING 'DCKER'
650 012732  004737  021046                      JSR     PC,LINE1        ;TYPE LINE 1 OF ERROR MESSAGE
651 012736  104414  072435                      DISPLY  ,EM43           ;'CAN'T MATCH DATA WITH PATTERN'
```

```
                                        JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
                                        JSR     PC,LINE3A       ;PRINT LINE 3A OF ERROR MESSAGE
                                        JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
                                        BR      2$              ;CONTINUE PROCESSING ERROR
                                1$:     DISPLY  ,EM43           ;'CAN'T MATCH DATA WITH PATTERN'
                                        DISPLY  ,$CRLF          ;CR-LF
                                2$:     DISPLY  ,LIN91          ,HEADER FOR DATA PRINTOUT
                                        MOV     R1,-(SP)        ;ADDRESS OF WORD 1
                                        JSR     PC,LINOCT       ;TYPE WORD 1
                                        DISPLY  ,BLNKS2         ;TYPE 2 BLANKS
                                        MOV     (R1)+,-(SP)     ;ADDRESS OF WORD 1
                                        JSR     PC,LINOCT       ;TYPE WORD 1
                                        DISPLY  ,$CRLF          ;CR-LF
                                        MOV     R1,-(SP)        ;ADDRESS OF WORD 2
                                        JSR     PC,LINOCT       ;TYPE WORD 2
                                        DISPLY  ,BLNKS2         ;TYPE 2 BLANKS
                                        MOV     (R1)+,-(SP)     ;ADDRESS OF WORD 2
                                        JSR     PC,LINOCT       ;TYPE WORD 2
                                        DISPLY  ,$CRLF          ;CR-LF
                                        MOV     R1,-(SP)        ;ADDRESS OF WORD 3
                                        JSR     PC,LINOCT       ;TYPE WORD 3
                                        DISPLY  ,BLNKS2         ;TYPE 2 BLANKS
                                        MOV     (R1)+,-(SP)     ;ADDRESS OF WORD 3
                                        JSR     PC,LINOCT       ;TYPE WORD 3
                                        DISPLY  ,$CRLF          ;CR-LF
                                        MOV     R1,-(SP)        ;ADDRESS OF WORD 4
                                        JSR     PC,LINOCT       ;TYPE WORD 4
                                        DISPLY  ,BLNKS2         ;TYPE 2 BLANKS
                                        MOV     (R1)+,-(SP)     ;ADDRESS OF WORD 4
                                        JSR     PC,LINOCT       ;TYPE WORD 4
                                        DISPLY  ,$CRLF          ;CR-LF
                                        ADD     #<252.*2.>,R1   ;INCREMENT BUFFER POINTER
                                        SUB     #256.,CMCNT     ;ADJUST WORD COUNT FOR MATCH
                                        BITB    #1,$CODE(R0)    ;HEADER OPERATION INVOLVED ?
                                        BEQ     3$              ;NO
                                        ADD     #4,R1           ;ADJUST BUFFER ADDRESS
                                        SUB     #2,CMCNT        ;ADJUST WORD COUNT
                                3$:     CLR     R2              ;CLEAR 'WORDS TO COMPARE' COUNT IN R2
                                        MOVB    #1,FRSTER       ;SET 'NOT FIRST ERROR' INDICATOR
                                        MOVB    #-1,FRSTER+1    ;SET ERROR FOUND INDICATOR
                                        MOV     CMPLMT,LIMIT    ;RESET THE COMPARE ERROR TYPEOUT LIMIT
                                        RTS     PC              ;RETURN

                                ;CHECK ERROR BITS IN THE RH/RM REGISTERS

                        CKERR:  BIT     #60000,$RMCS1(R0)       ;SEE IF 'TRE' OR 'MCPE' SET
                                BNE     1$                      ;BR IT EITHER SET
                                BIT     #177400,$RMCS2(R0)      ;SEE IF ERROR BITS IN CS2 SET
                                BNE     1$                      ;BR IF ANY SET
                                TST     $RMER1(R0)              ;ANY BITS SET IN ER1
                                BNE     1$                      ;BR IF ANY SET
                                TST     $RMER2(R0)              ;ANY BITS SET IN ER2 ?
                                BEQ     2$                      ;BR IF NONE SET
                        1$:     JSR     PC,LINE1                ;PRINT LINE 1 OF ERROR MESSAGE
                                DISPLY  ,EM44                   ;ERROR BITS SET. BUT 'SC' OR 'TRE' NOT SET
                                JSR     PC,LINE2                ;PRINT LINE 2 OF ERROR MESSAGE
                                JSR     PC,LINE3                ;PRINT LINE 3 OF ERROR MESSAGE
```

```
693 013252  004737  022236              JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
694 013256  004737  024724              JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
695 013262  004737  022710              JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
696 013266  000207             2$:     RTS     PC              ;RETURN
697
698                             ;CHECK BUS ADDRESS REGISTER & WORD COUNT REGISTER
699
700 013270  005760  002140     CKBUS:  TST     $RMWC(R0)       ;CHECK WORD COUNT
701 013274  001010              BNE     1$              ;BR IF NOT ZERO
702 013276  016046  000020              MOV     $WRDL(R0),-(SP) ;WORD LENGTH
703 013302  006316              ASL     (SP)            ;CHANGE INTO BYTE COUNT
704 013304  066016  000006              ADD     $BUF(R0),(SP)   ;ADD THE STARTING LOCATION
705 013310  022660  002142              CMP     (SP)+,$RMBA(R0) ;BUFFER ADDRESS PROPER ?
706 013314  001416              BEQ     2$              ;BR IF OK
707 013316  004737  021046     1$:     JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
708 013322  104414  072310              DISPLY  ,EM41           ;BUS ADDRESS OR WORD COUNT INCORRECT
709 013326  004737  021126              JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
710 013332  004737  021624              JSR     PC,LINE3D       ;PRINT LINE 3D OF ERROR MESSAGE
711 013336  004737  022236              JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
712 013342  004737  024724              JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
713 013346  004737  022710              JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
714 013352  000207             2$:     RTS     PC
```

```
                                        ;COMPARE THE BUFFER
                                        
013354  132760  000004  000024  CMPAR:  BITB    #4,$CODE(R0)    ;SEE IF READ COMMAND
013362  001001                          BNE     1$              ;BR IF IT IS
013364  000207                          RTS     PC              ;RETURN
013366  005037  001356          1$:     CLR     FRSTER          ;CLEAR 'FIRST ERROR' INDICATOR

013372                          CMPARD:
013372  005037  001364          1$:     CLR     ERCTR           ;CLEAR THE ERROR COUNTER
013376  016001  000006                  MOV     $BUF(R0),R1     ;BUFFER ADDRESS
013402  016037  000020  001370          MOV     $WRDL(R0),CMCNT ;WORD COUNT TO WORKING LOCATION
013410  066037  002140  001370          ADD     $RMWC(R0),CMCNT ;CALCULATE ACTUAL WORDS TRANSFERED
013416  001001                          BNE     2$
013420  000207                          RTS     PC              ;EXIT--NO WORDS XFERED
013422  016037  000012  001372  2$:     MOV     $CYL(R0),CMCYL  ;CYLINDER ADDRESS WORKING LOCATION
013430  052737  150000  001372          BIS     #150000,CMCYL   ;SET MFG, USER AND FMT BITS
013436  016037  000010  001374          MOV     $SEC(R0),CMSEC  ;SECTOR & TRACK ADDRESSES TO WORKING LOCNS
013444  013737  001446  001366          MOV     CMPLMT,LIMIT    ;DISPLAY LIMIT
013452  005237  001366                  INC     LIMIT           ;CONVERT PARAMETER INTO LIMIT VALUE
013456  012737  177777  001354  CMSTR:  MOV     #-1,ZROIND      ;CLEAR THE 'ZERO'S' INDICATOR
013464  005037  001360                  CLR     SAVER1          ;CLEAR THE R1 SAVE WORD
013470  005037  001362                  CLR     SAVER5          ;CLEAR THE R5 SAVE WORD
013474  023760  001370  000022          CMP     CMCNT,$SSEC(R0) ;IS BUFFER SIZE GREATER THAN ONE SECTOR ?
013502  101005                          BHI     1$              ;BR IF IT IS
013504  013702  001370                  MOV     CMCNT,R2        ;LESS THAN, USE REMAINING BUFFER
013510  005037  001370                  CLR     CMCNT           ;SET COUNTER TO 0
013514  000405                          BR      2$
013516  016002  000022          1$:     MOV     $SSEC(R0),R2    ;COMPARE SECTOR
013522  166037  000022  001370          SUB     $SSEC(R0),CMCNT ;DECREMENT WORD COUNT
013530  126027  000024  000005  2$:     CMPB    $CODE(R0),#5    ;READ HEADER & DATA?
013536  001026                          BNE     CMDAT           ;BR IF NOT

                                        ;COMPARE HEADER WORDS

013540  012705  001372          CMHED:  MOV     #CMCYL,R5       ;ADDRESS OF COMPARING CYLINDER
013544  052711  150000                  BIS     #150000,(R1)    ;SET BITS INCASE BAD SECTOR ENCOUNTER
013550  022521                          CMP     (R5)+,(R1)+     ;CHECK CYLINDER
013552  001402                          BEQ     1$              ;BR IF COMPARE OK
013554  004737  013602                  JSR     PC,CMSTR2       ;REPORT ERROR
013560  022521                  1$:     CMP     (R5)+,(R1)+     ;COMPARE SECTOR/TRACK
013562  001402                          BEQ     2$              ;BR IF EQ
013564  004737  013602                  JSR     PC,CMSTR2       ;REPORT ERROR
013570  162702  000002          2$:     SUB     #2,R2           ;SUBTRACT HEADER LENGTH FROM SIZE
013574  003007                          BGT     CMDAT           ;BR IF NOT FINISHED
013576  000137  014166                  JMP     CMPRX           ;COMPARE THE DATA PORTION

013602  005237  001364          CMSTR2: INC     ERCTR           ;INCREMENT THE ERROR COUNT
013606  004737  014174                  JSR     PC,CMPRT        ;REPORT THE COMPARISON ERROR
013612  000207                          RTS     PC              ;CHECK THE REST OF THE HEADER

                                        ;COMPARE DATA FIELD

013614  004737  027346          CMDAT:  JSR     PC,GETLMT       ;GET ADDRESS LIMITS
013620  004737  014516                  JSR     PC,MATCH        ;FIND THE PATTERN
013624  000403                          BR      1$              ;FOUND A PATTERN
013626  004737  012724                  JSR     PC,NOMTCH       ;RETURN HERE IF NO MATCH WITH PATTERN MADE
013632  000456                          BR      7$              ;BYPASS COMPARE ROUTINE
```

```
                                1$:     MOV     (R4),R5         ;ADDRESS OF PATTERN ADDRESS IN R4
                                        MOV     #16.,R3         ;R3 IS PATTERN POS COUNTER
                                2$:     CMP     (R1)+,(R5)+     ;COMPARE BUFFER WITH PATTERN
                                        BNE     4$              ;BR IF NOT EQUAL
                                        TST     ERCTR           ;ERRORS DETECTED ?
                                        BEQ     3$              ;BR IF NO ERRORS
                                        BIT     #SW3,@SWR       ;SWITCH 3 SET ?
                                        BEQ     3$              ;BR IF NOT SET
                                        JSR     PC,CMPRT        ;DISPLAY THE WORD
                                3$:     DEC     R2              ;DECREMENT SIZE COUNT
                                        BLE     7$              ;BR WHEN AT END
                                        DEC     R3              ;DECREMENT PATT POS COUNT
                                        BNE     2$              ;BR IF NOT AT END OF PATT
                                        BR      1$              ;RESTART THE PATTERN
                                4$:     TST     -2(R1)          ;IS MISCOMPARED CHARACTER-0
                                        BEQ     5$              ;BR IF YES
                                        MOV     #-1,ZROIND      ;SET NON-ZERO MISCOMPARED INDCATOR
                                        INC     ERCTR           ;INCREMENT THE ERROR COUNTER
                                        JSR     PC,CMPRT        ;REPORT ERROR
                                        BR      3$              ;CONTINUE COMPARE
                                5$:     TSTB    FRSTER          ;FIRST ERROR?
                                        BMI     6$              ;BR IF NOT
                                        CLR     ZROIND          ;SET THE ZERO INDICATOR
                                        MOV     R1,SAVER1       ;SAVE CURRENT R1
                                        MOV     R5,SAVER5       ;SAVE CURRENT R5
                                        BR      3$              ;CONTINUE COMPARE
                                6$:     TST     ZROIND          ;ANY MISCOMPARIONS NOT ZEROS ?
                                        BEQ     3$              ;BR IF NONE-ALL ERRORS-ZERO
                                        JSR     PC,CMPRT        ;REPORT ERROR
                                        BR      3$              ;CONTINUE COMPARING
                                7$:     CMP     CMCNT,#3        ;LAST 3 WORDS ?
                                        BLE     CMPRX           ;YES
                                        CMPB    $CODE(R0),#5    ;READ HEAD AND DATA ?
                                        BEQ     9$              ;YES
                                8$:     MOV     CMCNT,R2        ;SET COUNTER - REMAIN BUFFER LENGTH
                                        CMP     R2,#3           ;LAST 3 WORDS ?
                                        BLE     CMPRX           ;YES,EXIT
                                        SUB     #256.,CMCNT     ;GREATER THAN A SECTOR ?
                                        BLE     CMDAT           ;NO,RETURN TO COMPARE LOOP
                                        MOV     #256.,R2        ;SET COUNTER =SECTOR SIZE
                                        BR      CMDAT           ;RETURN TO COMPARE LOOP
                                9$:     INCB    CMSEC           ;INCREMENT COUNTER
                                        CMPB    CMSEC,SECLMT    ;MAX SECTOR # ?
                                        BLOS    10$             ;NO
                                        CLRB    CMSEC           ;RESET SECTOR #
                                        INCB    CMTRK           ;INCREMENT TRACK #
                                        CMPB    CMTRK,TRKLMT    ;MAX TRACK # ?
                                        BLOS    10$             ;NO
                                        CLRB    CMTRK           ;RESET TRACK #
                                        INC     CMCYL           ;INCREMENT CYLINDER NUMBER
                                        MOV     CMCYL,-(SP)     ;GET COMPARING CYLINDER
                                        BIC     #150000,(SP)    ;SAVE ONLY THE CYLINDER BITS
                                        CMP     (SP)+,CYLIMT    ;LAST CYLINDER ?
                                        BLOS    10$             ;NO
                                        BR      CMPRX           ;NORMAL RETURN,NOT WRAP AROUND

                                10$:    MOV     #CMCYL,R5       ;ADDRESS OF COMPARING CYLINDER
```

```
                                                ;SET BITS INCASE BAD SECTOR ENCOUNTER
                                                ;COMPARE 1ST HEADER WORD
                                                ;MATCH
                                                ;NOT MATCH
                                                ;SECOND WORD OF HEADER
                                                ;MATCH
                                                ;NOT MATCH
                                                ;ADJUST WORD COUNT
                                                ;COMPARE IS DONE
                                                ;RETURN TO COMPARE LOOP

                                                ;PRINT LAST LINE IF ERRORS

                                ;TYPE DATA COMPARE ERRORS

                                                ;PRINT SAVED VALUES ?
                                                ;BR IF YES
                                                ;FIRST ERROR?
                                                ;BR IF NOT
                                                ;PRINT INITIAL MESSAGE INFO
                                                ;PRINT REMAINDER OF MESSAGE
                                                ;EXIT

                                                ;;PUSH R1 ON STACK
                                                ;;PUSH R5 ON STACK
                                                ;DISPLAY SAVED R1
                                                ;DISPLAY SAVED R5
                                                ;PRINT INITIAL MESSAGE INFO
                                                ;PRINT SAVED VALUES
                                                ;CLEAR SAVED REGISTER INDICATORS
                                                ;CLEAR THE OTHER ONE
                                                ;;POP STACK INTO R5
                                                ;;POP STACK INTO R1
                                                ;PRINT REMAINDER OF MESSAGE
                                                ;RETURN

                                                ;FIRST ERROR ?
                                                ;BR IF NOT
                                                ;BR IF FIRST ERROR AND PROCESSING 'DCK' ERROR
                                                ;PRINT LINE 1 OF ERROR MESSAGE
                                                ;DATA COMPARE ERROR
                                                ;PRINT LINE 2 OF ERROR MESSAGE
                                                ;PRINT LINE 3A OF ERROR MESSAGE
                                                ;PRINT LINE 4 OF ERROR MESSAGE
                                                ;GO TO TYPE HEADER
                                                ;HEADER MESSAGE OF PROCESSING 'DCK' ERROR
                                                ;CR-LF
                                                ;DISPLAY HEADER
                                                ;SET FIRST ERROR FLAG
                                                ;RETURN

                                                ;TYPEOUT LIMIT REACHED ?
                                                ;BR IF IT HAS
                                                ;DECREMENT LIMIT COUNTER
                                                ;BR IF NOT AT LIMIT
                                                ;PRINT ALL DATA COMPARE ERRORS ?
```

```
                                        BNE     10$             ;BR IF YES
                                        RTS     PC              ;RETURN

                                10$:    MOV     R1,-(SP)        ;BUFFER ADDRESS
                                        SUB     #2,(SP)         ;ADJUST ADDRESS
                                        JSR     PC,LINOCT       ;TYPE IT
                                        DISPLY  ,BLNKS2         ;TYPE 2 BLANKS
                                        MOV     -2(R5),-(SP)    ;PUT GOOD DATA ON THE STACK
                                        JSR     PC,LINOCT       ;TYPE IT
                                        DISPLY  ,BLNKS2         ;TYPE 2 BLANKS
                                        MOV     -2(R1),-(SP)    ;BAD DATA
                                        JSR     PC,LINOCT       ;TYPE IT
                                        DISPLY  ,$CRLF          ;CR-LF
                                        RTS     PC              ;RETURN

                                ;LAST LINE OF COMPARE ERROR REPORTING

                                ENDCMP: TSTB    ERSTER+1        ;ANY COMPARE ERRORS FOUND ?
                                        BEQ     2$              ;BR IF NOT
                                        TST     ERCTR           ;SEE HOW MANY ERRORS
                                        BEQ     1$              ;BR IF ONLY CAN'T MATCH PATTERN
                                        DISPLY  ,LIN9E          ;'NUMBER OF ERRORS='
                                        MOV     ERCTR,-(SP)     ;NUMBER OF ERRORS
                                        JSR     PC,LINDEC       ;TYPE IT
                                        DISPLY  ,$CRLF          ;CR-LF
                                1$:     JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
                                        JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
                                2$:     RTS     PC              ;RETURN


                                ;ROUTINE TO MATCH THE DATA WITH A PATTERN, ONLY WHEN LOCATION 'PATTERN'
                                ;IS EQUAL TO 0 (RANDON DATA PATTERN MODE). OTHERWISE, THIS ROUTINE WILL
                                ;RETURN THE ADDRESS OF THE EXPECTED FIXED DATA PATTERN IN R4.
                                ;CALL:
                                ;       MOV     #BUFFER,R1      ;BUFFER ADDRESS
                                ;       JSR     PC,MATCH
                                ;       RETURN1                 ;PATTERN ADDRESS IN R4
                                ;       RETURN2                 ;COULDN'T MATCH PATTERN

                                MATCH:  MOV     R1,-(SP)        ;SAVE R1 ON THE STACK
                                        MOV     PATTERN,R4      ;WAS RANDOM PATTERN ENABLED ?
                                        BEQ     1$              ;BR IF YES
                                        ASL     R4              ;* 2
                                        BR      4$              ;USE KNOWN PATTERN
                                1$:     MOV     #44,R4          ;PATTERN TABLE INDEX
                                2$:     MOV     (SP),R1         ;RELOAD R1
                                        SUB     #2,R4           ;DECREMENT INDEX
                                        BEQ     5$              ;BR IF PATTERN NOT MATCH
                                        MOV     STNDAT(R4),R5   ;ADDRESS OF PATTERN ADDRESS
                                        MOV     #4,R3           ;NUMBER OF LOCATIONS TO CHECK
                                3$:     CMP     (R1)+,(R5)+     ;COMPARE THE BUFFER AGAINST THE PATTERN
                                        BNE     2$              ;BR IF NOT EQUAL, TRY NEXT PATTERN
                                        DEC     R3              ;FINISHED CHECKING?
                                        BNE     3$              ;BR IF NOT FINISHED
                                4$:     ADD     #STNDAT,R4      ;MAKE PATTERN ADDRESS ABSOLUTE
                                        BR      6$              ;EXIT
                                5$:     ADD     #2,2(SP)        ;INCREMENT RETURN ADDRESS
```

```
                                        6$:     MOV     (SP)+,R1        ;RESTORE R1
                                                RTS     PC              ;RETURN

                                        ;USE ECC TO CORRECT THE DATA ERROR

001400  ECC:    MOV     $RMBA(R0),ECSEC ;ADDRESS OF LAST LOCN XFERED
                MOV     $RMWC(R0),-(SP) ;ACT WORDS XFERED (2'S COMP)
                ADD     $WRDL(R0),(SP)  ;ADD WORDS REQUESTED
                BNE     1$
                TST     (SP)+           ;RESTORE STACK
                RTS     PC              ;EXIT--NO WORDS XFERRED
        1$:     CLR     -(SP)           ;CLEAR NEXT STACK LOCN
                MOV     $SSEC(R0),-(SP) ;SECTOR SIZE
                JSR     PC,$DIV         ;DIVIDE WORDS XFERED BY SECTOR SIZE
                TST     (SP)            ;PARTIAL SECTOR XFERED ?
                BEQ     2$              ;BR IF NOT
                ASL     (SP)            ;CONVERT INTO NUMBER OF BYTES
                SUB     (SP),ECSEC      ;SUBTRACT SECTOR RESIDUE
000024          CMPB    #5,$CODE(R0)    ;WAS OPERATION, READ HEAD & DATA
                BNE     3$              ;BR IF NOT
001400          ADD     #4,ECSEC        ;ADD HEADER SIZE (IN BYTES) BACK IN
                BR      3$              ;GO ADJUST THE STACK POINTER
001400  2$:     SUB     #256.*2,ECSEC   ;SUBTRACT SECTOR DATA FIELD SIZE (IN BYTES)
        3$:     ADD     #4,SP           ;ADJUST THE STACK POINTER
001376          MOV     $RMEC1(R0),ECBIT        ;ECC POSITION COUNT
                DEC     ECBIT           ;ADJUST BIT POSITION
001406          MOV     ECBIT,ECWRD     ;LOAD THE WORD COUNT LOCATION
001376          BIC     #^C17,ECBIT     ;SAVE THE BIT OFFSET COUNT
001406          BIC     #17,ECWRD       ;CLEAR THE BIT OFFSET
                ASR     ECWRD           ;CHANGE TO BYTE COUNT(DIVIDE BY 2)
                ASR     ECWRD           ;CHANGE TO BYTE COUNT(DIVIDE BY 4)
                ASR     ECWRD           ;CHANGE TO BYTE COUNT(DIVIDE BY 8.)
075154          DISPLY  ,LIN10A         ;'ERROR BURST BEGINS AT '
                MOV     ECWRD,-(SP)     ;PUT THE WORD COUNT ON THE STACK
                ASR     (SP)            ;GET STARTING WORD FOR MESSAGE(DIVIDE BY 16.)
033130          JSR     PC,$SB2D        ;CONVERT THE WORD COUNT TO DECIMAL
032234          JSR     PC,$SUPRL       ;AND PRINT IT
075210          DISPLY  ,LIN10B         ;' IN DATA FIELD OF ERROR SECTOR'
001400  001406  ADD     ECSEC,ECWRD     ;FIND THE BEGINNING OF THE ERROR BURST
002142  001406  CMP     $RMBA(R0),ECWRD ;SEE IF BURST WAS IN DATA READ
                BHI     4$              ;BR IF IN DATA READ
015340          JMP     ECC2            ;NOT IN DATA READ - REPORT IT

001402  4$:     MOV     $RMEC2(R0),ECMSK0       ;GET THE ERROR BIT MASK
001404          CLR     ECMSK1          ;CLEAR THE UPPER MASK WORD
001376  5$:     DEC     ECBIT           ;DECREMENT THE BIT OFFSET COUNT
                BLT     6$              ;BR IF DONE
001402          ASL     ECMSK0          ;SHIFT THE ERROR MASK
001404          ROL     ECMSK1          ;SHIFT THE LOWER INTO THE UPPER
                BR      5$              ;CONTINUE THE SHIFT

001412  6$:     MOV     @ECWRD,ECBAD0   ;SAVE THE INCORRECT WORD
                MOV     ECMSK0,-(SP)    ;PUT LOWER MASK ON STACK
                BIC     @ECWRD,(SP)     ;CLEAR ERRONEOUS ONE BITS FROM MASK
164300          BIC     ECMSK0,@ECWRD   ;CLEAR ERRONEOUS ONE BITS FROM BAD WORD
                BIS     (SP)+,@ECWRD    ;SET DROPPED BITS
```

```
                                                ;DOES ERROR GO INTO NEXT WORD ?
                                                ;BR IF NO
                                                ;DUPLICATE ADDRESS
                                                ;INCREMENT ERROR ADDRESS
                                                ;IS NEXT WORD IN THE BUFFER ?
                                                ;BR IF YES, ELSE,
                                                ;WAS ERROR IN FIRST WORD ?
                                                ;BR IF NO
                                                ;CLEAR 2ND WORD ADDRESS
                                                ;PRINT WORD CORRECTED

                                                ;SAVE THE SECOND BAD WORD
                                                ;PUT THE UPPER MASK ON THE STACK
                                                ;CLEAR ERRONEOUS ONE BITS FROM UPPER MASK
                                                ;CLEAR ERRONEOUS ONE BITS FROM DATA WORD
                                                ;SET DROPPED BITS

                                                ;HEADER
                                                ;PUT ECWRD ON THE STACK
                                                ;TYPE ECWRD
                                                ;TYPE 2 BLANKS
                                                ;PUT ECBAD0 ON THE STACK
                                                ;TYPE ECBAD0
                                                ;TYPE 2 BLANKS
                                                ;PUT @ECWRD ON THE STACK
                                                ;TYPE @ECWRD
                                                ;TYPE 2 BLANKS

                                                ;PRINT THE NEXT WORD ?
                                                ;BR IF NOT
                                                ;CR-LF
                                                ;PUT ECWRD1 ON THE STACK
                                                ;TYPE ECWRD1
                                                ;TYPE 2 BLANKS
                                                ;PUT ECBAD1 ON THE STACK
                                                ;TYPE ECBAD1
                                                ;TYPE 2 BLANKS
                                                ;PUT @ECWRD1 ON THE STACK
                                                ;TYPE @ECWRD1
                                                ;TYPE 2 BLANKS
                                                ;EXIT

                                                ;ERROR BURST WAS NOT TRANSFERED TO MEMORY
                                                ;CR-LF
                                                ;RETURN


                                                ;ROUTINE TO DISPLAY THE SECTOR WHICH GAVE THE HARD ERROR


                                                ;PRINT THE BAD SECTOR ?
                                                ;BR IF NOT
                                                ;PUT THE END ADDRESS INTO R1
                                                ;FIND THE BEGINNING OF THE SECTOR
                                                ;SUBTRACT THE WORDS NOT TRANSFERED
                                                ;RESTORE STACK
                                                ;EXIT--NO WORDS XFERRED
                                                ;MAKE THE UPPER DIVIDEND 0
```

```
                                                        ;DIVIDE THE WORDS XFERED BY THE SECTOR SIZE
342 015406  016046  000022          MOV     $SSEC(R0),-(SP)
343 015412  004737  031710          JSR     PC,$DIV         ;DIVIDE
344 015416  005716                  TST     (SP)            ;REMANDER = 0 ?
345 015420  001403                  BEQ     2$              ;BR IF IT IS - COMPLETE SECTOR TRANSFERED
346 015422  006316                  ASL     (SP)            ;CONVERT THE RESIDUAL SECTOR INTO BYTE COUNT
347 015424  161601                  SUB     (SP),R1         ;SUBTRACT IT FROM THE END ADDRESS
348 015426  000410                  BR      3$              ;FINISH THE SIZING
349 015430  162701  001000  2$:     SUB     #256.*2,R1      ;SUBTRACT FULL SECTOR FROM END ADDR (IN BYTES)
350 015434  122760  000005  000024  CMPB    #5,$CODF(R0)    ;WAS OPERATION READ HEADER & DATA ?
351 015442  001002                  BNE     3$              ;BR IF NOT
352 015444  162701  000004          SUB     #4,R1           ;SUBTRACT HEADER SIZE FROM ADDR
353 015450  062706  000004  3$:     ADD     #4,SP           ;RESTORE THE STACK POINTER
354 015454  104414  001203          DISPLY  ,$CRLF          ;CR-LF
355 015460  104414  075437          DISPLY  ,LIN11H         ;PRINT THE HEADER
356 015464  122760  000005  000024  CMPB    #5,$CODE(R0)    ;WAS OPERATION READ HEADER & DATA ?
357 015472  001021                  BNE     4$              ;BR IF NOT
358 015474  104414  075512          DISPLY  ,LIN11          ;TYPE 'ADDR    HEADER'
359 015500  010146                  MOV     R1,-(SP)        ;PUT THE ADDRESS ON THE STACK
360 015502  004737  023144          JSR     PC,LINOCT       ;TYPE THE ADDRESS
361 015506  104414  075553          DISPLY  ,BLNKS3         ;TYPE 3 BLANKS
362 015512  012146                  MOV     (R1)+,-(SP)     ;PUT WORD ON STACK
363 015514  004737  023144          JSR     PC,LINOCT       ;TYPE THE 1ST HEADER WORD
364 015520  104414  075555          DISPLY  ,BLNKS1         ;TYPE 1 BLANK
365 015524  012146                  MOV     (R1)+,-(SP)     ;PUT WORD ON STACK
366 015526  004737  023144          JSR     PC,LINOCT       ;TYPE THE 2ND HEADER WORD
367 015532  104414  001203          DISPLY  ,$CRLF          ;CR-LF
368
369 015536  104414  075533  4$:     DISPLY  ,LIN11A         ;TYPE 'ADDR    DATA'
370 015542  012702  000010  5$:     MOV     #8.,R2          ;8. DATA WORDS PER LINE
371 015546  010146                  MOV     R1,-(SP)        ;PUT THE ADDRESS ON THE STACK
372 015550  004737  023144          JSR     PC,LINOCT       ;TYPE THE ADDRESS
373 015554  104414  075554          DISPLY  ,BLNKS2         ;TYPE 2 BLANKS
374 015560  020160  002142  6$:     CMP     R1,$RMBA(R0)    ;PRINTED ALL THE SECTOR ?
375 015564  001412                  BEQ     7$              ;BR IF ALL PRINTED
376 015566  104414  075555          DISPLY  ,BLNKS1         ;TYPE 1 BLANK
377 015572  012146                  MOV     (R1)+,-(SP)     ;PUT THE DATA ON THE STACK
378 015574  004737  023144          JSR     PC,LINOCT       ;TYPE THE DATA
379 015600  005302                  DEC     R2              ;DECREMENT THE HORIZONTAL COUNT
380 015602  001366                  BNE     6$              ;BR IF NOT AT THE END OF THE LINE
381 015604  104414  001203          DISPLY  ,$CRLF          ;CR-LF
382 015610  000754                  BR      5$              ;RESTORE THE WORDS/LINE COUNT
383 015612  104414  001203  7$:     DISPLY  ,$CRLF          ;CR-LF
384 015616  104414  001203          DISPLY  ,$CRLF          ;CR-LF
385 015622  000207          8$:     RTS     PC              ;RETURN
386
387                                 ;ROUTINE TO DO AN RTC - DRIVE SELECTED IN R0
388                                 ;CALL:
389                                 ;       MOV     #DPB,R0         ;DPB ADDRESS
390                                 ;       JSR     PC,RTNCTR
391                                 ;       RETURN
392
393 015[24  111037  070322  RTNCTR: MOVB    (R0),GENDPB     ;MOVE THE DRIVE # TO THE GENERAL DPB
394 015[5[  112737  000117  070324          MOVB    #RTC,GENDPB+$COMND      ;COMMAND CODE
395 015630  004037  040714  1$:     JSR     R0,RM05         ;DRIVER ENTRANCE
396 015642  070322                  GENDPB                  ;DPB ADDRESS FOR COMMAND
397 015644  000774                  BR      1$              ;DRIVER DIDN'T ACCEPT COMMAND
398 015646  000207                  RTS     PC              ;RETURN
```

```
                                        ;ROUTINE TO DO A RECALIBRATE USING ACTIVE DPB
                                        ;CALL:
                                        ;       MOV     #DPB,R0         ;DPB ADDRESS
                                        ;       JSR     PC,RECALT
                                        ;       RETURN

015650  010037  015674          RECALT: MOV     R0,2$           ;LOAD THE DPB ADDRESS
015654  116060  002136  000027          MOVB    $RMCS1(R0),$PREVO(R0)   ;SAVE THE PREVIOUS COMMAND
015662  112760  000107  000002          MOVB    #RECAL,$COMND(R0)       ;LOAD THE NEW COMMAND
015670  004037  040714          1$:     JSR     R0,RM05         ;START THE RECALIBRATE
015674  000000                  2$:     .WORD   0               ;DPB ADDRESS
015676  000774                          BR      1$              ;DRIVER DIDN'T ACCEPT THE COMMAND
015700  005760  000016          3$:     TST     $STATUS(R0)     ;SEE IF FINISHED
015704  001775                          BEQ     3$              ;IF EQ NO
015706  004737  023216                  JSR     PC,READDR       ;DECREMENT THE ADDRESSES
015712  012660  000034                  MOV     (SP)+,$PREVA+2(R0)      ;MOVE THE CYLINDER ADDRESS
015716  112660  000033                  MOVB    (SP)+,$PREVA+1(R0)      ;MOVE THE TRACK ADDRESS
015722  112660  000032                  MOVB    (SP)+,$PREVA(R0)        ;MOVE THE SECTOR ADDRESS
015726  005060  000012                  CLR     $CYL(R0)        ;CLEAR THE CURRENT CYLINDER ADDRESS
015732  005060  000010                  CLR     $SEC(R0)        ;CLEAR THE CURRENT TRK/SEC ADDRESS
015736  000207                          RTS     PC              ;RETURN

                                        ;ROUTINE TO A RECAL WITH NO DPB ACTIVE
                                        ;CALL:
                                        ;       MOVB    #DRIVE,GENDPB   ;DRIVE ADDRESS
                                        ;       JSR     PC,RECALO
                                        ;       RETURN

015740  112737  000107  070324  RECALO: MOVB    #RECAL,GENDPB+$COMND    ;RELCALIBRATE COMMAND
015746  004037  040714          1$:     JSR     R0,RM05         ;DRIVER ENTRANCE
015752  070322                          GENDPB                  ;DPB ADDRESS FOR COMMAND
015754  000774                          BR      1$              ;DRIVER DIDN'T ACCEPT THE COMMAND
015756  005737  070340          2$:     TST     GENDPB+$TATUS   ;SEE IF FINISHED
015762  001775                          BEQ     2$              ;BR IF NOT FINISHED
015764  000207                          RTS     PC

                                        ;OFFSET THE DRIVE IN R0 (OFFSET CODE PRELOADED INTO 'RMOF')
                                        ;CALL:
                                        ;       MOVB    #OFFSET,GENDPB+$FMT     ;OFFSET CODE
                                        ;       MOV     #DPB,R0         ;DPB ADDRESS
                                        ;       JSR     PC,OFFST
                                        ;       RETURN

015766  111037  070322          OFFST:  MOVB    (R0),GENDPB     ;DRIVE # TO GENERAL DPB
015772  112737  000115  070324          MOVB    #OFFSET,GENDPB+$COMND   ;COMMAND
016000  004037  040714          1$:     JSR     R0,RM05         ;DRIVER ENTRANCE
016004  070322                          GENDPB                  ;DPB ADDRESS FOR COMMAND
016006  000774                          BR      1$              ;DRIVER DIDN'T ACCEPT COMMAND
016010  000207                          RTS     PC

                                        ;UTILITY READ HEADER ROUTINE
                                        ;CALL:
                                        ;       MOV     #DPB,R0         ;DPB ADDRESS
                                        ;       MOV     #SECTOR,-(SP)   ;SECTOR ADDRESS
                                        ;       MOV     #TRACK,-(SP)    ;TRACK ADDRESS
                                        ;       MOV     #CYLINDER,-(SP) ;CYLINDER ADDRESS
```

```
                                        ;       JSR     PC,READDR
                                        ;       RETURN

016012  116637  000004  070333  READHD: MOVB    4(SP),GENDPB+$TRK       ;TRACK ADDRESS
016020  116637  000006  070332          MOVB    6(SP),GENDPB+$SEC       ;SECTOR ADDRESS
016026  016637  000002  070334          MOV     2(SP),GENDPB+$CYL       ;CYLINDER ADDRESS
016034  111037  070322                  MOVB    (R0),GENDPB     ;DRIVE NUMBER
016040  112737  000173  070324          MOVB    #RDHD,GENDPB+$COMND     ;COMMAND
016046  012737  177776  070326          MOV     #-2,GENDPB+$WCNT        ;WORD CTR = 2
016054  004037  040714          1$:     JSR     R0,RM05         ;DRIVER ENTRANCE
016060  070322                          GENDPB                  ;DPB ADDRESS FOR COMMAND
016062  000774                          BR      1$              ;DRIVER DIDN'T ACCEPT COMMAND
016064  005737  070340          2$:     TST     GENDPB+$STATUS  ;FINISHED?
016070  001775                          BEQ     2$              ;BR IF NOT
016072  011666  000006                  MOV     (SP),6(SP)      ;ADJUST STACK FOR RETURN
016076  062706  000006                  ADD     #6,SP           ;ADJUST RETRUN POINTER
016102  000207                          RTS     PC              ;RETURN

                                        ;RETRY THE PRESENT OPERATION
                                        ;CALL:
                                        ;       MOV     #COUNT,RETRY    ;RETRY COUNT
                                        ;       JSR     PC,$RETRY
                                        ;       RETURN1                 ;RETRY UNSUCESSFUL
                                        ;       RETURN2                 ;SUCESSFUL RETRY
                                        ;                               ;NOTE: IF A DIFFERENT ERROR OCCURS DURING
                                        ;                               ;RETRY, THE ROUTINE EXITS TO 'ERPRC1'

016104  004737  017050          $RETRY: JSR     PC,GODRIV       ;RE-START COMMAND
016110  005760  000016          1$:     TST     $TATUS(R0)      ;COMMAND FINISHED?
016114  001775                          BEQ     1$              ;BR IF NOT
016116  100405                          BMI     2$              ;BR IF ERROR
016120  105237  001331                  INCB    RETRY+1         ;INCREMENT RETRY COUNT
016124  062716  000002                  ADD     #2,(SP)         ;INCREMENT RETURN
016130  000425                          BR      5$              ;GO TO EXIT
016132  032760  000200  000016  2$:     BIT     #BIT7,$TATUS(R0)        ;DID COMMAND TERMINATE NORMALLY ?
016140  001430                          BEQ     7$              ;BR IF NOT
016142  005737  001326                  TST     MASK            ;IS ERROR MASK 0 ?
016146  001004                          BNE     3$              ;BR IF NOT
016150  005760  002152                  TST     $RMER1(R0)      ;MAKE SURE THAT THE DRIVE ERROR REG IS CLEAR
016154  001014                          BNE     6$              ;BR IF NOT
016156  000404                          BR      4$              ;CONTINUE RETRY
016160  033760  001326  002152  3$:     BIT     MASK,$RMER1(R0) ;SAME ERROR?
016166  001407                          BEQ     6$              ;BR IF NOT
016170  105237  001331          4$:     INCB    RETRY+1         ;INCREMENT RETRY COUNT
016174  123737  001330  001331          CMPB    RETRY,RETRY+1   ;DONE ?
016202  001340                          BNE     $RETRY          ;BR IF NOT DONE
016204  000207                  5$:     RTS     PC              ;RETURN
016206  004737  023132          6$:     JSR     PC,LINE8        ;REPORT DIFFERENT ERROR
016212  004737  022710                  JSR     PC,LINE7        ;PRINT LINE 7
016216  005726                          TST     (SP)+           ;ADJUST STACK POINTER FOR DIRECT RETURN
016220  000207                          RTS     PC              ;RETURN
016222  104414  074732          7$:     DISPLY  ,LIN8M          ;'DIFFERENT ERROR DURING RETRY'
016226  000137  007254                  JMP     ERPRC1          ;REPORT THE ERROR
```

```
                                        ;ROUTINE TO UPDATE THE PERFORMANCE SUMMARY STATISTICS
                                        ;CALL:
                                        ;       MOV     #DPB,R0         ;DPB ADDRESS
                                        ;       JSR     PC,STATIS
                                        ;       RETURN

016232  032760  000300  000016  STATIS: BIT     #BIT07!BIT06,$STATUS(R0) ;CHECK FOR DATA TERMINATION
016240  001456                          BEQ     3$                      ;BR IF NOT DATA TERMINATION
016242  016037  002142  016400          MOV     $RMBA(R0),FACTOR        ;STORE THE FINAL BUFFER ADDRESS
016250  166037  000006  016400          SUB     $BJF(R0),FACTOR ;SUBTRACT THE INITIAL ADDRESS
016256  001447                          BEQ     3$              ;BR IF NO DATA TRANSFER
016260  006237  016400                  ASR     FACTOR          ;CONVERT TO A WORD COUNT

016264  122760  000002  000024          CMPB    #2,$CODE(R0)    ;SEE IF COMMAND WAS A WRITE
016272  001404                          BEQ     1$              ;BRANCH IF YES
016274  122760  000000  000024          CMPB    #0,$CODE(R0)    ;PRESENT OPERATION AN AUTO WRITE CHECK ?
016302  001012                          BNE     2$              ;BR IF NO
016304  063760  016400  000060  1$:     ADD     FACTOR,$WRITN(R0)       ;ADD WORDS WRITTEN DURING WRITE DATA
016312  005560  000062                  ADC     $WRITN+2(R0)    ;DID HIGH WORD OVFLO AFTER ADDING CARRY ?
016316  102004                          BVC     2$              ;BR IF NO
016320  005060  000062                  CLR     $WRITN+2(R0)    ;CLEAR HIGH WORD
016324  005260  000056                  INC     $WTOFL(R0)      ;AND COUNT WRITE OVERFLOW

016330  122760  000002  000024  2$:     CMPB    #2,$CODE(R0)    ;SEE IF COMMAND WAS A WRITE
016336  001417                          BEQ     3$              ;BRANCH IF YES
016340  063760  016400  000036          ADD     FACTOR,$ENDAT(R0)       ;END OF PASS DATA WORD COUNT
016346  005560  000040                  ADC     $ENDAT+2(R0)    ;ADD ANY CARRY
016352  063760  016400  000066          ADD     FACTOR,$READ(R0)        ;UPDATE THE READ WORD COUNT
016360  005560  000070                  ADC     $READ+2(R0)     ;DID HIGH WORD OVFLO AFTER ADDING CARRY ?
016364  102004                          BVC     3$              ;BR IF NO
016366  005060  000070                  CLR     $READ+2(R0)     ;CLEAR HIGH WORD
016372  005260  000064                  INC     $RDOFL(R0)      ;AND COUNT READ OVERFLOW
016376  000207                  3$:     RTS     PC

016400  000000                  FACTOR: .WORD   0               ;USED FOR WORDS TRANSFERED

                                        ;ROUTINE TO GET A BUFFER
                                        ;CALL:
                                        ;       MOV     #DPB,R0         ;DPB ADDRESS
                                        ;       CLR     -(SP)           ;CLEAR THE STACK
                                        ;       JSR     PC,GETBUF
                                        ;       RETURN                  ;BUFFER ADDRESS WILL BE ON THE STACK
                                        ;                               ;STACK WILL BE ZERO IF NO BUFFER AVAILABLE

016402  010146          GETBUF: MOV     R1,-(SP)        ;SAVE R1
016404  010246                  MOV     R2,-(SP)        ;SAVE R2
016406  010346                  MOV     R3,-(SP)        ;SAVE R3
016410  013702  001642          MOV     BUFTBL,R2       ;NUMBER OF SEPARATE BUFFERS
016414  001444                  BEQ     5$              ;BR IF NONE AVAILABLE
016416  012701  001644          MOV     #BUFTBL+2,R1    ;FIRST ADDRESS OF ALLOCATION TABLE
016422  026061  000020  000002  1$:     CMP     $WRDL(R0),2(R1) ;SEE IF THERE IS A BLOCK LARGE ENOUGH
016430  101405                  BLOS    2$              ;BRANCH IF IT IS
016432  005302                  DEC     R2              ;DECREMENT TABLE COUNT
016434  001434                  BEQ     5$              ;BR IF THROUGH TABLE
016436  062701  000004          ADD     #4,R1           ;INCREMENT TABLE POINTER
016442  000767                  BR      1$              ;CONTINUE LOOKING
016444  011166  000010          2$:     MOV     (R1),10(SP)     ;BUFFER ADDRESS TO STACK
```

```
 58 016450  166061  000020  000002          SUB     $WRDL(R0),2(R1) ;ADJUST BUFFER WRD CNT
 59 016456  001407                          BEQ     3$              ;BR IF DIFFERENCE IS ZERO
 60 016460  006360  000020                  ASL     $WRDL(R0)       ;CONVERT # WORDS TO BYTES
 61 016464  066011  000020                  ADD     $WRDL(R0),(R1)  ;MAKE NEW STARTING ADDRESS
 62 016470  006260  000020                  ASR     $WRDL(R0)       ;RETURN # BYTES TO WORDS
 63 016474  000414                          BR      5$              ;RETURN
 64 016476  005337  001642          3$:     DEC     BUFTBL          ;DECREMENT ENTRIES COUNT
 65 016502  001411                          BEQ     5$              ;BR IF ALLOCATION TABLE EMPTY
 66 016504  005302                          DEC     R2              ;DECREMENT TABLE COUNT
 67 016506  001407                          BEQ     5$              ;BR IF ITEM WERE LAST ENTRY
 68 016510  010103                          MOV     R1,R3           ;MOVE TABLE POINTER
 69 016512  062703  000004                  ADD     #4,R3           ;POINT TO NEXT ENTRY
 70 016516  012321          4$:     MOV     (R3)+,(R1)+     ;MOVE ITEMS
 71 016520  012321                          MOV     (R3)+,(R1)+
 72 016522  005302                          DEC     R2              ;DECREMENT TABLE COUNT
 73 016524  001374                          BNE     4$              ;CONTINUE IF NOT AT END OF TABLE
 74 016526  012603          5$:     MOV     (SP)+,R3        ;RESTORE R3
 75 016530  012602                          MOV     (SP)+,R2        ;RESTORE R2
 76 016532  012601                          MOV     (SP)+,R1        ;RESTORE R1
 77 016534  000207                          RTS     PC              ;RETURN
 78
 79
 80                                 ;ROUTINE TO PUT BUFFER BACK IN TABLE
 81                                 ;CALL:
 82                                 ;       MOV     #DPB,R0         ;DPB ADDRESS
 83                                 ;       JSR     PC,RELBUF
 84                                 ;       RETURN
 85
 86 016536  010146          RELBUF: MOV     R1,-(SP)        ;SAVE R1
 87 016540  010246                  MOV     R2,-(SP)        ;SAVE R2
 88 016542  010346                  MOV     R3,-(SP)        ;SAVE R3
 89 016544  010446                  MOV     R4,-(SP)        ;SAVE R4
 90 016546  010546                  MOV     R5,-(SP)        ;SAVE R5
 91 016550  012701  001644          MOV     #BUFTBL+2,R1    ;BEGINNING OF TABLE
 92 016554  013702  001642          MOV     BUFTBL,R2       ;ENTRY COUNT
 93 016560  001424                  BEQ     2$              ;BR IF EMPTY TABLE
 94 016562  016003  000110          MOV     $HLDWC(R0),R3   ;TRIAL ADDRESS
 95 016566  006303                  ASL     R3              ;CHANGE TO BYTE COUNT
 96 016570  066003  000006          ADD     $BUF(R0),R3     ;ADDRESS OF HIGHER ADJACENT BLOCK
 97 016574  021103          1$:     CMP     (R1),R3         ;UPPER ADJACENT BLOCK
 98 016576  001424                  BEQ     3$              ;BR IF YES
 99 016600  062701  000004          ADD     #4,R1           ;INCREMENT POINTER
100 016604  005302                  DEC     R2              ;DECREMENT ENTRY COUNT
101 016606  001372                  BNE     1$              ;CONTINUE SEARCHING
102 016610  016011  000006          MOV     $BUF(R0),(R1)   ;PUT THE BUFFER BLOCK INTO THE TABLE
103 016614  016061  000110  000002          MOV     $HLDWC(R0),2(R1)        ;BLOCK WRD CNT
104 016622  005237  001642          INC     BUFTBL          ;INCREMENT ENTRY COUNT
105 016626  005202                  INC     R2              ;INCREMENT R2 FOR USE LATER
106 016630  000414                  BR      4$              ;SEE IF A LOWER ADJACENT BLOCK IS IN THE TABLE
107 016632  016021  000006          2$:     MOV     $BUF(R0),(R1)+  ;BLOCK ADDRESS TO TABLE
108 016636  016021  000110                  MOV     $HLDWC(R0),(R1)+        ;WRD CNT TO TABLE
109 016642  005237  001642                  INC     BUFTBL          ;INCREMENT ENTRY COUNT
110 016646  000443                          BR      8$              ;EXIT
111 016650  016011  000006          3$:     MOV     $BUF(R0),(R1)   ;RELEASED BUFFER IS LOWER ADJACENT
112 016654  066061  000110  000002          ADD     $HLDWC(R0),2(R1)        ;INCREMENTED WRD CNT
113 016662  010246          4$:     MOV     R2,-(SP)        ;SAVE R2
114 016664  013702  001642                  MOV     BUFTBL,R2       ;ENTRY COUNT
```

```
                                                MOV     #BUFTBL+2,R5    ;BEGINNING OF TABLE
                                        5$:     MOV     2(R5),R4        ;BLOCK SIZE (IN WORDS)
                                                ASL     R4              ;CHANGE TO BYTE COUNT
                                                ADD     (R5),R4         ;ADD BLOCK BEGINNING ADDRESS
                                                CMP     R4,(R1)         ;R1 STILL POINTS TO INSERTED ENTRY
                                                BEQ     6$              ;LOWER ADJACENT IN TABLE
                                                ADD     #4,R5           ;INCREMENT POINTER
                                                DEC     R2              ;DECREMENT ENTRY COUNT
                                                BNE     5$              ;CONTINUE LOOKING
                                                TST     (SP)+           ;RESTORE STACK POINTER
                                                BR      8$              ;END
                                        6$:     MOV     (SP)+,R2        ;RESTORE R2
                                                ADD     2(R1),2(R5)     ;INCREMENT LOWER BLOCK LENGTH
                                                DEC     BUFTBL          ;DECREMENT ENTRY COUNT
                                                MOV     R1,R5           ;GET READY TO COMPRESS
                                                ADD     #4,R5           ;INCREMENT TO NEXT ENTRY
                                        7$:     MOV     (R5)+,(R1)+     ;COMPRESS TABLE
                                                MOV     (R5)+,(R1)+     ;MOVE SIZE FIELD DOWN
                                                DEC     R2              ;DECREMENT ENTRY COUNT
                                                BNE     7$              ;BR IF NOT FINISHED
                                        8$:     MOV     (SP)+,R5        ;RESTORE R5
                                                MOV     (SP)+,R4        ;RESTORE R4
                                                MOV     (SP)+,R3        ;RESTORE R3
                                                MOV     (SP)+,R2        ;RESTORE R2
                                                MOV     (SP)+,R1        ;RESTORE R1
                                                RTS     PC              ;RETURN

                                        ;FILL THE ASSIGNED BUFFER (IF WRITE OR WRITE CHECK COMMAND)
                                        ;CALL:
                                        ;       MOV     #DPB,R0         ;DPB ADDRESS
                                        ;       MOV     #BUFADR,$BUF(R0)        ;LOAD BUFFER ADDRESS INTO THE DPB
                                        ;       MOVB    #PATTERN,$PATTC(R0)     ;PATTERN CODE
                                        ;       JSR     PC,FILBUF
                                        ;       RETURN

                                        FILBUF: SAVREG                  ;SAVE THE REGISTERS
                                                BITB    #4,$CODE(R0)    ;SEE IF READ COMMAND
                                                BNE     4$              ;BR IF READ
                                        1$:     MOV     $BUF(R0),R1     ;BUFFER ADDRESS
                                                MOV     $WRDL(R0),R2    ;POSITIVE WORD COUNT
                                                MOVB    $PATTC(R0),R4   ;RELATIVE PATTERN ADDRESS
                                        2$:     MOV     STNDAT(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
```

```
                172
017050 010046           GODRIV: MOV     R0,-(SP)        ;SAVE R0
017052 010037 017062            MOV     R0,2$           ;CURRENT DPB ADDRESS
017056 004037 040714    1$:     JSR     R0,RM05         ;CALL THE DRIVE HANDLER
017062 000000           2$:     .WORD   0               ;DRIVE BLOCK ADDRESS GOES HERE
017064 000000                   HALT                    ;DRIVER REJECTED REQUEST
017066 012600                   MOV     (SP)+,R0        ;RESTORE R0
017070 062760 000001 000046     ADD     #1,$OPERC(R0)   ;INCREMENT THE OPERATION COUNT
017076 005560 000050            ADC     $OPERC+2(R0)
017102 026060 000034 000012     CMP     $PREVA+2(R0),$CYL(R0)   ;DID COMMAND REQUIRE A CYLINDER CHANGE ?
017110 001412                   BEQ     3$              ;BR IF NO
017112 062760 000001 000042     ADD     #1,$ENDSK(R0)   ;INCREMENT END OF PASS SEEK COUNT
017120 005560 000044            ADC     $ENDSK+2(R0)    ;ADD ANY CARRY
017124 062760 000001 000052     ADD     #1,$POSIT(R0)   ;INCREMENT SEEK COUNT
017132 005560 000054            ADC     $POSIT+2(R0)    ;ADD ANY CARRY
017136 000207           3$:     RTS     PC
```

```
                                        ;ROUTINE TO SETUP PARAMETERS FOR A SEQUENTIAL READ OR WRITE OF THE DISK
                                        ;CALL:
                                        ;       MOV     #DPB,R0         ;DPB ADDRESS
                                        ;-------------------------------;
                                        ;       MOV     #-2,$PACK(R0)   ;'WRITE PACK' & 'TEST' FLAG
                                        ;  OR
                                        ;       MOV     #-1,$PACK(R0)   ;'WRITE PACK' FLAG
                                        ;  OR
                                        ;       MOV     #1,$PACK(R0)    ;'READ PACK' FLAG
                                        ;-------------------------------;
                                        ;       JSR     PC,WRTPK        ;CALL READ OR WRITE PACK
                                        ;       RETURN

017140  004737  027346          WRTPK:  JSR     PC,GETLMT       ;GET ADDRESS LIMITS
017144  005760  000050                  TST     $OPERC+2(R0)    ;IS THIS THE FIRST OPERATION ?
017150  001003                          BNE     1$              ;BR IF NO
017152  005760  000046                  TST     $OPERC(R0)      ;IS THIS THE FIRST OPERATION ?
017156  001452                          BEQ     2$              ;BR IF YES

017160  116060  002136  000027  1$:     MOVB    $RMCS1(R0),$PREV0(R0)   ;SAVE CURRENT PARAMETERS
017166  016060  000010  000032          MOV     $SEC(R0),$PREVA(R0)     ;SAVE PREVIOUS TRACK/SECTOR ADDRESS
017174  016060  000012  000034          MOV     $CYL(R0),$PREVA+2(R0)   ;SAVE PREVIOUS CYLINDER ADDRESS
017202  016060  002144  000010          MOV     $RMDA(R0),$SEC(R0)      ;CURRENT SECTOR & TRACK ADDRESS
017210  016060  002172  000012          MOV     $RMDC(R0),$CYL(R0)      ;CURRENT CYLINDER ADDRESS

017216  012704  000010                  MOV     #$SEC,R4        ;GET INDEX TO SECTOR ADDR STORAGE IN DPB
017222  004737  020164                  JSR     PC,CKLMTS       ;GO CHECK DISK ADDRESS LIMITS
017226  000426                          BR      2$              ;BR IF NOT AT END OF SEQUENTIAL ADDRESSING
017230  116060  000136  000010          MOVB    MINSEC(R0),$SEC(R0)     ;RESET SECTOR ADDRESS
017236  116060  000132  000011          MOVB    MINTRK(R0),$TRK(R0)     ;RESET TRACK ADDRESS
017244  016060  000126  000012          MOV     MINCYL(R0),$CYL(R0)     ;RESET CYLINDER ADDRESS
017252  112760  000004  000024          MOVB    #4,$CODE(R0)    ;SET CODE TO READ DATA
017260  122760  177776  000026          CMPB    #-2,$PACK(R0)   ;WAS WRITE DATA PACK IN PROGESS ?
017266  001467                          BEQ     7$              ;BR IF YES (START TESTING)
017270  004737  031240                  JSR     PC,EOP2         ;DROP THE DRIVE (NORMAL TERMINATION)
017274  032777  000020  161652          BIT     #SW04,@SWR      ;IS SWITCH 4 SET ?
017302  001465                          BEQ     8$              ;BR IF NO

017304  012704  000010          2$:     MOV     #$SEC,R4        ;GET INDEX TO SECTOR STORAGE
017310  013705  001450                  MOV     WRDCNT,R5       ;WORD COUNT IS MAXIMUM
017314  004737  020316                  JSR     PC,CHKWC        ;CHECK WORD COUNT FOR MAXCYL/MAXTRK
017320  010560  000020                  MOV     R5,$WRDL(R0)    ;GET WORD COUNT
017324  042760  000377  000020          BIC     #377,$WRDL(R0)  ;IS IT LESS THAN ONE SECTOR WORD COUNT ?
017332  001002                          BNE     3$              ;NO
017334  105260  000021                  INCB    $WRDL+1(R0)     ;SET TO ONE SECTOR
017340  01606C  000020  000004  3$:     MOV     $WRDL(R0),$WCNT(R0)     ;STORE FOR 2'S COMPLEMENT WORD
017346  016060  000020  000110          MOV     $WRDL(R0),$HLDWC(R0)    ;HOLD WORD FOR 'RELBUF' ROUTINE
017354  005460  000004                  NEG     $WCNT(R0)       ;CHANGE WORD COUNT TO 2'S COMPLEMENT
017360  012760  000400  000022          MOV     #256.,$SSEC(R0) ;SECTOR SIZE FOR READ

017366  105760  000026                  TSTB    $PACK(R0)       ;READ OR WRITE PACK ?
017372  100407                          BMI     5$              ;BR IF WRITE
017374  112760  000004  000024  4$:     MOVB    #4,$CODE(R0)    ;CODE FOR READ DATA
017402  112760  000171  000002          MOVB    #RDDAT,$COMND(R0)       ;DRIVE CODE FOR OPERATION
017410  000412                          BR      6$              ;SET UP FOR EXIT
017412  112760  000002  000024  5$:     MOVB    #2,$CODE(R0)    ;CODE FOR WRTDAT
017420  112760  000161  000002          MOVB    #WRTDAT,$COMND(R0)      ;OP CODE
```

```
     58 017426  004737  020122                    JSR     PC,GETPAT       ;GET PATTERN CODE
     59 017432  110560  000030                    MOVB    R5,$PATTC(R0)   ;PATTERN CODE
     60 017436  012760  177777  000120  6$:       MOV     #-1,$NEXT(R0)   ;SET PARAMETERS SELECTED INDICATOR
     61 017444  000207                            RTS     PC              ;RETURN
     62
     63 017446  005037  001320           7$:      CLR     PACK            ;SET 'TEST' FLAG
     64 017452  105060  000026                    CLRB    $PACK(R0)       ;SET DPB 'TEST' FLAG
     65 017456  005060  000120           8$:      CLR     $NEXT(R0)       ;CLEAR 'PARAMETER SELECTED' INDICATOR
     66 017462  005726                            TST     (SP)+           ;CLEAR STACK LEVEL
     67 017464  000137  006142                    JMP     MAIN            ;JUMP TO MAIN BACKGROUND LOOP
```

```
                                        ;GENERATE PARAMETERS FOR THE OPERATION
                                        ;CALL:
                                        ;       MOV     #DPB,R0         ;DPB ADDRESS
                                        ;       JSR     PC,GENPAR
                                        ;       RETURN

017470  004737  027346          GENPAR: JSR     PC,GETLMT       ;GET ADDRESS LIMITS
017474  004737  036724                  JSR     PC,$RAND        ;CYCLE THE RANDOM NUMBER GENERATOR
017500  032777  000001  161446          BIT     #SWO,@SWR       ;SEE IF SWO SET
017506  001012                          BNE     1$              ;BR IF SET - READ ONLY
017510  012705  000010                  MOV     #8.,R5          ;READ/WRITE SELECTION DIVISOR
017514  004737  031664                  JSR     PC,GETREM       ;GET SELECTION VALUE
017520  020537  001464                  CMP     R5,RATIO        ;DETERMINE IF READ OR WRITE
017524  103003                          BHIS    1$              ;BR IF READ
017526  012705  000002                  MOV     #2,R5           ;SELECT WRITE DATA COMMAND
017532  000407                          BR      2$              ;SELECT ADDRESS

017534  013705  037024          1$:     MOV     $LONUM,R5       ;SELECT READ OPERATION CODE
017540  000305                          SWAB    R5              ;SWAP BYTES IN R5
017542  042705  177776                  BIC     #^C1,R5         ;MASK OUT ALL BUT BIT 0
017546  062705  000004                  ADD     #4,R5           ;TABLE OFFSET FOR READ CODE
017552  110560  000112          2$:     MOVB    R5,$NCODE(R0)   ;COMMAND SELECTION CODE TO CONTROL BLOCK
017556  016060  002144  000114          MOV     $RMDA(R0),$NSEC(R0)     ;SECTOR AND TRACK
017564  016060  002172  000116          MOV     $RMDC(R0),$NCYL(R0)     ;CYLINDER NUMBER

                                .ENABL  LSB

017572  005737  001474          THEAD:  TST     RANDOM          ;ENABLE RANDOM ADDRESS SELECT ?
017576  001426                          BEQ     RANCYL          ;YES
017600  005760  000050                  TST     $OPERC+2(R0)    ;IS THIS THE FIRST OPERATION ?
017604  001003                          BNE     THEAD1          ;BR IF NO
017606  005760  000046                  TST     $OPERC(R0)      ;IS THIS THE FIRST OPERATION ?
017612  001405                          BEQ     1$              ;BR IF YES

017614  012704  000114          THEAD1: MOV     #$NSEC,R4       ;GET INDEX TO SECTOR ADDR STORAGE IN DPB
017620  004737  020164                  JSR     PC,CKLMTS       ;GO CHECK DISK ADDRESS LIMITS
017624  000411                          BR      2$              ;BR IF NOT AT END OF SEQUENTIAL ADDRESSING
017626  116060  000136  000114  1$:     MOVB    MINSEC(R0),$NSEC(R0)    ;RESET SECTOR ADDRESS
017634  116060  000132  000115          MOVB    MINTRK(R0),$NTRK(R0)    ;RESET TRACK ADDRESS
017642  016060  000126  000116          MOV     MINCYL(R0),$NCYL(R0)    ;RESET CYLINDER ADDRESS
017650  000137  020000          2$:     JMP     RANSIZ          ;GO CHECK FOR RANDOM WORD SIZE

                                .DSABL  LSB
```

```
                                        ;GENERATE A RANDOM CYLINDER ADDRESS BETWEEN VALUES 'MINCYL' & 'MAXCYL'

017654  016005  000124        RANCYL: MOV     MAXCYL(R0),R5   ;GET MAXIMUM CYLINDER ADDRESS
017660  026005  000126                CMP     MINCYL(R0),R5   ;'MINCYL' AND 'MAXCYL' THE SAME ?
017664  001407                      BEQ     1$              ;BR IF THEY ARE
017666  166005  000126              SUB     MINCYL(R0),R5   ;GET NUMBER OF ALLOWABLE CYLINDERS
017672  005205                      INC     R5              ;INCREMENT DIFFERENCE TO USE AS DIVISOR
017674  004737  031664              JSR     PC,GETREM       ;GET THE RANDOM AUGMENT
017700  066005  000126              ADD     MINCYL(R0),R5   ;NEW CYLINDER ADDRESS
017704  010560  000116        1$:   MOV     R5,$NCYL(R0)    ;STORE CYLINDER ADDRESS IN DPB

                                        ;GENERATE A RANDOM TRACK ADDRESS BETWEEN VALUES 'MINTRK' & 'MAXTRK'

017710  016005  000130        RANTRK: MOV     MAXTRK(R0),R5   ;GET MAXIMUM TRACK ADDRESS
017714  026005  000132                CMP     MINTRK(R0),R5   ;'MINTRK' AND 'MAXTRK' THE SAME ?
017720  001407                      BEQ     1$              ;BR IF THEY ARE
017722  166005  000132              SUB     MINTRK(R0),R5   ;GET NUMBER OF ALLOWABLE TRACKS
017726  005205                      INC     R5              ;INCREMENT DIFFERENCE TO USE AS DIVISOR
017730  004737  031664              JSR     PC,GETREM       ;GET THE RANDOM AUGMENT
017734  066005  000132              ADD     MINTRK(R0),R5   ;NEW TRACK ADDRESS
017740  110560  000115        1$:   MOVB    R5,$NTRK(R0)    ;STORE TRACK ADDRESS IN DPB

                                        ;GENERATE A RANDOM SECTOR ADDRESS BETWEEN VALUES 'MINSEC' & 'MAXSEC'

017744  016005  000134        RANSEC: MOV     MAXSEC(R0),R5   ;GET MAXIMUM SECTOR ADDRESS
017750  026005  000136                CMP     MINSEC(R0),R5   ;'MINSEC' AND 'MAXSEC' THE SAME ?
017754  001407                      BEQ     1$              ;BR IF THEY ARE
017756  166005  000136              SUB     MINSEC(R0),R5   ;GET NUMBER OF ALLOWABLE SECTORS
017762  005205                      INC     R5              ;INCREMENT DIFFERENCE TO USE AS DIVISOR
017764  004737  031664              JSR     PC,GETREM       ;GET THE RANDOM AUGMENT
017770  066005  000136              ADD     MINSEC(R0),R5   ;NEW SECTOR ADDRESS
017774  110560  000114        1$:   MOVB    R5,$NSEC(R0)    ;STORE SECTOR ADDRESS IN DPB

                                        ;GENERATE A RANDOM BUFFER LENGTH BETWEEN 6 & THE VALUE IN 'WRDCNT'

020000  013705  001450        RANSIZ: MOV     WRDCNT,R5       ;GET MAX WORD COUNT
020004  005737  001462                TST     RANDWC          ;SELECT A RANDOM WORD COUNT ?
020010  001011                      BNE     2$              ;BR IF NOT
020012  005205                      INC     R5              ;INCREMENT THE MAXIMUM WRD CNT
020014  004737  031664              JSR     PC,GETREM       ;DIVIDE BY MAX VALUE
020020  020527  000006              CMP     R5,#6           ;WORD COUNT LESS THAN 6 ?
020024  002003                      BGE     2$              ;BR IF NO
020026  004737  036724        1$:   JSR     PC,$RAND        ;CYCLE THE RANDOM NUMBER GENERATOR
020032  000762                      BR      RANSIZ

020034  012704  000114        2$:   MOV     #$NSEC,R4       ;GET INDEX TO SECTOR STORAGE
020040  004737  020316              JSR     PC,CHKWC        ;SEE IF WORD COUNT IS TOO LARGE TO FIT
                                                            ;IN REMAINDER OF TRACK. IF SO, THEN ADJUST
                                                            ;WORD COUNT TO FIT ON TRACK.
020044  122760  000002  000112  3$:   CMPB    #2,$NCODE(R0)   ;WRITE OPERATION ?
020052  001005                      BNE     4$              ;BR IF NO
020054  042705  000377              BIC     #377,R5         ;WRITTING PARTIAL SECTOR ?
020060  001002                      BNE     4$              ;BR IF NO, ELSE,
020062  012705  000400              MOV     #256.,R5        ;WRITE AT LEAST ONE SECTOR
020066  010560  000020        4$:   MOV     R5,$WRDL(R0)    ;WORD COUNT

                                        ;GET A RANDOM PATTERN NUMBER
```

```
                58
020072  122760  000002  000112  RANPAT: CMPB    #2,$NCODE(R0)   ;WRITE OPERATION ?
020100  001004                          BNE     RANXIT          ;BR IF NO
020102  004737  020122                  JSR     PC,GETPAT       ;GET PATTERN CODE
020106  110560  000113                  MOVB    R5,$NPATC(R0)   ;MOVE PATTERN CODE TO CONTROL BLOCK
020112  012760  177777  000120  RANXIT: MOV     #-1,$NEXT(R0)   ;SET PARAMETERS SELECTED INDICATOR
020120  000207                          RTS     PC              ;RETURN
                65
                66                      ;ROUTINE TO SELECT A PATTERN
                67
020122  012705  000020          GETPAT: MOV     #16.,R5         ;SELECT PATTERN
020126  005737  001460                  TST     PATTERN         ;ENABLE RANDOM PATTERN SELECTION ?
020132  001403                          BEQ     1$              ;YES
020134  013705  001460                  MOV     PATTERN,R5      ;USE INDEXED PATTERN
020140  000407                          BR      2$              ;NO
020142  004737  031664          1$:     JSR     PC,GETREM       ;GET CODE
020146  005705                          TST     R5              ;WAS PATTERN ZERO SELECTED ?
020150  001003                          BNE     2$              ;BR IF NOT ZERO
020152  004737  036724                  JSR     PC,$RAND        ;CYCLE THE RANDOM NUMBER GENERATOR
020156  000761                          BR      GETPAT          ;TRY AGAIN
020160  006305          2$:     ASL     R5              ;MAKE CODE INTO TABLE INDEX
020162  000207                          RTS     PC
```

```
                                        ;THIS ROUTINE IS USED TO CHECK ADDRESS LIMITS BEFORE THE NEXT COMMAND
                                        ;IS PERFORMED. ALSO, IT WILL CHECK FOR MAXIMUM ADDRESS LIMITS TO LOOK
                                        ;FOR AN END TO THE SEQUENTIAL ADDRESSING.
                                        ;CALL:
                                        ;       MOV     #DPB,R0         ;DPB ADDRESS
                                        ;       MOV     #POINTER,R4     ;POINTER TO SECTOR STORAGE ($SEC OR $NSEC) IN DPB
                                        ;       JSR     PC,CKLMTS       ;CALL ADDRESS LIMITS ROUTINE
                                        ;       BR      ???             ;RETURN HERE IF NOT END OF SEQUENTIAL ADDRESSING
                                        ;       ------                  ;ELSE, RETURN HERE TO RESET DISK ADDRESS
                                        ;
                                        ;R0 = DPB ADDRESS BEFORE CALLING THE ROUTINE
                                        ;R4 = POINTER TO SECTOR STORAGE BEFORE CALLING THE ROUTINE

020164  060004                 CKLMTS: ADD     R0,R4           ;POINT TO SECTOR STORAGE POINT IN DPB
020166  026460  000002  000126          CMP     2(R4),MINCYL(R0)        ;IS CYLINDER ADDRESS BELOW MIN. ?
020174  002003                         BGE     1$              ;BR IF NO
020176  016064  000126  000002          MOV     MINCYL(R0),2(R4)        ;RESET CYLINDER TO MIN.
020204  126460  000001  000132  1$:     CMPB    1(R4),MINTRK(R0)        ;IS TRACK ADDRESS BELOW MIN. ?
020212  002003                         BGE     2$              ;BR IF NO
020214  116064  000132  000001          MOVB    MINTRK(R0),1(R4)        ;RESET TRACK TO MIN.
020222  121460  000136          2$:     CMPB    (R4),MINSEC(R0) ;IS SECTOR ADDRESS BELOW MIN. ?
020226  002002                         BGE     3$              ;BR IF NO
020230  116014  000136                 MOVB    MINSEC(R0),(R4) ;RESET SECTOR TO MIN.

                                        ;LOOK FOR MAXIMUM LIMITS AND END OF SEQUENTIAL ADDRESSING

020234  121460  000134          3$:     CMPB    (R4),MAXSEC(R0) ;IS SECTOR ADDRESS AT MAXIMUM ?
020240  003404                         BLE     4$              ;BR IF NO
020242  116014  000136                 MOVB    MINSEC(R0),(R4) ;RESET SECTOR ADDRESS
020246  105264  000001                 INCB    1(R4)           ;INCREMENT TO NEXT TRACK ADDRESS
020252  126460  000001  000130  4$:     CMPB    1(R4),MAXTRK(R0)        ;IS TRACK ADDRESS OVER MAXIMUM ?
020260  003407                         BLE     5$              ;BR IF NO
020262  116014  000136                 MOVB    MINSEC(R0),(R4) ;RESET SECTOR ADDRESS
020266  116064  000132  000001          MOVB    MINTRK(R0),1(R4)        ;RESET TRACK ADDRESS
020274  005264  000002                 INC     2(R4)           ;INCREMENT TO NEXT CYLINDER ADDRESS
020300  026460  000002  000124  5$:     CMP     2(R4),MAXCYL(R0)        ;IS CYLINDER ADDRESS OVER MAXIMUM ?
020306  003402                         BLE     6$              ;BR IF NO
020310  062716  000002                 ADD     #2,(SP)         ;ADJUST RETURN TO RESET DISK ADDRESS PARAMETERS
020314  000207                 6$.     RTS     PC              ;RETURN
```

```
                                        ;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

020316  060004                 CHKWC:  ADD     R0,R4           ;POINT TO SECTOR STORAGE POINT IN DPB
020320  105760  000136                  TSTB    MINSEC(R0)      ;ALLOW SPIRAL RD/WRT ?
020324  001023                          BNE     2$              ;BR IF NO
020326  126037  000134  001424          CMPB    MAXSEC(R0),SECLMT       ;ALLOW SPIRAL RD/WRT ?
020334  001017                          BNE     2$              ;BR IF NO
020336  105760  000132                  TSTB    MINTRK(R0)      ;ALLOW SPIRAL RD/WRT ?
020342  001010                          BNE     1$              ;BR IF NO
020344  126037  000130  001426          CMPB    MAXTRK(R0),TRKLMT       ;ALLOW SPIRAL RD/WRT ?
020352  001004                          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
020354  026064  000124  000002          CMP     MAXCYL(R0),2(R4)        ;ON MAXIMUM CYLINDER ?
020362  001021                          BNE     4$              ;BR IF NO
020364  126064  000130  000001  1$:     CMPB    MAXTRK(R0),1(R4)        ;ON MAXIMUM TRACK ?
020372  001015                          BNE     4$              ;BR IF NO
020374  111404                  2$:     MOVB    (R4),R4         ;GET STARTING SECTOR ADDRESS
020376  016046  000134                  MOV     MAXSEC(R0),-(SP)        ;GET MAXIMUM SECTOR
020402  160416                          SUB     R4,(SP)         ;GET NUMBER SECTORS TO BE XFERD
020404  005004                          CLR     R4              ;CLEAR R4
020406  062704  000400          3$:     ADD     #256.,R4        ;ADD 1 SECTOR OF WORDS TO R4
020412  005316                          DEC     (SP)            ;DONE ALL SECTORS YET ?
020414  002374                          BGE     3$              ;BR IF NO
020416  005726                          TST     (SP)+           ;RESTORE STACK
020420  020504                          CMP     R5,R4           ;TOO MANY WORDS FOR TRACK ?
020422  003401                          BLE     4$              ;BR IF NO
020424  010405                          MOV     R4,R5           ;YES, CHANGE WORD COUNT
020426  000207                  4$:     RTS     PC              ;RETURN
```

```
                                        ;ROUTINE TO GET THE PREVIOUSLY SELECTED PAPAMETER VALUES
                                        ;CALL:
                                        ;       MOV     #DPB,R0         ;DPB ADDRESS
                                        ;       JSR     PC,GENPAR       ;GENERATE THE PARAMETERS
                                        ;       JSR     PC,LODPAR       ;LOAD THE PARAMETERS JUST GENERATED
                                        ;       RETURN

020430  010546                  LODPAR: MOV     R5,-(SP)        ;SAVE R5
020432  105760  000026                  TSTB    $PACK(R0)       ;'R' OR 'W' COMMAND FOR THE DRIVE ?
020436  001106                          BNE     4$              ;BR IF YES
020440  116060  002136  000027          MOVB    $RMCS1(R0),$PREVO(R0)   ;SAVE CURRENT PARAMETERS
020446  142760  177701  000027          BICB    #^C76,$PREVO(R0)        ;STRIP GO,AND IE BITS
020454  132760  000006  000112          BITB    #6,$NCODE(R0)   ;SEE IF NEXT OPERATION IS READ OR WRITE
020462  001007                          BNE     1$              ;B, IF EITHER
020464  016060  000012  000034          MOV     $CYL(R0),$PREVA+2(R0)   ;SAVE STARTING CYLINDER
020472  016060  000010  000032          MOV     $SEC(R0),$PREVA(R0)     ;SAVE STARTING SECTOR AND TRACK
020500  000410                          BR      2$
020502  004737  023216          1$:     JSR     PC,READDR       ;GET THE DECREMENTED SECTOR AND TRACK ADDRESSES
020506  012660  000034                  MOV     (SP)+,$PREVA+2(R0)      ;CYLINDER ADDRESS
020512  112660  000033                  MOVB    (SP)+,$PREVA+1(R0)      ;TRACK ADDRESS
020516  112660  000032                  MOVB    (SP)+,$PREVA(R0)        ;SECTOR ADDRESS

020522  032777  000100  160424  2$:     BIT     #SW06,@SWR      ;SWITCH 6 SET ?
020530  001051                          BNE     4$              ;BR IF SET
020532  116060  000112  000024          MOVB    $NCODE(R0),$CODE(R0)    ;LOGICAL CODE FOR OPERATION
020540  116005  000112                  MOVB    $NCODE(R0),R5   ;LOAD R5 FOR USE AS TABLE INDEX
020544  116560  002064  000002          MOVB    COMTBL(R5),$COMND(R0)   ;COMMAND CODE
020552  122760  000151  000002          CMPB    #WCKD,$COMND(R0)        ;IS NEW COMMAND A WRITE CHECK DATA ?
020560  001012                          BNE     3$              ;BR IF NO
020562  122760  000060  000027          CMPB    #60,$PREVO(R0)  ;WAS PREVIOUS COMMAND A WRITE DATA ?
020570  001431                          BEQ     4$              ;BR IF YES
020572  112760  000171  000002          MOVB    #RDDAT,$COMND(R0)       ;CHANGE WRITE CHECK TO READ DATA COMMAND
020600  112760  000004  000024          MOVB    #4,$CODE(R0)    ;CODE NUMBER CHANGED TO READ DATA

020606  116060  000113  000030  3$:     MOVB    $NPATC(R0),$PATTC(R0)   ;PATTERN CODE
020614  016060  000114  000010          MOV     $NSEC(R0),$SEC(R0)      ;TRACK AND SECTOR ADDRESSES
020622  016060  000116  000012          MOV     $NCYL(R0),$CYL(R0)      ;CYLINDER ADDRESS
020630  012760  000400  000022          MOV     #256.,$SSEC(R0) ;INITIAL VALUE OF SECTOR SIZE
020636  132760  000001  000024          BITB    #1,$CODE(R0)    ;HEADER OPERATION ?
020644  001403                          BEQ     4$              ;BR IF NOT
020646  062760  000002  000022          ADD     #2,$SSEC(R0)    ;ADD HEADER SIZE
020654  016060  000020  000004  4$:     MOV     $WRDL(R0),$WCNT(R0)     ;GET WORD COUNT AND
020662  016060  000020  000110          MOV     $WRDL(R0),$HLDWC(R0)    ;HOLD WORD FOR 'RELBUF' ROUTINE
020670  005460  000004                  NEG     $WCNT(R0)       ;MAKE IT 2'S COMPLEMENT
020674  012605                          MOV     (SP)+,R5        ;RESTORE R5
020676  000207                          RTS     PC              ;RETURN
```

```
                                        ;ROUTINE TO COMPRESS A LIST
                                        ;CALL:
                                        ;       MOV     #ADDRS,R1       ;COMPRESS LIST STARTING AT THIS ADDRESS
                                        ;       JSR     PC,CMPRES
                                        ;       RETURN
                                        
020700  016111  000002          CMPRES: MOV     2(R1),(R1)      ;COMPRESS THE TABLE IN R1
020704  001402                          BEQ     1$              ;BR WHEN ZERO FOUND
020706  005721                          TST     (R1)+           ;INCREMENT R1
020710  000773                          BR      CMPRES          ;CONTINUE COMPRESSING TABLE
020712  000207                  1$:     RTS     PC              ;RETURN

                                        ;ROUTINE TO DETERMINE IF THE ERROR IS AT A LOCATION ON THE DISK, DEFINED
                                        ;IN THE BAD SECTOR TABLE FOR THE DRIVE.
                                        ;CALL:
                                        ;       JSR     PC,SPOTCK
                                        ;       RETURN1                 ;ERROR AT AN ADDRESS IN TABLE
                                        ;       RETURN2                 ;NO TABLE ENTRY FOR ERROR ADDRESS OR
                                        ;                               ;PARAMETER 'MESSAGE' IS 0

020714                          SPOTCK:
020714  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
020716  012701  000144                  MOV     #$BDSEC,R1      ;INCREMENT FOR BAD SECTOR TABLE
020722  060001                          ADD     R0,R1           ;ADD THE BLOCK'S STARTING ADDRESS
020724  004737  023216          1$:     JSR     PC,READDR       ;DECREMENT THE SECTOR/TRACK ADDRESS
020730  021126                          CMP     (R1),(SP)+      ;ON THE SAME CYLINDER ?
020732  001023                          BNE     5$              ;BRANCH IF NOT
020734  122761  177777  000003          CMPB    #-1,3(R1)       ;ALL BAD TRACKS ?
020742  001002                          BNE     2$              ;BR IF NO
020744  005726                          TST     (SP)+           ;ADJUST STACK AND
020746  000403                          BR      3$              ;GO CHECK SECTORS
020750  122661  000003          2$:     CMPB    (SP)+,3(R1)     ;COMPARE THE TRACK ADDRESS
020754  001013                          BNE     6$              ;BR IF IT IS NOT EQUAL
020756  122761  177777  000002  3$:     CMPB    #-1,2(R1)       ;ALL BAD SECTORS ?
020764  001002                          BNE     4$              ;BR IF NO
020766  005726                          TST     (SP)+           ;ADJUST STACK AND
020770  000413                          BR      8$              ;CHECK 'MESSAGE'
020772  122661  000002          4$:     CMPB    (SP)+,2(R1)     ;COMPARE THE SECTOR ADDRESS
020776  001003                          BNE     7$              ;BR IF NOT EQUAL
021000  000407                          BR      8$              ;CHECK 'MESSAGE'
021002  005726                  5$:     TST     (SP)+           ;CLEAR OFF THE STACK
021004  005726                  6$:     TST     (SP)+           ;INCREMENT THE STACK POINTER
021006  062701  000004          7$:     ADD     #4,R1           ;GO TO THE NEXT LOCATION IN THE TABLE
021012  005711                          TST     (R1)            ;EMPTY ENTRY OR TERMINATOR ?
021014  100407                          BMI     9$              ;BR IF YES
021016  000742                          BR      1$              ;TRY NEXT SECTOR
021020  005737  001472          8$:     TST     MESSAGE         ;PRINT THE ERROR ANYWAY ?
021024  001006                          BNE     10$             ;BR IF NOT
021026  012737  177777  001342          MOV     #-1,BADSEC      ;SET THE INDICATOR FOR THE IDENTIFICATION LINE
021034  062766  000002  000002  9$:     ADD     #2,2(SP)        ;INCREMENT THE RETURN
021042                          10$:
021042  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
021044  000207                          RTS     PC              ;RETURN
```

```
                                ;;*****************************************************************
                                .SBTTL  ERROR MESSAGE GENERATION ROUTINES
                                ;;*****************************************************************

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

021046  032777  002000  160100 LINE1:  BIT     #SW10,@SWR      ;SWITCH 10 SET ?
021054  001402                        BEQ     1$              ;BR IF NOT
021056  104401  001176                TYPE    ,$3ELL          ;RING THE BELL
021062  032777  020000  160064 1$:     BIT     #SW13,@SWR      ;INHIBIT TYPEOUT ?
021070  001405                        BEQ     2$              ;BR IF NOT
021072  104414  001203                DISPLY  ,$CRLF          ;CR-LF
021076  104414  001203                DISPLY  ,$CRLF          ;CR-LF
021102  000410                        BR      3$              ;EXIT
021104  104414  001203        2$:     DISPLY  ,$CRLF          ;CR-LF
021110  104414  001203                DISPLY  ,$CRLF          ;CR-LF
021114  004737  024750                JSR     PC,$TIME        ;TYPE THE TIME
021120  104414  075555                DISPLY  ,BLNKS1         ;TYPE 1 BLANK
021124  000207                3$:     RTS     PC              ;RETURN & TYPE DESCRIPTION

                                ;PRINT LINE 2 OF ERROR MESSAGE
                                :'PRSNT COMMAND = XXXX  PREV COMMAND = XXXX'
                                :'* ERROR AT BAD TRACK/SECTOR'
                                :'DRV RMCS1   RMCS2   RMDS1   RMER1   RMMR2   RMER2   RMEC1   RMEC2'
                                :'RMWC    RMBA    RMDA    RMAS    RMLA    RMDB   RMMR1   RMDT'
                                :'RMSN    RMOF    RMDC    RMCC    STATUS'
                                :'RMBAE   RMCS3'  (RH70 ONLY)
                                :'BUS ADDRESS OR WORD COUNT NOT CONSISTENT'
                                :'RMBA = XXXXXX   RMWC = XXXXXX'
                                :'BUFFER ADR = XXXXXX   WRD CNT = XXXX   ACTUAL NMBR WRDS XFRD   XXX'

021126                          LINE2:
021126  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
021130  010446                        MOV     R4,-(SP)        ;;PUSH R4 ON STACK
021132  010546                        MOV     R5,-(SP)        ;;PUSH R5 ON STACK
021134  104414  001203                DISPLY  ,$CRLF          ;CR-LF
021140  005037  021266                CLR     4$              ;CLEAR MESSAGE ADDRESS STORAGE
021144  005004                        CLR     R4              ;WORKING REGISTER
021146  012737  073666  021266        MOV     #LIN2C,4$       ;ADDRESS OF 'PRSNT COMMAND - ' MSC
021154  116004  002136                MOVB    $RMCS1(R0),R4   ;GET THE OPCODE
021160  042704  177701                BIC     #^C76,R4        ;SAVE ONLY SIGNIFICANT BITS
021164  004737  021222                JSR     PC,1$           ;TYPE THE FIRST MNEMONIC
021170  005737  021272                TST     5$              ;SEE IF MNEMONIC ENTRY FOUND
021174  001440                        BEQ     LINE2A          ;BR IF NOT
021176  012737  073706  021266        MOV     #LIN2P,4$       ;ADDRESS OF 'PREVS COMMAND = ' MSG
021204  116004  000027                MOVB    $PREVO(R0),R4   ;PREVIOUS OPERATION CODE
021210  042704  177701                BIC     #^C76,R4        ;SAVE ONLY SIGNIFICANT BITS
021214  004737  021222                JSR     PC,1$           ;TYPE THE PREVIOUS MNEMONIC
021220  000426                        BR      LINE2A          ;CONTINUE
021222  005005                1$:     CLR     R5              ;CLEAR THE TABLE INDEX
021224  126504  002072        2$:     CMPB    OPTBL(R5),R4    ;LOOK FOR THE OPCODE
021230  001405                        BEQ     3$              ;BR WHEN OPCODE COUNT EQUALS OPCODE
021232  105765  002072                TSTB    OPTBL(R5)       ;LOOK FOR END OF TABLE
021236  100402                        BMI     3$              ;BR IF END
021240  005205                        INC     R5              ;INCREMENT THE POINTER
021242  000770                        BR      2$              ;CONTINUE - NOT END OF TAB E
```

```
                                3$:     ASL     R5              ;SHIFT INDEX
                                        ASL     R5              ;SHIFT THE INDEX
                                        ASL     R5              ;SHIFT THE INDEX
021252  012737  002112  021272        MOV     #MNTBL,5$       ;ADDRESS OF ASCII TEXT TABLE
021260  060537  021272                ADD     R5,5$           ;ADD THE INDEX
021264  104414                        DISPLY                  ;TYPE IT
021266  000000                4$:     .WORD   0               ;ADDRESS OF 'PRESENT' OR 'PREVIOUS' MESSAGE
021270  104414                        DISPLY                  ;TYPE THE OPERATION MNEMONIC
021272  000000                5$:     .WORD   0               ;ADDRESS OF MESSAGE
021274  000207                        RTS     PC              ;RETURN TO MAIN ROUTINE

021276  005737  001342        LINE2A: TST     BADSEC          ;PRINT THE BAD SECTOR LINE ?
021302  001404                        BEQ     LINE2B          ;BR IF NOT
021304  104414  001203                DISPLY  ,$CRLF          ;CR-LF
021310  104414  073727                DISPLY  ,LIN2S          ;ERROR ADDRESS DEFINED AS BAD AREA
021314  104414  001203        LINE2B: DISPLY  ,$CRLF          ;CR-LF
021320  104414  073240                DISPLY  ,DH14           ;STANDARD RM REGISTER HEADER
021324  104414  075555                DISPLY  ,BLNKS1         ;TYPE 1 BLANK
021330  013746  001324                MOV     UNIT,-(SP)      ;PUT THE DRIVE NUMBER ON THE STACK
021334  004737  023176                JSR     PC,LINDEC       ;TYPE DRIVE NUMBER
021340  104414  075554                DISPLY  ,BLNKS2         ;TYPE 2 BLANKS
021344  012705  073600                MOV     #DT14,R5        ;REGISTER INDEXES
021350  004737  021532                JSR     PC,3$           ;PRINT THE REGISTERS
021354  032777  000040  157572        BIT     #SW05,@SWR      ;PRINT THE OPTIONAL REGISTERS ?
021362  001031                        BNE     1$              ;BR IF NOT
021364  104414  073342                DISPLY  ,DH15
021370  012705  073622                MOV     #DT15,R5        ;SECOND DATA LINE
021374  004737  021532                JSR     PC,3$           ;PRINT THEM
021400  104414  073440                DISPLY  ,DH16
021404  012705  073644                MOV     #DT16,R5        ;THIRD DATA LINE
021410  004737  021532                JSR     PC,3$           ;PRINT THE REGISTERS
021414  013746  001234                MOV     $CPUOP,-(SP)    ;CHECK THE CPU (RH) TYPE
021420  042716  003777                BIC     #^C174000,(SP)  ;LEAVE THE CPU BITS
021424  022726  030000                CMP     #30000,(SP)+    ;SEE IF RH70
021430  001006                        BNE     1$              ;BR IF NO
021432  104414  073510                DISPLY  ,DH17
021436  012705  073660                MOV     #DT17,R5        ;OPTIONAL FOURTH DATA LINE
021442  004737  021532                JSR     PC,3$           ;PRINT THE REGISTERS
021446  032760  000100  000016  1$:   BIT     #BIT6,$STATUS(R0)       ;DATA ERROR ?
021454  001422                        BEQ     2$              ;BR IF NOT
021456  016046  000020                MOV     $WRDL(R0),-(SP) ;TRANSFER WRD CNT
021462  066016  002140                ADD     $RMWC(R0),(SP)  ;ADD REMAINING WORD COUNT
021466  006316                        ASL     (SP)            ;CONVERT TO AN BYTE INCREMENT
021470  066016  000006                ADD     $BUF(R0),(SP)   ;BUFFER STARTING ADDRESS
021474  022660  002142                CMP     (SP)+,$RMBA(R0) ;CORRECT BUFFER ADDRESS ?
021500  001410                        BEQ     2$              ;BR IF YES
021502  104414  072665                DISPLY  ,EM46           ;'BUS ADDRESS AND WORD COUNT ARE NOT CONSISTENT'
021506  104414  001203                DISPLY  ,$CRLF          ;CR-LF
021512  004737  021624                JSR     PC,LINE3D       ;PRINT LINE 3D OF ERROR MESSAGE
021516  004737  022236                JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
021522                          2$:
021522  012605                        MOV     (SP)+,R5        ;;POP STACK INTO R5
021524  012604                        MOV     (SP)+,R4        ;;POP STACK INTO R4
021526  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
021530  000207                        RTS     PC              ;RETURN TO ERROR PROCESSING ROUTINE
021532  012546                3$:     MOV     (R5)+,-(SP)     ;PUT THE REGISTER INDEX ON THE STACK
021534  060016                        ADD     R0,(SP)         ;ADD DRIVE'S TABLE ADDRESS
```

```
                                        MOV     @(SP),-(SP)     ;VALUE
                                        JSR     PC,LINOCT       ;TYPE IT
                                        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   PREV 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
                                ;'RMBA = XXXXXX   RMWC = XXXXXX'

                        157322  LINE3D: BIT     #SW05,@SWR      ;SWITCH 5 SET ?
                                        BEQ     1$              ;BR IF IT IS
                                        DISPLY  ,LINB3          ;'RMBA = '
                                        MOV     $RMBA(R0),-(SP) ;BUFFER ADDR REG CONTENTS
                                        JSR     PC,LINOCT       ;CONVERT TO OCTAL AND TYPE IT
                                        DISPLY  ,LINW3          ;'  RMWC = '
                                        MOV     $RMWC(R0),-(SP) ;WORD COUNT REGISTER CONTENTS
                                        JSR     PC,LINOCT       ;CONVERT TO OCTAL AND TYPE IT
                                        DISPLY  ,$CRLF
                                1$:     RTS     PC

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

                                LINE3E: DISPLY  ,LINS3          ;'START CYL = '
                                        MOV     $CYL(R0),-(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(R0),(SP)   ;TRACK TO STACK
                                                JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
                                                DISPLY  ,BLNKS2         ;TYPE 2 BLANKS
                                                DISPLY  ,LINSS3         ;'START SEC = '
                                                CLR     -(SP)           ;CLEAR STACK
                                                MOVB    $SEC(R0),(SP)   ;SECTOR ADDR TO STACK
                                                JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
                                                DISPLY  ,$CRLF
                                                RTS     PC

                                        ;PRINT LINE 3F OF ERROR MESSAGE
                                        ;'RMDA = XXXXXX   RMCA - XXXXXX'

157166  LINE3F: BIT     #SW5,@SWR       ;SWITCH 5 SET ?
                BEQ     1$              ;BR IF NOT
                DISPLY  ,LINDA3         ;'RMDA = '
                MOV     $RMDA(R0),-(SP) ;PUT SECTOR/TRACK ADDRESS ON THE STACK
                JSR     PC,LINOCT       ;TYPE IT
                DISPLY  ,BLNKS2         ;TYPE 2 BLANKS
                DISPLY  ,LINCA3         ;' RMDC = '
                MOV     $RMDC(R0),-(SP) ;PUT DESIRED CYLINDER ADDRESS ON THE STACK
                JSR     PC,LINOCT       ;TYPE IT
                DISPLY  ,$CRLF
1$:             RTS     PC

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

LIN3.1: JSR     PC,READDR       ;DECREMENT TRACK AND SECTOR ADDRESS
        JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
        DISPLY  ,T              ;PRINT ' T'
        JSR     PC,LINDEC       ;TYPE TRACK IN DECIMAL
        DISPLY  ,S              ;PRINT ' S'
        JSR     PC,LINDEC       ;TYPE SECTOR ADDRESS
        DISPLY  ,LINP3          ;PRINT 'PREV ADDR'
        MOV     $PREVA+2(R0),-(SP)      ;PREVIOUS CYLINDER
        JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
        DISPLY  ,T              ;PRINT ' T'
        CLR     -(SP)           ;MAKE ROOM ON THE STACK
        MOVB    $PREVA+1(R0),(SP)       ;PREVIOUS TRACK ADDRESS
        JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
        DISPLY  ,S              ;PRINT ' S'
        CLR     -(SP)           ;MAKE ROOM ON THE STACK
        MOVB    $PREVA(R0),(SP) ;PREVIOUS SECTOR DDRESS
        JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
        DISPLY  ,$CRLF
        RTS     PC

;'START CYL = CCC   END CYL = CCC'

LIN3.3: DISPLY  ,LINS3          ;LINE '3B & 3C' ENTRANCE
        MOV     $PREVA+2(R0),-(SP)      ;PREVIOUS CYLINDER
        JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
        DISPLY  ,LINEN3         ;PRINT 'END CYL'
        MOV     $RMDC(R0),-(SP) ;PRESENT CYLINDER
        JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
        RTS     PC
```

```
                                        ;'ACTUAL CYL = CCC   TRK   TT'
                                        ;
022172  104414  074071        LIN3.4: DISPLY  ,LINA3          ;PRINT 'ACTUAL'
022176  013746  101066                MOV     CYLNDR,-(SP)    ;ACTUAL CYLINDER
022202  042716  010000                BIC     #BIT12,(SP)     ;CLEAR THE FORMAT BIT
022206  004737  023176                JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
022212  104414  074110                DISPLY  ,LINT3          ;PRINT TRACK
022216  005046                        CLR     -(SP)           ;CLEAR STACK WORD
022220  116016  002145                MOVB    $RMDA+1(R0),(SP)        ;PUT TRACK ON STACK
022224  004737  023176                JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
022230  104414  001203                DISPLY  ,$CRLF
022234  000207                        RTS     PC
                                        ;
                                        ;PRINT LINE 4 OF ERROR MESSAGE
                                        ;'BUFFER ADR = XXXXXX   WRD CNT = XXXX   ACTUAL NMBR WRDS XFRD   XXX'
                                        ;
022236  032760  000100  000016 LINE4:  BIT     #BIT06,$STATUS(R0)      ;DATA ERROR ?
022244  001427                        BEQ     1$              ;BR IF NOT
022246  104414  074210                DISPLY  ,LINM4          ;'PRINT BUFFER'
022252  016046  000006                MOV     $BUF(R0),-(SP)  ;BUFFER ADDR ON STACK
022256  004737  023144                JSR     PC,LINOCT       ;CONVERT TO OCTAL & PRINT
022262  104414  074226                DISPLY  ,LINS4          ;PRINT 'WRD CNT'
022266  016046  000020                MOV     $WRDL(R0),-(SP) ;WORD LENGTH SIZE(WORD COUNT)
022272  004737  023176                JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
022276  104414  074242                DISPLY  ,LINX4          ;'ACTUAL NMBR WRDS XFRD = '
022302  016046  002142                MOV     $RMBA(R0),-(SP) ;VALUE IN BUFFER ADDR REGISTER
022306  166016  000006                SUB     $BUF(R0),(SP)   ;SUBTRACT STARTING ADDRESS
022312  006216                        ASR     (SP)            ;CONVERT INTO A WORD COUNT
022314  004737  023176                JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
022320  104414  001203                DISPLY  ,$CRLF          ;CR-LF
022324  000207                1$:     RTS     PC              ;RETURN
                                        ;
                                        ;PRINT LINE 5 OF ERROR MESSAGE
                                        ;'GOOD DATA = XXXXXX   BAD DATA = XXXXXX   SECT POS = XXX'
                                        ;
022326  104414  074274        LINE5:  DISPLY  ,LIND5          ;PRINT 'GOOD DATA'
022332  162760  000002  002142        SUB     #2,$RMBA(R0)    ;BACK THE ADDRESS UP
022340  013746  001234                MOV     $CPUOP,-(SP)    ;CHECK THE CPU (RH) TYPE
022344  042716  003777                BIC     #^C174000,(SP)  ;LEAVE THE CPU BITS
022350  022726  030000                CMP     #30000,(SP)+    ;SEE IF RH70
022354  001012                        BNE     1$              ;BR IF NO
022356  162760  000004  002142        SUB     #4,$RMBA(R0)    ;BACKUP THE BUFFER POINTER
022364  032760  004000  002210        BIT     #BIT11,$RMCS3(R0)       ;SEE WHICH WORD HALF DIDN'T COMPARE
022372  001403                        BEQ     1$              ;IF EQ, EVEN HALF DIDN'T COMPARE
022374  162760  000002  002142        SUB     #2,$RMBA(R0)    ;BACKUP THE BUFFER POINTER AGAIN
022402  017046  002142        1$:     MOV     @$RMBA(R0),-(SP)        ;'GOOD' DATA - AT THE BUFFER LOCATION
022406  004737  023144                JSR     PC,LINOCT       ;TYPE IT
022412  104414  074310                DISPLY  ,LINB5          ;PRINT 'BAD DATA'
022416  016046  002160                MOV     $RMDB(R0),-(SP) ;BAD DATA FROM BUFFER
022422  004737  023144                JSR     PC,LINOCT       ;TYPE IT
022426  016046  002140                MOV     $RMWC(R0),-(SP) ;WORD LENGTH ON STACK
022432  066016  000020                ADD     $WRDL(R0),(SP)  ;MAKE INTO A POSITIVE NUMBER
022436  005046                        CLR     -(SP)           ;UPPER DIVIDEND TO ZERO
022440  016046  000022                MOV     $SSEC(R0),-(SP) ;SECTOR SIZE ON THE STACK
022444  004737  031710                JSR     PC,$DIV         ;DIVIDE WORDS XFERED BY SECTOR SIZE
022450  012616                        MOV     (SP)+,(SP)      ;MOVE REMAINDER UP THE STACK
022452  104414  074325                DISPLY  ,LINP5          ;PRINT 'SECT POS'
```

```
                                        JSR     PC,LINDEC       ;TYPE THE POSITION
                                        DISPLY  ,$CRLF
                                        RTS     PC

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

                                LINE5A:
                                2$:     DISPLY  ,LINS5          ;'HEADER CONTENTS OF ERROR SECTOR'
                                        MOV     CYLNDR,-(SP)    ;HEADER POSITION
                                        JSR     PC,LINOCT       ;TYPE IT
                                        DISPLY  ,BLNKS2         ;TYPE 2 BLANKS
                                        MOV     CYLNDR+2,-(SP)  ;HEADER POSITION +2
                                        JSR     PC,LINOCT       ;TYPE IT
                                        DISPLY  ,BLNKS2         ;TYPE 2 BLANKS
                                        DISPLY  ,LINX5          ;APPENDING INFO 1/23/77
                                3$:     DISPLY  ,$CRLF
                                        RTS     PC

                                ;PRINT LINE 5B OF ERROR MESSAGE
                                ;'RMEC1 = XXXXXX   RMEC2 = XXXXXX'

                                LINE5B: DISPLY  ,LINEP5         ;'RMEC1 = '
                                        MOV     $RMEC1(R0),-(SP)        ;PUT REGISTER CONTENTS ON THE STACK
                                        JSR     PC,LINOCT       ;TYPE IT
                                        DISPLY  ,BLNKS2         ;TYPE 2 BLANKS
                                        DISPLY  ,LINE05         ;' RMEC2 = '
                                        MOV     $RMEC2(R0),-(SP)        ;PUT REGISTER CONTENTS ON THE STACK
                                        JSR     PC,LINOCT       ;TYPE IT
                                        DISPLY  ,$CRLF
                                        RTS     PC              ;RETURN

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

                                LINE6:  DISPLY  ,LIN86          ;ECC CORRECTABLE
                                        DISPLY  ,$CRLF
                                        RTS     PC

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

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

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

                                LINE6B: DISPLY  ,LIN86          ;PRINT 'SECTOR IS ECC CORRECTABLE '
                                        BR      LIN6.1

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

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

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

022634  104414  074531         LINE6D: DISPLY  ,LINU06          ;'UNCORRECTABLE AFTER N RETRIES'
022640  000411                         BR      LIN6.2          ;FINISH

                                ;TYPE THE OFFSET VALUE IN MICRO-INCHES

022642  006301                 LIN6.1: ASL     R1              ;DOUBLE THE OFFSET TABLE INDEX
022644  016137  002420  022654         MOV     OFMTBL(R1),1$   ;ADDRESS OF OFFSET POSITION MESSAGE
022652  104414                         DISPLY
022654  000000                 1$:     .WORD   0               ;OFFSET VALUE
022656  104414  001203                 DISPLY  ,$CRLF
022662  000207                         RTS     PC

                                ;RETRY COUNT TYPEOUT

022664  005046                 LIN6.2: CLR     -(SP)           ;CLEAR STACK
022666  113716  001331                 MOVB    RETRY+1,(SP)    ;RETRY COUNT
022672  004737  023176                 JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
022676  104414  074517                 DISPLY  ,LINR6          ;'RETRY'
022702  104414  001203                 DISPLY  ,$CRLF
022706  000207                         RTS     PC

                                ;PRINT LINE 7 OF THE ERROR MESSAGE
                                ;'TOTAL ERRORS:XXX   WOFL:X WRDS WRITN:XXXXXXX   ROFL:0 WRDS READ:XXXXXXX'

022710  104414  074643         LINE7:  DISPLY  ,LIN7T          ;TOTAL ERRORS
022714  016046  000072                 MOV     $TOTAL(R0),-(SP)        ;TO STACK
022720  004737  023176                 JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
022724  104414  074661                 DISPLY  ,LIN7OX         ;PRINT 'WTOFL'
022730  016046  000056                 MOV     $WTOFL(R0),-(SP)        ;;PUSH $WTOFL(R0) ON STACK
022734  004737  023176                 JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
022740  104414  074671                 DISPLY  ,LIN7X          ;PRINT 'WRDS WRITN'
022744  012746  000060                 MOV     #$WRITN,-(SP)   ;ADDRESS OF LOW WORD ON STACK
022750  060016                         ADD     R0,(SP)
022752  004737  037122                 JSR     PC,$DB2D        ;CONVERT
022756  004737  032234                 JSR     PC,$SUPRL       ;PRINT
022762  104414  074706                 DISPLY  ,LIN7OR         ;PRINT 'ROFL'
022766  016046  000064                 MOV     $RDOFL(R0),-(SP)        ;;PUSH $RDOFL(R0) ON STACK
022772  004737  023176                 JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
022776  104414  074716                 DISPLY  ,LIN7R          ;'WRDS READ'
023002  012746  000066                 MOV     #$READ,-(SP)    ;LOW WORD ADDRESS
023006  060016                         ADD     R0,(SP)
023010  004737  037122                 JSR     PC,$DB2D        ;CONVERT
023014  004737  032234                 JSR     PC,$SUPRL       ;PRINT IT
023020  104414  001203                 DISPLY  ,$CRLF          ;CR-LF
023024  032777  100000  156122         BIT     #SW15,@SWR      ;SEE IF 'HALT ON ERROR' - SWITCH 15
023032  001401                         BEQ     1$              ;BR IF NOT
023034  000000                         HALT                    ;SWITCH 15 HALT
023036  000207                 1$:     RTS     PC

                                ;PRINT LINE 7A OF ERROR MESSAGE
                                ;'TOTAL SEEKS=XXXXX   TOTAL MISFOS ERR = XXX   TOTAL SKI- XXX'

023040  104414  074603         LINE7A: DISPLY  ,LIN7P          ;'TOTAL SEEKS = '
023044  012746  000052                 MOV     #$POSIT,-(SP)   ;TOTAL SEEKS
```

```
                        ADD     R0,(SP)         ;DEVICE TABLE ADDRESS
                        JSR     PC,$DB2D        ;CONVERT THE SEEK COUNT
                        JSR     PC,$SUPRL       ;PRINT IT
                        DISPLY  ,LIN7M          ;'  TOTAL MISPOS ERR - '
                        MOV     $MISPO(R0),-(SP)        ;TOTAL ERRORS
                        JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
                        DISPLY  ,LIN7S          ;'  TOTAL SKI ERR = '
                        MOV     $SKI(R0),-(SP)  ;CONVERT & PRINT IT
                        JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
                        DISPLY  ,$CRLF          ;CR-LF
                        BIT     #SW15,@SWR      ;SEE IF HALT ON ERROR - SWITCH 15 SET
                        BEQ     1$              ;BR IF NOT
                        HALT                    ;SWITCH 15 HALT
                1$:     RTS     PC

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

                LINE8:  DISPLY  ,LIN8M
                        JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
                        RTS     PC

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

                LINOCT: MOV     2(SP),-(SP)     ;PUT NUMBER IN PROPER LOCATION ON STACK
                        JSR     PC,$SB2O        ;CONVERT THE NUMBER TO OCTAL
                        MOV     (SP)+,1$        ;GET THE ADDRESS OF THE ASCII STRING
                        ADD     #5.,1$          ;ADDRESS THE LAST 6 ASCII DIGITS
                        DISPLY                  ;TYPE IT
                1$:     .WORD   0               ;ADDRESS
                        MOV     (SP)+,(SP)      ;CORRECT THE STACK
                        RTS     PC              ;RETURN

                ;ROUTINE TO CONVERT THE INPUT NUMBER TO DECIMAL AND TYPE IT WITH
                ;LEADING ZERO SUPRESSION
                ;CALL:
                ;       MOV     NUM,-(SP)       ;PUT THE NUMBER ON THE STACK
                ;       JSR     PC,LINDEC
                ;       RETURN

                LINDEC: MOV     2(SP),-(SP)     ;SET UP STACK FOR CONVERT
                        JSR     PC,$SB2D        ;CONVERT IT TO DECIMAL
                        JSR     PC,$SUPRL       ;TYPE IT (WITH LEADING ZEROS SUPRESSED)
                        MOV     (SP)+,(SP)      ;RESTORE STACK POINTER
                        RTS     PC
```

```
                                        .SBTTL  GENERAL SUPPORT SUBROUTINES
                                        ;DECREMENT THE SECTOR-TRACK ADDRESS
                                        ;CALL:
                                        ;       MOV     #DPB,R0         ;DPB ADDRESS
                                        ;       JSR     PC,READDR
                                        ;       RETURN
                                        ;
                                        ;ON RETURN THE STACK CONTAINS THE FOLLOWING:
                                        ;       4(SP) = SECTOR ADDRESS
                                        ;       2(SP) = TRACK ADDRESS
                                        ;        (SP) = CYLINDER ADDRESS
                                        ;
023216  004737  027346          READDR: JSR     PC,GETLMT       ;GET ADDRESS LIMITS
023222  162706  000006                  SUB     #6,SP           ;DECREMENT THE STACK POINTER
023226  016616  000006                  MOV     6(SP),(SP)      ;MOVE THE RETURN ADDR DOWN THE STACK
023232  005066  000006                  CLR     6(SP)           ;CLEAR STACK FOR SECTOR
023236  005066  000004                  CLR     4(SP)           ;CLEAR STACK FOR TRACK
023242  116066  002144  000006          MOVB    $RMDA(R0),6(SP) ;SECTOR ON STACK
023250  116066  002145  000004          MOVB    $RMDA+1(R0),4(SP)       ;TRACK ADDRESS
023256  016066  002172  000002          MOV     $RMDC(R0),2(SP) ;CYLINDER ADDRESS
023264  005766  000006          1$:     TST     6(SP)           ;SECTOR 0 ?
023270  001403                          BEQ     2$              ;BRANCH IF SO
023272  105366  000006                  DECB    6(SP)           ;DECREMENT ONE SECTOR
023276  000421                          BR      4$              ;BRANCH TO EXIT
023300  005766  000004          2$:     TST     4(SP)           ;ALSO ON TRACK 0 ?
023304  001406                          BEQ     3$              ;BRANCH IF SO
023306  113766  001424  000006          MOVB    SECLMT,6(SP)    ;LAST SECTOR
023314  105366  000004                  DECB    4(SP)           ;DECREMENT ONE TRACK
023320  000410                          BR      4$              ;EXIT
023322  113766  001424  000006  3$:     MOVB    SECLMT,6(SP)    ;LAST SECTOR
023330  113766  001426  000004          MOVB    TRKLMT,4(SP)    ;GET LAST TRACK
023336  005366  000002                  DEC     2(SP)           ;DECREMENT ONE CYLINDER COUNT
023342  000207                  4$:     RTS     PC              ;RETURN

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

023344  012737  177777  001312  CKCLK:  MOV     #-1,CLKFLG      ;CLEAR CLOCK AVAILABILITY FLAG
023352  012737  177777  001310          MOV     #-1,PCLOCK      ;CLEAR KW11-P CLOCK AVAILABILITY FLAG
023360  013746  000004                  MOV     ERRVEC,-(SP)    ;;PUSH ERRVEC ON STACK
023364  012737  023440  000004          MOV     #CKCLK1,ERRVEC  ;SET UP VECTOR FOR CLOCK CHECK
023372  005777  155700                  TST     @$LKCSR         ;CHECK FOR KW11-P
023376  005037  001312                  CLR     CLKFLG          ;SET CLOCK AVAILABILITY FLAG
023402  005037  001310                  CLR     PCLOCK          ;SET KW11-P CLOCK FLAG
023406  013701  001302                  MOV     $LPVEC,R1       ;KW11-P VECTOR ADDRESS
023412  012721  025074                  MOV     #CLOCK,(R1)+    ;SET UP KW11-P VECTOR
023416  012711  000300                  MOV     #300,(R1)       ;PSW - PRI 6
023422  012777  174575  155650          MOV     #-1667.,@$LKCSB ;LOAD COUNTER BUFFER WITH 16.67
023430  012777  000131  155640          MOV     #131,@$LKCSR    ;SET CLOCK - CNT UP, 10US, CONT INT
023436  000441                          BR      CKCLK3

023440  012716  023446          CKCLK1: MOV     #1$,(SP)        ;SETUP RETURN ADDRESS
023444  000002                          RTI
023446  012737  023510  000004  1$:     MOV     #CKCLK2,ERRVEC  ;CHANGE ERROR VECTOR TO CHECK FOR KW11-L
023454  005777  155624                  TST     @$LKS           ;LOOK FOR KW11-L
023460  005037  001312                  CLR     CLKFLG          ;SET CLOCK FLAG
023464  013701  001306                  MOV     $LLVEC,R1       ;KW11-L VECTOR ADDRESS
```

```
                                                ;SET UP KW11-L VECTOR
                                                ;PSW - PRI 6
                                                ;SET KW11-L INTERRUPT
                                                ;
                                                ;SETUP RETURN ADDRESS
                                                ;'P OR L CLOCK MUST BE ON SYSTEM'
                                                ;RUNNING IN AUTO MODE ?
                                                ;BR IF NOT
                                                ;ABORT PROGRAM
                                                ;HALT
                                                ;TRY AGAIN
                                                ;RESTORE THE ERROR VECTOR
```

58	023470	012721	025074			MOV	#CLOCK,(R1)+	
59	023474	012711	000300			MOV	#300,(R1)	
60	023500	012777	000100	155576		MOV	#100,@$LKS	
61	023506	000415				BR	CKCLK3	
62								
63	023510	012716	023516		CKCLK2:	MOV	#1$,(SP)	
64	023514	000002				RTI		
65	023516	104401	076422		1$:	TYPE	,NEDCLK	
66	023522	105737	001150			TSTB	$AUTOB	
67	023526	001402				BEQ	2$	
68	023530	000137	031612			JMP	$GET42	
69	023534	000000			2$:	HALT		
70	023536	000137	003634			JMP	START	
71	023542	012637	000004		CKCLK3:	MOV	(SP)+,ERRVEC	
72	023546	000207				RTS	PC	

```
                                ;ROUTINE TO DISPLAY STATISTICS FOR ASSIGNED DRIVES
                                ;CALL:
                                ;       JSR     PC,STATPR
                                ;       RETURN

023550                          STATPR: MOV     R0,-(SP)        ;;PUSH R0 ON STACK
023550  010046                         MOV     R4,-(SP)        ;;PUSH R4 ON STACK
023552  010446
023554  005737  001530                 TST     ASNLST          ;ANY DRIVES ASSIGNED ?
023560  001423                         BEQ     5$              ;BR IF NOT
023562  104401  001203         2$:     TYPE    ,$CRLF          ;CR-LF
023566  004737  023664                 JSR     PC,SHDTYP       ;TYPE THE HEADING
023572  005004                         CLR     R4              ;CLEAR THE DRIVE INDEX
023574  006304                 3$:     ASL     R4              ;CHANGE TO INDEX WORDS
023576  016400  002044                 MOV     BLKADR(R4),R0   ;GET THE DRIVE'S BLOCK ADDRESS
023602  006204                         ASR     R4              ;RESTORE R4
023604  136437  040142  001530         BITB    ATABIT(R4),ASNLST       ;IS THIS DRIVE ASSIGNED ?
023612  001402                         BEQ     4$              ;BR IF NOT
023614  004737  023674                 JSR     PC,SDETAL       ;TYPE THE PERFORMANCE SUMMARY
023620  005204                 4$:     INC     R4              ;INCREMENT THE INDEX
023622  020427  000010                 CMP     R4,#8.          ;FINISHED ?
023626  001362                         BNE     3$              ;BR IF NO
023630                          5$:
023630  012604                         MOV     (SP)+,R4        ;;POP STACK INTO R4
023632  012600                         MOV     (SP)+,R0        ;;POP STACK INTO R0
023634  000207                         RTS     PC              ;RETURN

                                ;ROUTINE TO TYPE THE PERFORMANCE SUMMARY (STATISTICS) FOR AN INDIVIDUAL
                                ;DRIVE.
                                ;CALL:
                                ;       MOV     #DPB,R0         ;DPB ADDRESS
                                ;       JSR     PC,SUMARY
                                ;       RETURN

023636  010046                  SUMARY: MOV     R0,-(SP)        ;SAVE R0
023640  010446                         MOV     R4,-(SP)        ;SAVE R4
023642  004737  023664                 JSR     PC,SHDTYP       ;TYPE THE HEADING
023646  005004                         CLR     R4              ;CLEAR R4 FOR DRIVE NUMBER
023650  111004                         MOVB    (R0),R4         ;DRIVE NUMBER
023652  004737  023674                 JSR     PC,SDETAL       ;TYPE THE STATISTICS
023656  012604                         MOV     (SP)+,R4        ;RESTORE R4
023660  012600                         MOV     (SP)+,R0        ;RESTORE R0
023662  000207                         RTS     PC              ;RETURN

                                ;TYPE THE HEADER FOR THE DRIVE PERFORMANCE SUMMARY TYPEOUT
                                ;CALL:
                                ;       JSR     PC,SHDTYP
                                ;       RETURN

023664  004537  032444         SHDTYP: JSR     R5,TYPRI4       ;TYPE SUMMARY HEADER
023670  076164                         SUMHD                   ;HEADER
023672  000207                         RTS     PC              ;RETURN

                                ;TYPE THE PERFORMANCE SUMMARY
                                ;CALL:
                                ;       MOV     #DRIVE,R4       ;DRIVE NUMBER
```

```
                                        ;       MOV     #DPB,R0         ;DPB ADDRESS
                                        ;       RETURN
                                        SDETAL:
023674                                          TYPE    ,65$            ;;TYPE ASCIZ STRING
023674  104401  023702                          BR      64$             ;;GET OVER THE ASCIZ
023700  000404                          ;;65$:  .ASCIZ  <CRLF>/TIME /
023712                                  64$:
023712  004737  024750                          JSR     PC,$TIME        ;TYPE ELAPSED TIME

                                        ;TYPE LINE 2 OF SUMMARY
023716  104401  001203                          TYPE    ,$CRLF          ;CR-LF
023722  104401  075625                          TYPE    ,UNTMSG         ;TYPE 'DRIVE'
023726  010446                          MOV     R4,-(SP)        ;;SAVE R4 FOR TYPEOUT
023730  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
023732     002                          .BYTE   2               ;;TYPE 2 DIGIT(S)
023733     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
023734  104401  075555                  TYPE    ,BLNKS1         ;TYPE 1 BLANK
023740  104401  075621                  TYPE    ,DASH           ;TYPE '-'
023744  104401  075555                  TYPE    ,BLNKS1         ;TYPE 1 BLANK
023750  012737  076051  024014          MOV     #$RM03,2$       ;ADDRESS OF RM03 MESSAGE
023756  122764  000004  040046          CMPB    #4,DRVTYP(R4)   ;IS DEVICE AN RM03 ?
023764  001412                          BEQ     1$              ;BR IF YES
023766  012737  076044  024014          MOV     #$RM02,2$       ;ADDRESS OF RM02 MESSAGE
023774  122764  000005  040046          CMPB    #5,DRVTYP(R4)   ;IS DEVICE AN RM02 ?
024002  001403                          BEQ     1$              ;BR IF YES
024004  012737  076056  024014          MOV     #$RM05,2$       ;ADDRESS OF RM05 MESSAGE
024012  104401                  1$:     TYPE                    ;TYPE THE DRIVE TYPE MESSAGE
024014  000000                  2$:     .WORD   0               ;MESSAGE ADDRESS HERE
024016  104401  075617                  TYPE    ,COMMA          ;TYPE ','

024022  104401  024030                  TYPE    ,67$            ;;TYPE ASCIZ STRING
024026  000404                          BR      66$             ;;GET OVER THE ASCIZ
024040                          ;;67$:  .ASCIZ  / PASS /
024040                          66$:
024040  016046  000104                  MOV     $PASSC(R0),-(SP)        ;PUT THE PASS COUNT ON THE STACK
024044  004737  033130                  JSR     PC,$SB2D        ;CONVERT IT
024050  004537  032340                  JSR     R5,REPLZ        ;TYPE IT
024054  000003                          .WORD   3               ;TYPE 3 DIGITS
024056  104401  076477                  TYPE    ,PERIOD         ;TYPE '.'
024062  104401  075555                  TYPE    ,BLNKS1         ;TYPE 1 BLANK
024066  004737  032472                  JSR     PC,TYMBA        ;TYPE MBA SERIAL NUMBER
024072  104401  075555                  TYPE    ,BLNKS1         ;TYPE 1 BLANK
024076  004737  032522                  JSR     PC,TYPACK       ;TYPE PACK SERIAL NUMBER

                                        ;TYPE LINE 3 OF SUMMARY
024102  104401  024110                  TYPE    ,69$            ;;TYPE ASCIZ STRING
024106  000407                          BR      68$             ;;GET OVER THE ASCIZ
024126                          ;;69$:  .ASCIZ  <CRLF><LF>/WT OFLOW  /
024126                          68$:
024126  016046  000056                  MOV     $WTOFL(R0),-(SP)        ;;SAVE $WTOFL(R0) FOR TYPEOUT
024132  104405                          TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
024134  104401  076477                  TYPE    ,PERIOD         ;TYPE '.'
024140  104401  024146                  TYPE    ,71$            ;;TYPE ASCIZ STRING
024144  000407                          BR      70$             ;;GET OVER THE ASCIZ
024164                          ;;71$:  .ASCIZ  / WRDS WRITN /
                                        70$:
```

```
                        MOV     R0,-(SP)        ;GET ADDRESS OF DPB
                        ADD     #$WRITN,(SP)    ;POINT TO LOW NUMBER OF WRDS WRITTEN
                        JSR     PC,$DB2D        ;CONVERT DECIMAL NUMBER
                        JSR     PC,SUPRS        ;SUPPRESS LEADING ZEROS AND TYPE
                        TYPE    ,PERIOD         ;TYPE '.'

                ;TYPE LINE 4 OF SUMMARY
                        TYPE    ,73$            ;;TYPE ASCIZ STRING
                        BR      72$             ;;GET OVER THE ASCIZ
                ;;73$:  .ASCIZ  <CRLF>/RD OFLOW  /
                72$:
                        MOV     $RDOFL(R0),-(SP)        ;;SAVE $RDOFL(R0) FOR TYPEOUT
                        TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
                        TYPE    ,PERIOD         ;TYPE '.'
                        TYPE    ,75$            ;;TYPE ASCIZ STRING
                        BR      74$             ;;GET OVER THE ASCIZ
                ;;75$:  .ASCIZ  / WRDS READ  /
                74$:
                        MOV     R0,-(SP)        ;GET ADDRESS OF DPB
                        ADD     #$READ,(SP)     ;POINT TO LOW NUMBER OF WRDS READ
                        JSR     PC,$DB2D        ;CONVERT DECIMAL NUMBER
                        JSR     PC,SUPRS        ;SUPPRESS LEADING ZEROS AND TYPE
                        TYPE    ,PERIOD         ;TYPE '.'

                ;TYPE LINE 5 OF SUMMARY
                        TYPE    ,77$            ;;TYPE ASCIZ STRING
                        BR      76$             ;;GET OVER THE ASCIZ
                ;;77$:  .ASCIZ  <CRLF>/SEEKS /
                76$:
                        MOV     R0,-(SP)        ;PUT $POSIT ON THE STACK
                        ADD     #$POSIT,(SP)    ;POINT TO LOW NUMBER OF SEEK COUNT
                        JSR     PC,$DB2D        ;CONVERT DECIMAL NUMBER
                        JSR     PC,SUPRS        ;SUPPRESS LEADING ZEROS AND TYPE
                        TYPE    ,PERIOD         ;TYPE '.'

                ;TYPE LINES 6 AND 7 OF SUMMARY
                        TYPE    ,79$            ;;TYPE ASCIZ STRING
                        BR      78$             ;;GET OVER THE ASCIZ
                ;;79$:  .ASCIZ  <CRLF>/ERRORS:/
                78$:
                        TYPE    ,81$            ;;TYPE ASCIZ STRING
                        BR      80$             ;;GET OVER THE ASCIZ
                ;;81$:  .ASCIZ  <CRLF>/SOFT /
                80$:
                        MOV     $SOFT(R0),-(SP) ;;SAVE $SOFT(R0) FOR TYPEOUT
                        TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
                        TYPE    ,PERIOD         ;TYPE '.'
                        TYPE    ,83$            ;;TYPE ASCIZ STRING
                        BR      82$             ;;GET OVER THE ASCIZ
                ;;83$:  .ASCIZ  / HARD /
                82$:
                        MOV     $HARD(R0),-(SP) ;;SAVE $HARD(R0) FOR TYPEOUT
                        TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
                        TYPE    ,PERIOD         ;TYPE '.'
                        TYPE    ,85$            ;;TYPE ASCIZ STRING
                        BR      84$             ;;GET OVER THE ASCIZ
                ;;85$:  .ASCIZ  / SKI /
```

```
                024464
147 024464  016046  000100          84$:
                                            MOV     $SKI(R0),-(SP)  ;;SAVE $SKI(R0) FOR TYPEOUT
    024470  104405                         TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
148 024472  104401  076477              TYPE    ,PERIOD             ;TYPE '.'
149 024476  104401  024504              TYPE    ,87$                ;;TYPE ASCIZ STRING
    024502  000404                      BR      86$                 ;;GET OVER THE ASCIZ
                                    ;;87$:  .ASCIZ  / MISP /
                024514              86$:
150 024514  016046  000102                  MOV     $MISPO(R0),-(SP)        ;;SAVE $MISPO(R0) FOR TYPEOUT
    024520  104405                          TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
151 024522  104401  076477              TYPE    ,PERIOD             ;TYPE '.'
152 024526  104401  024534              TYPE    ,89$                ;;TYPE ASCIZ STRING
    024532  000404                      BR      88$                 ;;GET OVER THE ASCIZ
                                    ;;89$:  .ASCIZ  / OTHER /
                024544              88$:
153 024544  016046  000072                  MOV     $TOTAL(R0),-(SP)        ;CALCULATE NUMBER OF OTHER ERRORS
156 024550  166016  000074                  SUB     $SOFT(R0),(SP)  ;SUBTRACT $SOFT FROM $TOTAL
    024554  166016  000076                  SUB     $HARD(R0),(SP)  ;SUBTRACT $HARD FROM $TOTAL
    024560  166016  000100                  SUB     $SKI(R0),(SP)   ;SUBTRACT $SKI FROM $TOTAL
    024564  166016  000102                  SUB     $MISPO(R0),(SP) ;SUBTRACT $MISPO FROM $TOTAL
157 024570  104405                          TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
158 024572  104401  076477              TYPE    ,PERIOD             ;TYPE '.'
159 024576  104401  001203              TYPE    ,$CRLF              ;CR-LF
160 024602  000207                      RTS     PC
```

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

024604  005737  001342  INCSOF: TST     BADSEC          ;SEE IF BAD TRK/SEC INDICATOR SET
024610  001006                  BNE     3$              ;BR IF IT'S SET, DON'T INCREMENT COUNT
024612  026027  000074  077777        CMP     $SOFT(R0),#77777        ;IS $SOFT ALREADY AT MAXIMUM ?
024620  103002                  BHIS    3$              ;BR IF IT IS
024622  005260  000074          INC     $SOFT(R0)               ;INCREMENT $SOFT
024626  000207          3$:     RTS     PC              ;RETURN

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

024630  005737  001342  INCHRD: TST     BADSEC          ;SEE IF BAD TRK/SEC INDICATOR SET
024634  001006                  BNE     3$              ;BR IF IT'S SET, DON'T INCREMENT COUNT
024636  026027  000076  077777        CMP     $HARD(R0),#77777        ;IS $HARD ALREADY AT MAXIMUM ?
024644  103002                  BHIS    3$              ;BR IF IT IS
024646  005260  000076          INC     $HARD(R0)               ;INCREMENT $HARD
024652  000207          3$:     RTS     PC              ;RETURN

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

024654  005737  001342  INCSKI: TST     BADSEC          ;SEE IF BAD TRK/SEC INDICATOR SET
024660  001006                  BNE     3$              ;BR IF IT'S SET, DON'T INCREMENT COUNT
024662  026027  000100  077777        CMP     $SKI(R0),#77777 ;IS $SKI ALREADY AT MAXIMUM ?
024670  103002                  BHIS    3$              ;BR IF IT IS
024672  005260  000100          INC     $SKI(R0)                ;INCREMENT $SKI
024676  000207          3$:     RTS     PC              ;RETURN

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

024700  005737  001342  INCMIS: TST     BADSEC          ;SEE IF BAD TRK/SEC INDICATOR SET
024704  001006                  BNE     3$              ;BR IF IT'S SET, DON'T INCREMENT COUNT
024706  026027  000102  077777        CMP     $MISPO(R0),#77777       ;IS $MISPO ALREADY AT MAXIMUM ?
024714  103002                  BHIS    3$              ;BR IF IT IS
024716  005260  000102          INC     $MISPO(R0)              ;INCREMENT $MISPO
024722  000207          3$:     RTS     PC              ;RETURN

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

024724  005737  001342  INCTOT: TST     BADSEC          ;SEE IF BAD TRK/SEC INDICATOR SET
024730  001006                  BNE     3$              ;BR IF IT'S SET, DON'T INCREMENT COUNT
024732  026027  000072  077777        CMP     $TOTAL(R0),#77777       ;IS $TOTAL ALREADY AT MAXIMUM ?
024740  103002                  BHIS    3$              ;BR IF IT IS
024742  005260  000072          INC     $TOTAL(R0)              ;INCREMENT $TOTAL
024746  000207          3$:     RTS     PC              ;RETURN
```

```
                                        ;ROUTINE TO TYPE THE TIME
                                $TIME:  TST     CLKFLG          ;CLOCK ON THE SYSTEM ?
                                        BNE     3$              ;BR IF NOT
                                        MOV     #2,2$           ;ASSUME 2 DIGITS TO TYPE
                                        MOV     HOUR,-(SP)      ;PUT 'HOURS' ON THE STACK
                                        CMP     (SP),#100.      ;100. HOURS OR MORE ?
                                        BLT     1$              ;BR IF NO
                                        INC     2$              ;TYPE 3 DIGITS
                                        CMP     (SP),#1000.     ;1000. HOURS OR MORE ?
                                        BLT     1$              ;BR IF NO
                                        INC     2$              ;TYPE 4 DIGITS
                                1$:     JSR     PC,$SE2D        ;CONVERT TO DECIMAL
                                        JSR     R5,FILLZ        ;TYPE IT
                                2$:     .WORD   0               ;NUMBER OF HOUR DIGITS TO TYPE
                                        TYPE    ,COLON          ;':'
                                        MOV     MINUTE,-(SP)    ;PUT 'MINUTES' ON THE STACK
                                        JSR     PC,$SB2D        ;CONVERT TO DECIMAL
                                        JSR     R5,FILLZ        ;TYPE IT
                                        .WORD   2               ;TYPE 2 DIGITS
                                        TYPE    ,COLON          ;':'
                                        MOV     SECOND,-(SP)    ;PUT SECONDS ON THE STACK
                                        JSR     PC,$SB2D        ;CONVERT TO DECIMAL
                                        JSR     R5,FILLZ        ;TYPE IT
                                        .WORD   2               ;TYPE 2 DIGITS
                                3$:     RTS     PC

                                        ;CLOCK HANDLER ROUTINE

                                CLOCK:  DEC     ONESEC          ;INCREMENT THE ONE SECOND COUNTER
                                        BNE     1$              ;BR IF A SECOND NOT COUNTED
                                        MOV     HZ,ONESEC       ;RESTORE THE VALUE
                                        INC     SECOND          ;COUNT THE SECOND
                                        CMP     #60.,SECOND     ;AT MAXIMUM ?
                                        BNE     1$              ;BR IF NOT
                                        CLR     SECOND          ;CLEAR THE SECOND'S COUNTER
                                        INC     INTRVL+2        ;COUNT THE PERFORMANCE SUMMARY INTERVAL
                                        INC     MINUTE          ;COUNT THE MINUTE
                                        CMP     #60.,MINUTE     ;AT MAXIMUM ?
                                        BNE     1$              ;BR IF NOT
                                        CLR     MINUTE          ;CLEAR THE MINUTE'S COUNTER
                                        INC     HOUR            ;COUNT THE HOURS
                                1$:     CMP     #50.,HZ         ;CPU RUNNING @ 50HZ ?
                                        BEQ     2$              ;BR IF YES
                                        MOV     #16.,-(SP)      ;16MS ON THE STACK @ 60HZ
                                        BR      3$
                                2$:     MOV     #20.,-(SP)      ;20MS ON THE STACK @ 50HZ
                                3$:     JSR     PC,RMTMR        ;DRIVER TIMER ROUTINE
                                        TST     INTRVL          ;DISPLAY THE PERFORMANCE SUMMARY ?
                                        BEQ     4$              ;BR IF NOT
                                        CMP     INTRVL,INTRVL+2 ;DISPLAY INTERVAL FINISHED ?
                                        BNE     4$              ;BR IF NOT
                                        MOV     #-1,STATIN      ;SET PERFORMANCE SUMMARY DISPLAY FLAG
                                        CLR     INTRVL+2        ;CLEAR THE PERFORMANCE INTERVAL COUNTER
                                4$:     RTI
```

```
                                        ;COMMAND DECODE ROUTINE
                                        ;CALL:
                                        ;       MOV     #1,CFLAG        ;'CFLAG' IS NORMALLY SET BY THE TTY SERVICE
                                        ;                               ;ROUTINE IN INTERRUPT MODE
                                        ;       JSR     PC,KSR
                                        ;       RETURN1                 ;SYSTEM BUSY RETURN
                                        ;       RETURN2                 ;RETURN AFTER KEYBOARD SERVICED

025240  005737  040140          KSR:  TST     DTJW            ;ANY DATA TRANSFERS UNDER WAY ?
025244  100375                  KSR1: BPL     KSR             ;BR IF YES
025246  104412                        SAVREG                  ;SAVE THE REGISTERS
025250  012737  000200  177776        MOV     #PR4,PS         ;SET PRIORITY TO 4
025256  005037  001340          1$:   CLR     CFLAG           ;CLEAR THE 'CONTROL C' FLAG
025262  104401  001203                TYPE    ,$CRLF          ;CR-LF
025266  004737  024750                JSR     PC,$TIME        ;TYPE THE TIME
025272  004737  033226                JSR     PC,$TKINT       ;INITIALIZE TTY KEYBOARD
025276  104401  076567                TYPE    ,ENTCOM         ;'ENTER COMMAND'

025302  104411                        RDLIN                   ;READ THE KEYBOARD
025304  012605                        MOV     (SP)+,R5        ;GET ADDRESS OF INPUT STRING
025306  005737  001340                TST     CFLAG           ;CHECK THE CONTROL C FLAG
025312  001405                        BEQ     2$              ;BR IF NO 'CONTROL C' ENTERED
025314  005737  001530                TST     ASNLST          ;ANY DRIVES ASSIGNED ?
025320  001130                        BNE     13$             ;BR IF YES
025322  000137  003634                JMP     START           ;JUMP TO START

025326  005205                  2$:   INC     R5              ;POINT TO SECOND CHARACTER
025330  122715  000124                CMPB    #'T,(R5)        ;EQ TO A 'T' ?
025334  001462                        BEQ     9$              ;YES
025336  122715  000101                CMPB    #'A,(R5)        ;EQ TO AN 'A'
025342  001410                        BEQ     3$              ;BR IF IT IS
025344  121527  000067                CMPB    (R5),#'7        ;DRIVE NUMBER GREATER THAN AN ASCII 7 ?
025350  101111                        BHI     12$             ;BR IF IT IS
025352  121527  000060                CMPB    (R5),#'0        ;DRIVE NUMBER LESS THAN AN ASCII 0 ?
025356  103506                        BLO     12$             ;BR IF IT IS
025360  142715  177770                BICB    #^C7,(R5)       ;LEAVE ONLY LOWER 3 BITS IF CHAR NOT 'A'
025364  122765  000124  177777  3$:   CMPB    #'T,-1(R5)      ;EQ TO 'T'
025372  001003                        BNE     4$              ;BR IF NOT EQ
025374  004737  026530                JSR     PC,NEWASN       ;ASSIGN DRIVE FOR TEST
025400  000500                        BR      13$             ;EXIT
025402  122765  000104  177777  4$:   CMPB    #'D,-1(R5)      ;EQ TO 'D' ?
025410  001003                        BNE     5$              ;BR IF NOT EQ
025412  004737  026330                JSR     PC,DEASGN       ;DEASSIGN DRIVE
025416  000471                        BR      13$             ;EXIT
025420  122765  000123  177777  5$:   CMPB    #'S,-1(R5)      ;EQ TO 'S'
025426  001003                        BNE     6$              ;BR IF NOT EQ
025430  004737  026436                JSR     PC,SCMND        ;TYPE STATISTICS
025434  000462                        BR      13$             ;EXIT
025436  122765  000127  177777  6$:   CMPB    #'W,-1(R5)      ;EQ TO 'W'
025444  001007                        BNE     8$              ;BR IF NOT EQ
025446  032777  000001  153500        BIT     #SW0,@SWR       ;IS SWITCH 0 SET ?
025454  001044                        BNE     11$             ;BR IF SET, CAN'T DO 'W' COMMAND
025456  004737  026552          7$:   JSR     PC,DATAPK       ;WRITE A DATA PACK
025462  000447                        BR      13$             ;EXIT
025464  122765  000122  177777  8$:   CMPB    #'R,-1(R5)      ;EQ TO 'R' ?
025472  001040                        BNE     12$             ;BR IF NOT EQ
```

```
 58 025474  004737  026540                    JSR     PC,REDAPK       ;READ A DATA PACK
 59 025500  000440                            BR      13$             ;EXIT
 60 025502  122765  000127  177777  9$:       CMPB    #'W,-1(R5)      ;WT COMMAND ?
 61 025510  001031                            BNE     12$             ;NO
 62 025512  032777  000001  153434            BIT     #SW0,@SWR       ;IS SWITCH 0 SET ?
 63 025520  001022                            BNE     11$             ;BR IF SET, CAN'T DO 'W' COMMAND
 64 025522  122765  000101  000001            CMPB    #'A,1(R5)       ;ALL DRIVES ?
 65 025530  001413                            BEQ     10$             ;YES
 66 025532  126527  000001  000067            CMPB    1(R5),#'7       ;GREAT THAN 7
 67 025540  101015                            BHI     12$             ;YES
 68 025542  126527  000001  000060            CMPB    1(R5),#'0       ;LESS THAN 0
 69 025550  103411                            BLO     12$             ;YES
 70 025552  142765  177770  000001            BICB    #^C7,1(R5)      ;CHOP OFF THE HIGHER BITS
 71 025560  004737  026564          10$:      JSR     PC,WATPAK       ;ASSIGN DRIVES WITH WT COMMAND
 72 025564  000406                            BR      13$
 73 025566  104401  076501          11$:      TYPE    ,MSWRO          ;TYPE 'CAN'T WRITE IN READ ONLY MODE'
 74 025572  000631                            BR      1$              ;TRY AGAIN
 75 025574  104401  076544          12$:      TYPE    ,INVLD          ;TYPE 'INVALID COMMAND' MESSAGE
 76 025600  000626                            BR      1$              ;TRY AGAIN
 77 025602  104413                  13$:      RESREG                  ;RESTORE R0 - R5
 78 025604  005777  153352                    TST     @$TKB           ;CLEAR THE TTY BUFFER
 79 025610  052777  000100  153342            BIS     #BIT06,@$TKS    ;SET TTY INTERRUPT ENABLE
 80 025616  005037  177776                    CLR     PS              ;SET PRIORITY BACK TO ZERO
 81 025622  000207                            RTS     PC              ;RETURN
 82
 83                                 ;ROUTINE TO PROCESS THE ASSIGN REQUEST ('T', 'R', OR 'W' COMMANDS)
 84
 85 025624  111504                  ASSIGN: MOVB    (R5),R4         ;PUT DRIVE # IN R4
 86 025626  005037  001340          1$:     CLR     CFLAG           ;CLEAR CONTROL C FLAG
 87 025632  005037  001430                  CLR     DRVPAR          ;ASSUME CHANGING DRIVE PARAMETERS
 88 025636  104401  077315                  TYPE    ,MSPRM          ;TYPE 'CHANGE DRIVE PARAMETERS ?'
 89 025642  10441?                          RDLIN                   ;READ THE ENTRY
 90 025644  012600                          MOV     (SP)+,R0        ;SAVE ADDRESS OF RESPONSE
 91 025646  005737  001340                  TST     CFLAG           ;WAS IT CONTROL C ?
 92 025652  001365                          BNE     1$              ;BR IF YES
 93 025654  105710                          TSTB    (R0)            ;WAS RESPONSE A CARRIAGE RETURN (DEFAULT 'N')?
 94 025656  001414                          BEQ     3$              ;BR IF YES
 95 025660  105760  000001                  TSTB    1(R0)           ;WAS IT TERMINATED WITH CARRIAGE RETURN ?
 96 025664  001006                          BNE     2$              ;BR IF NO
 97 025666  122710  000131                  CMPB    #'Y,(R0)        ;WAS IT A 'Y' RESPONSE ?
 98 025672  001410                          BEQ     4$              ;BR IF YES
 99 025674  122710  000116                  CMPB    #'N,(R0)        ;WAS IT A 'N' RESPONSE ?
100 025700  001403                          BEQ     3$              ;BR IF YES
101 025702  104401  076677          2$:     TYPE    ,BADENT         ;TYPE BAD ENTRY MESSAGE
102 025706  000747                          BR      1$              ;TRY AGAIN
103 025710  005237  001430          3$:     INC     DRVPAR          ;DO NOT CHANGE DRIVE PARAMETERS
104 025714  122704  000101          4$:     CMPB    #'A,R4          ;ASSIGN ALL DRIVES ?
105 025720  001431                          BEQ     ASGN2           ;BR IF YES
106
107 025722  012737  075676  031050  ASGN1:  MOV     #UNTASN,ASNMSG  ;'DRIVE ASSIGNED' MESSAGE ADDRESS
108 025730  005737  001432                  TST     XXDP            ;LOADED FROM THIS DEVICE ?
109 025734  001412                          BEQ     1$              ;BR IF NO
110 025736  123704  001432                  CMPB    XXDP,R4         ;LOADED FROM THIS DRIVE ?
111 025742  001007                          BNE     1$              ;BR IF NO
112 025744  146437  040142  001530          BICB    ATABIT(R4),ASNLST       ;DELETE THE DRIVE FROM THE ASSIGNED LIST
113 025752  012737  076011  031050          MOV     #LODEV,ASNMSG   ;'LOAD DEVICE' MESSAGE ADDRESS
114 025760  000407                          BR      2$
```

```
115 025762 136437 040142 001530 1$:     BITB    ATABIT(R4),ASNLST       ;DRIVE ALREADY ASSIGNED ?
116 025770 001003                       BNE     2$                      ;BR IF IT IS
117 025772 004737 026102                JSR     PC,ASGN3                ;SEE IF DRIVE ON THE SYSTEM
118 025776 000207                       RTS     PC                      ;RETURN
119 026000 000137 031024        2$:     JMP     ASNERR                  ;EXIT ERROR
120
121 026004 005004               ASGN2:  CLR     R4                      ;START WITH DRIVE 0
122 026006 012737 075676 031050 1$:     MOV     #UNTASN,ASNMSG          ;ERROR MESSAGE
123 026014 005737 001432                TST     XXDP                    ;LOADED FROM THIS DEVICE ?
124 026020 001412                       BEQ     2$                      ;BR IF NO
125 026022 123704 001432                CMPB    XXDP,R4                 ;LOADED FROM THIS DRIVE ?
126 026026 001007                       BNE     2$                      ;BR IF NO
127 026030 146437 040142 001530         BICB    ATABIT(R4),ASNLST       ;DELETE THE DRIVE FROM THE ASSIGNED LIST
128 026036 012737 076011 031050         MOV     #LODEV,ASNMSG           ;'LOAD DEVICE' MESSAGE ADDRESS
129 026044 000413                       BR      4$
130 026046 136437 040142 001530 2$:     BITB    ATABIT(R4),ASNLST       ;ALREADY ASSIGNED ?
131 026054 001007                       BNE     4$                      ;YES
132 026056 004737 026102                JSR     PC,ASGN3                ;ASSIGN THE DRIVE
133 026062 005204               3$:     INC     R4                      ;INCREMENT DRIVE #
134 026064 020427 000007                CMP     R4,#7                   ;ALL DRIVE CHECKED ?
135 026070 003746                       BLE     1$                      ;NO
136 026072 000207                       RTS     PC                      ;YES
137 026074 004737 031024        4$:     JSR     PC,ASNERR               ;ERROR MESSAGE
138 026100 000770                       BR      3$                      ;TO LOOP
139
140 026102 136437 040142 001530 ASGN3:  BITB    ATABIT(R4),ASNLST       ;DRIVE ALREADY ASSIGNED ?
141 026110 001060                       BNE     ASGN4                   ;BR IF IT IS
142 026112 110437 070322                MOVB    R4,GENDPB               ;GET DRIVE NUMBER
143 026116 006304                       ASL     R4                      ;MAKE R4 WORD INDEX
144 026120 016400 002044                MOV     BLKADR(R4),R0           ;PUT BLOCK'S ADDR INTO R0
145 026124 004737 015740                JSR     PC,RECAL0               ;RECALIBRATE DRIVE
146 026130 006204                       ASR     R4                      ;MAKE R4 BYTE INDEX
147 026132 105764 040036                TSTB    DRVSTA(R4)              ;DRIVE AVAILABLE?
148 026136 001453                       BEQ     ASGN7                   ;BR IF DRIVE OFFLINE OR NONEXISTENT
149 026140 100445                       BMI     ASGN6                   ;BR IF DRIVE UNSAFE
150 026142 004737 026602                JSR     PC,CLRDPB               ;CLEAR BLOCK FOR DRIVE JUST ASSIGNED
151 026146 004737 027420                JSR     PC,GETID                ;GET DRIVE (MBA) SERIAL NUMBER
152 026152 004537 027520                JSR     R5,GETADR               ;RETRIEVE BAD SECTOR FILE
153 026156 005737 001430                TST     DRVPAR                  ;CHANGE DRIVE PARAMETERS ?
154 026162 001017                       BNE     1$                      ;BR IF NO
155 026164 104401 076372                TYPE    ,DRNUM                  ;TYPE DRIVE MESSAGE
156 026170 010446                       MOV     R4,-(SP)                ;;SAVE R4 FOR TYPEOUT
    026172 104403                       TYPOS                           ;;GO TYPE--OCTAL ASCII
    026174    002                       .BYTE   2                       ;;TYPE 2 DIGIT(S)
    026175    000                       .BYTE   0                       ;;SUPPRESS LEADING ZEROS
157 026176 104401 001203                TYPE    ,$CRLF                  ;CR-LF
158 026202 004737 032472                JSR     PC,TYMBA                ;TYPE MBA (DRIVE) SERIAL NUMBER
159 026206 104401 075623                TYPE    ,TAB                    ;TYPE TAB CONTROL
160 026212 004737 032522                JSR     PC,TYPACK               ;TYPE PACK SERIAL NUMBER
161 026216 104401 001203                TYPE    ,$CRLF                  ;CR-LF
162 026222 006304               1$:     ASL     R4                      ;MAKE R4 WORD INDEX
163 026224 004737 027020                JSR     PC,DRVPRM               ;GET THE DRIVE'S ADDRESS LIMITS
164 026230 004737 030124                JSR     PC,MANTER               ;MANUALLY ENTER BAD SECTOR INFORMATION
165 026234 016464 002044 001554         MOV     BLKADR(R4),NEWUNT(R4)   ;DPB ADDRESS
166 026242 113760 001320 000026         MOVB    PACK,$PACK(R0)          ;SET READ/WRITE DATA PACK INDICATOR
167 026250 006204                       ASR     R4                      ;MAKE R4 BYTE INDEX
168 026252 000207               ASGN4:  RTS     PC                      ;RETURN
```

```
169
170 026254  012737  076001  031050  ASGN6:  MOV     #NOTSAF,ASNMSG  ;'UNSAFE' MESSAGE ADDRESS
171 026262  000137  031024                  JMP     ASNERR          ;TO ERROR ROUTINE
172
173 026266  105764  040046          ASGN7:  TSTB    DRVTYP(R4)      ;DRIVE PRESENT?
174 026272  001405                          BEQ     1$              ;BR IF NOT
175 026274  100010                          BPL     2$              ;BR IF DRIVE OFFLINE
176 026276  012737  075724  031050          MOV     #NOTRM,ASNMSG   ;ADDRESS OF 'NOT RM05/3/2' MSG
177 026304  000407                          BR      3$              ;EXIT
178 026306  012737  075745  031050  1$:     MOV     #NOTPRS,ASNMSG  ;ADDRESS OF 'NOT PRESENT' MSG
179 026314  000403                          BR      3$              ;EXIT
180 026316  012737  075633  031050  2$:     MOV     #UNTOFF,ASNMSG  ;ADDRESS OF 'DRIVE OFFLINE' MESSAGE
181 026324  000137  031024          3$:     JMP     ASNERR          ;TO ERROR ROUTINE
```

```
                                        ;'D' COMMAND (ROUTINE TO DEASSIGN A DRIVE)
                                        ;
026330  005004                  DEASGN: CLR     R4              ;START WITH DRIVE 0
026332  012703  000010                  MOV     #8.,R3          ;COUNTER
026336  122715  000101                  CMPB    #'A,(R5)        ;DEASSIGN ALL DRIVES ?
026342  001403                          BEQ     1$              ;BR IF YES
026344  111504                          MOVB    (R5),R4         ;GET DRIVE NUMBER
026346  012703  000001                  MOV     #1,R3           ;SET R3 FOR ONE UNIT
026352  136437  040142  001530  1$:     BITB    ATABIT(R4),ASNLST       ;DRIVE ASSIGNED ?
026360  001417                          BEQ     3$              ;BR IF NOT
026362  146437  040142  001530          BICB    ATABIT(R4),ASNLST       ;DELETE THE DRIVE FROM THE ASSIGNED LIST
026370  146437  040142  031662          BICB    ATABIT(R4),AUTLST       ;DELETE DRIVE FROM AUTO ASSIGN LIST
026376  006304                          ASL     R4              ;MAKE ADDR INTO A WORD INDEX
026400  016464  002044  001532          MOV     BLKADR(R4),DUNIT(R4)    ;PUT ADDRESS IN DEASSIGN LIST
026406  006204                          ASR     R4
026410  005303                  2$:     DEC     R3              ;ANY MORE DRIVES ?
026412  0014 0                         BEQ     4$              ;BR IF NOT
026414  005204                          INC     R4
026416  000755                          BR      1$
026420  012737  075654  031050  3$:     MOV     #UNTNOT,ASNMSG  ;ADDR OF 'NOT ASSIGNED' MESSAGE
026426  004737  031024                  JSR     PC,ASNERR       ;REPORT IT
026432  000766                          BR      2$
026434  000207                  4$:     RTS     PC
                                        ;'S' COMMAND (ROUTINE TO TYPE DRIVE PERFORMANCE SUMMARY)
                                        ;
026436                          SCMND:
026436  013746  001530                  MOV     ASNLST,-(SP)    ;;PUSH ASNLST ON STACK
026442  122715  000101                  CMPB    #'A,(R5)        ;ALL STATISTICS ?
026446  001416                          BEQ     2$              ;BR IF YES
026450  111504                          MOVB    (R5),R4         ;GET DRIVE NUMBER
026452  136416  040142                  BITB    ATABIT(R4),(SP) ;IS THIS DRIVE ASSIGNED ?
026456  001404                          BEQ     1$              ;BR IF NO
026460  116437  040142  001530          MOVB    ATABIT(R4),ASNLST       ;GET DRIVE ASSIGN BIT
026466  000411                          BR      3$
                                        ;
026470  012737  075654  031050  1$:     MOV     #UNTNOT,ASNMSG  ;ADDR OF 'NOT ASSIGNED' MSG
026476  004737  031024                  JSR     PC,ASNERR       ;TYPE ERROR MESSAGE
026502  000407                          BR      4$              ;EXIT
026504  105737  001530          2$:     TSTB    ASNLST          ;ANY DRIVE ASSIGNED ?
026510  001404                          BEQ     4$              ;BR IF NO
026512  004737  023550          3$:     JSR     PC,STATPR       ;TYPE ALL STATISTICS
026516  104401  076155                  TYPE    ,STAR5          ;TYPE '****...ETC'
026522                          4$:
026522  012637  001530                  MOV     (SP)+,ASNLST    ;;POP STACK INTO ASNLST
026526  000207                          RTS     PC
                                        ;'T' COMMAND (ROUTINE TO TEST A DRIVE)
                                        ;
026530  005037  001320          NEWASN: CLR     PACK            ;SET 'T' COMMAND INDICATOR
026534  000137  025624                  JMP     ASSIGN          ;GO TO THE ASSIGN ROUTINE
                                        ;'R' COMMAND (ROUTINE TO READ A DATA PACK)
                                        ;
026540  012737  000001  001320  READPK: MOV     #1,PACK         ;SET THE 'READ' INDICATOR
026546  000137  025624                  JMP     ASSIGN          ;ASSIGN THE REQUESTED DRIVE
```

```
                                        ;'W' COMMAND (ROUTINE TO WRITE A DATA PACK)

026552  012737  177777  001320  DATAPK: MOV     #-1,PACK        ;SET THE 'W' COMMAND INDICATOR
026560  000137  025624                  JMP     ASSIGN          ;ASSIGN REQUESTED DRIVE

                                        ;'WT' COMMAND (TO WRITE A PACK AND TEST A DRIVE)

026564  116515  000001          WATPAK: MOVB    1(R5),(R5)      ;ADJUST DRIVE NUMBER ADDRESS
026570  012737  177776  001320          MOV     #-2,PACK        ;PACK WRITE COMMAND
026576  000137  025624                  JMP     ASSIGN          ;JUMP TO ASSIGN ROUTINE
```

```
                                        ;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

 026602                                 CLRDPB:
 026602  010146                                MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 026604  010346                                MOV     R3,-(SP)        ;;PUSH R3 ON STACK
 026606  010446                                MOV     R4,-(SP)        ;;PUSH R4 ON STACK
 026610  010546                                MOV     R5,-(SP)        ;;PUSH R5 ON STACK
 026612  010004                                MOV     R0,R4           ;GET THE DPB ADDRESS
 026614  062704  000002                        ADD     #2,R4           ;ADDRESS OF FIRST LOCN TO BE CLEARED
 026620  012703  000012                        MOV     #<$CYL-$COMND>+2,R3     ;NUMBER OF LOCNS TO BE CLEARED
 026624  005024                         1$:    CLR     (R4)+           ;CLEAR THE LOCATION
 026626  162703  000002                        SUB     #2,R3           ;DONE CLEARING YET ?
 026632  001374                                BNE     1$              ;BR IF NO
 026634  062704  000002                        ADD     #2,R4           ;SKIP OVER THE '$REG' LOCATION
 026640  012703  000104                        MOV     #<$NEXT-$TATUS>+2,R3    ;NUMBER OF LOCNS TO BE CLEARED
 026644  005024                         2$:    CLR     (R4)+           ;CLEAR THE LOCATION
 026646  162703  000002                        SUB     #2,R3           ;DONE CLEARING YET ?
 026652  001374                                BNE     2$              ;BR IF NO
 026654  062704  002004                        ADD     #<$MBASN-$FIRST>,R4     ;SKIP OVER FIRST FLAG, MIN/MAX ADRS
                                                                        ;LIMITS AND BAD SECTOR TABLE
 026660  012703  000064                        MOV     #<$RMCS3-$MBASN>+2,R3   ;NUMBER OF LOCNS TO BE CLEARED
 026664  012764  177777  177776                MOV     #-1,-2(R4)      ;INITIALIZE TERMINATOR FOR BAD SECTOR TABLE
 026672  005024                         3$:    CLR     (R4)+           ;CLEAR A LOCATION
 026674  162703  000002                        SUB     #2,R3           ;DONE CLEARING YET ?
 026700  001374                                BNE     3$              ;BR IF NO
 026702  113760  001502  000024                MOVB    BEGCOD,$CODE(R0)        ;INITIAL COMMAND CODE
 026710  013701  001502                        MOV     BEGCOD,R1       ;GET THE ACTUAL OP CODE
 026714  116160  002064  000002                MOVB    COMTBL(R1),$COMND(R0)   ;OPERATION CODE
 026722  113760  001500  000030                MOVB    BEGPAT,$PATTC(R0)       ;PATTERN CODE
 026730  106360  000030                        ASLB    $PATTC(R0)      ;CONVERT CODE TO A TABLE INDEX
 026734  013760  001504  000020                MOV     BEGWC,$WRDL(R0) ;BEGINNING WORD COUNT
 026742  013760  001504  000004                MOV     BEGWC,$WCNT(R0) ;VALUE FOR DATA TRANSFER
 026750  005460  000004                        NEG     $WCNT(R0)       ;MAKE IT INTO 2'S COMPLEMENT
 026754  012760  000400  000022                MOV     #256.,$SSEC(R0) ;INITIAL VALUE OF SECTOR SIZE
 026762  012760  000001  000104                MOV     #1,$PASSC(R0)   ;PRESET PASS COUNT TO 1
 026770  132760  000001  000024                BITB    #1,$CODE(R0)    ;HEADER COMMAND ?
 026776  001403                                BEQ     4$              ;BR IF NOT
 027000  062760  000002  000022                ADD     #2,$SSEC(R0)    ;ADD HEADER SIZE TO SECTOR SIZE
 027006                                 4$:
 027006  012605                                MOV     (SP)+,R5        ;;POP STACK INTO R5
 027010  012604                                MOV     (SP)+,R4        ;;POP STACK INTO R4
 027012  012603                                MOV     (SP)+,R3        ;;POP STACK INTO R3
 027014  012601                                MOV     (SP)+,R1        ;;POP STACK INTO R1
 027016  000207                                RTS     PC              ;RETURN

                                        ;ROUTINE TO GET ADDRESS LIMITS FROM THE OPERATOR
                                        ;CALL:
                                        ;       MOV     #DPB,R0         ;DPB ADDRESS
                                        ;       JSR     PC,DRVPRM       ;CALL ROUTINE
                                        ;
```

```
                                        ;R0 = DPB ADDRESS BEFORE CALLING THE ROUTINE

027020  010346                DRVPRM: MOV     R3,-(SP)        ;SAVE R3
027022  010446                        MOV     R4,-(SP)        ;SAVE R4
027024  105737  001150                TSTB    $AUTOB          ;RUNNING IN AUTO MODE ?
027030  001010                        BNE     1$              ;BR IF YES
027032  005737  001336                TST     CHGADR          ;PROGRAM STARTED AT 200 ?
027036  003005                        BGT     1$              ;BR IF YES
027040  005737  001430                TST     DRVPAR          ;CHANGE DRIVE PARAMETERS ?
027044  001002                        BNE     1$              ;BR IF NO
027046  104401  076610                TYPE    ,ENTLMT         ;'ENTER ADDRESS LIMITS'

027052  004737  027346        1$:     JSR     PC,GETLMT       ;GET ADDRESS LIMITS
027056  062760  177777  000122        ADD     #-1,$FIRST(R0)  ;SEE IF FIRST TIME STARTED
027064  103417                        BCS     2$              ;BR IF NOT
027066  013760  001422  000124        MOV     CYLIMT,MAXCYL(R0)       ;LOAD MAXIMUM CYLINDER
027074  013760  001426  000130        MOV     TRKLMT,MAXTRK(R0)       ;LOAD MAXIMUM TRACK
027102  013760  001424  000134        MOV     SECLMT,MAXSEC(R0)       ;LOAD MAXIMUM SECTOR
027110  005060  000126                CLR     MINCYL(R0)      ;CLEAR MINIMUM CYLINDER
027114  005060  000132                CLR     MINTRK(R0)      ;CLEAR MINIMUM TRACK
027120  005060  000136                CLR     MINSEC(R0)      ;CLEAR MINIMUM SECTOR

027124  105737  001150        2$:     TSTB    $AUTOB          ;RUNNING IN AUTO MODE ?
027130  001072                        BNE     5$              ;BR IF YES
027132  005737  001336                TST     CHGADR          ;PROGRAM STARTED AT 200 ?
027136  003067                        BGT     5$              ;BR IF YES
027140  005737  001430                TST     DRVPAR          ;CHANGE DRIVE PARAMETERS ?
027144  001064                        BNE     5$              ;BR IF NO
027146  016403  077734                MOV     TABLE(R4),R3    ;PARAMETER TABLE ADDRESS
027152  013763  001422  000002        MOV     CYLIMT,2(R3)    ;LOAD CYLINDER LIMIT FOR MINCYL
027160  013763  001422  000010        MOV     CYLIMT,10(R3)   ;LOAD CYLINDER LIMIT FOR MAXCYL
027166  013763  001426  000016        MOV     TRKLMT,16(R3)   ;LOAD TRACK LIMIT FOR MINTRK
027174  013763  001426  000024        MOV     TRKLMT,24(R3)   ;LOAD TRACK LIMIT FOR MAXTRK
027202  013763  001424  000032        MOV     SECLMT,32(R3)   ;LOAD SECTOR LIMIT FOR MINSEC
027210  013763  001424  000040        MOV     SECLMT,40(R3)   ;LOAD SECTOR LIMIT FOR MAXSEC
027216  004737  030700                JSR     PC,PARENT       ;GET THE DRIVE'S PARAMETERS

027222  016003  000126                MOV     MINCYL(R0),R3   ;STORE MINCYL VALUE
027226  016004  000124                MOV     MAXCYL(R0),R4   ;STORE MAXCYL VALUE
027232  020304                        CMP     R3,R4           ;IS MIN. LESS THAN OR EQUAL TO MAX. ?
027234  003404                        BLE     3$              ;BR IF YES
027236  010360  000124                MOV     R3,MAXCYL(R0)   ;SWAP MIN. TO MAX.
027242  010460  000126                MOV     R4,MINCYL(R0)   ;SWAP MAX. TO MIN.
027246  016003  000132        3$:     MOV     MINTRK(R0),R3   ;STORE MINTRK VALUE
027252  016004  000130                MOV     MAXTRK(R0),R4   ;STORE MAXTRK VALUE
027256  020304                        CMP     R3,R4           ;IS MIN. LESS THAN OR EQUAL TO MAX. ?
027260  003404                        BLE     4$              ;BR IF YES
027262  010360  000130                MOV     R3,MAXTRK(R0)   ;SWAP MIN. TO MAX.
027266  010460  000132                MOV     R4,MINTRK(R0)   ;SWAP MAX. TO MIN.
027272  016003  000136        4$:     MOV     MINSEC(R0),R3   ;STORE MINSEC VALUE
027276  016004  000134                MOV     MAXSEC(R0),R4   ;STORE MAXSEC VALUE
027302  020304                        CMP     R3,R4           ;IS MIN. LESS THAN OR EQUAL TO MAX. ?
027304  003404                        BLE     5$              ;BR IF YES
027306  010360  000134                MOV     R3,MAXSEC(R0)   ;SWAP MIN. TO MAX.
027312  010460  000136                MOV     R4,MINSEC(R0)   ;SWAP MAX. TO MIN.

027316  016000  000126  000012 5$:     MOV     MINCYL(R0),$CYL(R0)     ;INITIAL CYLINDER VALUE
```

```
111 027324  116060  000132  000011          MOVB    MINTRK(R0),$TRK(R0)     ;INITIAL TRACK VALUE
112 027332  116060  000136  000010          MOVB    MINSEC(R0),$SEC(R0)     ;INITIAL SECTOR VALUE
113 027340  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
    027342  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
114 027344  000207                          RTS     PC              ;RETURN
115
116                                 ;ROUTINE TO GET THE ADDRESS LIMITS FOR THE CURRENT DRIVE TYPE
117                                 ;CALL:
118                                 ;       JSR     PC,GETLMT       ;CALL ROUTINE
119                                 ;
120                                 ;R0 = DPB ADDRESS BEFORE CALLING THE ROUTINE
121
122 027346                          GETLMT:
    027346  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
123 027350  005001                          CLR     R1              ;START FRESH
124 027352  012737  001465  001422          MOV     #821.,CYLLMT    ;GET CYLINDER LIMIT
125 027360  012737  000037  001424          MOV     #31.,SECLMT     ;GET SECTOR LIMIT
126 027366  111001                          MOVB    (R0),R1         ;GET DRIVE NUMBER
127 027370  012737  000022  001426          MOV     #18.,TRKLMT     ;ASSUME LAST TRACK FOR AN RM05
128 027376  122761  000007  040046          CMPB    #7,DRVTYP(R1)   ;IS DRIVE AN RM05 ?
129 027404  001403                          BEQ     1$              ;BR IF YES
130 027406  012737  000004  001426          MOV     #4,TRKLMT       ;GET LAST TRACK FOR AN RM03/2
131 027414                          1$:
    027414  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
132 027416  000207                          RTS     PC              ;RETURN
```

```
                                ;ROUTINE TO GET THE DRIVE (MBA) SERIAL NUMBER FROM RMSN REGISTER
                                ;THIS NUMBERS CONTAINED IN THE REGISTER ARE ONLY THE 4 LSD'S OF THE
                                ;SERIAL NUMBER.
                                ;CALL:
                                ;       MOV     #DPB,R0         ;DPB ADDRESS
                                ;       JSR     PC,GETID        ;CALL ROUTINE
                                ;
                                ;R0 - DPB ADDRESS BEFORE CALLING THE ROUTINE

027420                          GETID:
027420  010046                         MOV     R0,-(SP)        ;;PUSH R0 ON STACK
027422  010146                         MOV     R1,-(SP)        ;;PUSH R1 ON STACK
027424  010246                         MOV     R2,-(SP)        ;;PUSH R2 ON STACK
027426  010546                         MOV     R5,-(SP)        ;;PUSH R5 ON STACK
027430  010002                         MOV     R0,R2           ;GET INDEX TO DPB
027432  004737  045230                 JSR     PC,SVRH70       ;SAVE ALL REGISTERS
027436  012702  000004                 MOV     #4,R2           ;FOUR DIGITS TO STORE
027442  016001  002166                 MOV     $RMSN(R0),R1    ;SERIAL NUMBER
027446  005005                 1$:     CLR     R5              ;ZERO
027450  006101                         ROL     R1              ;PUT THE NEXT DIGIT
027452  006105                         ROL     R5              ;INTO R5
027454  006101                         ROL     R1
027456  006105                         ROL     R5
027460  006101                         ROL     R1
027462  006105                         ROL     R5
027464  006101                         ROL     R1
027466  006105                         ROL     R5
027470  062705  000060                 ADD     #'0,R5          ;MAKE IT ASCII
027474  110560  002126                 MOVB    R5,$MBASN(R0)   ;SAVE DRIVE (MBA) SERIAL NUMBER DIGIT
027500  005200                         INC     R0              ;GET NEXT INDEX FOR DRIVE (MBA) SERIAL NUMBER
027502  005302                         DEC     R2              ;ALL DIGITS TYPED?
027504  003360                         BGT     1$              ;NO -- BRANCH
027506  012605                         MOV     (SP)+,R5        ;;POP STACK INTO R5
027510  012602                         MOV     (SP)+,R2        ;;POP STACK INTO R2
027512  012601                         MOV     (SP)+,R1        ;;POP STACK INTO R1
027514  012600                         MOV     (SP)+,R0        ;;POP STACK INTO R0
027516  000207                         RTS     PC              ;RETURN
```

```
                        .SBTTL  READ DEC144 FILE

                        ;THIS ROUTINE IS USED TO READ THE DEC144 BAD SECTOR FILE FROM CYLINDER
                        ;558. TRACK 13. AND TO STORE THE FILE IN IT'S RESPECTIVE DPB TABLE.
                        ;THE DPB TABLE HAS ENOUGH ROOM TO SAVE THE ENTIRE MFG AND USR PORTIONS
                        ;OF THE DEC144 FILE. (MFG=126. ENTRIES AND USR=126. ENTRIES) EVERY TIME
                        ;THE DRIVE IS ASSIGNNED THE DEC144 FILE IS READ TO DETERMINE THE STATUS
                        ;OF THE CURRENT PACK SERIAL NUMBER. BUT, IN ORDER TO INITIALIZE THE
                        ;BAD SECTOR ENTRY TABLE, AT LEAST ONE OF FOLLOWING STATEMENTS MUST BE VALID.
                        ;
                        ;1)     FIRST TIME PROGRAM WAS STARTED
                        ;   OR
                        ;2)     LOCATION 'BADBLK' IS EQUAL TO 1
                        ;   OR
                        ;3)     LOCATION 'BADBLK' IS EQUAL TO 0 AND THE PACK
                        ;       SERIAL NUMBER CHANGED SINCE THE LAST TIME IT WAS
                        ;       READ. (DEFAULT)
                        ;
                        ;NOTE: IF THE SERIAL NUMBER HAS CHANGED, THIS MOST LIKELY MEANS THAT THE
                        ;      PACK OR DRIVE HAD BEEN REPLACED WHILE THE DRIVE WAS DEASSIGNED.
                        ;
                        ;THIS ROUTINE CHECKS THAT THE TWO SERIAL NUMBER WORDS ARE NOT ZERO
                        ;AND ARE POSITIVE NUMBERS. ALSO, WORDS 3 AND 4 ARE CHECKED TO BE ALL
                        ;ZERO WORDS. IF THE DEC144 FILE DOES NOT COMPLY WITH THIS STUCTURE,
                        ;AN ERROR MESSAGE IS TYPED AND THE ROUTINE IS EXITED.
                        ;
                        ;CALL
                        ;       MOV     #DPB,R0         ;DPB ADDRESS
                        ;       JSR     R5,GETADR       ;READ DEC144 BAD SECTOR FILES
                        ;
                        ;R0 - DPB ADDRESS BEFORE CALLING THE ROUTINE

                        GETADR:
027520                                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
027520  010146
027522  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
027524  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
027526  004737  027346                  JSR     PC,GETLMT       ;GET ADDRESS LIMITS
027532  010001                          MOV     R0,R1           ;DPB ADDRESS
027534  062701  000144                  ADD     #$BDSEC,R1      ;ADDRESS OF BAD SECTOR TABLE
027540  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
027542  111037  070322                  MOVB    (R0),GENDPB     ;DRIVE NUMBER
027546  012737  001466  070334          MOV     #822.,GENDPB+$CYL       ;LAST CYLINDER
027554  113737  001426  070333          MOVB    TRKLMT,GENDPB+$TRK      ;GET LAST TRACK
027562  112737  000000  070332          MOVB    #0,GENDPB+$SEC  ;GET STARTING SECTOR OF 16 BIT MFG FILE
027570  012737  177400  070326          MOV     #-256.,GENDPB+$WCNT     ;ONE SECTOR WORD COUNT
027576  112737  000171  070324          MOVB    #RDDAT,GENDPB+$COMND    ;READ DATA COMMAND
027604  012737  000010  001270          MOV     #8.,$CDW2       ;GET LAST SECTOR OF 16 BIT MFG FILE
027612  012703  101066          1$:     MOV     #CYLNDR,R3      ;GET READ BUFFER ADDRESS
027616  004037  040714                  JSR     R0,RM05         ;READ CURRENT SECTOR
027622  070322                          GENDPB
027624  000772                          BR      1$              ;WAIT FOR QUE
027626  005737  070340          2$:     TST     GENDPB+$STATUS  ;READ DONE YET ?
027632  001775                          BEQ     2$              ;BR IF NO
027634  100010                          BPL     3$              ;BR IF NO ERROR, ELSE
027636  062737  000002  070332          ADD     #2,GENDPB+$SEC  ;INCREMENT NEXT SECTOR TO READ
027644  123737  001270  070332          CMPB    $CDW2,GENDPB+$SEC       ;WERE ALL SECTORS TRIED ?
027652  103357                          BHIS    1$              ;BR IF NO
```

```
 55 027654  000470                    BR      10$             ;BR IF UNSUCCESSFUL ON RETRIES
 56 027656  005723             3$:    TST     (R3)+           ;ARE LSB'S OF SERIAL NUMBER VALID ?
 57 027660  005723                    TST     (R3)+           ;ARE MSB'S OF SERIAL NUMBER VALID ?
 58 027662  100462                    BMI     9$              ;BR IF MINUS
 59 027664  001003                    BNE     4$              ;BR IF PLUS
 60 027666  005763  177774            TST     -4(R3)          ;ARE SERIAL NUMBERS ZERO ?
 61 027672  001456                    BEQ     9$              ;BR IF YES
 62 027674  005723             4$:    TST     (R3)+           ;IS 3RD WORD ALL 0'S ?
 63 027676  001054                    BNE     9$              ;BR IF NO
 64 027700  005723                    TST     (R3)+           ;IS 4TH WORD ALL 0'S ?
 65 027702  001052                    BNE     9$              ;BR IF NO
 66 027704  123727  070332  000012    CMPB    GENDPB+$SEC,#10.        ;READING USR BAD FILE ?
 67 027712  103021                    BHIS    6$              ;BR IF YES
 68 027714  005737  001476            TST     BADBLK          ;INIT. BAD SECTOR TABLE ENTRIES ?
 69 027720  001010                    BNE     5$              ;BR IF YES
 70 027722  026360  177770  000140    CMP     -10(R3),$PSNL(R0)       ;ARE LSB'S OF S/N SAME AS BEFORE ?
 71 027730  001004                    BNE     5$              ;BR IF NO
 72 027732  026360  177772  000142    CMP     -6(R3),$PSNM(R0)        ;ARE MSB'S OF S/N SAME AS BEFORE ?
 73 027740  001464                    BEQ     13$             ;BR IF YES
 74 027742  016360  177770  00014(    5$:    MOV     -10(R3),$PSNL(R0)       ;STORE PACK SERIAL NUMBER
 75 027750  016360  177772  000142           MOV     -6(R3),$PSNM(R0)
 76
 77 027756  012702  000176             6$:    MOV     #126.,R2        ;NUMBER OF ENTRIES PER FILE (MFG/USR)
 78 027762  012321                     7$:    MOV     (R3)+,(R1)+     ;STORE BAD CYLINDER ADDRESS
 79 027764  012321                            MOV     (R3)+,(R1)+     ;STORE BAD TRK/SEC ADDRESS
 80 027766  005302                            DEC     R2              ;DONE WITH ENTRIES ?
 81 027770  001374                            BNE     7$              ;BR IF NO
 82 027772  123727  070332  000012            CMPB    GENDPB+$SEC,#10.        ;USR BAD FILE DONE YET ?
 83 030000  103044                            BHIS    13$             ;BR IF YES
 84 030002  112737  000012  070332            MOVB    #10.,GENDPB+$SEC        ;GET STARTING SECTOR OF USR FILE
 85 030010  012737  000036  001270            MOV     #30.,$CDW2      ;GET LAST SECTOR OF USR FILE
 86 030016  011601                            MOV     (SP),R1         ;GET BEGINNING OF $BDSEC TABLE
 87 030020  005721                     8$:    TST     (R1)+           ;IS THIS TERMINATOR ?
 88 030022  100376                            BPL     8$              ;BR IF NO
 89 030024  005741                            TST     -(R1)           ;FOUND TERMINATOR, BACKUP 1 WORD
 90 030026  000671                            BR      1$
 91
 92 030030  104401  077000             9$:    TYPE    ,MERR2          ;REPORT, INVALID DEC144 FILE STRUCTURE
 93 030034  000402                            BR      11$
 94 030036  104401  076720            10$:    TYPE    ,MERR1          ;REPORT, FAILED TO RETRIEVE DEC144 FILES
 95 030042  104401  075625            11$:    TYPE    ,UNTMSG         ;TYPE 'ON DRIVE'
 96 030046  011046                            MOV     (R0),-(SP)      ;SAVE (R0) FOR TYPEOUT
    030050  104403                            TYPOS                   ;GO TYPE--OCTAL ASCII
    030052     002                            .BYTE   2               ;TYPE 2 DIGIT(S)
    030053     000                            .BYTE   0               ;SUPPRESS LEADING ZEROS
 97 030054  104401  001203                    TYPE    ,$CRLF          ;CR-LF
 98 030060  012601                            MOV     (SP)+,R1        ;POP STACK INTO R1
 99 030062  012702  000374                    MOV     #252.,R2        ;TOTAL NUMBER OF ENTRIES ALLOWED
100 030066  012721  177777            12$:    MOV     #-1,(R1)+       ;INITIALIZE CYLINDER LOCATIONS TO -1
101 030072  012721  177777                    MOV     #-1,(R1)+       ;INITIALIZE TRK/SEC LOCATIONS TO -1
102 030076  005302                            DEC     R2              ;DONE YET ?
103 030100  001372                            BNE     12$             ;BR IF NO
104 030102  012760  177777  000142            MOV     #-1,$PSNM(R0)   ;INDICATE SERIAL NUMBER IS UNKNOWN
105 030110  000401                            BR      14$
106 030112  005726                    13$:    TST     (SP)+           ;RESTORE STACK
107 030114                            14$:
    030114  012603                            MOV     (SP)+,R3        ;POP STACK INTO R3
```

```
        030116  012602                  MOV     (SP)+,R2        ;;POP STACK INTO R2
        030120  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
    108 030122  000205                  RTS     R5              ;EXIT
```

```
                        .SBTTL  ENTER BAD SECTOR ROUTINE
                        ;ROUTINE TO ENTER BAD SECTOR INFORMATION MANUALLY
                        ;CALL:
                        ;       MOV     #DPB,R0         ;DPB ADDRESS
                        ;       JSR     PC,MANTER       ;CALL ROUTINE
                        ;
                        ;R0 = DPB ADDRESS BEFORE CALLING THE ROUTINE

030124                  MANTER:
030124  010146                 MOV     R1,-(SP)        ;;PUSH R1 ON STACK
030126  010246                 MOV     R2,-(SP)        ;;PUSH R2 ON STACK
030130  010346                 MOV     R3,-(SP)        ;;PUSH R3 ON STACK
030132  010446                 MOV     R4,-(SP)        ;;PUSH R4 ON STACK
030134  105737  001150         TSTB    $AUTOB          ;RUNNING IN AUTO MODE ?
030140  001162                 BNE     19$             ;BRANCH IF SO
030142  005737  000430         TST     DRVPAR          ;CHANGE DRIVE PARAMETERS ?
030146  001157                 BNE     19$             ;BR IF NO
030150  005037  001340 1$:     CLR     CFLAG           ;CLEAR THE CONTROL-C FLAG
030154  104401  076640         TYPE    ,ENTADR         ;MESSAGE TO ENTER...

030160  012704  000144         MOV     #$BDSEC,R4      ;INDEX VALUE OF TABLE ADDRESS
030164  060004                 ADD     R0,R4           ;TABLE STARTING ADDRESS
030166  012701  000374         MOV     #<126.*2>,R1    ;256. TOTAL BAD SECTORS ALLOWED
030172  022714  177777 2$:     CMP     #-1,(R4)        ;ENTRY IN THE TABLE ?
030176  001407                 BEQ     3$              ;BRANCH IF SO
030200  062704  000004         ADD     #4,R4           ;ADJUST THE TABLE ENTRY POINTER
030204  005301                 DEC     R1              ;DECREMENT THE BAD SECTOR COUNT
030206  001371                 BNE     2$              ;BR IF TO NEXT ENTRIES POSITION
030210  104401  077044         TYPE    ,MSFULL         ;TYPE 'BAD SECTOR TABLE IS FULL'
030214  000534                 BR      19$             ;EXIT..

030216  010146         3$:     MOV     R1,-(SP)        ;SAVE THE COUNTER AND FIRST
030220  010446                 MOV     R4,-(SP)        ;ENTRY POINTER PAIR
030222  012714  177777 4$:     MOV     #-1,(R4)        ;RESET CYLINDER TO -1
030226  012764  177777  000002 MOV     #-1,2(R4)       ;RESET TRACK/SECTOR FIELD TO -1
030234  104401  077102         TYPE    ,MSGCTS         ;TYPE 'CYL,TRK,SEC = '
030240  104411                 RDLIN                   ;READ THE ADDRESS
030242  012601                 MOV     (SP)+,R1        ;READ IN TEXT ADDRESS
030244  005737  001340         TST     CFLAG           ;CONTROL-C ENTERED ?
030250  001011                 BNE     5$              ;BRANCH IF YES
030252  105761  000001         TSTB    1(R1)           ;WAS IT TERMINATED WITH CARRIAGE RETURN ?
030256  001021                 BNE     7$              ;BR IF NO
030260  122711  000114         CMPB    #'L,(R1)        ;WAS CHARACTER AN 'L' ?
030264  001016                 BNE     7$              ;BR IF NO
030266  004737  030520         JSR     PC,TYLIST       ;TYPE BAD SECTOR LIST FOR USER
030272  000753                 BR      4$
030274  012604         5$:     MOV     (SP)+,R4        ;RETRIEVE THE ENTRY POINTER
030276  012601                 MOV     (SP)+,R1        ;RETRIEVE THE COUNT
030300  012724  177777 6$:     MOV     #-1,(R4)+       ;RESET THE TABLE
030304  012724  177777         MOV     #-1,(R4)+       ;TO -1
030310  005301                 DEC     R1              ;ALL DONE ?
030312  001372                 BNE     6$              ;BRANCH IF NOT
030314  104401  077232         TYPE    ,ALOST          ;TYPE ' * ALL CURRENT ENTRIES LOST *'
030320  000713                 BR      1$              ;ENTER AGAIN FROM THE FIRST POINTER

030322          7$:
```

```
                030322  013702  001422                  MOV     CYLIMT,R2       ;UPPER LIMIT OF INPUT
                030326  004537  032772                  JSR     R5,CK.DIG       ;CHECK THE DIGIT(S)
                030332  030502                          18$                     ;CARRIAGE RETURN ONLY ENTERED
                030334  030502                          18$                     ;PERIOD ONLY ENTERED
                030336  030474                          17$                     ;ILLEGAL INPUT
                030340  030346                          8$                      ;TERMINATED WITH A CARRIAGE RETURN
                030342  030356                          10$                     ;TERMINATED WITH A ','
                030344  030352                          9$                      ;TERMINATED WITH A '.'
     54         030346  010214                  8$:     MOV     R2,(R4)         ;CYLINDER ADDRESS
     55         030350  000444                          BR      16$             ;FINISH WITH THE CURRENT ADDRESS
     56         030352  010214                  9$:     MOV     R2,(R4)         ;CYLINDER ADDRESS
     57         030354  000452                          BR      18$             ;EXIT,PERIOD ENTERED
     58         030356  010214                  10$:    MOV     R2,(R4)         ;CYLINDER ADDRESS FOLLOWED BY ','
     59
     60         030360  013702  001426                  MOV     TRKLMT,R2       ;UPPER LIMIT OF INPUT
                030364  004537  032772                  JSR     R5,CK.DIG       ;CHECK THE DIGIT(S)
                030370  030502                          18$                     ;CARRIAGE RETURN ONLY ENTERED
                030372  030502                          18$                     ;PERIOD ONLY ENTERED
                030374  030474                          17$                     ;ILLEGAL INPUT
                030376  030404                          11$                     ;TERMINATED WITH A CARRIAGE RETURN
                030400  030420                          13$                     ;TERMINATED WITH A ','
                030402  030412                          12$                     ;TERMINATED WITH A '.'
     61         030404  110264  000003          11$:    MOVB    R2,3(R4)        ;TRACK ADDRESS
     62         030410  000424                          BR      16$             ;TRACK NUMBER FOLLOWED BY CR
     63         030412  110264  000003          12$:    MOVB    R2,3(R4)        ;TRACK ADDRESS
     64         030416  000431                          BR      18$             ;EXIT, TRACK NUMBER FOLLOWED BY '.'
     65         030420  110264  000003          13$:    MOVB    R2,3(R4)        ;TRACK ADDRESS FOLLOWED BY ','
     66
     67         030424  013702  001424                  MOV     SECLMT,R2       ;UPPER LIMIT OF INPUT
                030430  004537  032772                  JSR     R5,CK.DIG       ;CHECK THE DIGIT(S)
                030434  030502                          18$                     ;CARRIAGE RETURN ONLY ENTERED
                030436  030502                          18$                     ;PERIOD ONLY ENTERED
                030440  030474                          17$                     ;ILLEGAL INPUT
                030442  030456                          15$                     ;TERMINATED WITH A CARRIAGE RETURN
                030444  030474                          17$                     ;TERMINATED WITH A ','
                030446  030450                          14$                     ;TERMINATED WITH A '.'
     68         030450  110264  000002          14$:    MOVB    R2,2(R4)        ;SECTOR ADDRESS
     69         030454  000412                          BR      18$             ;EXIT,SECTOR ADDRESS FOLLOWED BY '.'
     70         030456  110264  000002          15$:    MOVB    R2,2(R4)        ;SECTOR ADDRESS
     71
     72         030462  005303                  16$:    DEC     R3              ;MORE ENTRYS ?
     73         030464  001406                          BEQ     18$             ;BRANCH IF EXHAUSTED
     74         030466  062704  000004                  ADD     #4,R4           ;ADJUST FOR THE NEXT TABLE ENTRY
     75         030472  000653                          BR      4$              ;ENTER NEXT SECTOR ADDRESS
     76
     77         030474  104401  076677          17$:    TYPE    ,BADENT         ;MESSAGE BAD ENTRY
     78         030500  000650                          BR      4$              ;ENTER SECTOR ADDRESS AGAIN
     79         030502  062706  000004          18$:    ADD     #4,SP           ;CLEAR OFF THE STACK POINT
     80         030506                          19$:
                030506  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
                030510  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
                030512  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
                030514  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
     81         030516  000207                          RTS     PC              ;EXIT
     82
     83                                         .SBTTL  TYPE BAD SECTOR LIST
     84
```

```
                                        ;ROUTINE TO LIST BAD SECTORS ON THE TERMINAL IN DECIMAL NUMBERS
                                        ;FORMAT IS: CYL,TRK,SEC
                                        ;CALL:
                                        ;       MOV     #DPB,R0         ;DPB ADDRESS
                                        ;       JSR     PC,TYLIST       ;CALL ROUTINE
                                        ;
                                        ;R0 = DPB ADDRESS BEFORE CALLING THE ROUTINE

030520                                  TYLIST:
030520  010146                                 MOV     R1,-(SP)        ;;PUSH R1 ON STACK
030522  104401  077166                         TYPE    ,LSTHDR         ;TYPE 'DEC144 AND MANUAL BAD SECTOR LIST'
030526  012701  000144                         MOV     #$BDSEC,R1      ;INDEX VALUE OF TABLE ADDRESS
030532  060001                               ADD     R0,R1           ;TABLE STARTING ADDRESS
030534  010146                               MOV     R1,-(SP)        ;SAVE ADDRESS FOR LATER
030536  022711  177777                 1$:   CMP     #-1,(R1)        ;TERMINATOR OR NO ENTRY IN THE TABLE ?
030542  001444                               BEQ     6$              ;BRANCH IF YES
030544  011146                               MOV     (R1),-(SP)      ;GET CYLINDER NUMBER
030546  004737  033130                       JSR     PC,$SB2D        ;CONVERT NUMBER
030552  004737  032134                       JSR     PC,SUPRSL       ;LEFT JUSTIFY AND TYPE
030556  005046                               CLR     -(SP)           ;CLEAR HI BYTE AND PUSH STACK
030560  116116  000003                       MOVB    3(R1),(SP)      ;GET TRACK NUMBER
030564  100407                               BMI     2$              ;BR IF ALL BAD
030566  104401  075617                       TYPE    ,COMMA          ;TYPE ','
030572  004737  033130                       JSR     PC,$SB2D        ;CONVERT NUMBER
030576  004737  032134                       JSR     PC,SUPRSL       ;LEFT JUSTIFY AND TYPE
030602  000401                               BR      3$
030604  005726                         2$:   TST     (SP)+           ;RESTORE STACK
030606  005046                         3$:   CLR     -(SP)           ;CLEAR HI BYTE AND PUSH STACK
030610  116116  000002                       MOVB    2(R1),(SP)      ;GET SECTOR NUMBER
030614  100407                               BMI     4$              ;BR IF ALL BAD
030616  104401  075617                       TYPE    ,COMMA          ;TYPE ','
030622  004737  033130                       JSR     PC,$SB2D        ;CONVERT NUMBER
030626  004737  032134                       JSR     PC,SUPRSL       ;LEFT JUSTIFY AND TYPE
030632  000401                               BR      5$
030634  005726                         4$:   TST     (SP)+           ;RESTORE STACK
030636  104401  001203                 5$:   TYPE    ,$CRLF          ;CR-LF
030642  062701  000004                       ADD     #4,R1           ;INCREMENT POINTER
030646  005737  001340                       TST     CFLAG           ;CONTROL-C ENTERED ?
030652  001731                               BEQ     1$              ;BRANCH IF NO
030654  022601                         6$:   CMP     (SP)+,R1        ;ANY ENTRIES ?
030656  001002                               BNE     7$              ;BR IF YES
030660  104401  077146                       TYPE    ,NOENTY         ;TYPE 'NO ENTRIES'
030664  104401  001203                 7$:   TYPE    ,$CRLF          ;CR-LF
030670  005037  001340                       CLR     CFLAG           ;CLEAR CONTROL FLAG
030674  012601                               MOV     (SP)+,R1        ;;POP STACK INTO R1
030676  000207                               RTS     PC              ;RETURN
```

```
                                .SBTTL  PARAMETER ENTRY ROUTINE

                                ;PARAMETER ENTRY ROUTINE
                                ;CALL:
                                ;       MOV     #ADR,R3         ;PARAMETER TABLE ADDRESS
                                ;       JSR     PC,PARENT       ;GET THE PARAMETERS

 030700  010346                 PARENT: MOV     R3,-(SP)        ;SAVE THE PARAMETER TABLE ADDRESS
 030702  005037  001340                 CLR     CFLAG           ;CLEAR THE 'CONTROL C' FLAG
 030706  012337  030716         1$:     MOV     (R3)+,2$        ;ADDRESS OF PARAMETER NAME
 030712  001442                         BEQ     7$              ;BR IF AT END OF TABLE
 030714  104401                         TYPE                    ;TYPE THE PARAMETER NAME
 030716  000000                 2$:     .WORD   0               ;ADDRESS OF PARAMETER NAME TEXT
 030720  012302                         MOV     (R3)+,R2        ;MAXIMUM PARAMETER VALUE
 030722  012305                         MOV     (R3)+,R5        ;ADDRESS OF PARAMETER
 030724  011546                         MOV     (R5),-(SP)      ;CURRENT VALUE OF PARAMETER
 030726  104405                         TYPDS                   ;TYPE THE CURRENT VALUE OF THE PARAMETER
 030730  104401  077464                 TYPE    ,SLASH          ;' / '
 030734  104411                         RDLIN                   ;READ THE KEYBOARD
 030736  012601                         MOV     (SP)+,R1        ;INPUT ASCII STRING ADDRESS
 030740  005737  001340                 TST     CFLAG           ;'CONTROL C' ENTERED ?
 030744  001021                         BNE     6$              ;BR IF IT WAS
 030746  004537  032772                 JSR     R5,CK.DIG       ;CHECK THE DIGIT(S)
 030752  030706                         1$                      ;CARRIAGE RETURN ONLY ENTERED
 030754  031020                         7$                      ;PERIOD ONLY ENTERED
 030756  030772                         4$                      ;ILLEGAL INPUT
 030760  030766                         3$                      ;TERMINATED WITH A CARRIAGE RETURN
 030762  030772                         4$                      ;TERMINATED WITH A ','
 030764  031004                         5$                      ;TERMINATED WITH A '/'
 030766  010215                 3$:     MOV     R2,(R5)         ;MOVE NEW VALUE TO PARAMETER LOCATION
 030770  000746                         BR      1$              ;GET MORE PARAMETERS
 030772  104401  076677         4$:     TYPE    ,BADENT         ;'BAD ENTRY'
 030776  162703  000006                 SUB     #6,R3           ;DECREMENT THE TABLE POINTER
 031002  000741                         BR      1$              ;TRY AGAIN
 031004  010215                 5$:     MOV     R2,(R5)         ;NEW VALUE
 031006  000404                         BR      7$              ;EXIT
 031010  005037  001340         6$:     CLR     CFLAG           ;CLEAR THE 'CONTROL C' FLAG
 031014  011603                         MOV     (SP),R3         ;RELOAD THE PARAMETER TABLE ADDRESS
 031016  000733                         BR      1$              ;TRY AGAIN
 031020  005726                 7$:     TST     (SP)+           ;CORRECT THE STACK POINTER
 031022  000207                         RTS     PC              ;RETURN
```

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

031024  104401  001203                ASNERR: TYPE    ,$CRLF          ;CR-LF
031030  104401  075613                        TYPE    ,QUES           ;'?'
031034  104401  075625                        TYPE    ,UNTMSG         ;TYPE 'DRIVE'
031040  010446                        ;       MOV     R4,-(SP)        ;;SAVE R4 FOR TYPEOUT
                                        ;                               ;;TYPE DRIVE NUMBER
031042  104403                        ;       TYPOS                   ;;GO TYPE--OCTAL ASCII
031044     002                          ;       .BYTE   2               ;;TYPE 2 DIGIT(S)
031045     000                          ;       .BYTE   0               ;;SUPPRESS LEADING ZEROS
031046  104401                        ;       TYPE                    ;TYPE SPECIFIC MESSAGE
031050  000000                        ASNMSG: .WORD   0               ;MESSAGE ADDRESS
031052  000207                                RTS     PC

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

031054  005004                        DROP:   CLR     R4              ;CLEAR R4 FOR DRIVE NUMBER
031056  111004                                MOVB    (R0),R4         ;MOVE DRIVE NUMBER TO R4
031060  146437  040142  001530                BICB    ATABIT(R4),ASNLST       ;REMOVE DRIVE FROM ASSIGNED LIST
031066  146437  040142  031662                BICB    ATABIT(R4),AUTLST       ;DELETE DRIVE FROM AUTO ASSIGN LIST
031074  006304                                ASL     R4              ;MAKE DRIVE NUMBER INTO A TABLE INDEX
031076  010064  001532                        MOV     R0,DUNIT(R4)    ;PUT DRIVE I' DROP LIST
031102  104401  001203                        TYPE    ,$CRLF
031106  104401  076236                        TYPE    ,DROPNG         ;TYPE 'FATAL OR EXCESSIVE ERRORS'
031112  104401  076311                        TYPE    ,MSGON          ;TYPE 'ON'
031116  104401  075625                        TYPE    ,UNTMSG         ;TYPE 'DRIVE'
031122  006204                                ASR     R4              ;DRIVE NUMBER
031124  010446                                MOV     R4,-(SP)        ;;SAVE R4 FOR TYPEOUT
                                                                        ;;TYPE DRIVE NUMBER
031126  104403                                TYPOS                   ;;GO TYPE--OCTAL ASCII
031130     002                                .BYTE   2               ;;TYPE 2 DIGIT(S)
031131     000                                .BYTE   0               ;;SUPPRESS LEADING ZEROS
031132  000207                        1$:     RTS     PC

                                        ;ROUTINE TO DEASSIGN DRIVE IF ERRORS BECOMES EXCESSIVE

031134  032777  000020  150012        ABNRML: BIT     #SW04,@SWR      ;SEE IF SWITCH 4 SET
031142  001006                                BNE     1$              ;BR IF IT'S SET
031144  023760  001444  000072                CMP     MAXER,$TOTAL(R0)        ;CHECK TOTAL ERROR VALUE
031152  101002                                BHI     1$              ;BR IF ERRORS DO NOT EXCEED MAX
031154  000137  031054                        JMP     DROP            ;DEASSING THE DRIVE
031160  000207                        1$:     RTS     PC              ;RETURN

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

```
                        .SBTTL  END OF PASS ROUTINE

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

031162                  $EOP:
031162  005737  001466          TST     ENDING          ;END OF PASS DETERMINED BY SEEKS OR DATA WORDS ?
031166  001412                  BEQ     EOP1            ;BR IF SEEKS
031170  026037  000040  001436  CMP     $ENDAT+2(R0),ENDCON+2   ;CHECK MSW OF WORDS DATA COUNT
031176  101020                  BHI     EOP2            ;BR IF MSW GREATER THAN LIMIT
031200  103404                  BLO     1$              ;BR IF MSW LESS THAN LIMIT
031202  026037  000036  001434  CMP     $ENDAT(R0),ENDCON       ;CHECK LSW AGAINST LIMIT
031210  103013                  BHIS    EOP2            ;BR IF EQUAL OR GREATER
031212  000207          1$:     RTS     PC

031214  026037  000044  001442  EOP1:   CMP     $ENDSK+2(R0),ENDSEK+2   ;CHECK MSW OF SEEK COUNT
031222  101006                  BHI     EOP2            ;BR IF MSW GREATER THAN LIMIT
031224  103404                  BLO     1$              ;EXIT IF MSW LESS THAN LIMIT
031226  026037  000042  001440  CMP     $ENDSK(R0),ENDSEK       ;CHECK LSW OF SEEK COUNT
031234  103001                  BHIS    EOP2            ;BR IF EQUAL OR GREATER
031236  000207          1$:     RTS     PC

031240  010446          EOP2:   MOV     R4,-(SP)        ;SAVE R4
031242  032777  000400  147704  BIT     #SW08,@SWR      ;INHIBIT END OF PASS TYPEOUT ?
031250  001023                  BNE     1$              ;BR IF YES
031252  104401  001203          TYPE    ,$CRLF          ;CR-LF
031256  104401  076272          TYPE    ,ENDPAS         ;END OF PASS FOR THE DRIVE
031262  016046  000104          MOV     $PASSC(R0),-(SP)        ;;SAVE $PASSC(R0) FOR TYPEOUT
031266  104405                  TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
031270  111037  001324          MOVB    (R0),UNIT       ;STORE THE DRIVE NUMBER
031274  104401  076311          TYPE    ,MSGON          ;TYPE 'ON'
031300  104401  075625          TYPE    ,UNTMSG         ;'DRIVE '
031304  013746  001324          MOV     UNIT,-(SP)      ;;SAVE UNIT FOR TYPEOUT
031310  104403                  TYPOS                   ;;GO TYPE--OCTAL ASCII
031312     002                  .BYTE   2               ;;TYPE 2 DIGIT(S)
031313     000                  .BYTE   0               ;;SUPPRESS LEADING ZEROS
031314  104401  001203          TYPE    ,$CRLF          ;CR-LF
031320  111004          1$:     MOVB    (R0),R4         ;MOVE DRIVE NUMBER

031322  105737  001150          TSTB    $AUTOB          ;RUNNING IN AUTO MODE ?
031326  001410                  BEQ     2$              ;BR IF NO
031330  136437  040142  031662  BITB    ATABIT(R4),AUTLST       ;IS DRIVE ALREADY ASSIGNED TO AUTO LIST ?
031336  001071                  BNE     6$              ;BR IF YES
031340  156437  040142  031662  BISB    ATABIT(R4),AUTLST       ;ADD DRIVE TO AUTO ASSIGN LIST
031346  000443                  BR      3$

031350  026037  000104  001456  2$:     CMP     $PASSC(R0),PASSES       ;SEE IF AT END OF TEST
031356  103437                  BLO     3$              ;BR IF NOT
031360  032777  000020  147566  BIT     #SW04,@SWR      ;TYPE END OF TEST MESSAGE ?
031366  001033                  BNE     3$              ;BR IF NO
031370  104401  076316          TYPE    ,ENDTST         ;TYPE 'END OF TEST'
031374  104401  076334          TYPE    ,MSGFOR         ;TYPE 'FOR'
031400  104401  075625          TYPE    ,UNTMSG         ;'DRIVE '
031404  013746  001324          MOV     UNIT,-(SP)      ;;SAVE UNIT FOR TYPEOUT
031410  104403                  TYPOS                   ;;GO TYPE--OCTAL ASCII
```

```
031412  002                                     .BYTE   2               ;;TYPE 2 DIGIT(S)
031413  000                                     .BYTE   0               ;;SUPPRESS LEADING ZEROS
031414  146437  040142  001530                 BICB    ATABIT(R4),ASNLST       ;DELETE DRIVE FROM ASSIGNED LIST
031422  006304                                  ASL     R4              ;MAKE DRIVE NUMBER INTO TABLE INDEX
031424  010064  001532                         MOV     R0,DUNIT(R4)    ;PUT BLOCK ADDRESS INTO DROP LIST
031430  105737  001530                         TSTB    ASNLST          ;ALL DRIVES ARE DEASSIGNED ?
031434  001041                                  BNE     7$              ;BR IF NO
031436  005237  001216                         INC     $DEVCT          ;INCREMENT DEVICE COUNT
031442  005237  001214                         INC     $PASS           ;INCREMENT THE PASS COUNT
031446  042737  100000  001214                 BIC     #100000,$PASS   ;AVOID NEGATIVE NUMBER
031454  000431                                  BR      7$

031456  032777  000400  147470  3$:     BIT     #SW08,@SWR      ;INHIBIT END OF PASS TYPEOUT ?
031464  001002                                  BNE     4$              ;BR IF YES
031466  004737  023636                         JSR     PC,SUMARY       ;TYPE THE DRIVE'S STATISTICS SUMMARY
031472  010346                  4$:     MOV     R3,-(SP)        ;SAVE R3
031474  010004                          MOV     R0,R4           ;DRIVE'S BLOCK ADDRESS
031476  062704  000036                 ADD     #$ENDAT,R4      ;ADD THE STARTING ADDR OF SECTIONS TO CLEAR
031502  012703  000006                 MOV     #6,R3           ;NUMBER OF LOCNS TO BE CLEARED
                                                                ;(CLEAR $ENDAT, $ENDSK AND $OPERC COUNTERS)
031506  005024                  5$:     CLR     (R4)+           ;CLEAR THE LOCN
031510  005303                          DEC     R3              ;DECREMENT THE LOCATION COUNTER
031512  001375                          BNE     5$              ;BR IF MORE TO GO
031514  012603                          MOV     (SP)+,R3        ;RESTORE R3
031516  005260  000104                 INC     $PASSC(R0)      ;INCREMENT THE PASS COUNT

031522  105737  001150          6$:     TSTB    $AUTOB          ;RUNNING IN AUTO MODE ?
031526  001404                          BEQ     7$              ;BR IF NO
031530  023737  001530  031662         CMP     ASNLST,AUTLST   ;HAVE ALL DRIVES COMPLETED PASS IN AUTO MODE ?
031536  001402                          BEQ     8$              ;BR IF YES
031540  012604                  7$:     MOV     (SP)+,R4        ;RESTORE R4
031542  000207                          RTS     PC              ;RETURN

031544  005237  031662          8$:     INC     AUTLST          ;CLEAR AUTO ASSIGN LIST FOR NEXT PASS AND
031550  001375                          BNE     8$              ;WAIT FOR TTY
031552  012737  000340  177776         MOV     #PR7,PS         ;DON'T ALLOW ANY INTERRUPTS
031560  005237  001216                 INC     $DEVCT          ;INCREMENT DEVICE COUNT
031564  005237  001214                 INC     $PASS           ;;INCREMENT THE PASS NUMBER
031570  042737  100000  001214         BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
031576  005327                          DEC     (PC)+           ;;LOOP?
031600  000001                  $EOPCT: .WORD   1
031602  003013                          BGT     $DOAGN          ;;YES
031604  012737                          MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
031606  000001                  $ENDCT: .WORD   1
031610  031600                          $EOPCT
031612  013700  000042          $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
031616  001405                          BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
031620  000005                          RESET                   ;;CLEAR THE WORLD
031622  004710                  $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
031624  000240                          NOP                     ;;SAVE ROOM
031626  000240                          NOP                     ;;FOR
031630  000240                          NOP                     ;;ACT11
031632                          $DOAGN:
031632  000137                          JMP     @(PC)+          ;;RETURN
031634  031636                  $RTNAD: .WORD   RTURN

2
3 031636  012706  001100          RTURN:  MOV     #STACK,SP       ;RESTORE STACK
```

```
      4 031642  005237  001212                    INC     $TESTN          ;INCREMENT THE TEST NUMBER IN THE MAIL BOX
      5 031646  004737  033226                    JSR     PC,$TKINT       ;MAKE SURE KEYBOARD INTERRUPT AND
      6 031652  004737  023344                    JSR     PC,CKCLK        ;SYSTEM CLOCK ARE STILL ON.
      7 031656  000137  006142                    JMP     MAIN            ;RETURN TO LOOP
      8
      9 031662  000000                  AUTLST: .WORD   0               ;AUTO ASSIGN LIST (USED IN AUTO RUN MODE)
```

```
                                        ;ROUTINE TO GET THE REMAINDER OF THE RANDOM NUMBER
                                        ;CALL:
                                        ;       MOV     NUMBER,R5       ;DIVISOR INTO R5
                                        ;       JSR     PC,GETREM
                                        ;       RETURN                  ;REMAINDER IS IN R5
                                        ;
031664  013746  037024                  GETREM: MOV     $LONUM,-(SP)    ;STORE RANDOM NUMBER ON THE STACK FOR DIVIDE
031670  013746  037022                          MOV     $HINUM,-(SP)    ;UPPER PART
031674  010546                          MOV     R5,-(SP)        ;PUT THE DIVISOR ONTO THE STACK
031676  004737  031710                  JSR     PC,$DIV         ;DIVIDE THE RANDOM NUMBERS
031702  012605                          MOV     (SP)+,R5        ;PUT THE REMAINDER INTO R5
031704  005726                          TST     (SP)+           ;ADJUST THE STACK POINTER
031706  000207                          RTS     PC
                                        ;
                                        .SBTTL  INTEGER DIVIDE ROUTINE
                                        ;
                                        ;;**********************************************************************
                                        ;*THIS ROUTINE WILL DIVIDE A 32-BIT TWO'S COMPLEMENT INTEGER
                                        ;*DIVIDEND BY A 16-BIT TWO'S COMPLEMENT INTEGER DIVISOR GIVING
                                        ;*A 16-BIT TWO'S COMPLEMENT INTEGER QUOTIENT AND A 16-BIT REMAINDER.
                                        ;*DIVISION WILL BE PERFORMED SO THAT THE REMAINDER IS OF THE
                                        ;*SAME SIGN AS THE DIVIDEND.
                                        ;*CALL:
                                        ;*      MOV     LOW DIVIDEND,-(SP)      ;;THE HIGH DIVIDEND MUST BE < 1/2
                                        ;*      MOV     HIGH DIVIDEND,-(SP)     ;;AS LARGE AS THE DIVISOR
                                        ;*      MOV     DIVISOR,-(SP)
                                        ;*      JSR     PC,$DIV
                                        ;*      RETURN                          ;;QUOTIENT & REMAINDER ARE ON THE STACK
                                        ;*
                                        ;*      STACK   NO ERROR        OVERFLOW        DIVIDE BY ZERO
                                        ;*      -----   --------        --------        -------------
                                        ;*      TOP     REMAINDER       ALL ZEROS       ALL ONES
                                        ;*      +2      QUOTIENT        ALL ZEROS       ALL ONES
                                        ;*
                                        ;*
                                        ;*NOTE: THIS ROUTINE WILL LINK TO THE DIVISION SUBROUTINE ('M.DPID').
                                        ;
031710  104412                          $DIV:   SAVREG                  ;STORE R0 - R5
031712  016605  000026                          MOV     26(SP),R5       ;DIVISOR
031716  005004                          CLR     R4              ;OTHER DIVISOR WORD
031720  016602  000030                  MOV     30(SP),R2       ;UPPER DIVIDEND WORD
031724  016603  000032                  MOV     32(SP),R3       ;LOWER DIVIDEND WORD
031730  005000                          CLR     R0              ;CLEAR OTHER DIVIDEND REGISTERS
031732  005001                          CLR     R1
031734  004737  031756                  JSR     PC,M.DPID       ;GO TO THE DIVIDE ROUTINE
031740  010166  000030                  MOV     R1,30(SP)       ;REMAINDER ON THE STACK
031744  010366  000032                  MOV     R3,32(SP)       ;QUOTIENT ON THE STACK
031750  104413                          RESREG                  ;RESTORE R0 - R5
031752  012616                          MOV     (SP)+,(SP)      ;MOVE RETURN UP THE STACK
031754  000207                          RTS     PC
                                        ;
                                        .SBTTL  DOUBLE PRECISION DIVISION SUBROUTINE
                                        ;
                                        ;CALL:
                                        ;       JSR     PC,M.DPID
                                        ;
```

```
                                        ;       DIVIDEND = R0-R1-R2-R3 (R0=MSD)
                                        ;       DIVISOR = R4-R5 (R4=MSD)
                                        ;
                                        ;RETURN
                                        ;
                                        ;       REMAINDER AFTER DIVISION = R0-R1 (R0=MSD)
                                        ;       QUOTIENT AFTER DIVISION = R2-R3 (R2=MSD)
                                        ;
032056  012746  000040          M.DPID: MOV     #40.,-(SP)      ;COUNTER FOR DIVISION CYCLES
032062  010446                          MOV     R4,-(SP)        ;HIGH ORDER
032064  010546                          MOV     R5,-(SP)        ;LOW ORDER DIVISOR TO THE STACK
032066  005466  000002                  NEG     2(SP)           ;FORM NEGATIVE
032072  005416                          NEG     @SP             ;VERSION OF THE DIVISOR
032074  005666  000002                  SBC     2(SP)
032000  061601                          ADD     @SP,R1
032002  005500                          ADC     R0              ;PERFORM THE INITIAL SUBTRACTION
032004  066600  000002                  ADD     2(SP),R0
032010  103445                          BCS     M.DP50          ;IF CARRY THEN OVERFLOW HAS OCCURRED
032012  005046                          CLR     -(SP)           ;THIS IS A LONGER LASTING CARRY BIT
032014  006103                  M.DP40: ROL     R3
032016  006102                          ROL     R2
032020  006101                          ROL     R1
032022  006100                          ROL     R0
032024  005716                          TST     @SP             ;TEST "CARRY" INDICATOR
032026  001410                          BEQ     M.DP41          ;IF NO "CARRY" THEN ADD ELSE SUBTRACT
032030  005016                          CLR     @SP             ;CLEAR UP FOR NEXT TIME
032032  066601  000002                  ADD     2(SP),R1
032036  005500                          ADC     R0              ;ADD -(DIVISOR)
032040  005516                          ADC     @SP     ;  I    ;SET "CARRY"
032042  066600  000004                  ADD     4(SP),R0;<-
032046  000404                          BR      M.DP42
032050  060501                  M.DP41: ADD     R5,R1
032052  005500                          ADC     R0              ;ADD +(DIVISOR)
032054  005516                          ADC     @SP     ;  I    ;SET "CARRY"
032056  060400                          ADD     R4,R0   ;<-
032060  005516                  M.DP42: ADC     @SP             ;SET "CARRY"
032062  005716                          TST     @SP             ;TEST THE UPDATE INDICATOR
032064  001401                          BEQ     .+4     ;->     ;IF ZERO FORGET IT
032066  005203                          INC     R3      ;  I    ;NO CARRY POSSIBLE HERE
032070  005366  000006                  DEC     6(SP)   ;<-     ;DECREMENT COUNTER
032074  003347                          BGT     M.DP40          ;BRANCH IF MORE TO DO
032076  006003                          ROR     R3
032100  103404                          BCS     M.DP44
032102  060501                          ADD     R5,R1
032104  005500                          ADC     R0
032106  060400                          ADD     R4,R0
032110  000241                          CLC
032112  006103                  M.DP44: ROL     R3
032114  062706  000010                  ADD     #10,SP          ;ADJUST STACK BY 4 WORDS
032120  000242                          CLV
032122  000207                          RTS     PC
032124  062706  000006          M.DP50: ADD     #6,SP
032130  000262                          SEV
032132  000207                          RTS     PC
```

```
                                        .SBTTL  SUPRS - TYPE ASCIZ, REPLACE LEADING 0'S WITH BLANKS
                                        .SBTTL  SUPRSL -TYPE ASCIZ, LEFT JUSTIFY
                                        ;
                                        ;;*****************************************************************
                                        ;CALL:
                                        ;       MOV     #NUMADR,-(SP)   ;FIRST ADDRESS OF ASCIZ STRING
                                        ;       JSR     PC,SUPRS
                                        ;  OR
                                        ;       MOV     #NUMADR,-(SP)   ;FIRST ADDRESS OF ASCIZ STRING
                                        ;       JSR     PC,SUPRSL
                                        ;
032134  0 0046                  SUPRSL: MOV     R0,-(SP)        ;SAVE R0
032136  016600  000004                  MOV     4(SP),R0        ;GET POINTER TO MESSAGE
032142  005037  032224                  CLR     SUPR2
032146  000405                          BR      SUPR1
                                        ;
032150  010046                  SUPRS:  MOV     R0,-(SP)        ;SAVE R0
032152  016600  000004                  MOV     4(SP),R0        ;GET POINTER TO MESSAGE
032156  010037  032224                  MOV     R0,SUPR2        ;GET POINTER FOR TYPING
032162                          SUPR1:
032162  105710                  1$:     TSTB    (R0)            ;TEST FOR TERMINATOR
032164  001406                          BEQ     2$              ;YES
032166  122710  000060                  CMPB    #'0,(R0)        ;IS THIS A '0' ?
032172  001006                          BNE     3$              ;NO
032174  112720  000040                  MOVB    #40,(R0)+       ;REPLACE IT WITH A 'BLANK'
032200  000770                          BR      1$              ;NEXT CHAR.
032202  005300                  2$:     DEC     R0              ;BACKUP 1
032204  112710  000060                  MOVB    #'0,(R0)        ;MAKE IT '0'
032210  005737  032224          3$:     TST     SUPR2           ;LEFT JUSTIFY ?
032214  001002                          BNE     4$              ;NO
032216  010037  032224                  MOV     R0,SUPR2        ;YES
032222  104401                  4$:     TYPE
032224  000000                  SUPR2:  .WORD   0
032226  012600                          MOV     (SP)+,R0        ;RESTORE R0
032230  012616                          MOV     (SP)+,(SP)      ;RESTORE STACK
032232  000207                          RTS     PC
```

```
                                        .SBTTL  $SUPRS - TYPE ASCIZ, REPLACE LEADING 0'S WITH BLANKS
                                        .SBTTL  $SUPRL - TYPE ASCIZ, LEFT JUSTIFY

                                        ;;*****************************************************************
                                        ;
                                        ;THIS ROUTINE IS SAME AS 'SUPRSL' AND 'SUPRS', EXCEPT THAT IT
                                        ;WILL SUPPRESS THE ERROR TYPEOUT IF SW13=1. THIS ACCOMPLISHED BY
                                        ;USED TRAP CALL 'DISPLY', INSTEAD OF 'TYPE'.
                                        ;CALL:
                                        ;       MOV     #NUMADR,-(SP)   ;FIRST ADDRESS OF ASCIZ STRING
                                        ;       JSR     PC,$SUPRS
                                        ;  OR
                                        ;       MOV     #NUMADR,-(SP)   ;FIRST ADDRESS OF ASCIZ STRING
                                        ;       JSR     PC,$SUPRL

032234  010046                  $SUPRL: MOV     R0,-(SP)        ;SAVE R0
032236  016600  000004                  MOV     4(SP),R0        ;GET POINTER TO MESSAGE
032242  005037  032324                  CLR     $SUPR2
032246  000405                          BR      $SUPR1

032250  010046                  $SUPRS: MOV     R0,-(SP)        ;SAVE R0
032252  016600  000004                  MOV     4(SP),R0        ;GET POINTER TO MESSAGE
032256  010037  032324                  MOV     R0,$SUPR2       ;GET POINTER FOR TYPING
032262                          $SUPR1:
032262  105710                  1$:     TSTB    (R0)            ;TEST FOR TERMINATOR
032264  001406                          BEQ     2$              ;YES
032266  122710  000060                  CMPB    #'0,(R0)        ;IS THIS A '0' ?
032272  001006                          BNE     3$              ;NO
032274  112720  000040                  MOVB    #40,(R0)+       ;REPLACE IT WITH A 'BLANK'
032300  000770                          BR      1$              ;NEXT CHAR.
032302  005300                  2$:     DEC     R0              ;BACKUP 1
032304  112710  000060                  MOVB    #'0,(R0)        ;MAKE IT '0'
032310  005737  032324          3$:     TST     $SUPR2          ;LEFT JUSTIFY ?
032314  001002                          BNE     4$              ;NO
032316  010037  032324                  MOV     R0,$SUPR2       ;YES
032322  104414                  4$:     DISPLY                  ;TYPE, UNLESS SW13-1
032324  000000                  $SUPR2: .WORD   0
032326  012600                          MOV     (SP)+,R0        ;RESTORE R0
032330  012616                          MOV     (SP)+,(SP)      ;RESTORE STACK
032332  000207                          RTS     PC
```

```
                                        ;ROUTINE TO REPLACE LEADING ZEROS IN A NUMERIC STRING WITH SPACES
                                        ;CALL:
                                        ;       MOV     #ADR,-(SP)      ;ADDRESS OF NUMBER (IN ASCII)
                                        ;       JSR     R5,REPLZ        ;REPLACE PRECEDING ZEROS WITH BLANKS
                                        ;       .WORD   N               ;'N' IS NUMBER OF DIGITS TO BE TYPED
                                        ;  OR
                                        ;       MOV     #ADR,-(SP)      ;ADDRESS OF NUMBER (IN ASCII)
                                        ;       JSR     R5,FILLZ        ;TYPE PRECEDING ZEROS
                                        ;       .WORD   N               ;'N' IS NUMBER OF DIGITS TO BE TYPED

032334  005237  032442      FILLZ:  INC     FILLO           ;LEAVE ZERO'S

032340  010046              REPLZ:  MOV     R0,-(SP)        ;SAVE R0
032342  016600  000004              MOV     4(SP),R0        ;ADDRESS OF NUMBER TO R0
032346  005737  032442              TST     FILLO           ;LEAVE PRECEDING ZEROS ?
032352  001014                      BNE     3$              ;BR IF YES
032354  122710  000060      1$:     CMPB    #'0,(R0)        ;BYTE EQUAL TO ASCII '0' ?
032360  001004                      BNE     2$              ;BR IF NOT
032362  112710  000040              MOVB    #40,(R0)        ;REPLACE THE ZERO WITH A SPACE
032366  005200                      INC     R0              ;INCREMENT THE BYTE ADDRESS
032370  000771                      BR      1$              ;GO BACK AND LOOK FOR MORE LEADING ZEROS
032372  105710      2$:     TSTB    (R0)            ;SEE IF ZERO BYTE TERMINATOR
032374  001003                      BNE     3$              ;BR IF NOT
032376  005300                      DEC     R0              ;BACKUP STRING POINTER
032400  112710  000060              MOVB    #'0,(R0)        ;PUT A ZERO BACK IN
032404  016600  000004      3$:     MOV     4(SP),R0        ;PUT ADDRESS OF FIRST CHARACTER ON STACK
032410  105720              4$:     TSTB    (R0)+           ;SEE IF ZERO BYTE TERMINATOR
032412  001376                      BNE     4$              ;BR IF NOT
032414  005300                      DEC     R0              ;BACKUP STRING POINTER
032416  162500                      SUB     (R5)+,R0        ;ADJUST ADDRESS
032420  010037  032426              MOV     R0,5$           ;GET ADDRESS FOR TYPEOUT
032424  104401                      TYPE                    ;TYPE THE NUMBER
032426  000000      5$:     .WORD   0               ;ADDRESS OF NUMBER
032430  012600                      MOV     (SP)+,R0        ;;POP STACK INTO R0
032432  012616                      MOV     (SP)+,(SP)      ;RESTORE STACK
032434  005037  032442              CLR     FILLO           ;RESET FILL FLAG
032440  000205                      RTS     R5              ;RETURN

032442  000000      FILLO:  .WORD   0               ;IF SET, LEAVE PRECEDING ZEROS FOR TYPE

                                        ;ROUTINE TO TYPE AT PRIORITY 4

032444  013746  177776      TYPRI4: MOV     @#PS,-(SP)      ;SAVE THE PRESENT STATUS
032450  012737  000200  177776      MOV     #200,@#PS       ;CHANGE THE PRIORITY TO 4
032456  012537  032466              MOV     (R5)+,1$        ;MESSAGE ADDRESS
032462  004737  035450              JSR     PC,$TYPE        ;TYPE THE MESSAGE
032466  000000      1$:     .WORD   0               ;MESSAGE ADDRESS GOES HERE
032470  000205                      RTS     R5              ;RETURN

                                        ;ROUTINE TO TYPE THE MBA (DRIVE) SERIAL NUMBER IN DECIMAL
                                        ;CALL:
                                        ;       MOV     #DPB,R0         ;ADDRESS OF DRIVE PARAMETER BLOCK
                                        ;       JSR     PC,TYMBA        ;CALL ROUTINE
                                        ;   OR
                                        ;       MOV     #DPB,R0         ;ADDRESS OF DRIVE PARAMETER BLOCK
                                        ;       JSR     PC,TYPMBA       ;CALL ROUTINE(WITH NO HEADER MESSAGE)
                                        ;
```

```
                                        ;R0 = DPB ADDRESS BEFORE CALLING THE ROUTINE
                                        ;
032472  104401  077303          TYMBA:  TYPE    ,MBASN          ;TYPE 'MBA S/N:'
032476  010037  032512          TYPMBA: MOV     R0,1$           ;ADDRESS OF DPB
032502  062737  002126  032512          ADD     #$MBASN,1$      ;INDEX TO DRIVE (MBA) SERIAL NUMBER
032510  104401                          TYPE                    ;TYPE THE DRIVE (MBA) SERIAL NUMBER
032512  000000          1$:             .WORD   0               ;ADDRESS OF DRIVE (MBA) SERIAL NUMBER FIELD
032514  104401  076477                  TYPE    ,PERIOD         ;TYPE '.'
032520  000207                          RTS     PC              ;RETURN
                                        ;
                                        ;ROUTINE TO TYPE THE PACK SERIAL NUMBER IN OCTAL
                                        ;CALL:
                                        ;       MOV     #DPB,R0         ;ADDRESS OF DRIVE PARAMETER BLOCK
                                        ;       JSR     PC,TYPACK       ;CALL ROUTINE
                                        ;   OR
                                        ;       MOV     #DPB,R0         ;ADDRESS OF DRIVE PARAMETER BLOCK
                                        ;       JSR     PC,TYPCK        ;CALL ROUTINE(WITH NO HEADER MESSAGE)
                                        ;
                                        ;R0 = DPB ADDRESS BEFORE CALLING THE ROUTINE
                                        ;
032522  104401  077270          TYPACK: TYPE    ,PACKSN         ;TYPE 'PACK S/N:'
032526  005760  000142          TYPCK:  TST     $PSNM(R0)       ;IS SERIAL NUMBER VALID ?
032532  002003                          BGE     1$              ;BR IF VALID
032534  104401  076672                  TYPE    ,NONE           ;TYPE 'NONE'
032540  000425                          BR      3$
032542  016046  000142          1$:     MOV     $PSNM(R0),-(SP) ;GET HI NUMBER (MSB)
032546  001414                          BEQ     2$              ;BR IF ZERO
032550  004737  033160                  JSR     PC,$SB20        ;CONVERT TO OCTAL NUMBER
032554  004737  032134                  JSR     PC,SUPRSL       ;AND TYPE IT LEFT JUSTIFIED
032560  016046  000140                  MOV     $PSNL(R0),-(SP) ;GET LOW NUMBER (LSB)
032564  004737  033160                  JSR     PC,$SB20        ;CONVERT TO OCTAL NUMBER
032570  004537  032334                  JSR     R5,FILLZ        ;TYPE WITH PRECEEDING ZEROS
032574  000005                          .WORD   5               ;5 DIGITS
032576  000406                          BR      3$
032600  016016  000140          2$:     MOV     $PSNL(R0),(SP)  ;GET LOW NUMBER (LSB)
032604  004737  033160                  JSR     PC,$SB20        ;CONVERT TO OCTAL NUMBER
032610  004737  032134                  JSR     PC,SUPRSL       ;AND TYPE IT LEFT JUSTIFIED
032614  000207          3$:             RTS     PC              ;RETURN
                                        ;
                                        ;ROUTINE TO TYPE ERRORS
                                        ;CALL:
                                        ;       DISPLY                  ;MUST DEFINED IN 'TRAP' TABLE
                                        ;       MESADR                  ;ADDRESS OF MESSAGE
                                        ;       RETURN
                                        ;
032616  032777  020000  146330  $DSPLY: BIT     #BIT13,@SWR     ;INHIBIT ERROR TYPEOUT ?
032624  001004                          BNE     1$              ;BR IF YES
032626  005037  177776                  CLR     @#PS            ;SET PRIORITY TO ZERO
032632  000137  035450                  JMP     $TYPE           ;TYPE THE MESSAGE
032636  062716  000002          1$:     ADD     #2,(SP)         ;INCREMENT THE RETURN
032642  000002                          RTI                     ;RETURN
```

```
                                        ;THIS ROUTINE IS USED TO CHECK IF AN
                                        ;ASCII CHARACTER IS A DIGIT BETWEEN 0 AND 7.
                                        ;CALL:
                                        ;       MOV     #ADR,R1         ;ADDRESS OF ASCII CHARACTER
                                        ;       JSR     R5,CK.OCT       ;CHECK THE CHARACTER
                                        ;       RETURN1                 ;CHARACTER IS NOT BETWEEN 0-7
                                        ;       RETURN2                 ;CHARACTER IS IN R2 AS A
                                        ;                               ;OCTAL DIGIT

032644  121127  000060          CK.OCT: CMPB    (R1),#'0        ;LESS THAN ZERO?
032650  103407                          BLO     1$              ;YES -- BRANCH
032652  121127  000067                  CMPB    (R1),#'7        ;GREATER THAN SEVEN?
032656  101004                          BHI     1$              ;YES -- BRANCH
032660  111102                          MOVB    (R1),R2         ;GET THE CHARACTER
032662  042702  177770                  BIC     #^C7,R2         ;STRIP AWAY THE ASCII
032666  005725                          TST     (R5)+           ;ADJUST FOR RETURN
032670  000205                  1$:     RTS     R5              ;RETURN

                                        ;THIS ROUTINE IS USED TO CHECK AN ASCII CHARACTER
                                        ;AND DETERMINE IF IT IS A DIGIT BETWEEN 0 AND 9.
                                        ;CALL:
                                        ;       MOV     #ADR,R1         ;ADDRESS OF ASCII CHARACTER
                                        ;       JSR     R5,CK.DEC       ;CHECK THE CHARACTER
                                        ;       RETURN1                 ;NOT BETWEEN 0 AND 9
                                        ;       RETURN2                 ;BETWEEN 0 AND 9
                                        ;                               ;R2 = DIGIT

032672  121127  000060          CK.DEC: CMPB    (R1),#'0        ;LESS THAN ZERO?
032676  103407                          BLO     1$              ;YES -- BRANCH
032700  121127  000071                  CMPB    (R1),#'9        ;GREATER THAN NINE?
032704  101004                          BHI     1$              ;YES -- BRANCH
032706  111102                          MOVB    (R1),R2         ;GET THE CHARACTER
032710  042702  000060                  BIC     #'0,R2          ;STRIP AWAY THE ASCII
032714  005725                          TST     (R5)+           ;ADJUST FOR RETURN
032716  000205                  1$:     RTS     R5              ;RETURN

                                        ;THIS ROUTINE WILL CHECK AN ASCII CHARACTER TO
                                        ;DETERMINE WHAT IT IS.
                                        ;CALL:
                                        ;       MOV     #ADR,R1         ;ADDRESS OF ASCII CHARACTER
                                        ;       JSR     R5,CK.CHR       ;CHECK CHARACTER
                                        ;       RETURN  ADR1            ;UNKNOWN CHARACTER
                                        ;       RETURN  ADR2            ;CARRIAGE RETURN * (R1)-ADR+'
                                        ;       RETURN  ADR3            ;COMMA * (R1)=ADR+1
                                        ;       RETURN  ADR4            ;PERIOD * (R1)=ADR+1
                                        ;       RETURN  ADR5            ;DIGIT BETWEEN 0 AND 7.
                                        ;       RETURN  ADR6            ;DIGIT BETWEEN 8 AND 9.
                                        ;                               ;R2 = DIGIT * (R1)=ADR+'

032720  105711                  CK.CHR: TSTB    (R1)            ;''CARRIAGE RETURN''?
032722  001417                          BEQ     3$              ;YES -- BRANCH
032724  121127  000054                  CMPB    (R1),#',        ;''COMMA''?
032730  001413                          BEQ     2$              ;YES -- BRANCH
032732  121127  000056                  CMPB    (R1),#'.        ;''PERIOD''?
032736  001407                          BEQ     1$              ;YES -- BRANCH
032740  004537  032672                  JSR     R5,CK.DEC       ;''DIGIT''?
032744  000410                          BR      4$              ;NO -- BRANCH
```

```
                                        JSR     R5,CK.OCT       ;OCTAL ?
                                        TST     (R5)+           ;DIGIT BETWEEN 8-9
                                        TST     (R5)+           ;DIGIT BETWEEN 0-7
                                1$:     TST     (R5)+           ;PERIOD
                                2$:     TST     (R5)+           ;COMMA
                                3$:     TST     (R5)+           ;CARRIAGE RETURN
                                        INC     R1              ;MOVE POINTER TO NEXT CHARACTER
                                4$:     MOV     (R5),R5         ;UNKNOWN CHARACTER
                                        RTS     R5              ;RETURN

                                ;THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL
                                ;CHARACTERS AND FORMS A DECIMAL VALUE BINARY NUMBER IN R2.
                                ;CALL:
                                ;       MOV     #ADR,R1         ;ADDRESS OF ASCIZ STRING
                                ;       MOV     #NUM,R2         ;MAX. MAGNITUDE OF INPUT NUMBER
                                ;       JSR     R5,CK.DIG       ;CHECK DIGITS
                                ;       RETURN  ADR1            ;''CR'' ONLY ENTERED -- R2=0
                                ;       RETURN  ADR2            ;''PERIOD'' ONLY ENTERED -- R2=0
                                ;       RETURN  ADR3            ;ILLEGAL CHARACTER OR INPUT TOO LARGE -- R2=?
                                ;       RETURN  ADR4            ;''CR'' -- R2 = NUMBER
                                ;       RETURN  ADR5            ;''COMMA'' -- R2 = NUMBER
                                ;       RETURN  ADR6            ;''PERIOD'' -- R2 = NUMBER

                                CK.DIG: MOV     R4,-(SP)        ;SAVE R4
                                        MOV     R3,-(SP)        ;SAVE R3
                                        MOV     R2,-(SP)        ;SAVE THE MAX. SIZE ON THE STACK
                                        CLR     R2              ;START WITH 0
                                        CLR     R3
                                        CLR     R4
                                        JSR     R5,CK.CHR       ;CHECK ONE CHARACTER
                                        6$                      ;ILLEGAL CHARACTER
                                        9$                      ;CARRIAGE RETURN
                                        6$                      ;''
                                        7$                      ;,
                                        1$                      ;DIGIT 0-7
                                        1$                      ;DIGIT 8-9
                                1$:     ADD     #4,R5           ;STEP RETURN POINTER PAST ''CR'' & ''PERIOD'' RETURNS
                                2$:     ASL     R3              ;INPUT NUMBER *2
                                        MOV     R3,-(SP)        ;SAVE *2
                                        ASL     R3              ;*4
                                        ASL     R3              ;*8
                                        ADD     (SP)+,R3        ;(*2)+(*8)    *10
                                        ADD     R2,R3           ;UPDATE THE INPUT NUMBER
                                        JSR     R5,CK.CHR       ;CHECK ONE CHARACTER
                                        8$                      ;ILLEGAL CHARACTER
                                        5$                      ;CARRIAGE RETURN
                                        4$                      ;''
                                        3$                      ;,
                                        2$                      ;DIGIT 0-7
                                        2$                      ;DIGIT 8-9
                                3$:     TSTB    (R1)            ;DOES A ''CR'' FOLLOW THE ''PERIOD''
                                        BNE     8$              ;BR IF NOT
                                        TST     (R4)+           ;INCREMENT THE RETURN
                                4$:     TST     (R4)+           ;INCREMENT THE RETURN
                                5$:     TST     (R4)+           ;INCREMENT THE RETURN
                                        CMP     R3,(SP)         ;CHECK THE MAGNITUDE OF THE NUMBER
                                        BHI     9$              ;BR IF ENTERED NUMBER TOO LARGE
```

```
                                                        BR      8$              ;BYPASS INCREMENT
                                                6$:     TST     (R5)+           ;INCREMENT RETURN PAST INVALID RETURN
                                                7$:     TST     (R5)+           ;INCREMENT RETURN
                                                8$:     ADD     R4,R5           ;SETUP RETURN POINTER
                                                9$:     MOV     R3,R2           ;ENTERED VALUE
                                                        TST     (SP)+           ;CLEAN MAX. SIZE OFF OF STACK
                                                        MOV     (SP)+,R3        ;RESTORE R3
                                                        MOV     (SP)+,R4        ;RESTORE R4
                                                        MOV     (R5),R5         ;GET RETURN ADDRESS
                                                        RTS     R5              ;RETURN

                                                ;THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
                                                ;UNSIGNED DECIMAL ASCIZ NUMBER.
                                                ;CALL:
                                                ;       MOV     NUMBER,-(SP)    ;PUT THE NUMBER ON THE STACK
                                                ;       JSR     PC,$SB2D        ;CALL:
                                                ;       RETURN                  ;ADDRESS OF THE 1ST ASCIZ CHAR IS ON THE STACK
                                                ;
                                                ;NOTE:  THE PROGRAM REQUIRES THIS FORM OF '$SB2D', NOT THE VERSION ON
                                                ;       THE SYSMAC LIBRARY, REV C AND LATER

033130  016637  000002  033154  $SB2D:  MOV     2(SP),1$        ;SAVE THE BINARY NUMBER
033136  012746  033154                  MOV     #1$,-(SP)       ;SET THE POINTER
033142  004737  037122                  JSR     PC,$DB2D        ;CALL THE DOUBLE LENGTH CONVERT
033146  012666  000002                  MOV     (SP)+,2(SP)     ;PICKUP THE POINTER
033152  000207                          RTS     PC              ;RETURN
033154  000000  000000          1$:     .WORD   0,0

                                                ;THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
                                                ;UNSIGNED OCTAL ASCIZ NUMBER.
                                                ;CALL:
                                                ;       MOV     NUMBER,-(SP)    ;PUT THE NUMBER ON THE STACK
                                                ;       JSR     PC,$SB2O        ;CALL:
                                                ;       RETURN                  ;ADDRESS OF THE 1ST ASCIZ CHAR IS ON THE STACK
                                                ;
                                                ;NOTE:  THE PROGRAM REQUIRES THIS FORM OF '$SB2O', NOT THE VERSION ON
                                                ;       THE SYSMAC LIBRARY, REV C AND LATER

033160  016637  000002  033204  $SB2O:  MOV     2(SP),1$        ;SAVE THE BINARY NUMBER
033166  012746  033204                  MOV     #1$,-(SP)       ;SET THE POINTER
033172  004737  037316                  JSR     PC,$DB2O        ;CALL THE DOUBLE LENGTH CONVERT
033176  012666  000002                  MOV     (SP)+,2(SP)     ;PICKUP THE POINTER
033202  000207                          RTS     PC              ;RETURN
033204  000000  000000          1$:     .WORD   0,0
```

```
                                .SBTTL  TTY INPUT ROUTINE

                                ;;*******************************************************************
                                ;
                                .ENABL  LSB
033210  000000                  $TKCNT: .WORD   0               ;;NUMBER OF ITEMS IN QUEUE
033212  000000                  $TKQIN: .WORD   0               ;;INPUT POINTER
033214  000000                  $TKQOUT: .WORD  0               ;;OUTPUT POINTER
033216                          $TKQSRT: .BLKB  7               ;;TTY KEYBOARD QUEUE
        033225                  $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
                                ;
033226  005037  033210          $TKINT: CLR     $TKCNT          ;;CLEAR COUNT OF ITEMS IN QUEUE
033232  012737  033216  033212          MOV     #$TKQSRT,$TKQIN ;;MOVE THE STARTING ADDRESS OF THE
033240  013737  033212  033214          MOV     $TKQIN,$TKQOUT  ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
033246  012737  033276  000060          MOV     #$TKSRV,@#TKVEC ;;INITIALIZE THE KEYBOARD VECTOR
033254  012737  000200  000062          MOV     #200,@#TKVEC+2  ;;'BR' LEVEL 4
033262  005777  145674                  TST     @$TKB           ;;CLEAR DONE FLAG
033266  012777  000100  145664          MOV     #100,@$TKS      ;;ENABLE TTY KEYBOARD INTERRUPT
033274  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)
                                ;
033276  117746  145660          $TKSRV: MOVB    @$TKB,-(SP)     ;;PICKUP THE CHARACTER
033302  042716  177600                  BIC     #^C177,(SP)     ;;STRIP THE JUNK
033306  021627  000021                  CMP     (SP),#$XON      ;;IS IT A RANDOM XON?
033312  001002                  BNE     30$             ;;BRANCH IF NO
033314  005726                  TST     (SP)+           ;;CLEAN RANDOM XON OFF STACK
033316  000002                  RTI                     ;;RETURN
033320                          30$:
03332C  021627  000003                  CMP     (SP),#3         ;;IS IT A CONTROL C?
033324  001007                  BNE     1$              ;;BRANCH IF NO
033326  104401  034433                  TYPE    ,$CNTLC         ;;TYPE A CONTROL-C (^C)
033332  004737  033226                  JSR     PC,$TKINT       ;;INIT THE KEYBOARD
033336  005726                  TST     (SP)+           ;;CLEAN UP STACK
033340  000137  034474                  JMP     CTRAP           ;;CONTROL C RESTART
033344  021627  000007          1$:     CMP     (SP),#7         ;;IS IT A CONTROL G?
033350  001004                  BNE     2$              ;;BRANCH IF NO
033352  022737  000176  001154          CMP     #SWREG,SWR      ;;IS SOFT-SWR SELECTED?
033360  001500                  BEQ     6$              ;;GO TO SWR CHANGE

033362                          2$:
033362  022737  000007  033210          CMP     #7,$TKCNT       ;;IS THE QUEUE FULL?
033370  001004                  BNE     3$              ;;BRANCH IF NO
033372  104401  001176                  TYPE    ,$BELL          ;;RING THE TTY BELL
```

```
033376  005726                          TST     (SP)+           ;;CLEAN CHARACTER OFF OF STACK
033400  000451                          BR      5$              ;;EXIT
033402  021627  000023          3$:     CMP     (SP),#23        ;;IS IT A CONTROL-S?
033406  001021                          BNE     32$             ;;BRANCH IF NO
033410  005077  145544                  CLR     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
033414  005726                          TST     (SP)+           ;;CLEAN CHAR OFF STACK
033416  105777  145536          31$:    TSTB    @$TKS           ;;WAIT FOR A CHAR
033422  100375                          BPL     31$             ;;LOOP UNTIL ITS THERE
033424  117746  145532                  MOVB    @$TKB,-(SP)     ;;GET THE CHARACTER
033430  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
033434  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
033440  001366                          BNE     31$             ;;BRANCH IF NO
033442  012777  000100  145510          MOV     #100,@$TKS      ;;REENABLE TTY KEYBOARD INTERRUPTS
033450  000002                          RTI                     ;;RETURN
033452  005237  033210          32$:    INC     $TKCNT          ;;COUNT THIS CHARACTER
033456  021627  000140                  CMP     (SP),#140       ;;IS IT UPPER CASE?
033462  002405                          BLT     4$              ;;BRANCH IF YES
033464  021627  000175                  CMP     (SP),#175       ;;IS IT A SPECIAL CHAR?
033470  003002                          BGT     4$              ;;BRANCH IF YES
033472  042716  000040                  BIC     #40,(SP)        ;;MAKE IT UPPER CASE
033476  112677  177510          4$:     MOVB    (SP)+,@$TKQIN   ;;AND PUT IT IN QUEUE
033502  005237  033212                  INC     $TKQIN          ;;UPDATE THE POINTER
033506  023727  033212  033225          CMP     $TKQIN,#$TKQEND ;;GO OFF THE END?
033514  001003                          BNE     5$              ;;BRANCH IF NO
033516  012737  033216  033212          MOV     #$TKQSRT,$TKQIN ;;RESET THE POINTER
033524  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.
033526  022737  000176  001154  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED
033534  001124                          BNE     15$             ;;EXIT IF NOT
033536  105777  145416                  TSTB    @$TKS           ;;IS A CHAR WAITING?
033542  100121                          BPL     15$             ;;IF NOT, EXIT
033544  117746  145412                  MOVB    @$TKB,-(SP)     ;;YES
033550  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT  7-BIT ASCII
033554  021627  000007                  CMP     (SP),#7         ;;IS IT A CONTROL-G?
033560  001300                          BNE     2$              ;;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.
033562  123727  001150  000001  6$:     CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
033570  001674                          BEQ     2$              ;;BRANCH IF YES
033572  005726                          TST     (SP)+           ;;CLEAR CONTROL-G OFF STACK
033574  004737  033226                  JSR     PC,$TKINT       ;;FLUSH THE TTY INPUT QUEUE
033600  005077  145354                  CLR     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
033604  112737  000001  001151          MOVB    #1,$INTAG       ;;SET INTERRUPT MODE INDICATOR

033612  104401  034445                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
033616  104401  034452  $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
033622  013746  000176                  MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
033626  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
```

```
033630  104401  034463          TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
033634  005046          19$:    CLR     -(SP)           ;;CLEAR COUNTER
033636  005046                  CLR     -(SP)           ;;THE NEW SWR
033640  105777  145314    7$:   TSTB    @$TKS           ;;CHAR THERE?
033644  100375                  BPL     7$              ;;IF NOT TRY AGAIN

033646  117746  145310          MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
033652  042716  177600          BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII

033656  021627  000003          CMP     (SP),#3         ;;IS IT A CONTROL-C?
033662  001015                  BNE     9$              ;;BRANCH IF NOT
033664  104401  034433          TYPE    ,$CNTLC         ;;YES, ECHO CONTROL-C (^C)
033670  062706  000006          ADD     #6,SP           ;;CLEAN UP STACK
033674  123727  001151  000001  CMPB    $INTAG,#1       ;;REENABLE TTY KEYBOARD INTERRUPTS?
033702  001003                  BNE     8$              ;;BRANCH IF NO
033704  012777  000100  145246  MOV     #100,@$TKS      ;;ALLOW TTY KEYBOARD INTERRUPTS
033712  000137  034474    8$:   JMP     CTRAP           ;;CONTROL-C RESTART

033716  021627  000025    9$:   CMP     (SP),#25        ;;IS IT A CONTROL-U?
033722  001005                  BNE     10$             ;;BRANCH IF NOT
033724  104401  034440          TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
033730  062706  000006    20$:  ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
033734  000737                  BR      19$             ;;LET'S TRY IT AGAIN

033736  021627  000015    10$:  CMP     (SP),#15        ;;IS IT A <CR>?
033742  001022                  BNE     16$             ;;BRANCH IF NO
033744  005766  000004          TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
033750  001403                  BEQ     11$             ;;BRANCH IF YES
033752  016677  000002  145174  MOV     2(SP),@SWR      ;;SAVE NEW SWR
033760  062706  000006    11$:  ADD     #6,SP           ;;CLEAR UP STACK
033764  104401  001203    14$:  TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
033770  123727  001151  000001  CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
033776  001003                  BNE     15$             ;;BRANCH IF NOT
034000  012777  000100  145152  MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
034006  000002    15$:  RTI             ;;RETURN
034010  004737  035662    16$:  JSR     PC,$TYPEC       ;;ECHO CHAR
034014  021627  000060          CMP     (SP),#60        ;;CHAR < 0?
034020  002420                  BLT     18$             ;;BRANCH IF YES
034022  021627  000067          CMP     (SP),#67        ;;CHAR > 7?
034026  003015                  BGT     18$             ;;BRANCH IF YES
034030  042726  000060          BIC     #60,(SP)+       ;;STRIP-OFF ASCII
034034  005766  000002          TST     2(SP)           ;;IS THIS THE FIRST CHAR
034040  001403                  BEQ     17$             ;;BRANCH IF YES
034042  006316                  ASL     (SP)            ;;NO, SHIFT PRESENT
034044  006316                  ASL     (SP)            ;;    CHAR OVER TO MAKE
034046  006316                  ASL     (SP)            ;;    ROOM FOR NEW ONE.
034050  005266  000002    17$:  INC     2(SP)           ;;KEEP COUNT OF CHAR
034054  056616  177776          BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
034060  000667                  BR      7$              ;;GET THE NEXT ONE
034062  104401  001202    18$:  TYPE    ,$QUES          ;;TYPE ?<CR><LF>
034066  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
                                ;

034070  011646                  $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC AND
034072  016666  000004  000002          MOV     4(SP),2(SP)     ;;THE PS
034100  005066  000004                  CLR     4(SP)           ;;GET READY FOR A CHARACTER
034104  005046                          CLR     -(SP)           ;;PUT NEW PS ON STACK
034106  012746  034114                  MOV     #64$,-(SP)      ;;PUT NEW PC ON  STACK
034112  000002                          RTI                     ;;POP NEW PC AND PS
034114                          64$:
034114  005737  033210          1$:     TST     $TKCNT          ;;WAIT ON A CHARACTER
034120  001775                          BEQ     1$
034122  005337  033210                  DEC     $TKCNT          ;;DECREMENT THE COUNTER
034126  117766  177062  000004        MOVB    @$TKQOUT,4(SP)  ;;GET ONE CHARACTER
034134  005237  033214                  INC     $TKQOUT         ;;UPDATE THE POINTER
034140  023727  033214  033225        CMP     $TKQOUT,#$TKQEND ;;DID IT GO OFF OF THE END?
034146  001003                          BNE     2$              ;;BRANCH IF NO
034150  012737  033216  033214        MOV     #$TKQSRT,$TKQOUT ;;RESET THE POINTER
034156  000002                  2$:     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

034160  010346                  $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
034162  005046                          CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
034164  012703  034414          1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
034170  022703  034433          2$:     CMP     #$TTYIN+15.,R3  ;;BUFFER FULL?
034174  101456                          BLOS    4$              ;;BR IF YES
034176  104410                          RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
034200  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
034202  122713  000177          10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
034206  001022                          BNE     5$              ;;BR IF NO
034210  005716                          TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
034212  001007                          BNE     6$              ;;BR IF NO
034214  112737  000134  034412        MOVB    #'\,9$          ;;TYPE A BACK SLASH
034222  104401  034412                  TYPE    ,9$
034226  012716  177777                  MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
034232  005303                  6$:     DEC     R3              ;;BACKUP BY ONE
034234  020327  034414                  CMP     R3,#$TTYIN      ;;STACK EMPTY?
034240  103434                          BLO     4$              ;;BR IF YES
034242  111337  034412                  MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR.
034246  104401  034412                  TYPE    ,9$             ;;GO TYPE
034252  000746                          BR      2$              ;;GO READ ANOTHER CHAR.
034254  005716                  5$:     TST     (SP)            ;;RUBOUT KEY SET?
034256  001406                          BEQ     7$              ;;BR IF NO
034260  112737  000134  034412        MOVB    #'\,9$          ;;TYPE A BACK SLASH
034266  104401  034412                  TYPE    ,9$
034272  005016                          CLR     (SP)            ;;CLEAR THE RUBOUT KEY
034274  122713  000025          7$:     CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
034300  001003                          BNE     8$              ;;BR IF NO
```

```
034302  104401  034440                  TYPE    ,$CNTLU         ;;TYPE A CONTROL 'U'
034306  000726                          BR      1$              ;;GO START OVER
034310  122713  000022          8$:     CMPB    #22,(R3)        ;;IS CHARACTER A '^R'?
034314  001011                          BNE     3$              ;;BRANCH IF NO
034316  105013                          CLRB    (R3)            ;;CLEAR THE CHARACTER
034320  104401  001203                  TYPE    ,$CRLF          ;;TYPE A 'CR' & 'LF'
034324  104401  034414                  TYPE    ,$TTYIN         ;;TYPE THE INPUT STRING
034330  000717                          BR      2$              ;;GO PICKUP ANOTHER CHACTER
034332  104401  001202          4$:     TYPE    ,$QUES          ;;TYPE A '?'
034336  000712                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
034340  111337  034412          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
034344  104401  034412                  TYPE    ,9$
034350  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
034354  001305                          BNE     2$              ;;LOOP IF NOT RETURN
034356  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
034362  104401  001204                  TYPE    ,$LF            ;;TYPE A LINE FEED
034366  005726                          TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
034370  012603                          MOV     (SP)+,R3        ;;RESTORE R3
034372  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
034374  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
034402  012766  034414  000004          MOV     #$TTYIN,4(SP)
034410  000002                          RTI                     ;;RETURN
034412     000          9$:     .BYTE   0                       ;;STORAGE FOR ASCII CHAR. TO TYPE
034413     000                  .BYTE   0                       ;;TERMINATOR
034414                  $TTYIN: .BLKB   15.                     ;;RESERVE 15. BYTES FOR TTY INPUT
034433     136  103  015  $CNTLC: .ASCIZ  /^C/<15><12>         ;;CONTROL 'C'
034440     136  125  015  $CNTLU: .ASCIZ  /^U/<15><12>         ;;CONTROL 'U'
034445     136  107  015  $CNTLG: .ASCIZ  /^G/<15><12>         ;;CONTROL 'G'
034452     015  012  123  $MSWR:  .ASCIZ  <15><12>/SWR = /
034463     040  040  116  $MNEW:  .ASCIZ  /  NEW = /
2
3                                       ;THIS ROUTINE WILL PROCESS THE (^C) CHARACTER
4
5  034474  012737  000001  001340  CTRAP:  MOV     #1,CFLAG        ;SET THE 'CONTROL C' FLAG
6  034502  005237  033210                  INC     $TKCNT          ;COUNT THIS CHARACTER
7  034506  112777  000015  176476          MOVB    #15,@$TKQIN     ;PUT 'RETURN' CHARACTER IN QUEUE
8  034514  005237  033212                  INC     $TKQIN          ;UPDATE THE POINTER
9  034520  023727  033212  033225          CMP     $TKQIN,#$TKQEND ;GO OFF THE END ?
10 034526  001003                          BNE     1$              ;BR IF YES
11 034530  012737  033216  033212          MOV     #$TKQSRT,$TKQIN ;RESET THE POINTER
12 034536  000002                  1$:     RTI                     ;RETURN
```

```
                                .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

034540  105037  035126          $ERROR: CLRB    IBSAVE          ;;CLEAR THE ITEM BYTE SAVE LOCATION
034544  104407                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
034546  010337  001322                  MOV     R3,ATTN         ;SAVE THE ATTENTION REGISTER CONTENTS
034552  010137  001220                  MOV     R1,DRIVE        ;DRIVE NUMBER
034556  032777  020000  144370          BIT     #SW13,@SWR      ;INHIBIT PRINTOUTS ?
034564  001004                          BNE     .+12            ;BR IF YES
034566  104401  001203                  TYPE    ,$CRLF          ;CR-LF
034572  104401  001203                  TYPE    ,$CRLF          ;CR-LF
034576  004737  024750                  JSR     PC,$TIME        ;TYPE THE TIME
034602  105237  001117          7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
034606  001775                          BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
034610  013777  001116  144340          MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
034616  032777  002000  144330          BIT     #BIT10,@SWR     ;;BELL ON ERROR?
034624  001402                          BEQ     1$              ;;NO - SKIP
034626  104401  001176                  TYPE    ,$BELL          ;;RING BELL
034632  005237  001126          1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
034636  011637  001132                  MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
034642  162737  000002  001132          SUB     #2,$ERRPC
034650  117737  144256  001130          MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
034656  032777  001000  144270          BIT     #BIT09,@SWR     ;;SEE IF LOOP ON ERROR IS SET
034664  001060                          BNE     1004$           ;;BRANCH AROUND ROUTINE IF SO
034666  122737  000177  001130          CMPB    #177,$ITEMB     ;;SEE IF THIS IS THE POWER FAIL CALL
034674  001454                          BEQ     1004$           ;;BRANCH AROUND ROUTINE IF IT IS
034676  105737  035126                  TSTB    IBSAVE          ;;SEE IF THIS IS THE 2ND ERROR CALL IN THIS ROUTINE
034702  001047                          BNE     1003$           ;;BRANCH IF SO
034704  022737  177777  035124          CMP     #-1,CPSAVE      ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
034712  001445                          BEQ     1004$           ;;BRANCH IF SO
034714  013746  000004                  MOV     ERRVEC,-(SP)    ;;SAVE CONTENTS OF ERROR VECTOR
034720  012737  034736  000004          MOV     #1000$,ERRVEC   ;;SETUP 'TRAP' RETURN ADDRESS
034726  013737  177766  035124          MOV     177766,CPSAVE   ;;MOVE CPU ERROR REGISTER TO CPSAVE FOR TEST
034734  000406                          BR      1001$
034736  012737  177777  035124  1000$:  MOV     #-1,CPSAVE      ;;SET CPU ERROR REGISTER TIMEOUT INDICATOR
034744  012716  034752                  MOV     #1001$,(SP)     ;;SETUP RETURN ADDRESS
034750  000002                          RTI
034752  012637  000004          1001$:  MOV     (SP)+,ERRVEC    ;;RESTORE CONTENTS OF ERROR VECTOR

034756  022737  177777  035124  1002$:  CMP     #-1,CPSAVE      ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
034764  001420                          BEQ     1004$           ;;BRANCH IF SO
034766  032737  000001  035124          BIT     #BIT00,CPSAVE   ;;SEE IF POWER MONITOR BIT IS SET IN CPU ERR REG
034774  001414                          BEQ     1004$           ;;BRANCH IF OK
034776  042737  000001  177766          BIC     #BIT00,177766   ;;CLEAR THE BIT FOUND SET
035004  113737  001130  035126          MOVB    $ITEMB,IBSAVE   ;;MAKE IBSAVE NON-ZERO FOR DUAL ERROR CALL
035012  112737  000177  001130          MOVB    #177,$ITEMB     ;;SET $ITEMB TO SPECIAL POWER FAIL POINTER
035020  000402                          BR      1004$           ;;BRANCH OVER IBSAVE CLEARING
```

```
035022  105037  035126          1003$:  CLRB    IBSAVE          ;;CLEAR IBSAVE SO 2ND TIME THROUGH EXITS
035026                          1004$:
035026  032777  020000  144120          BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
035034  001004                          BNE     20$             ;;SKIP TYPEOUTS
035036  004737  035130                  JSR     PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
035042  104401  001203                  TYPE    ,$CRLF
035046                          20$:
035046  122737  000001  001226          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
035054  001007                          BNE     2$              ;;NO,SKIP APT ERROR REPORT
035056  113737  001130  035070          MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
035064  004737  036474                  JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
035070     000                  21$:    .BYTE   0
035071     000                          .BYTE   0
035072  000777                  22$:    BR      22$             ;;APT ERROR LOOP
035074  105737  035126          2$:     TSTB    IBSAVE          ;;SEE IF IBSAVE IS LOADED
035100  001005                          BNE     3$              ;;BRANCH IF NOT - NO HALT ON PWR MON BIT ERROR
035102  005777  144046                  TST     @SWR            ;;HALT ON ERROR
035106  100002                          BPL     3$              ;;SKIP IF CONTINUE
035110  000000                          HALT                    ;;HALT ON ERROR!
035112  104407                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
035114                          3$:
035114  105737  035126                  TSTB    IBSAVE          ;;SEE IF ITEM BYTE SAVE LOCATION HAS AN ERROR CALL
035120  001230                          BNE     7$              ;;BRANCH BACK TO CALL ORIGINAL ERROR
035122  000002                          RTI                     ;;RETURN
035124  000000                  CPSAVE: .WORD   0               ;;LOCATION TO SAVE CPU ERROR REG CONTENTS
035126  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.

035130                  $ERRTYP:
035130  104401  001203          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & 'LINE FEED"
035134  010046                  MOV     R0,-(SP)        ;;SAVE R0
035136  005000                  CLR     R0              ;;PICKUP THE ITEM INDEX
035140  153700  001130          BISB    @#$ITEMB,R0
035144  001004                  BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
                                                        ;;TYPE THE PC OF THE ERROR
035146  013746  001132          MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
                                                        ;;ERROR ADDRESS
035152  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
035154  000437                  BR      6$              ;;GET OUT
035156  122700  000177  1$:     CMPB    #177,R0         ;;SEE IF THIS ERROR CALL IS SPECIAL POWER FAIL CALL
035162  001006                  BNE     1000$           ;;BRANCH IF NOT
035164  013737  001212  035446  MOV     $TESTN,PFTSTN   ;;GET TEST NUMBER
035172  012700  035306          MOV     #PFECH,R0       ;;MOVE POWER FAIL ERROR CALL TABLE TO R0
035176  000406                  BR      1001$           ;;BRANCH TO CALL ERROR
035200  005300  1000$:  DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
035202  006300                  ASL     R0              ;;      WORK FOR THE ERROR TABLE
035204  006300                  ASL     R0
035206  006300                  ASL     R0
035210  062700  003472          ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
035214  012037  035224  1001$:  MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
035220  001404                  BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
035222  104401                  TYPE                    ;;TYPE THE "ERROR MESSAGE"
035224  000000  2$:     .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
035226  104401  001203          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
035232  012037  035242  3$:     MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
035236  001404                  BEQ     5$              ;;SKIP TYPEOUT IF 0
035240  104401                  TYPE                    ;;TYPE THE "DATA HEADER"
035242  000000  4$:     .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
035244  104401  001203          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
035250  011000  5$:     MOV     (R0),R0         ;;PICKUP "DATA TABLE" POINTER
035252  001004                  BNE     7$              ;;GO TYPE THE DATA
035254  012600  6$:     MOV     (SP)+,R0        ;;RESTORE R0
035256  104401  001203          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
035262  000207                  RTS     PC              ;;RETURN
035264                  7$:
035264  013046                  MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
035266  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
035270  005710                  TST     (R0)            ;;IS THERE ANOTHER NUMBER?
035272  001770                  BEQ     6$              ;;BR IF NO
035274  104401  035302          TYPE    ,8$             ;;TYPE TWO(2) SPACES
035300  000771                  BR      7$              ;;LOOP
035302    040   040   000  8$:     .ASCIZ  /  /            ;;TWO(2) SPACES
                                .EVEN
035306  035316  035400  035432  PFECH:  PFECH1,PFECH2,PFECH3,PFECH4 ;;WORDS DEFINING TABLES BELOW
035316    120   117   127  PFECH1: .ASCIZ  ?POWER MONITOR BIT IN CPU ERROR REGISTER FOUND SET?
035400    124   105   123  PFECH2: .ASCIZ  ?TESTNO  ERR PC  CPUERREG?
                                .EVEN
035432  035446  001132  035124  PFECH3: .WORD   PFTSTN,$ERRPC,CPSAVE,0
```

```
035442  000  000  000  PFECH4: .BYTE  0,0,0,0
035446  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
                        ;*

035450  105737  001173          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
035454  100002                          BPL     1$              ;;BR IF YES
035456  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
035460  000430                          BR      3$              ;;LEAVE
035462  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
035464  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
035470  122737  000001  001226          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
035476  001011                          BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
035500  132737  000100  001227          BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
035506  001405                          BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
035510  010037  035520                  MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
035514  004737  036464                  JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
035520  000000                  61$:    .WORD   0               ;;MESSAGE ADDRESS
035522  132737  000040  001227  62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
035530  001003                          BNE     60$             ;;YES,SKIP TYPE OUT
035532  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
035534  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
035536  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
035540  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
035542  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
035546  000002                          RTI                     ;;RETURN
035550  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
035554  001430                          BEQ     8$
035556  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
035562  001006                          BNE     5$
035564  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
035566  104401                          TYPE                    ;;TYPE A CR AND LF
035570  001203                          $CRLF
035572  105037  036000                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
035576  000755                          BR      2$              ;;GET NEXT CHARACTER
035600  004737  035662          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
035604  123726  001172          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
035610  001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
035612  013746  001170                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                ;;AND THE NULL CHAR.
035616  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
035622  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
035624  004737  035662                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
035630  105337  036000                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
035634  000770                          BR      7$              ;;LOOP
```

```
                                ;HORIZONTAL TAB PROCESSOR

035636  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
035642  004737  035662          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
035646  132737  000007  036000          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
035654  001372                          BNE     9$              ;;TAB STOP
035656  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
035660  000724                          BR      2$              ;;GET NEXT CHARACTER
035662                          $TYPEC:
035662  105777  143272                  TSTB    @$TKS           ;;CHAR IN KYBD BUFFER?
035666  100022                          BPL     10$             ;;BR IF NOT
035670  017746  143266                  MOV     @$TKB,-(SP)     ;;GET CHAR
035674  042716  177600                  BIC     #177600,(SP)    ;;STRIP EXTRANEOUS BITS
035700  122716  000023                  CMPB    #$XOFF,(SP)     ;;WAS CHAR XOFF
035704  001012                          BNE     102$            ;;BR IF NOT
035706                          101$:
035706  105777  143246                  TSTB    @$TKS           ;;WAIT FOR CHAR
035712  100375                          BPL     101$
035714  117716  143242                  MOVB    @$TKB,(SP)      ;;GET CHAR
035720  042716  177600                  BIC     #177600,(SP)    ;;STRIP IT
035724  122716  000021                  CMPB    #$XON,(SP)      ;;WAS IT XON?
035730  001366                          BNE     101$            ;;BR IF NOT
035732                          102$:
035732  005726                          TST     (SP)+           ;;FIX STACK
035734                          10$:
035734  105777  143224                  TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
035740  100375                          BPL     10$
035742  116677  000002  143216          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
035750  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
035756  001003                          BNE     1$              ;;BRANCH IF NO
035760  105037  036000                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
035764  000406                          BR      $TYPEX          ;;EXIT
035766  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
035774  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
035776  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
036000  000000                  $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
036002  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 LAS
                        ;*$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

036004  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
036010  116637  000001  036227          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
036016  112637  036231                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
036022  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
036026  000406                          BR      $TYPON
036030  112737  000001  036227  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
036036  112737  000006  036231          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
036044  112737  000005  036226  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
036052  010346                          MOV     R3,-(SP)        ;;SAVE R3
036054  010446                          MOV     R4,-(SP)        ;;SAVE R4
036056  010546                          MOV     R5,-(SP)        ;;SAVE R5
036060  113704  036231                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
036064  005404                          NEG     R4
036066  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
036072  110437  036230                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
036076  113704  036227                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
036102  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
036106  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
036110  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
036112  000404                          BR      3$              ;;GO DO MSB
036114  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
036116  006105                          ROL     R5
036120  006105                          ROL     R5
036122  010503                          MOV     R5,R3
036124  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
036126  105337  036230                  DECB    $OMODE          ;;TYPE THIS DIGIT?
036132  100016                          BPL     7$              ;;BR IF NO
036134  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
036140  001002                          BNE     4$              ;;TEST FOR 0
036142  005704                          TST     R4              ;;SUPPRESS THIS 0?
036144  001403                          BEQ     5$              ;;BR IF YES
036146  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
```

```
036150  052703  000060            BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
036154  052703  000040     5$:    BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
036160  110337  036224            MOVB    R3,8$           ;;SAVE FOR TYPING
036164  104401  036224            TYPE    ,8$             ;;GO TYPE THIS DIGIT
036170  105337  036226     7$:    DECB    $OCNT           ;;COUNT BY 1
036174  003347                    BGT     2$              ;;BR IF MORE TO DO
036176  002402                    BLT     6$              ;;BR IF DONE
036200  005204                    INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
036202  000744                    BR      2$              ;;GO DO THE LAST DIGIT
036204  012605     6$:    MOV     (SP)+,R5        ;;RESTORE R5
036206  012604            MOV     (SP)+,R4        ;;RESTORE R4
036210  012603            MOV     (SP)+,R3        ;;RESTORE R3
036212  016666  000002  000004    MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
036220  012616            MOV     (SP)+,(SP)
036222  000002            RTI                     ;;RETURN
036224     000     8$:    .BYTE   0               ;;STORAGE FOR ASCII DIGIT
036225     000            .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
036226     000     $OCNT: .BYTE   0               ;;OCTAL DIGIT COUNTER
036227     000     $OFILL: .BYTE  0               ;;ZERO FILL SWITCH
036230  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

036232                          $TYPDS:
036232  010046                         MOV     R0,-(SP)        ;;PUSH R0 ON STACK
036234  010146                         MOV     R1,-(SP)        ;;PUSH R1 ON STACK
036236  010246                         MOV     R2,-(SP)        ;;PUSH R2 ON STACK
036240  010346                         MOV     R3,-(SP)        ;;PUSH R3 ON STACK
036242  010546                         MOV     R5,-(SP)        ;;PUSH R5 ON STACK
036244  012746  020200                 MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
036250  016605  000020                 MOV     20(SP),R5       ;;GET THE INPUT NUMBER
036254  100004                         BPL     1$              ;;BR IF INPUT IS POS.
036256  005405                         NEG     R5              ;;MAKE THE BINARY NUMBER POS.
036260  112766  000055  000001         MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
036266  005000                1$:      CLR     R0              ;;ZERO THE CONSTANTS INDEX
036270  012703  036446                 MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
036274  112723  000040                 MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
036300  005002                2$:      CLR     R2              ;;CLEAR THE BCD NUMBER
036302  016001  036436                 MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
036306  160105                3$:      SUB     R1,R5           ;;FORM THIS BCD DIGIT
036310  002402                         BLT     4$              ;;BR IF DONE
036312  005202                         INC     R2              ;;INCREASE THE BCD DIGIT BY 1
036314  000774                         BR      3$
036316  060105                4$:      ADD     R1,R5           ;;ADD BACK THE CONSTANT
036320  005702                         TST     R2              ;;CHECK IF BCD DIGIT=0
036322  001002                         BNE     5$              ;;FALL THROUGH IF 0
036324  105716                         TSTB    (SP)            ;;STILL DOING LEADING 0'S?
036326  100407                         BMI     7$              ;;BR IF YES
036330  106316                5$:      ASLB    (SP)            ;;MSD?
036332  103003                         BCC     6$              ;;BR IF NO
036334  116663  000001  177777         MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
036342  052702  000060         6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
036346  052702  000040         7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
036352  110223                         MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
036354  005720                         TST     (R0)+           ;;JUST INCREMENTING
036356  020027  000010                 CMP     R0,#10          ;;CHECK THE TABLE INDEX
036362  002746                         BLT     2$              ;;GO DO THE NEXT DIGIT
036364  003002                         BGT     8$              ;;GO TO EXIT
036366  010502                         MOV     R5,R2           ;;GET THE LSD
036370  000764                         BR      6$              ;;GO CHANGE TO ASCII
036372  105726                8$:      TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
036374  100003                         BPL     9$              ;;BR IF NO
036376  116663  177777  177776         MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
036404  105013                9$:      CLRB    (R3)            ;;SET THE TERMINATOR
036406  012605                         MOV     (SP)+,R5        ;;POP STACK INTO R5
036410  012603                         MOV     (SP)+,R3        ;;POP STACK INTO R3
036412  012602                         MOV     (SP)+,R2        ;;POP STACK INTO R2
036414  012601                         MOV     (SP)+,R1        ;;POP STACK INTO R1
```

```
036416  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
036420  104401  036446                  TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
036424  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
036432  012616                          MOV     (SP)+,(SP)
036434  000002                          RTI                     ;;RETURN TO USER
036436  023420                  $DTBL:  10000.
036440  001750                          1000.
036442  000144                          100.
036444  000012                          10.
036446                          $DBLK:  .BLKW   4
```

```
                                .SBTTL  APT COMMUNICATIONS ROUTINE

                                ;;*****************************************************************
036456  112737  000001  036722  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROP
036464  112737  000001  036720  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
036472  000403                          BR      $ATYC
036474  112737  000001  036722  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
036502                          $ATYC:
036502  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
036504  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
036506  105737  036720                  TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
036512  001450                          BEQ     5$              ;;IF NOT:  BR
036514  122737  000001  001226          CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
036522  001031                          BNE     3$              ;;IF NOT:  BR
036524  132737  000100  001227          BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
036532  001425                          BEQ     3$              ;;IF NOT:  BR
036534  017600  000004                  MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
036540  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
036546  005737  001206          1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
036552  001375                          BNE     1$              ;;IF NOT:  WAIT
036554  010037  001222                  MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
036560  105720                  2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
036562  001376                          BNE     2$
036564  163700  001222                  SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
036570  006200                          ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
036572  010037  001224                  MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
036576  012737  000004  001206          MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
036604  000413                          BR      5$
036606  017637  000004  036632  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
036614  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
036622  013746  177776                  MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
036626  004737  035450                  JSR     PC,$TYPE        ;;CALL TYPE MACRO
036632  000000                  4$:     .WORD   0
036634                          5$:
036634  105737  036722          10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
036640  001416                          BEQ     12$             ;;IF NOT:  BR
036642  005737  001226                  TST     $ENV            ;;RUNNING UNDER APT?
036646  001413                          BEQ     12$             ;;IF NOT:  BR
036650  005737  001206          11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
036654  001375                          BNE     11$             ;;IF NOT:  WAIT
036656  017637  000004  001210          MOV     @4(SP),$FATAL   ;;GET ERROR #
036664  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
036672  005237  001206                  INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
036676  105037  036722          12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
036702  105037  036721                  CLRB    $LFLG           ;;CLEAR LOG FLAG
036706  105037  036720                  CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
036712  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
036714  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
036716  000207                          RTS     PC              ;;RETURN
036720     000          $MFLG:  .BYTE   0               ;;MESSG. FLAG
036721     000          $LFLG:  .BYTE   0               ;;LOG FLAG
036722     000          $FFLG:  .BYTE   0               ;;FATAL FLAG
                                .EVEN
        000200          APTSIZE = 200
        000001          APTENV  = 001
        000100          APTSPOOL= 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

036724                          $RAND:
036724  010046                         MOV     R0,-(SP)        ;;PUSH R0 ON STACK
036726  010146                         MOV     R1,-(SP)        ;;PUSH R1 ON STACK
036730  010246                         MOV     R2,-(SP)        ;;PUSH R2 ON STACK
036732  013700  037024                 MOV     $LONUM,R0       ;;SET R0 WITH LOW
036736  013701  037022                 MOV     $HINUM,R1       ;;SET R1 WITH HIGH
036742  012702  177771                 MOV     #-7,R2          ;;SET SHIFT COUNT
036746  006300                  1$:    ASL     R0              ;;SHIFT R0 LEFT AND
036750  006101                         ROL     R1              ;;ROTATE CARRY INTO R1 AND
036752  005202                         INC     R2              ;;CHECK FOR DONE
036754  001374                         BNE     1$              ;;CONTINUE SHIFT LOOP
036756  063700  037024                 ADD     $LONUM R0       ;;ADD NUMBER      MAKE X 129
036762  005501                         ADC     R1              ;;PROPOGATE CARRY
036764  063701  037022                 ADD     $HINUM,R1       ;;ADD NUMBER TO MAKE X 129
036770  062700  001057                 ADD     #1057,R0        ;;ADD LOW CONSTANT
036774  005501                         ADC     R1              ;;PROPOGATE CARRY
036776  062701  047401                 ADD     #47401,R1       ;;ADD HIGH CONSTANT
037002  010037  037024                 MOV     R0,$LONUM       ;;SAVE R0
037006  010137  037022                 MOV     R1,$HINUM       ;;SAVE R1
037012  012602                         MOV     (SP)+,R2        ;;POP STACK INTO R2
037014  012601                         MOV     (SP)+,R1        ;;POP STACK INTO R1
037016  012600                         MOV     (SP)+,R0        ;;POP STACK INTO R0
037020  000207                         RTS     PC              ;;RETURN
037022  176543                  $HINUM: .WORD   176543
037024  123456                  $LONUM: .WORD   123456
```

```
                                .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES

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

037026                          $SAVREG:
037026  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
037030  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
037032  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
037034  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
037036  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
037040  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
037042  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
037046  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
037052  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
037056  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
037062  000002                          RTI

                                ;*RESTORE R0-R5
                                ;*CALL:
                                ;*      RESREG
037064                          $RESREG:
037064  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
037070  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
037074  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
037100  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
037104  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
037106  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
037110  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
037112  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
037114  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
037116  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
037120  000002                          RTI
```

```
                        .SBTTL  DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

                        ;;*****************************************************************
                        ;*THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
                        ;*DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
                        ;*POSITIVE.
                        ;*CALL
                        ;*      MOV     #PNTR,-(SP)     ;;POINTER TO LOW WORD OF BINARY NUMBER
                        ;*      JSR     PC,@#$DB2D
                        ;*      RETURN                  ;;THE FIRST ADDRESS OF ASCIZ
                                                        ;;IS ON THE STACK


037122  104412          $DB2D:  SAVREG                  ;;SAVE REGISTERS
037124  016602  000002          MOV     2(SP),R2        ;;PICKUP THE DATA POINTER
037130  012700  037302          MOV     #$DECVL,R0      ;;GET ADDRESS OF "$DECVL" STRING
037134  010066  000002          MOV     R0,2(SP)        ;;PUT ADDRESS OF ASCIZ STRING ON STACK
037140  012201                  MOV     (R2)+,R1        ;;PICKUP THE BINARY NUMBER
037142  012202                  MOV     (R2)+,R2
037144  012737  000012  037220  MOV     #10.,4$         ;;SET UP TO DO 10 CONVERSIONS
037152  012704  037232          MOV     #$TNPWR,R4      ;;ADDRESS OF TEN POWER
037156  012705  037234          MOV     #$TNPWR+2,R5
037162  005003          1$:     CLR     R3              ;;CLEAR PARTIAL
037164  161401          2$:     SUB     (R4),R1         ;;SUBTRACT TEN POWER
037166  005602                  SBC     R2
037170  161502                  SUB     (R5),R2
037172  002402                  BLT     3$              ;;BR IF TEN POWER TO LARGE
037174  005203                  INC     R3              ;;ADD 1 TO PARTIAL
037176  000772                  BR      2$              ;;LOOP
037200  062401          3$:     ADD     (R4)+,R1        ;;RESTORE SUBTRACTED VALUE
037202  005502                  ADC     R2
037204  062402                  ADD     (R4)+,R2
037206  022525                  CMP     (R5)+,(R5)+     ;;MOVE TO NEXT TEN POWER
037210  052703  000060          BIS     #'0,R3          ;;CHANGE PARTIAL TO ASCII
037214  110320                  MOVB    R3,(R0)+        ;;SAVE IT
037216  005327                  DEC     (PC)+           ;;DONE?
037220  000000          4$:     .WORD   0
037222  001357                  BNE     1$              ;;BR IF NO
037224  105020                  CLRB    (R0)+           ;;TERMINATOR
037226  104413                  RESREG                  ;;RESTORE REGISTERS
037230  000207                  RTS     PC              ;;RETURN
037232  145000          $TNPWR: 145000                  ;;1.0E09
037234  035632                  35632
037236  160400                  160400                  ;;1.0E08
037240  002765                  2765
037242  113200                  113200                  ;;1.0E07
037244  000230                  230
037246  041100                  041100                  ;;1.0E06
037250  000017                  17
037252  103240                  103240                  ;;1.0E05
037254  000001                  1
037256  023420                  23420                   ;;1.0E04
037260  000000                  0
037262  001750                  1750                    ;;1.0E03
037264  000000                  0
037266  000144                  144                     ;;1.0E02
037270  000000                  0
```

```
037272  000012                          12                      ;;1.0E01
037274  000000                          0
037276  000001                          1                       ;;1.0E00
037300  000000                          0
037302                          $DECVL: .BLKB   12.             ;;RESERVE STORAGE FOR ASCIZ STRING
```

```
                        .SBTTL  DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE

                        ;;*****************************************************************
                        ;*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
                        ;*UNSIGNED OCTAL ASCIZ NUMBER.
                        ;*CALL
                        ;*      MOV     #PNTR,-(SP)     ;;POINTER TO LOW WORD OF BINARY NUMBER
                        ;*      JSR     PC,@#$DB20      ;;CALL THE ROUTINE
                        ;*      RETURN                  ;;THE ADDRESS OF THE FIRST ASCIZ CHAR. IS ON THE STACK

037316  104412          $DB20:  SAVREG                  ;;SAVE ALL REGISTERS
037320  016601  000002          MOV     2(SP),R1        ;;PICKUP THE POINTER TO LOW WORD
037324  012705  037435          MOV     #$OCTVL+13.,R5  ;;POINTER TO DATA TABLE
037330  012704  000014          MOV     #12.,R4         ;;DO ELEVEN CHARACTERS
037334  012703  177770          MOV     #^C7,R3         ;;MASK
037340  012100                  MOV     (R1)+,R0        ;;LOWER WORD
037342  012101                  MOV     (R1)+,R1        ;;HIGH WORD
037344  005002                  CLR     R2              ;;TERMINATOR
037346  110245          1$:     MOVB    R2,-(R5)        ;;PUT CHARACTER IN DATA TABLE
037350  010002                  MOV     R0,R2           ;;GET THIS DIGIT
037352  005304                  DEC     R4              ;;COUNT THIS CHARACTER
037354  003007                  BGT     3$              ;;BR IF NOT THE LAST DIGIT
037356  001405                  BEQ     2$              ;;BR IF IT IS THE LAST DIGIT
037360  005205                  INC     R5              ;;ALL DIGITS DONE-ADJUST POINTER FOR FIRST
037362  010566  000002          MOV     R5,2(SP)        ;;ASCIZ CHAR. & PUT IT ON THE STACK
037366  104413                  RESREG                  ;;RESTORE ALL REGISTERS
037370  000207                  RTS     PC              ;;RETURN TO USER
037372  006203          2$:     ASR     R3              ;;POSITION THE MASK FOR THE LAST DIGIT
037374  006001          3$:     ROR     R1              ;;POSITION THE BINARY NUMBER FOR
037376  006000                  ROR     R0              ;;      THE NEXT OCTAL DIGIT
037400  006001                  ROR     R1
037402  006000                  ROR     R0
037404  006001                  ROR     R1
037406  006000                  ROR     R0
037410  040302                  BIC     R3,R2           ;;MASK OUT ALL JUNK
037412  062702  000060          ADD     #'0,R2          ;;MAKE THIS CHAR. ASCII
037416  000753                  BR      1$              ;;GO PUT IT IN THE DATA TABLE
037420                  $OCTVL: .BLKB   14.             ;;RESERVE DATA TABLE
```

```
                                .SBTTL  POWER DOWN AND UP ROUTINES

                                ;;*****************************************************************
                                ;POWER DOWN ROUTINE
037436  012737  037610  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
037444  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
037452  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
037454  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
037456  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
037460  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
037462  010446                        MOV     R4,-(SP)        ;;PUSH R4 ON STACK
037464  010546                        MOV     R5,-(SP)        ;;PUSH R5 ON STACK
037466  017746  141462                MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
037472  010637  037614                MOV     SP,$SAVR6       ;;SAVE SP
037476  012737  037510  000024        MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
037504  000000                        HALT
037506  000776                        BR      .-2             ;;HANG UP

                                ;;*****************************************************************
                                ;POWER UP ROUTINE
037510  012737  037610  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
037516  013706  037614                MOV     $SAVR6,SP       ;;GET SP
037522  005037  037614                CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
037526  005237  037614          1$:   INC     $SAVR6          ;;WAIT FOR THE INC
037532  001375                        BNE     1$              ;;OF  WORD
037534  005337  037616          2$:   DEC     PWRFLG          ;WAIT AND SET POWER FAIL FLAG
037540  003775                        BLE     2$              ;WAIT FOR FLAG= 1
037542  012677  141406                MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
037546  012605                        MOV     (SP)+,R5        ;;POP STACK INTO R5
037550  012604                        MOV     (SP)+,R4        ;;POP STACK INTO R4
037552  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
037554  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
037556  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
037560  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
037562  012737  037436  000024        MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
037570  012737  000340  000026        MOV     #340,@#PWRVEC+2 ;;PRIO:7
037576  104401                        TYPE                    ;;REPORT THE POWER FAILURE
037600  037620                $PWRMG: .WORD   $POWER          ;;POWER FAIL MESSAGE POINTER
037602  012716                        MOV     (PC)+,(SP)      ;;RESTART AT SATPOW
037604  037636                $PWRAD: .WORD   SATPOW          ;;RESTART ADDRESS
037606  000002                        RTI
037610  000000                $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
037612  000776                        BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
037614  000000                $SAVR6: 0                       ;;PUT THE SP HERE
037616  000000                PWRFLG: .WORD   0               ;INDICATES POWER FAIL OCCURRED WHEN SE
037620      200     042     120 $POWER: .ASCIZ  <CRLF>/'POWER UP'/<CRLF>
                                 EVEN


                                ;POWER UP ROUTINE ,WAIT TWO MINUTES,
                                ;THEN AUTO STARTS AT SIZMEM

037636  005227  000000        SATPOW: INC     #0              ;TTY LOOP, WAIT FOR INCREMENT
037642  001375                        BNE     .-4             ;OF WORD
037644  000005                        RESET                   ;CLEAR THE WORLD
037646  005037  001346                CLR     MINUTE          ;RESET MINUTE COUNT
037652  005037  001350                CLR     SECOND          ;RESET SECOND COUNT
037656  005037  001454                CLR     INTRVL+2        ;RESET THE INTERVAL COUNT
```

```
;15 037662  005037  177776                    CLR     @#PS            ;CLEAR PSW
;16 037666  004737  023344                    JSR     PC,CKCLK        ;START THE CLOCK
;17 037672  022737  000002  001454  1$:       CMP     #2,INTRVL+2     ;TWO MINUTES YET ?
;18 037700  101374                            BHI     1$              ;WAIT IF NOT
;19 037702  012737  000400  001336            MOV     #400,CHGADR     ;FUDGE THE AUTO START
;20 037710  012705  001506                    MOV     #ORDERQ,R5      ;CLEAR UP THE QUE AND BUFFER
;21 037714  005025                    2$:     CLR     (R5)+
;22 037716  022705  002044                    CMP     #BLKADR,R5      ;ALL DONE ?
;23 037722  001374                            BNE     2$              ;BRANCH IF NOT
;24 037724  000137  005112                    JMP     SIZMEM          ;LOOP BACK
```

```
                        .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.

037730  010046          $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
037732  016600  000002          MOV     2(SP),R0        ;;GET TRAP ADDRESS
037736  005740                  TST     -(R0)           ;;BACKUP BY 2
037740  111000                  MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
037742  006300                  ASL     R0              ;;POSITION FOR INDEXING
037744  016000  037764          MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
037750  000200                  RTS     R0              ;;GO TO ROUTINE


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

037752  011646          $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
037754  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
037762  000002                  RTI                     ;;RESTORE THE PSW

                        .SBTTL  TRAP TABLE

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

                        ;       ROUTINE
                        ;       -------

037764  037752  $TRPAD: .WORD   $TRAP2
037766  035450          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
037770  036030          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
037772  036004          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
037774  036044          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
037776  036232          $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)

040000  033616          $GTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING

040002  033526          $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
040004  034070          $RDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
040006  034160          $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
040010  037026          $SAVREG ;;CALL=SAVREG   TRAP+12(104412) SAVE R0-R5 ROUTINE
040012  037064          $RESREG ;;CALL=RESREG   TRAP+13(104413) RESTORE R0-R5 ROUTINE
040014  032616          $DSPLY  ;;CALL=DISPLY   TRAP+14(104414) ROUTINE TO TYPE ERROR MESSAGE
000032  $TERM = .-$TRPAD
```

```
                                .SBTTL  SINGLE/DUAL PORT RH/RM DRIVER (REV 6.5) 1981

                                ;NEW DRIVE TYPE ID FOR RM02 **********
                                ;10-AUG-77 **********
                                ;10-MAR-78 THE SC, SC5 CHANGES
                                ;NEW DRIVE TYPE ID FOR RM05 **********
                                ;1980 **********

                                ;COPYRIGHT (C) 1977,1981
                                ;DIGITAL EQUIPMENT CORP.
                                ;MAYNARD, MA 01754
                                ;AUTHOR(S): JIM LACEY/CHUCK HESS
                                ;REVISED BY: MIKE LEAVITT       11-APR-80, 27-MAR-81

                                ;;*****************************************************************

                                ;STORAGE FOR RMDS, RMER1, RMER2, AND RMMR2 ON AN ERROR "2"
                                        ;RMERRS    - RMDS
                                        ;RMERRS+2 = RMER1
                                        ;RMERRS+4 = RMER2
                                        ;RMERRS+6 = RMMR2

040016  000000  000000  000000  RMERRS: .WORD   0,0,0,0

                                ;TABLE OF DRIVE ACTIVE INDICATORS (DRVACT=8 BYTES)
                                        ;DRVACT=0 IF DRIVE IS IDLE
                                        ;DRVACT>0 IF DRIVE IS ACTIVE WITH A COMMAND
                                        ;DRVACT<0 IF DRIVE IS ACTIVE WITH AN ERROR RECOVERY OPERATION

040026      000                 DRVACT: .BYTE   0               ;DRIVE 0
040027      000                         .BYTE   0               ;DRIVE 1
040030      000                         .BYTE   0               ;DRIVE 2
040031      000                         .BYTE   0               ;DRIVE 3
040032      000                         .BYTE   0               ;DRIVE 4
040033      000                         .BYTE   0               ;DRIVE 5
040034      000                         .BYTE   0               ;DRIVE 6
040035      000                         .BYTE   0               ;DRIVE 7

                                ;TABLE OF DRIVE STATUS INDICATORS (DRVSTA=8 BYTES)
                                        ;DRVSTA=0 IF DRIVE IS OFFLINE OR NONEXSITENT
                                        ;DRVSTA>0 IF DRIVE IS ONLINE
                                        ;DRVSTA<0 IF DRIVE IS UNSAFE

040036      000                 DRVSTA: .BYTE   0               ;DRIVE 0
040037      000                         .BYTE   0               ;DRIVE 1
040040      000                         .BYTE   0               ;DRIVE 2
040041      000                         .BYTE   0               ;DRIVE 3
040042      000                         .BYTE   0               ;DRIVE 4
040043      000                         .BYTE   0               ;DRIVE 5
040044      000                         .BYTE   0               ;DRIVE 6
040045      000                         .BYTE   0               ;DRIVE 7

                                ;TABLE OF DRIVE TYPES (DRVTYP=8 BYTES)
                                        ;DRVTYP=0 IF DRIVE IS NONEXISTENT (DRVSTA=0, ALSO)
                                        ;DRVTYP=7 IF DRIVE IS RM05 ******
                                        ;DRVTYP=5 IF DRIVE IS RM02 ******
                                        ;DRVTYP=4 IF DRIVE IS RM03
```

```
                                        ;DRVTYP -1 IF NO1 RM05/3/2

040046  000             DRVTYP: .BYTE   0               ;DRIVE 0
040047  000                     .BYTE   0               ;DRIVE 1
040050  000                     .BYTE   0               ;DRIVE 2
040051  000                     .BYTE   0               ;DRIVE 3
040052  000                     .BYTE   0               ;DRIVE 4
040053  000                     .BYTE   0               ;DRIVE 5
040054  000                     .BYTE   0               ;DRIVE 6
040055  000                     .BYTE   0               ;DRIVE 7

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

040056  000             DPINT:  .BYTE   0               ;DRIVE 0
040057  000                     .BYTE   0               ;DRIVE 1
040060  000                     .BYTE   0               ;DRIVE 2
040061  000                     .BYTE   0               ;DRIVE 3
040062  000                     .BYTE   0               ;DRIVE 4
040063  000                     .BYTE   0               ;DRIVE 5
040064  000                     .BYTE   0               ;DRIVE 6
040065  000                     .BYTE   0               ;DRIVE 7

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

040066  000             DPRQS:  .BYTE   0               ;DRIVE 0
040067  000                     .BYTE   0               ;DRIVE 1
040070  000                     .BYTE   0               ;DRIVE 2
040071  000                     .BYTE   0               ;DRIVE 3
040072  000                     .BYTE   0               ;DRIVE 4
040073  000                     .BYTE   0               ;DRIVE 5
040074  000                     .BYTE   0               ;DRIVE 6
040075  000                     .BYTE   0               ;DRIVE 7

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

040076  000000          TRNSWT: .WORD   0

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

040100  000000          SRCHWT: .WORD   0

                        ;RM DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)
                                ;ACTDRV=0 IF DRIVER IS INACTIVE
                                ;ACTDRV>0 IF DRIVER IS ACTIVE

040102  000             ACTDRV: .BYTE   0
```

```
                                ;SOFTWARE TIMER ROUTINE ACTIVE FLAG (ACTSTR=1 BYTE)
                                        ;ACTSTR=0 IF SOFTWARE TIMER ROUTINE IS INACTIVE
                                        ;ACTSTR>0 IF SOFTWARE TIMER ROUTINE IS ACTIVE

040103      000                 ACTSTR: .BYTE   0

                                ;UNLOAD FLAG (ULDFLG=8 BYTES)
                                        ;ULDFLG=0 IF NO UNLOAD COMMAND
                                        ;ULDFLG>0 IF UNLOAD COMMAND IN PROGRESS
                                        ;ULDFLG<0 IF UNLOAD COMMAND IN WAIT QUEUE

040104      000                 ULDFLG: .BYTE   0               ;DRIVE 0
040105      000                         .BYTE   0               ;DRIVE 1
040106      000                         .BYTE   0               ;DRIVE 2
040107      000                         .BYTE   0               ;DRIVE 3
040110      000                         .BYTE   0               ;DRIVE 4
040111      000                         .BYTE   0               ;DRIVE 5
040112      000                         .BYTE   0               ;DRIVE 6
040113      000                         .BYTE   0               ;DRIVE 7

                                ;SAVE REGISTERS FLAG (SAVEFG =1 WORD)
                                        ;SAVEFG <0 IF SAVE THE RH/RM REGISTERS WHEN THE
                                        ;OPERATION IS COMPLETED AS PER (DPB+14).
                                        ;SAVEFG=0 IF SAVE THE RH/RM REGISTERS, AS PER
                                        ;(DPB+14), AFTER AN ERROR.

040114  000000              SAVEFG: .WORD   0

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

040116  177777              SEEKFG: .WORD   -1

                                ;TIMEOUT TABLE (TIMER=8 WORDS)
                                        ;THIS TABLE CONTAINS THE TIME ALLOWED FOR AN OPERATION

040120  177777              TIMER:  .WORD   -1              ;DRIVE 0
040122  177777                     .WORD   -1              ;DRIVE 1
040124  177777                     .WORD   -1              ;DRIVE 2
040126  177777                     .WORD   -1              ;DRIVE 3
040130  177777                     .WORD   -1              ;DRIVE 4
040132  177777                     .WORD   -1              ;DRIVE 5
040134  177777                     .WORD   -1              ;DRIVE 6
040136  177777                     .WORD   -1              ;DRIVE 7

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

040140  177777              DTUW:   .WORD   -1

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

```
                                ABIT: .BYTE   1       ;DRIVE 0
                                      .BYTE   2       ;DRIVE 1
                                      .BYTE   4       ;DRIVE 2
                                      .BYTE   10      ;DRIVE 3
                                      .BYTE   20      ;DRIVE 4
                                      .BYTE   40      ;DRIVE 5
                                      .BYTE   100     ;DRIVE 6
                                      .BYTE   200     ;DRIVE 7

                              ;NUMBER OF 'MASSBUS CONTROL PARITY ERRORS'' (MCPE) ALLOWED BEFORE
                              ;CALLING IT FATAL (MCPEMX=1 WORD)

                              MCPEMX: .WORD   3

                              ;STORAGE FOR RMADR (THE FIRST ADDRESS (776700) OF THE RH/RM),
                              ;RMVEC (THE VECTOR ADDRESS (254)), AND RMVEC+2 (THE BR LEVEL (5)).

                              RMADR:  .WORD   176700
                              RMVEC:  .WORD   254,5*32.
                              RHEXT:  .WORD   50              ;OFFSET TO RMBAE

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

                              ;DEFINITIONS OF THE RH/RM ADDRESS INDEXES

                              RMCS1   = 0                     ;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 0)
                              RMWC    = 2                     ;WORD COUNT REGISTER (NOT A DRIVE REG)
                              RMBA    = 4                     ;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)
                              RMDA    = 6                     ;DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 5)
                              RMCS2   = 10                    ;CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
                              RMDS    = 12                    ;DRIVE STATUS REGISTER (DRIVE REG 1)
                              RMER1   = 14                    ;ERROR REGISTER #1 (DRIVE REG. 2)
                              RMAS    = 16                    ;ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 4)
                              RMLA    = 20                    ;LOOK AHEAD REGISTER (DRIVE REG. 7)
                              RMDB    = 22                    ;DATA BUFFER REGISTER (NOT A DRIVE REG.)
                              RMMR1   = 24                    ;MAINTAINABILITY REGISTER (DRIVE REG. 3)
                              RMDT    = 26                    ;DRIVE TYPE REGISTER (DRIVE REG. 6)
                              RMSN    = 30                    ;SERIAL NUMBER REGISTER (DRIVE REG. 10)
                              RMOF    = 32                    ;OFFSET REGISTER (DRIVE REG. 11)
                              RMDC    = 34                    ;DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
                              RMHR    = 36                    ;DUMMY  ADDRESS REGISTER (DRIVE REG. 13)
                              RMMR2   = 40                    ;MAINTENANCE REGISTER #2
                              RMER2   = 42                    ;ERROR REGISTER #2 (DRIVE REG. 15)
                              RMEC1   = 44                    ;ECC POSITION REGISTER (DRIVE REG. 16)
                              RMEC2   = 46                    ;ECC PATTERN REGISTER (DRIVE REG.  17)
                              RMBAE   = 50                    ;BUS ADDRESS EXTENTION REGISTER
                              RMCS3   = 52                    ;CONTROL AND STATUS REGISTER #3

                              .SBTTL  RH/RM DRIVER INITIALIZATION CODE

                              ;THIS ROUTINE WILL DETERMINE WHICH RM DRIVES ARE
                              ;AVAILABLE FOR TESTING AND SET THE DRVSTA INDICATOR
                              ;TO THE PROPER STATE FOR EACH DRIVE.
                              ;NOTE: THIS ROUTINE CALLS DRVINT
                              ;
```

```
                                        ;CALL
                                        ;
                                        ;       JSR     PC,RMINIT
                                        ;       RETURN
                                        ;
                                        ;NOTE: THE 'P' OR 'L' CLOCK MUST BE STARTED
                                        ;
040166  104412                  RMINIT: SAVREG                  ;SAVE R0 - R5
040170  013746  177776                  MOV     PS,-(SP)        ;SAVE THE PRESENT PROCESSOR STATUS
040174  012737  000240  177776          MOV     #<5*32.>,PS     ;CHANGE THE PRIORITY TO 5
040202  004737  045740                  JSR     PC,CLRQUE       ;CLEAR ALL REQUEST QUEUES
040206  012701  040016                  MOV     #RMERRS,R1      ;FIRST ADDRESS TO BE CLEARED
040212  012702  040116                  MOV     #SEEKFG,R2      ;LAST ADDRESS TO BE CLEARED
040216  005021                  1$:     CLR     (R1)+           ;CLEAR
040220  020102                          CMP     R1,R2           ;ARE WE DONE?
040222  103775                          BLO     1$              ;BR IF NO
040224  012702  040140                  MOV     #DTUW,R2        ;LAST ADDRESS
040230  012721  177777          2$:     MOV     #-1,(R1)+       ;INITIALIZE
040234  020102                          CMP     R1,R2           ;DONE?
040236  101774                          BLOS    2$              ;LOOP IF NO
040240  005037  040036                  CLR     DRVSTA          ;SET ALL DRIVES TO OFFLINE
040244  005037  040040                  CLR     DRVSTA+2
040250  005037  040042                  CLR     DRVSTA+4
040254  005037  040044                  CLR     DRVSTA+6
040260  013703  040156                  MOV     RMVEC,R3        ;SETUP THE RH/RM VECTOR
040264  012723  042614                  MOV     #ISR,(R3)+
040270  013713  040160                  MOV     RMVEC+2,(R3)
040274  013704  040154                  MOV     RMADR,R4        ;FIRST ADDRESS OF RH/RM
040300  012764  000040  000010          MOV     #CLR,RMCS2(R4)  ;MASSBUS INIT
040306  005001                          CLR     R1              ;START WITH DRIVE 0
040310  004037  040400          3$:     JSR     R0,DRVINT       ;INIT THE DRIVE
040314  000401                          BR      4$              ;'DVA' NOT SET OR PARITY ERROR
040316  000402                          BR      5$              ;NORMAL RETURN
040320  105061  040036          4$:     CLRB    DRVSTA(R1)      ;SET DRIVE STATUS TO OFFLINE
040324  005201                  5$:     INC     R1              ;GO TO NEXT DRIVE
040326  042701  177770                  BIC     #^C7,R1         ;MASK OUT UNUSED BITS
040332  001366                          BNE     3$              ;BR IF MORE DRIVES TO GO
040334  012701  000007                  MOV     #7,R1           ;START WITH DRIVE 7
040340  005037  177776                  CLR     PS              ;CLEAR THE PROCESSOR STATUS
040344  105761  040056          6$:     TSTB    DPINT(R1)       ;WAITING FOR DRIVE TO SWITCH PORTS ?
040350  001405                          BEQ     8$              ;BR NOT WAITING
040352  004737  045374                  JSR     PC,SET.IE       ;SET INTERRUPT
040356  105761  040056          7$:     TSTB    DPINT(R1)       ;DRIVE SWITCHED PORTS ?
040362  001375                          BNE     7$              ;BR IF NOT
040364  005301                  8$:     DEC     R1              ;GO TO THE NEXT DRIVE
040366  100366                          BPL     6$              ;CHECK NEXT DRIVE
040370  012637  177776                  MOV     (SP)+,PS        ;RESTORE THE PROCESSOR STATUS
040374  104413                          RESREG                  ;RESTORE R0 - R5
040376  000207                          RTS     PC              ;BYE-BYE
                                        ;
                                        ;DRIVE INITILIZATION ROUTINE
                                        ;THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
                                        ;AN RM05/3/2. IF IT IS, A "READ-IN PRESET" IS ISSUED AND FMT16
                                        ;IS SET TO A "1". THEN MOL, DPR, DRY, AND VV ARE CHECKED TO
                                        ;INSURE THEY ARE ALL ON A "1". AND DEPENDING ON THEIR STATE,
                                        ;DRVSTA IS SET TO THE PROPER CONDITION.
                                        ;CALL
```

```
                                        ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
                                        ;       MOV     RMADR,R4        ;UNIBUS ADDRESS OF RH/RM (RMCS1)
                                        ;       JSR     R0,DRVINT       ;CALLED BY A JSR
                                        ;       RETURN1                 ;ERROR OCCURRED (PARITY)
                                        ;       RETURN2                 ;NORMAL RETURN
                                        ;
040400  010546                  DRVINT: MOV     R5,-(SP)        ;SAVE R5
040402  105061  040036                  CLRB    DRVSTA(R1)      ;START DRIVE STATUS AS OFFLINE
040406  105061  040046                  CLRB    DRVTYP(R1)      ;CLEAR THE DRIVE TYPE INDICATOR
040412  105061  040104                  CLRB    ULDFLG(R1)      ;CLEAR THE UNLOAD FLAG
040416  010164  000010                  MOV     R1,RMCS2(R4)    ;SELECT A DRIVE
040422  112764  000111  000000          MOVB    #111,RMCS1(R4)  ;DO A DRIVE CLEAR COMMAND (& SEIZE DRIVE)
040430  032764  010000  000010          BIT     #BIT12,RMCS2(R4)        ;NONEXISTENT DRIVE?
040436  001403                          BEQ     1$              ;NO
040440  004737  045374                  JSR     PC,SET.IE       ;GO SET ''IE'' WITHOUT A ''TRE''
040444  000520                          BR      4$              ;LEAVE THIS ROUTINE
                                
040446  105061  040036          1$:     CLRB    DRVSTA(R1)      ;SET DRIVE STATUS TO OFFLINE
040452  032764  004000  000000          BIT     #BIT11,RMCS1(R4)        ;SEE IF DRIVE AVAILABLE
040460  001512                          BEQ     4$              ;BR IF DRIVE NOT AVAILABLE
040462  004037  044666                  JSR     R0,RD.RM        ;READ THE DRIVE TYPE REG.
040466  000026                          RMDT
040470  040710                          5$                      ;ERROR RETURN ADDRESS
040472  012605                          MOV     (SP)+,R5        ;PUT DRIVE TYPE IN R5
040474  112761  000004  040046          MOVB    #4,DRVTYP(R1)   ;SET RM03 INDICATOR
040502  022705  020024                  CMP     #20024,R5       ;SINGLE PORT RM03 ?
040506  001431                          BEQ     2$              ;BR IF YES
040510  022705  024024                  CMP     #24024,R5       ;DUAL PORT RM03 ?
040514  001426                          BEQ     2$              ;BR IF YES
040516  112761  000005  040046          MOVB    #5,DRVTYP(R1)   ;SET RM02 INDICATOR
040524  022705  020025                  CMP     #20025,R5       ;SINGLE PORT RM02 ?
040530  001420                          BEQ     2$              ;BR IF SO
040532  022705  024025                  CMP     #24025,R5       ;DUAL PORT RM02 ?
040536  001415                          BEQ     2$              ;BR IF SO
040540  112761  000007  040046          MOVB    #7,DRVTYP(R1)   ;SET RM05 INDICATOR
040546  022705  020027                  CMP     #20027,R5       ;SINGLE PORT RM05 ?
040552  001407                          BEQ     2$              ;BR IF YES
040554  022705  024027                  CMP     #24027,R5       ;DUAL PORT RM05 ?
040560  001404                          BEQ     2$              ;BR IF YES
040562  112761  177777  040046          MOVB    #-1,DRVTYP(R1)  ;SET INDICATOR TO 'OTHER'
040570  000446                          BR      4$              ;EXIT
                                
040572  012746  000121          2$:     MOV     #121,-(SP)      ;DO A 'READ-IN PRESET''
040576  004037  045042                  JSR     R0,WRT.RM
040602  000000                          RMCS1
040604  040710                          5$
040606  012746  010000                  MOV     #BIT12,-(SP)    ;SET FMT16=1
040612  004037  045042                  JSR     R0,WRT.RM
040616  000032                          RMOF
040620  040710                          5$
040622  004037  044666                  JSR     R0,RD.RM        ;READ RMDS
040626  000012                          RMDS
040630  040710                          5$
040632  012605                          MOV     (SP)+,R5        ;AND SAVE IT IN R5
040634  100015                          BPL     3$              ;BR IF ATA=0
040636  116164  040142  000016          MOVB    ATABIT(R1),RMAS(R4)     ;CLEAR ATTENTION BIT
```

```
040644  004037  044666                  JSR     R0,RD.RM        ;FIND OUT WHY ATA 1
040650  000014                          RMER1
040652  040710                          5$
040654  006126                          ROL     (SP)+           ;IS IT UNSAFE?
040656  100004                          BPL     3$              ;BR IF NOT
040660  112761  177777  040036          MOVB    #-1,DRVSTA(R1)  ;SET UNSAFE INDICATOR
040666  000407                          BR      4$              ;EXIT

040670  005105                  3$:     COM     R5              ;CHECK MOL, DPR, DRY, AND VV
040672  042705  167077                  BIC     #^C<BIT12!BIT08!BIT07!BIT06>,R5
040676  001003                          BNE     4$              ;BR IF MOL, DPR, DRY, OR VV IS CLEAR
040700  112761  000001  040036          MOVB    #1,DRVSTA(R1)   ;SET DRIVE STATUS TO ONLINE
040706  005720                  4$:     TST     (R0)+           ;STEP OVER THE ERROR RETURN
040710  012605                  5$:     MOV     (SP)+,R5        ;RESTORE R5
040712  000200                          RTS     R0              ;EXIT

                                ;REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
                                ;
                                ;CALL
                                ;
                                ;       JSR     R0,RM05         ;CALL THE RM05 DRIVER
                                ;       PNTADR                  ;ADDRESS OF POINTER OF DRIVES PARAMETER BLOCK
                                ;       RETURN1                 ;RETURN HERE IF QUEUE IS FULL
                                ;       RETURN2                 ;RETURN HERE IF REQUEST IS IN QUEUE OR THERE
                                ;                               ;IS AN ERROR CONDITION

040714  013746  177776          RM05:   MOV     PS,-(SP)        ;SAVE THE CALLING STATUS
040720  013737  040160  177776          MOV     RMVEC+2,PS      ;DON'T ALLOW ANY RM INTERRUPTS
040726  112737  000001  040102          MOVB    #1,ACTDRV       ;SET ''ACTIVE DRIVER'' FLAG
040734  104412                          SAVREG                  ;SAVE R0 - R5
040736  011002                          MOV     (R0),R2         ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
040740  005062  000016                  CLR     16(R2)          ;CLEAR THE STATUS/ERROR INDICATOR
040744  111201                          MOVB    (R2),R1         ;PICKUP THE DRIVE NUMBER
040746  013704  040154                  MOV     RMADR,R4        ;UNIBUS ADDRESS OF RMCS1
040752  105761  040036                  TSTB    DRVSTA(R1)      ;CHECK DRIVES STATUS
040756  003014                          BGT     1$              ;BR IF ONLINE
040760  105761  040104                  TSTB    ULDFLG(R1)      ;UNLOAD COMMAND IN QUEUE?
040764  001036                          BNE     3$              ;BR IF YES
040766  105761  040056                  TSTB    DPINIT(R1)      ;TRYING TO INIT THE DRIVE
040772  001042                          BNE     5$              ;BR IF YES
040774  004037  040400                  JSR     R0,DRVINT       ;GO INIT. THE DRIVE
041000  000434                          BR      4$              ;ERROR RETURN
041002  105761  040036                  TSTB    DRVSTA(R1)      ;IS DRIVE STATUS ONLINE?
041006  003445                          BLE     6$              ;BR IF NOT
041010  105761  040066          1$:     TSTB    DPRQS(R1)       ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
041014  001031                          BNE     5$              ;BR IF YES
041016  010164  000010                  MOV     R1,RMCS2(R4)    ;SELECT THE DRIVE
041022  004037  046036                  JSR     R0,DRVQUE       ;PUT THIS REQUEST IN QUEUE
041026  000460                          BR      9$              ;QUEUE IS FULL
041030  122762  000103  000002          CMPB    #103,2(R2)      ;IS THIS REQ. FOR AN UNLOAD?
041036  001003                          BNE     2$              ;BR IF NO
041040  112761  177777  040104          MOVB    #-1,ULDFLG(R1)  ;SET THE 'UNLOAD IN QUEUE'' FLAG
041046  105761  040026          2$:     TSTB    DRVACT(R1)      ;IS THIS DRIVE ACTIVE?
041052  001043                          BNE     8$              ;BR IF YES
041054  004737  041206                  JSR     PC,OPT          ;CALL THE OPTIMIZER
041060  000440                          BR      8$
041062  012762  120000  000016  3$:     MOV     #BIT15.BIT13,16(R2)     ;SET THE 'UNLOAD IN QUEUE'' ERROR FLAG
```

```
041070  000434                          BR      8$              ;EXIT
041072  004737  042264          4$:     JSR     PC,CI7          ;GO HANDLE THE PARITY ERROR
041076  000431                          BR      8$
041100  004037  046036          5$:     JSR     R0,DRVQUE       ;PUT REQUEST IN QUEUE
041104  000431                          BR      9$              ;QUEUE IS FULL
041106  032714  000100                  BIT     #BIT06,(R4)     ;IE BIT SET ?
041112  001023                          BNE     8$              ;YES
041114  004737  045374                  JSR     PC,SET.IE       ;SET THE INTERRUPT
041120  000420                          BR      8$              ;RETURN
041122  105761  040036          6$:     TSTB    DRVSTA(R1)      ;SEE IF DRIVE OFFLINE OR UNSAFE
041126  002412                          BLT     7$              ;BR IF UNSAFE
041130  012762  140000  000016          MOV     #BIT15.BIT14,16(R2)     ;SET OFFLINE ERROR INDICATOR
041136  105761  040046                  TSTB    DRVTYP(R1)      ;SEE IF OFFLINE OR NONEXISTENT
041142  001007                          BNE     8$              ;BR IF OFFLINE
041144  012762  100002  000016          MOV     #BIT15.BIT01,16(R2)     ;REPORT DRIVE NONEXISTENT
041152  000403                          BR      8$              ;GO TO EXIT
041154  012762  110000  000016  7$:     MOV     #BIT15.BIT12,16(R2)     ;DRIVE IS UNSAFE
041162  104413                  8$:     RESREG                  ;RESTORE R0 - R5
041164  005720                          TST     (R0)+           ;SETUP FOR NORMAL RETURN
041166  000401                          BR      10$             ;FINISH UP, THEN EXIT
041170  104413                  9$:     RESREG                  ;RESTORE R0 - R5
041172  005720                  10$:    TST     (R0)+           ;CORRECT THE RETURN ADDRESS
041174  105037  040102                  CLRB    ACTDRV          ;CLEAR "ACTIVE DRIVER" FLAG
041200  012637  177776                  MOV     (SP)+,PS        ;RETURN "PS" TO USER LEVEL
041204  000200                          RTS     R0              ;RETURN TO CALLER

                                ;OPTIMIZER-CALLED FOR A PARTICULAR DRIVE
                                ;
                                ;CALL
                                ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
                                ;       JSR     PC,OPT          ;SETUP A COMMAND

041206  104412                  OPT:    SAVREG                  ;SAVE R0 - R5
041210  013746  177776                  MOV     PS,-(SP)        ;SAVE PROC. STATUS
041214  146137  040142  040100          BICB    ATABIT(R1),SRCHWT       ;CLEAR LA SEACH FLAG
041222  105061  040066                  CLRB    DPRQS(R1)       ;RESET THE PORT REQ FLAG ****
041226  004737  046112                  JSR     PC,GETREQ       ;GET "DPB" POINTER OF REQUEST
041232  005702                          TST     R2              ;IS THERE A REQUEST IN QUEUE?
041234  001466                          BEQ     7$              ;NO--BR TO EXIT
041236  010164  000010                  MOV     R1,RMCS2(R4)    ;LOAD THE DRIVE ADDRESS *******
041242  012764  000111  000000          MOV     #111,RMCS1(R4)  ;CLEAR THE DRIVE
041250  032764  004000  000000          BIT     #BIT11,RMCS1(R4)        ;DVA SET ?
041256  001442                          BEQ     5$              ;TO PORT REQUEST ,IF NOT
041260  105761  040036          1$:     TSTB    DRVSTA(R1)      ;IS DRIVE ONLINE?
041264  003014                          BGT     2$              ;YES
041266  004737  046134                  JSR     PC,POPQUE       ;NO--REMOVE REQUEST FROM QUEUE
041272  012762  140000  000016          MOV     #BIT15!BIT14,16(R2)     ;SET OFFLINE STATUS/ERROR INDICATOR
041300  105761  040036                  TSTB    DRVSTA(R1)      ;IS DRIVE UNSAFE ?
041304  100047                          BPL     8$              ;BR TO EXIT IF NOT
041306  012762  110000  000016          MOV     #BIT15!BIT12,16(R2)     ;SET UNSAFE STATUS/ERROR INDICATOR
041314  000443                          BR      8$              ;BR TO EXIT
041316                          2$:
041316  122762  000150  000002          CMPB    #150,2(R2)      ;IS THE REQUEST FOR I/O?
041324  002403                          BLT     3$              ;YES
041326  004737  041650                  JSR     PC,CI4          ;CALL THE COMMAND INITIATOR
041332  000434                          BR      8$              ;BR TO EXIT
041334  005737  040140          3$:     TST     DTUW            ;DATA TRANSFER UNDERWAY?
```

```
 441 041340  002006                          BGE     4$              ;YES--GO START A SEARCH
 442 041342  005737  040116                  TST     SEFKFG          ;DO IMPLIED SEEKS?
 443 041346  100003                          BPL     4$              ;NO, DO A SEARCH
 444 041350  004737  041434                  JSR     PC,CI1          ;START A DATA TRANSFER
 445 041354  000423                          BR      8$
 446 041356  004737  041542          4$:     JSR     PC,CI3          ;START A SEARCH
 447 041362  000420                          BR      8$              ;GO TO THE EXIT
 448 041364  112761  177777  040066  5$:     MOVB    #-1,DPRQS(R1)   ;SET PORT REQUEST INDICATOR
 449 041372  010103                          MOV     R1,R3           ;SET UP TO ADDRESS WORDS
 450 041374  006303                          ASL     R3              ;CONVERT TO WORD INDEX
 451 041376  012763  035230  040120          MOV     #15000.,TIMER(R3)       ;START 15. SECOND TIMER
 452 041404  000402                          BR      7$              ;EXIT
 453 041406  004737  042264          6$:     JSR     PC,CI7          ;PROCESS THE PARITY ERROR
 454 041412  032714  000100          7$:     BIT     #BIT06,(R4)     ;SEE IF 'IE' ALREADY SET
 455 041416  001002                          BNE     8$              ;BR IF SET
 456 041420  004737  045374                  JSR     PC,SET.IE       ;SET ''IE'' WITHOUT A ''TRE''
 457 041424  012637  177776          8$:     MOV     (SP)+,PS        ;RESTORE PROC. STATUS
 458 041430  104413                          RESREG                  ;RESTORE R0 - R5
 459 041432  000207                          RTS     PC
 460
 461                                 ;COMMAND INITIATOR
 462                                 ;
 463                                 ;CALL
 464                                 ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER
 465                                 ;       MOV     #DPB,R2         ;ADDRESS OF DPB
 466                                 ;       JSR     PC,CI?          ;CI?= CI1,CI3, OR CI4
 467                                 ;                               ;WHERE:
 468                                 ;                               ;CI1=DATA TRANSFER
 469                                 ;                               ;CI3=SEARCH REQUESTED BY DATA XFER
 470                                 ;                               ;CI4=NOT DATA TRANSFER
 471
 472 041434  004737  046134  CI1:     JSR     PC,POPQUE       ;REMOVE REQUEST FROM 'DRIVES WAIT'' QUEUE
 473 041440  010237  040076          MOV     R2,TRNSWT       ;PUT REQ. IN TRANSFER WAIT QUEUE
 474 041444  010203                  MOV     R2,R3           ;DPB ADDRESS TO R3
 475 041446  013704  040154          MOV     RMADR,R4        ;RMCS1 ADDRESS
 476 041452  010164  000010          MOV     R1,RMCS2(R4)    ;SELECT DRIVE
 477 041456  062703  000004          ADD     #4,R3           ;DESIRED WORD COUNT
 478 041462  062704  000002          ADD     #2,R4           ;RMWC ADDRESS
 479 041466  012324                  MOV     (R3)+,(R4)+     ;LOAD WORD COUNT
 480 041470  012324                  MOV     (R3)+,(R4)+     ;LOAD BUFFER ADDRESS
 481 041472  012346                  MOV     (R3)+,-(SP)     ;LOAD SECTOR AND TRACK
 482 041474  004037  045042          JSR     R0,WRT.RM       ;CALL THE LOAD(WRITE) ROUTINE
 483 041500  000006                  RMDA                    ;INDEX OF REGISTER TO LOAD
 484 041502  042264                  CI7                     ;ERROR RETURN ADDRESS
 485 041504  012346                  MOV     (R3)+,-(SP)     ;LOAD CYLINDER ADDRESS
 486 041506  004037  045042          JSR     R0,WRT.RM
 487 041512  000034                  RMDC
 488 041514  042264                  CI7
 489 041516  016246  000002          MOV     2(R2),-(SP)     ;LOAD ''COMMAND+GO'', ''A17&A16'', AND ''PSEL''
 490 041522  004037  045042          JSR     R0,WRT.RM
 491 041526  000000                  RMCS1
 492 041530  042264                  CI7
 493 041532  010137  040140          MOV     R1,DTUW         ;SET 'DATA TRANSFER UNDERWAY''
 494 041536  000137  042226          JMP     CI5
 495
 496 041542  013704  040154  CI3:     MOV     RMADR,R4        ;RMCS1 ADDRESS
 497 041546  010164  000010          MOV     R1,RMCS2(R4)    ;SELECT DRIVE
```

;

```
 498 041552  016246  000012                    MOV     12(R2),-(SP)    ;DESIRED CYLINDER ADDRESS
 499 041556  004037  045042                    JSR     R0,WRT.RM
 500 041562  000034                            RMDC
 501 041564  042264                            CI7
 502 041566  116203  000010                    MOVB    10(R2),R3       ;PICKUP SECTOR ADDRESS
 503 041572  163703  040164                    SUB     MXWNDW,R3       ;BACKUP BY MAX. SEARCH FOR I/O WINDOW
 504 041576  002002                            BGE     1$
 505 041600  062703  000040                    ADD     #32.,R3
 506 041604  010346                1$:         MOV     R3,-(SP)        ;COMBINE THE ADJUSTED SECTOR WITH
 507 041606  116266  000011  000001            MOVB    11(R2),1(SP)    ;THE DESIRED TRACK
 508 041614  004037  045042                    JSR     R0,WRT.RM       ;LOAD DESIRED TRACK & SECTOR
 509 041620  000006                            RMDA
 510 041622  042264                            CI7
 511 041624  012746  000131                    MOV     #131,-(SP)      ;START A SEARCH
 512 041630  004037  045042                    JSR     R0,WRT.RM
 513 041634  000000                            RMCS1
 514 041636  042264                            CI7
 515 041640  156137  040142  040100            BISB    ATABIT(R1),SRCHWT       ;SET "SEARCH WAIT" KEY
 516 041646  000567                            BR      CI5
 517
 518 041650  013704  040154          CI4:      MOV     RMADR,R4        ;RMCS1 ADDRESS
 519 041654  010164  000010                    MOV     R1,RMCS2(R4)    ;SELECT DRIVE
 520 041660  116203  000002                    MOVB    2(R2),R3        ;PICKUP THE REQUESTED COMMAND
 521 041654  122703  000131                    CMPB    #131,R3         ;IS IT A SEARCH COMMAND?
 522 041670  001007                            BNE     1$              ;BR IF NO
 523 041672  016246  000010                    MOV     10(R2),-(SP)    ;LOAD DESIRED TRACK & SECTOR
 524 041676  004037  045042                    JSR     R0,WRT.RM
 525 041702  000006                            RMDA
 526 041704  042264                            CI7
 527 041706  000403                            BR      2$              ;GO LOAD CYLINDER
 528 041710  122703  000105          1$:       CMPB    #105,R3         ;IS IT A SEEK COMMAND
 529 041714  001007                            BNE     3$              ;BR IF NO
 530 041716  016246  000012          2$:       MOV     12(R2),-(SP)    ;LOAD DESIRED CYLINDER
 531 041722  004037  045042                    JSR     R0,WRT.RM
 532 041726  000034                            RMDC
 533 041730  042264                            CI7
 534 041732  000546                            BR      CI6
 535 041734  122703  000115          3$:       CMPB    #115,R3         ;IS IT AN "OFFSET" COMMAND?
 536 041740  001013                            BNE     4$              ;BR IF NO
 537 041742  004037  044666                    JSR     R0,RD.RM        ;MERGE THE OFFSET VALUE INTO RMOF
 538 041746  000032                            RMOF                    ;BUT DON'T CHANGE THE UPPER
 539 041750  042264                            CI7
 540 041752  116216  000001                    MOVB    1(R2),(SP)      ;BYTE WHEN LOADING THE
 541 041756  004037  045042                    JSR     R0,WRT.RM       ;REGISTER (RMOF)
 542 041762  000032                            RMOF
 543 041764  042264                            CI7
 544 041766  000530                            BR      CI6             ;GO START THE COMMAND
 545 041770  122703  000107          4$:       CMPB    #107,R3         ;IS IT A "RECALIBRATE" COMMAND?
 546 041774  001525                            BEQ     CI6             ;BR IF YES
 547 041776  122703  000117                    CMPB    #117,R3         ;IS IT A RETURN TO CENTER?
 548 042002  001522                            BEQ     CI6             ;BR IF YES
 549 042004  122703  000103                    CMPB    #103,R3         ;IS IT AN "UNLOAD" COMMAND?
 550 042010  001016                            BNE     5$              ;BR IF NO
 551 042012  112761  000001  040026            MOVB    #1,DRVACT(R1)   ;SET THE DRIVE ACTIVE INDICATOR
 552 042020  105061  040036                    CLRB    DRVSTA(R1)      ;PUT DRIVE STATUS TO OFFLINE
 553 042024  112761  000001  040104            MOVB    #1,ULDFLG(R1)   ;SET "UNLOAD IN PROGRESS" FLAG
 554 042032  010346                            MOV     R3,-(SP)        ;START THE "UNLOAD" COMMAND
```

```
555 042034  004037  045042            JSR     R0,WRT.RM
556 042040  000000                    RMCS1
557 042042  042264                    CI7
558 042044  000207                    RTS     PC              ;RETURN TO USER
559 042046  122703  000143      5$:   CMPB    #143,R3         ;IS IT A ''SET FORMAT'' COMMAND?
560 042052  001014                    BNE     6$              ;BR IF NO
561 042054  004037  044666            JSR     R0,RD.RM        ;READ THE OFFSET REGISTER
562 042060  000032                    RMOF
563 042062  042264                    CI7
564 042064  116266  000001  000001    MOVB    1(R2),1(SP)     ;COMBINE ''FMT16'',''ECI'', AND ''HCI''
565 042072  004037  045042            JSR     R0,WRT.RM       ;LOAD ''FMT16'', ''ECI'', AND/OR ''HCI''.
566 042076  000032                    RMOF
567 042100  042264                    CI7
568 042102  000436                    BR      12$
569 042104  122703  000141      6$:   CMPB    #141,R3         ;IS IT A ''GET REGISTER'' COMMAND?
570 042110  001023                    BNE     10$             ;BR IF NO
571 042112  016203  000006      7$:   MOV     6(R2),R3        ;POINTS TO 1ST ADDRESS OF WHERE
572                                                           ;TO PUT THE REGISTER(S)
573 042116  116237  000010  042134    MOVB    10(R2),9$       ;INIT. THE INDEX FOR THE FIRST REG.
574 042124  116205  000011            MOVB    11(R2),R5       ;INDEX OF LAST REG. TO MOVE
575 042130  004037  044666      8$:   JSR     R0,RD.RM        ;READ RH/RM REGISTER
576 042134  000000      9$:         RMCS1                     ;INDEX OF REG. TO READ
577 042136  042264                    CI7
578 042140  012623                    MOV     (SP)+,(R3)+     ;GET THE CONTENTS OF RH/RM REG.
579 042142  023705  042134            CMP     9$,R5           ;LAST REG. BEEN READ?
580 042146  001414                    BEQ     12$             ;GET OUT IF YES
581 042150  062737  000002  042134    ADD     #2,9$           ;INCREASE THE INDEX BY 2
582 042156  000764                    BR      8$              ;LOOP--MORE TO READ
583 042160  122703  000145     10$:   CMPB    #145,R3         ;IS IT A ''SELECT DRIVE'' COMMAND?
584 042164  001405                    BEQ     12$             ;BR IF YES
585 042166  010346      11$:         MOV     R3,-(SP)        ;LOAD THE COMMAND
586 042170  004037  045042            JSR     R0,WRT.RM
587 042174  000000                    RMCS1
588 042176  042264                    CI7
589 042200  004737  046134     12$:   JSR     PC,POPQUE       ;REMOVE REQ. FROM QUEUE
590 042204  052762  000200  000016    BIS     #BIT07,16(R2)   ;SET THE ''DONE'' BIT
591 042212  005737  040114            TST     SAVEFG          ;SAVE THE RH/RM REGISTERS?
592 042216  100002                    BPL     13$             ;BR IF NO
593 042220  004737  045230            JSR     PC,SVRH70       ;YES--GO SAVE THE REGISTERS
594 042224  000207      13$:         RTS     PC              ;RETURN TO USER
595
596 042226  006301      CI5:         ASL     R1
597 042230  012761  023420  040120    MOV     #10000.,TIMER(R1)       ;START 10. SECOND TIMER
598 042236  006201                    ASR     R1
599 042240  112761  000001  040026    MOVB    #1,DRVACT(R1)   ;SET THE DRIVE ACTIVE
600 042246  000207                    RTS     PC              ;RETURN TO THE USER
601
602 042250  010346      CI6:         MOV     R3,-(SP)        ;LOAD THE COMMAND
603 042252  004037  045042            JSR     R0,WRT.RM
604 042256  000000                    RMCS1
605 042260  042264                    CI7
606 042262  000761                    BR      CI5
607
608 042264  032764  010000  000010  CI7:   BIT     #BIT12,RMCS2(R4)        ;DRIVE NON-EXISTENT ?
612 042272  005702      1$:          TST     R2              ;ANYTHING IN QUEUE ?
616 042274  001001                    BNE     2$              ;BR IF QUEUE IS THERE
617 042276  000207                    RTS     PC              ;OTHERWISE EXIT
```

```
                                                ;SET ''PARITY'' ERROR INDICATOR
                                                ;DO A 'DRIVE CLEAR''
                                                ;EMPTY THE QUEUE
                                                ;CLEAR THE PORT REQUEST FLAG
                                                ;CLEAR THE UNLOAD IN QUEUE FLAG
                                                ;DRIVE IS IDLE
                                                ;IF THIS DRIVE HAD AN I/O REQUEST
                                                ;IN PROGRESS CLEAR ALL OF THE FLAGS
                                                ;SAVE R0 - R5
                                                ;DRIVE ACTIVE?
                                                ;BR IF IN ACTIVE
                                                ;PORT REQUEST
                                                ;BR IF NOT
                                                ;GET THE ''TRANSFER WAIT'' QUEUE
                                                ;DID THIS DRIVE HAVE AN I/O IN PROGRESS?
                                                ;BR IF YES
                                                ;GET THE DPB POINTER
                                                ;QUEUE ENTRY FOR DRIVE ?
                                                ;BR IF NOT
                                                ;'NED' SET ?
                                                ;BR IF NOT
                                                ;SET 'DRIVE NON-EXISTENT' INDICATOR
                                                ;CONTINUE
                                                ;SET 'NON-CLEARABLE PARITY'' ERROR INDICATOR
                                                ;STOP THE TIMER
                                                ;SET 'DRIVE ACTIVE'' TO IDLE
                                                ;CLEAR PORT REQUEST FLAG
                                                ;IS THIS DRIVE SETUP FOR A TRANSFER
                                                ;BR IF NOT
                                                ;RESET THE INDICATOR
                                                ;CLEAR THE TRANSFER QUEUE
                                                ;CLEAR UNLOAD FLAG
                                                ;'NED' SET ?
                                                ;MOVE TO THE NEXT DRIVE
                                                ;BR IF MORE DRIVES
                                                ;NO DATA TRANSFERS UNDERWAY
                                                ;CLEAR THE 'TRANSFER WAIT' QUEUE
                                                ;CLEAR ALL OF THE REQUEST QUEUES
                                                ;DO A MASSBUS INIT.
                                                ;CONTINUE
                                                ;CLEAR THE DRIVE'S QUEUE
                                                ;SET DRIVE TO OFFLINE
                                                ;CLEAR THE DRIVE TYPE INDICATOR
                                                ;SET ''IE'' WITHOUT ''TRE''
                                                ;RESTORE R0 - R5
```

```
687 042612  000207                          RTS     PC              ;RETURN
688
689                                 ;INTERRUPT SERVICE ROUTINE
690
691 042614  112737  000001  040102  ISR:    MOVB    #1,ACTDRV       ;SET ''ACTIVE DRIVER'' FLAG
692 042622  104412                          SAVREG                  ;SAVE R0 - R5
693 042624  013704  040154                  MOV     RMADR,R4        ;ADDRESS OF RMCS1
694 042630  013701  040140                  MOV     DTUW,R1         ;GET 'DATA TRANSFER UNDERWAY'' INDICATOR
695 042634  002402                          BLT     1$              ;BR IF NO DATA TRANSFER UNDERWAY
696 042636  004737  042656                  JSR     PC,TD           ;CALL TRANSFER DONE
697 042642  004737  043120          1$:     JSR     PC,SC           ;CALL SPECIAL CONDITIONS
698 042646  104413                  2$:     RESREG                  ;RESTORE R0 - R5
699 042650  105037  040102                  CLRB    ACTDRV          ;CLEAR ''ACTIVE DRIVER'' FLAG
700 042654  000002                          RTI                     ;RETURN
701
702                                 ;TRANSFER DONE ROUTINE
703
704 042656  105061  040026          TD:     CLRB    DRVACT(R1)      ;SET DRIVE ACTIVE INDICATOR TO IDLE
705 042662  012737  177777  040140          MOV     #-1,DTUW        ;NO DATA TRANSFERS UNDERWAY
706 042670  006301                          ASL     R1
707 042672  012761  177777  040120          MOV     #-1,TIMER(R1)   ;CANCEL TIMEOUT
708 042700  006201                          ASR     R1
709 042702  013702  040076                  MOV     TRNSWT,R2       ;GET 'DPB'' ADDRESS FROM THE
710 042706  005037  040076                  CLR     TRNSWT          ;TRANSFER WAIT QUEUE--CLEAR QUEUE
711 042712  052762  000200  000016          BIS     #BIT07,16(R2)   ;SET DONE
712 042720  010164  000010                  MOV     R1,RMCS2(R4)    ;SELECT THE DRIVE
713 042724  004037  044666                  JSR     R0,RD.RM        ;TRANSFER ERROR(TRE=1)?
714 042730  000000                          RMCS1
715 042732  042264                          CI7
716 042734  006126                          ROL     (SP)+           ;IS TRE=1 ?
717 042736  100421                          BMI     3$              ;BR IF YES
718 042740  005737  040114                  TST     SAVEFG          ;SAVE THE RH/RM REGISTERS?
719 042744  100002                          BPL     1$              ;BR IF NO
720 042746  004737  045230                  JSR     PC,SVRH70       ;YES--SAVE THE REGISTERS
722 042752  004737  043032          1$:     JSR     PC,WC.HK        ;SEE IF WRITE CHECK TO BE PUT IN QUEUE
723 042756  004737  046112                  JSR     PC,GETREQ       ;GET DPB POINTER
727 042762  005702                          TST     R2              ;ENTRY FOR DRIVE ?
728 042764  001403                          BEQ     2$              ;BR IF NOT
729 042766  004737  041206                  JSR     PC,OPT          ;CALL OPTIMIZER
730 042772  000207                          RTS     PC              ;RETURN
731 042774  012714  000113          2$:     MOV     #113,(R4)       ;RELEASE THE DRIVE
732 043000  000207                          RTS     PC              ;RETURN
733
734 043002  052762  100100  000016  3$:     BIS     #BIT15!BIT06,16(R2)     ;SET DATA ERROR FLAG
735 043010  004737  046016                  JSR     PC,EMPTYQ       ;EMPTY THE 'DRIVE'S WAIT'' QUEUE
736 043014  004737  045230                  JSR     PC,SVRH70       ;SAVE THE RH/RM REGISTERS
737 043020  012714  040111                  MOV     #40111,(R4)     ;ISSUE A 'DRIVE CLEAR''
738 043024  012714  000113                  MOV     #113,(R4)       ;ISSUE A RELEASE TO THE DRIVE
739 043030  000207                          RTS     PC              ;RETURN
740
742                                 ;FORCED WRITE CHECK ROUTINE
743
745 043032  005737  001470          WC.HK:  TST     WRTCHK          ;DO WRITE CHECK ?
746 043036  001427                          BEQ     2$              ;BR IF NOT
747 043040  122762  000161  000002          CMPB    #WRTDAT,$COMND(R2)  ;LAST OPERATION A WRITE ORDER ?
751 043046  001023                          BNE     2$              ;BR IF NOT
752 043050  004037  046036          1$:     JSR     R0,DRVQUE       ;PUT THE OPERATION IN THE QUEUE
```

```
                        BR      2$              ;QUEUE IS FULL
                        CLR     $TATUS(R2)      ;CLEAR 'DONE' BIT IN DPB
                        MOVB    $RMCS1(R2),$PREVO(R2)   ;SAVE WRITE OPERATION CODE
                        MOV     $CYL(R2),$PREVA+2(R2)   ;SAVE CYLINDER
                        MOV     $SEC(R2),$PREVA(R2)     ;SAVE SECTOR AND TRACK ADDRESSES
                        CLRB    $CODE(R2)       ;CHANGE WRITE DATA TO WRITE CHECK
                        MOVB    #WCKD,$COMND(R2)  ;CHANGE FUNCTION CODE TO WRITE CHECK
                2$:     RTS     PC              ;EXIT

                ;SPECIAL CONDITION ROUTINE

                SC:     MOVB    RMAS(R4),R3     ;READ 'RMAS'
                        BNE     2$              ;BR IF ANY 'ATA' BITS SET
                        JSR     R0,RD.RM        ;READ CONTROL AND STATUS REGISTER
                        RMCS1
                        CIB
                        ROLB    (SP)+           ;IS ''IE''=1?
                        BMI     1$              ;YES, NO DRIVES TO CHECK
                        EMT     1               ;REPORT AN ILLEGAL INTERRUPT
                        JSR     PC,SET.IE       ;SET INTERRUPT ENABLE
                1$:     RTS     PC              ;RETURN
                2$:     CLR     -(SP)           ;PROCESS ALL DRIVES THAT HAVE
                        MOVB    R3,(SP)         ;AN ''ATA''=1
                        MOV     #1,R3
                        CLR     R1

                SC3:    BIT     R3,(SP)         ;ATA=1?
                        BNE     SC5             ;YES

                SC4:    INC     R1              ;MOVE TO THE NEXT DRIVE
                        ASLB    R3
                        BNE     SC3             ;BR IF MORE TO CHECK?
                        TST     (SP)+           ;CLEAN OFF THE STACK
                        RTS     PC              ;RETURN TO USER

                SC5:    TSTB    DPINT(R1)       ;INITIALIZING THE DRIVE ?
                        BEQ     1$              ;BR IF NOT
                        JMP     SC13            ;PROCESS THE DRIVE
                1$:     TSTB    DPRQS(R1)       ;PORT REQUEST OUTSTANDING ?
                        BEQ     2$              ;BR IF NOT
                        JMP     SC13            ;START THE OUTSTANDING COMMAND
                2$:     TSTB    DRVSTA(R1)      ;CHECK THE DRIVE STATUS
                        BGT     4$              ;BR IF ONLINE
                        TSTB    ULDFLG(R1)      ;UNLOAD IN PROGRESS?
                        BLE     4$              ;BR IF NOT
                        JSR     PC,GETREQ       ;GET DPB POINTER
                        JSR     PC,SVRH70       ;SAVE THE RH/RM REGISTERS
                        JSR     PC,SC12         ;SAVE RMDS, RMER1, RMER2, AND RMMR2
                                                ;ALSO DO A DRIVE INIT (DRVINT)
                        TSTB    DRVSTA(R1)      ;DID DRIVE COME ONLINE?
                        BLE     5$              ;NO
                        BIT     #BIT14,RMERRS   ;WAS THERE AN ERROR?
                        BNE     3$              ;BR IF ERROR
                3$:     MOV     RMERRS+2,R5     ;YES -- PICKUP RMER1 AND
                        BR      SC6A            ;GO PROCESS THE ERROR
                4$:     TSTB    DRVACT(R1)      ;DRIVE ACTIVE WITH COMMAND OR ERROR RECOVERY ?
                        BNE     SC6             ;BR IF EITHER
```

```
814 043310  004737  044046                  JSR     PC,SC12         ;SAVE RMDS, RMER1, RMER2, AND RMMR2
815                                                                 ;ALSO DO A DRVINT
816 043314  105761  040056          5$:     TSTB    DPINT(R1)       ;TRYING TO INIT THE DRIVE ?
817 043320  001323                          BNE     SC4             ;BR IF YES, CHECK ON MORE DRIVES
818 043322  105761  040036                  TSTB    DRVSTA(R1)      ;CHECK ON DRIVE'S STATUS
819 043326  100412                          BMI     6$              ;BR IF UNSAFE
820 043330  032737  020000  040022          BIT     #BIT13,RMERRS+4 ;ADDRESS PLUG CHANGED ?
821 043336  001011                          BNE     7$              ;BR IF YES
825 043340  012746  000111                  MOV     #111,-(SP)      ;DRIVE CLEAR
826 043344  004037  045042                  JSR     R0,WRT.RM       ;WRITE THE COMMAND INTO RMCS1
827 043350  000000                          RMCS1                   ;REGISTER INDEX
828 043352  043706                          SC8                     ;PARITY EXIT ADDRESS
829 043354  011605                  6$:     MOV     (SP),R5         ;PICKUP (RMAS) BEFORE THE ERROR CALL
830 043356  104002                          EMT     2               ;REPORT THE UNEXPECTED ATTENTION
831 043360  000703                          BR      SC4             ;GO CHECK FOR MORE ATA'S
832 043362                          7$:
    043362  104005                          EMT     5               ;REPORT THE ADDRESS PLUG CHANGE
833 043364  000701                          BR      SC4             ;CHECK FOR MORE DRIVES
834
835 043366  006301                  SC6:    ASL     R1              ;SETUP TO ADDRESS WORDS
836 043370  012761  177777  040120          MOV     #-1,TIMER(R1)   ;STOP THE TIMER
837 043376  006201                          ASR     R1              ;RESTORE THE DRIVE ADDRESS
838 043400  004737  046112                  JSR     PC,GETREQ       ;GET THE DPB POINTER FROM THE QUEUE
839 043404  010164  00001C                  MOV     R1,RMCS2(R4)    ;SELECT DRIVE
840 043410  000137  043736                  JMP     SC11            ;PROCESS THE SEARCH
841 043414  004037  044666                  JSR     R0,RD.RM        ;READ THE RM'S STATUS REG.
842 043420  000012                          RMDS
843 043422  043706                          SC8
844 043424  011605                          MOV     (SP),R5         ;AND PUT IT IN R5
845 043426  006126                          ROL     (SP)+           ;WAS THERE AN ERROR?
846 043430  100407                          BMI     1$              ;BR IF ERROR
847 043432  105761  040026                  TSTB    DRVACT(R1)      ;CHECK DRIVE'S STATE
848 043436  003137                          BGT     SC11            ;BR IF DRIVE ACTIVE WITH ORDER
849 043440  052762  100210  000016          BIS     #BIT15!BIT07!BIT03,16(R2)       ;INFORM USER OF ERROR RECOVER COMPLETION
850 043446  000470                          BR      SC7
851 043450  004037  044666          1$:     JSR     R0,RD.RM        ;READ ERROR REGISTER #1
852 043454  000014                          RMER1
853 043456  043706                          SC8
854 043460  012605                          MOV     (SP)+,R5        ;AND SAVE IT IN R5
855 043462  004737  045230                  JSR     PC,SVRH70       ;SAVE RH/RM REGISTERS
856 043466  012746  000111                  MOV     #111,-(SP)      ;ISSUE A DRIVE CLEAR
857 043472  004037  045042                  JSR     R0,WRT.RM
858 043476  000000                          RMCS1
859 043500  043706                          SC8
860
861 043502  006105                  SC6A:   ROL     R5              ;WAS 'UNSAFE'' CONDITION =1?
862 043504  100406                          BMI     1$              ;BR IF YES
863 043506  005702                          TST     R2              ;ANYTHING IN QUEUE ?
864 043510  001447                          BEQ     SC7             ;BR IF NOT
865 043512  052762  100240  000016          BIS     #BIT15.BIT07.BIT05,16(R2)       ;INFORM USER OF ERROR
866 043520  000443                          BR      SC7
867 043522  004037  044666          1$:     JSR     R0,RD.RM        ;READ DRIVE STATUS REG. #1
868 043526  000012                          RMDS
869 043530  043706                          SC8
870 043532  011605                          MOV     (SP),R5         ;SAVE RMDS IN R5
871 043534  006126                          ROL     (SP)+           ;'ERR'=1?
872 043536  10001                           BPL     2$              ;BR IF NO--UNSAFE CLEARED
```

```
873 043540  112761  177777  040036          MOVB    #-1,DRVSTA(R1)  ;DRIVE IS UNSAFE
874 043546  004737  045230                  JSR     PC,SVRH70       ;SAVE RH/RM REGISTERS
875 043552  052762  110000  000016          BIS     #BIT15.BIT12,16(R2)     ;INFORM USER OF UNSAFE ERROR
876 043560  000423                          BR      SC7
877 043562  032705  010000          2$:     BIT     #BIT12,R5       ;'MOL'' = 1 ?
878 043566  001015                          BNE     3$              ;BR IF YES
879 043570  112761  177777  040026          MOVB    #-1,DRVACT(R1)  ;ACTIVE ERROR RECOVER
880 043576  112761  000001  040036          MOVB    #1,DRVSTA(R1)   ;ONLINE
881 043604  006301                          ASL     R1
882 043606  012761  035230  040120          MOV     #15000.,TIMER(R1)       ;START 15. SECOND TIMER
883 043614  006201                          ASR     R1
884 043616  000137  043170                  JMP     SC4
885 043622  052762  100220  000016  3$:     BIS     #BIT15.BIT07.BIT04,16(R2)       ;INFORM USER OF ERROR
886
887 043630  105061  040026          SC7:    CLRB    DRVACT(R1)      ;DRIVE IS IDLE
891 043634  004737  046134                  JSR     PC,POPQUE       ;REMOVE THE QUEUE
892 043640  105761  040104                  TSTB    ULDFLG(R1)      ;UNLOAD IN PROGRESS OR QUEUE?
893 043644  003002                          BGT     1$              ;BR IF NOT
894 043646  105061  040104                  CLRB    ULDFLG(R1)      ;CLEAR UNLOAD FLAG
895 043652  116164  040142  000016  1$:     MOVB    ATABIT(R1),RMAS(R4)     ;CLEAR ATTENTION BIT
896 043660  105761  040036                  TSTB    DRVSTA(R1)      ;IS THE DRIVE UNSAFE ?
897 043664  100406                          BMI     2$              ;BR IF IT IS
901 043666  012746  000111                  MOV     #111,-(SP)      ;DRIVE CLEAR COMMAND
902 043672  004037  045042                  JSR     R0,WRT.RM       ;WRITE THE COMMAND INTO RPCS1
903 043676  000000                          RMCS1                   ;REGISTER INDEX
904 043700  043706                          SC8                     ;PARITY EXIT ADDRESS
905 043702  000137  043170          2$:     JMP     SC4             ;CHECK FOR MORE DRIVES
906
907 043706  105761  040026          SC8:    TSTB    DRVACT(R1)      ;IS DRIVE IDLE?
908 043712  001405                          BEQ     1$              ;YES
909 043714  004737  046112                  JSR     PC,GETREQ       ;GET DPB POINTER
910 043720  004737  042264                  JSR     PC,CI7          ;PROCESS THE PARITY ERROR
911 043724  000402                          BR      2$              ;CONTINUE
912 043726                          1$:
916 043726  004737  042306                  JSR     PC,CI7B         ;PROCESS THE UNCORRECTABLE PARITY ERROR
917 043732  000137  043170          2$:     JMP     SC4             ;CHECK MORE DRIVES
918
919 043736  105761  040104          SC11:   TSTB    ULDFLG(R1)      ;'UNLOAD IN PROGRESS''?
920 043742  003402                          BLE     1$              ;BR IF NO
921 043744  105061  040104                  CLRB    ULDFLG(R1)      ;CLEAR UNLOAD FLAG
922 043750  105061  040026          1$:     CLRB    DRVACT(R1)      ;SET DRIVE IDLE
923 043754  136137  040142  040100          BITB    ATABIT(R1),SRCHWT       ;DOING A SEARCH OPERATION FOR
924                                                                     ;AN I/O COMMAND?
925 043762  001012                          BNE     2$              ;BR IF YES
926 043764  004737  046134                  JSR     PC,POPQUE       ;REMOVE REQUEST FROM QUEUE
927 043770  052762  000200  000016          BIS     #BIT07,16(R2)   ;SET 'DONE'' BIT
928 043776  005737  040114                  TST     SAVEFG          ;SAVE THE REGISTERS?
929 044002  100002                          BPL     2$              ;BR IF NO
930 044004  004737  045230                  JSR     PC,SVRH70       ;YES--SAVE ALL OF THE RH/RM REG'S
931 044010  116164  040142  000016  2$:     MOVB    ATABIT(R1),RMAS(R4)     ;CLEAR ATTENTION BIT
932 044016  146137  040142  040100          BICB    ATABIT(R1),SRCHWT       ;CLEAR IMPLIED SEEK SET
933 044024  006301                          ASL     R1              ;WORD INDEX
934 044026  012761  177777  040120          MOV     #-1,TIMER(R1)   ;STOP CLOCK
935 044034  006201                          ASR     R1              ;RESTORE R1
936 044036  004737  041206                  JSR     PC,OPT          ;START A REQUEST
937 044042  000137  043170                  JMP     SC4             ;CHECK FOR MORE DRIVES
938
```

```
                                SC12:   MOV     R1,RMCS2(R4)    ;SELECT DRIVE
                                        MOV     RMDS(R4),RMERRS ;SAVE THE FOUR REGISTERS THAT
                                        MOV     RMER1(R4),RMERRS+2      ;WILL TELL US SOMETHING
                                        MOV     RMER2(R4),RMERRS+4
                                        MOV     RMMR2(R4),RMERRS+6
                                        JSR     R0,DRVINT       ;INIT. THE STATE OF THE DRIVE
                                        BR      1$              ;TAKE ERROR EXIT
                                        RTS     PC              ;RETURN
                                1$:     TST     (SP)+           ;POP PC OFF OF THE STACK
                                        BR      SC8             ;PROCESS THE PARITY ERROR

                                SC13:   ASL     R1              ;SETUP TO ADDRESS WORDS
                                        MOV     #-1,TIMER(R1)   ;STOP THE TIMER
                                        ASR     R1              ;
                                        MOV     R1,RMCS2(R4)    ;SELECT THE DRIVE
                                        MOVB    ATABIT(R1),RMAS(R4)     ;CLEAR THE ATTENTION BIT
                                1$:     TSTB    DPINT(R1)       ;INITIALIZING THE DRIVE ?
                                        BEQ     2$              ;BR IF NOT
                                        CLRB    DPINT(R1)       ;CLEAR THE INIT INDICATOR
                                        JSR     R0,DRVINT       ;GO INIT THE DRIVE
                                        NOP                     ;DUMMY PARITY ERROR RETURN
                                        TSTB    DRVSTA(R1)      ;DRIVE ONLINE ?
                                        BGT     2$              ;BR IF YES -- START ORDER
                                        TST     R2              ;QUEUE ENTRY FOR THE DRIVE
                                        BEQ     3$              ;BR IF NOT
                                        JSR     PC,GETREQ       ;GET DPB ADDRESS
                                        BIS     #BIT15!BIT14,16(R2)     ;INFORM USER THAT DRIVE OFFLINE
                                        JSR     PC,SVRH70       ;SAVE THE REGISTERS
                                        JSR     PC,POPQUE       ;REMOVE THE QUEUE
                                        BR      3$
                                2$:     BIT     #BIT11,RMCS1(R4)        ;DVA SET ?
                                        BNE     4$              ;SET THEN CALL OPT
                                        ASL     R1
                                        MOV     #15000.,TIMER(R1)       ;START 15. SECOND TIMER
                                        ASR     R1
                                        BR      3$
                                4$:     JSR     PC,OPT          ;START THE PENDING REQUEST
                                3$:     JMP     SC4             ;PROCESS OTHER DRIVES

                                ;RM TIMER ROUTINE
                                ;CALL
                                ;       MOV     #TIME,-(SP)     ;ELASPED TIME IN MILLISECONDS ON THE STACK
                                ;       JSR     PC,RMTMR        ;CALL RM05 TIME ROUTINE

                                RMTMR:  TST     ACTDRV          ;CHECK "ACTDRV & ACTSTR"
                                        BNE     4$              ;IF NON ZERO EXIT
                                        MOVB    #1,ACTSTR       ;SET "ACTSTR"
                                        SAVREG                  ;SAVE R0 - R5
                                        CLR     R1              ;START WITH DRIVE 0
                                        CLR     R3
                                1$:     TST     TIMER(R3)       ;IS THE TIMER RUNNING?
                                        BLT     2$              ;BR IF NO
                                        SUB     2(SP),TIMER(R3) ;COUNT THE INTERVAL
                                        BGT     2$              ;BR IF NO SOFTWARE TIMEOUT
                                        JSR     PC,STO          ;CALL SOFTWARE TIMEOUT ROUTINE
                                2$:     INC     R1              ;MOVE TO NEXT DRIVE
                                        TST     (R3)+
```

```
 999 044324  022701  000010                CMP     #8.,R1          ;OUT OF DRIVES?
1000 044330  003362                        BGT     1$              ;BR IF NO
1001 044332  104413                3$:     RESREG                  ;RESTORE R0 - R5
1002 044334  105037  040103                CLRB    ACTSTR          ;ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
1003 044340  012616                4$:     MOV     (SP)+,(SP)      ;ADJUST THE STACK
1004 044342  000207                        RTS     PC              ;RETURN
1005
1006                               ;SOFTWARE TIMEOUT ROUTINE
1007                               ;
1008                               ;NOTE: THIS ROUTINE MUST BE ENTERED AT PRIORITY 6
1009                               ;       OR GREATER
1010                               ;
1011                               ;CALL:  STO
1012                               ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER
1013                               ;       JSR     PC,STO          ;CALL
1014                               ;       RETURN
1015
1016 044344  010146                STO:    MOV     R1,-(SP)        ;SAVE R1
1017 044346  010246                        MOV     R2,-(SP)        ;SAVE R2
1018 044350  010346                        MOV     R3,-(SP)        ;SAVE R3
1019 044352  010446                        MOV     R4,-(SP)        ;SAVE R4
1020 044354  013704  040154                MOV     RMADR,R4        ;GET ADDRESS OF ''RMCS1''
1021 044360  010164  000010                MOV     R1,RMCS2(R4)    ;SELECT THE DRIVE
1022 044364  004037  044666                JSR     R0,RD.RM        ;READ ''DRIVE STATUS REG''
1023 044370  000012                        RMDS
1027 044372  044654                        STO9
1028 044374  105726                        TSTB    (SP)+           ;IS ''DRY''=1?
1029 044376  100436                        BMI     STO2            ;BR IF YES
1030 044400  105761  040056        STO1:   TSTB    DPINT(R1)       ;TRYING TO INTIALIZE THE DRIVE ?
1031 044404  001033                        BNE     STO2            ;BR IF YES
1032 044406  105761  040066                TSTB    DPRQS(R1)       ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
1033 044412  001030                        BNE     STO2            ;BR IF YES
1034 044414  013702  040076                MOV     TRNSWT,R2       ;PICKUP TRANSFER WAIT QUEUE
1035 044420  020137  040140                CMP     R1,DTUW         ;TRANSFER UNDERWAY ON THIS DRIVE?
1036 044424  001404                        BEQ     1$              ;BR IF YES
1037 044426  000137  044654                JMP     STO9            ;IF NOT DON'T BOTHER DRIVES
1038 044432  004737  046112                JSR     PC,GETREQ       ;GET DPB ADDRESS
1039 044436  052762  101000  000016  1$:   BIS     #BIT15!BIT09,16(R2)     ;SET THE ERROR FLAGS
1040 044444  004737  045230                JSR     PC,SVRH70       ;SAVE RH/RM REGISTERS
1044 044450  105061  040026                CLRB    DRVACT(R1)      ;DRIVE IS IDLE
1045 044454  105061  040104                CLRB    ULDFLG(R1)      ;CLEAR THE UNLOAD FLAG
1046 044460  005037  040076                CLR     TRNSWT          ;CLEAR DPB ADDRESS
1047 044464  012737  177777  040140        MOV     #-1,DTUW        ;CLEAR THE TRANSFER DRIVE #
1048 044472  000470                        BR      STO9            ;DON'T BOTHER OTHER DRIVES
1049
1050 044474  116405  000016        STO2:   MOVB    RMAS(R4),R5     ;READ ATTENTION REG
1051 044500  136105  040142                BITB    ATABIT(R1),R5   ;IS ATTENTION FOR THIS DRIVE UP ?
1052 044504  001007                        BNE     STO3            ;YES
1053 044506  105761  040056                TSTB    DPINT(R1)       ;TRYING TO INTIALIZE THE DRIVE ?
1054 044512  001021                        BNE     STO6            ;BR IF YES - DRIVE NOT ONLINE
1055 044514  105761  040066                TSTB    DPRQS(R1)       ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
1056 044520  001035                        BNE     STO7            ;BR IF YES - NO RESPONSE TO REQUEST
1057 044522  000454                        BR      STO9            ;OTHER WISE EXIT
1058
1059 044524  105761  040056        STO3:   TSTB    DPINT(R1)       ;INITIALIZING THE DRIVE ?
1060 044530  001003                        BNE     1$              ;BR IF INIT PENDING
1061 044532  105761  040066                TSTB    DPRQS(R1)       ;PORT REQUEST PENDING ?
```

```
1062 044536  001446                          BEQ     ST09            ;BR IF NOT
1063 044540  012763  177777  040120  1$:     MOV     #-1,TIMER(R3)   ;STOP THE TIMER
1064 044546  000442                          BR      ST09            ;EXIT
1065
1066 044550  004737  042364          ST05:   JSR     PC,CI8          ;GO HANDLE THE PARITY ERROR
1067 044554  000437                          BR      ST09
1068
1069 044556  105061  040056          ST06:   CLRB    DPINT(R1)       ;CLEAR THE INITIALIZE INDICATOR
1070 044562  105061  040036                  CLRB    DRVSTA(R1)      ;SET UNIT OFFLINE
1071 044566  012763  177777  040120          MOV     #-1,TIMER(R3)   ;STOP THE TIMER
1072 044574  004737  046112                  JSR     PC,GETREQ       ;GET THE DPB ADDRESS
1073 044600  005702                          TST     R2              ;REQUEST IN QUEUE ?
1074 044602  001424                          BEQ     ST09            ;BR IF NOT
1075 044604  052762  140000  000016          BIS     #BIT15!BIT14,16(R2)     ;INFORM THE USER DRIVE NOT AVAILABLE
1076 044612  000414                          BR      ST08            ;FINISH
1077
1078 044614  012763  177777  040120  ST07:   MOV     #-1,TIMER(R3)   ;STOP THE TIMER
1079 044622  105061  040066                  CLRB    DPRQS(R1)       ;CLEAR PORT REQUEST INDICATOR
1080 044626  004737  046112                  JSR     PC,GETREQ       ;GET DPB ADDRESS
1081 044632  005702                          TST     R2              ;QUEUE ENTRY FOR DRIVE ?
1082 044634  001407                          BEQ     ST09            ;BR IF NONE
1083 044636  012762  100004  000016          MOV     #BIT15!BIT2,16(R2)      ;INFORM USER OF PORT REQUEST ERROR
1084 044644  004737  046016          ST08:   JSR     PC,EMPTYQ       ;CLEAR THE QUEUE FOR THE DRIVE
1085 044650  004737  045230                  JSR     PC,SVRH70       ;SAVE THE REGISTERS
1086 044654  012604                  ST09:   MOV     (SP)+,R4        ;RESTORE R4
1087 044656  012603                          MOV     (SP)+,R3        ;RESTORE R3
1088 044660  012602                          MOV     (SP)+,R2        ;RESTORE R2
1089 044662  012601                          MOV     (SP)+,R1        ;RESTORE R1
1090 044664  000207                          RTS     PC              ;RETURN
1091
1092                                 ;ROUTINE TO READ A RH/RM REGISTER
1093                                 ;
1094                                 ;CALL
1095                                 ;       JSR     R0,RD.RM        ;GO READ A REGISTER
1096                                 ;       INDEX                   ;REG. INDEX FROM BASE
1097                                 ;       ERRADR                  ;ERROR ADDRESS--PROCESS ERROR STARTING
1098                                 ;                               ;AT THIS ADDRESS
1099                                 ;       RETURN                  ;CONTENTS OF REG. IS ON THE STACK
1100                                 ;
1101 044666  013737  040152  045030  RD.RM:   MOV     MCPEMX,RD.RM2   ;MAX. RETRYS ALLOWED
1102 044674  011646                          MOV     (SP),-(SP)      ;SAVE R0 FOR RETURN
1103 044676  013737  040154  044712          MOV     RMADR,RD.ADR    ;FORM THE DESIRED ADDRESS
1104 044704  062037  044712                  ADD     (R0)+,RD.ADR    ;USING THE BASE AND THE INDEX
1105 044710  013727                  RD.RM1: MOV     @(PC)+,(PC)+    ;READ THE DESIRED REGISTER OF THE RM DRIVE
1106 044712  000000                  RD.ADR: .WORD   0               ;ADDRESS IS FORMED HERE
1107 044714  000000                  RD.WRD: .WORD   0               ;REG. CONTENTS PUT HERE
1108 044716  013766  044714  000002          MOV     RD.WRD,2(SP)    ;RETURN IT TO THE USER
1109 044724  013746  040154                  MOV     RMADR,-(SP)     ;PUT THE ADDRESS ON THE STACK
1110 044730  062716  000010                  ADD     #RMCS2,(SP)     ;FORM THE ADDRESS OF RMCS2
1111 044734  032736  010000                  BIT     #BIT12,@(SP)+   ;CHECK THE 'NED' BIT
1112 044740  001035                          BNE     RD.RM3          ;BR IF DRIVE NON-EXISTENT
1113 044742  017746  173206                  MOV     @RMADR,-(SP)    ;READ RMCS1
1114 044746  032716  020000                  BIT     #BIT13,(SP)     ;DID MCPE SET?
1115 044752  001002                          BNE     1$              ;BR IF YES
1116 044754  022620                          CMP     (SP)+,(R0)+     ;ADJUST FOR RETURN
1117 044756  000430                          BR      RD.RM4          ;EXIT
1118 044760                          1$:
```

```
      044760  104003                          EMT     3               ;REPORT 'MCPE' ERROR
1119  044762  005737  040140                  TST     DTUW            ;DATA TRANSFER UNDERWAY?
1120  044766  100405                          BMI     2$              ;NO
1121  044770  032716  040000                  BIT     #BIT14,(SP)     ;'TRE' = 1 ?
1122  044774  001402                          BEQ     2$              ;NO
1123  044776  005726                          TST     (SP)+           ;YES--CLEAN OFF THE STACK AND
1124  045000  000415                          BR      RD.RM3          ;TAKE THE FATAL ERROR EXIT
1125  045002  052716  040000          2$:     BIS     #BIT14,(SP)     ;CLEAR 'MCPE' BY SENDING A '1' TO 'TRE'
1126  045006  000316                          SWAB    (SP)            ;POSITION BEFORE WRITING
1127  045010  013737  040154  045024          MOV     RMADR,3$        ;FORM ADDRESS OF HIGH BYTE
1128  045016  005237  045024                  INC     3$
1129  045022  112637                          MOVB    (SP)+,@(PC)+    ;WRITE THE HIGH BYTE OF RMCS1
1130  045024  000000          3$:     .WORD   0                       ;ADDRESS STORAGE
1131  045026  005327                          DEC     (PC)+           ;EXCEEDED MAX. RETRYS
1132  045030  000003          RD.RM2: .WORD   3
1133  045032  002326                          BGE     RD.RM1          ;BR IF NO
1134  045034  011000          RD.RM3: MOV     (R0),R0                 ;FATAL ERROR EXIT
1135  045036  012616                          MOV     (SP)+,(SP)
1136  045040  000200          RD.RM4: RTS     R0
1137
1138                          ;ROUTINE TO WRITE A REGISTER
1139                          ;
1140                          ;CALL
1141                          ;       MOV     DATA,-(SP)      ;DATA TO BE LOADED ON THE STACK
1142                          ;       JSR     R0,WRT.RM       ;CALL THE ROUTINE TO LOAD(WRITE) THE REG.
1143                          ;       INDEX                   ;INDEX OF THE REGISTER TO BE LOADED
1144                          ;       ERRADR                  ;ADDRESS TO RETURN TO ON AN ERROR
1145                          ;       RETURN                  ;ERROR FREE RETURN
1146
1147  045042  013737  040152  045214  WRT.RM: MOV     MCPEMX,WRT.R2   ;MAX RETRYS ALLOWED
1148  045050  016637  000002  045130          MOV     2(SP),WRT.WD    ;SAVE THE WORD TO WRITE
1149  045056  012616                          MOV     (SP)+,(SP)      ;ADJUST THE STACK
1150  045060  012037  045132                  MOV     (R0)+,WRT.AD    ;GET INDEX OF REGISTER TO BE WRITTEN
1151  045064  001015                          BNE     1$              ;BR IF NOT RMCS1
1152  045066  122737  000150  045130          CMPB    #150,WRT.WD     ;IS THE COMMAND FOR DATA TRANSFERS?
1153  045074  002411                          BLT     1$              ;YES--DON'T GET THE OLD A16 & A17, & PSEL
1154  045076  004037  044666                  JSR     R0,RD.RM        ;NO---COMBINE A16&A17, & PSEL WITH
1155  045102  000000                          RMCS1                   ;THE COMMAND BEFORE SENDING IT TO
1156  045104  045220                          WRT.R3                  ;THE RH/RM
1157  045106  000316                          SWAB    (SP)
1158  045110  042716  177770                  BIC     #^C7,(SP)
1159  045114  112637  045131                  MOVB    (SP)+,WRT.WD+1
1160  045120  063737  040154  045132  1$:     ADD     RMADR,WRT.AD    ;FORM THE ADDRESS OF THE DISK REG.
1161  045126  012737                  WRT.R1: MOV     (PC)+,@(PC)+    ;LOAD THE DESIRED REG.
1162  045130  000000                  WRT.WD: .WORD   0               ;WORD TO WRITE GOES HERE
1163  045132  000000                  WRT.AD: .WORD   0               ;ADDRESS IS FORMED HERE
1164  045134  013746  040154                  MOV     RMADR,-(SP)     ;PUT THE ADDRESS ON THE STACK
1165  045140  062716  000010                  ADD     #RMCS2,(SP)     ;FORM THE ADDRESS OF RMCS2
1166  045144  032736  010000                  BIT     #BIT12,@(SP)+   ;CHECK THE 'NED' BIT
1167  045150  001023                          BNE     WRT.R3          ;BR IF DRIVE NON-EXISTENT
1168  045152  004037  044666                  JSR     R0,RD.RM        ;CHECK FOR PARITY ERROR ON WRITE
1169  045156  000014                          RMER1
1170  045160  045220                          WRT.R3
1171  045162  032726  000010                  BIT     #BIT03,(SP)+
1172  045166  001416                          BEQ     WRT.R4          ;BR IF 'PAR=0'
1173  045170  016037  177776  045202          MOV     -2(R0),1$       ;PICKUP THE INDEX
1174  045176  004037  044666                  JSR     R0,RD.RM        ;READ THE REG.
```

```
1175 045202 000000                  1$:     .WORD   0               ;REG. INDEX
1176 045204 045220                          WRT.R3                  ;RETURN TO THIS ADDRESS ON ERROR
1177 045206 104004                          EMT     4               ;REPORT THE PARITY ON WRITE ERROR
1178 045210 005726                          TST     (SP)+           ;CLEAR OFF THE STACK
1179 045212 005327                          DEC     (PC)+           ;DECREMENT THE ERROR COUNT
1180 045214 000003                  WRT.R2: .WORD   3               ;RETRY COUNTER
1181 045216 002343                          BGE     WRT.R1          ;TRY AGAIN IF NOT FINISHED
1182 045220 011000                  WRT.R3: MOV     (R0),R0         ;TAKE THE 'PARITY ON WRITE'' ERROR EXIT
1183 045222 000401                          BR      WRT.R5          ;EXIT
1184 045224 005720                  WRT.R4: TST     (R0)+           ;ADJUST FOR ERROR FREE EXIT
1185 045226 000200                  WRT.R5: RTS     R0
1186
1187                                ;ROUTINE TO SAVE THE RH/RM REGISTERS AS PER DPB+14
1188                                ;
1189                                ;CALL
1190                                ;       MOV     #DPBNUM,R2      ;DPB POINTER TO R2
1191                                ;       JSR     PC,SVRH70       ;SAVE THE DRIVES REG'S
1192
1193 045230 104412                  SVRH70: SAVREG                  ;SAVE R0 - R5
1194 045232 005702                          TST     R2              ;QUEUE ENTRY FOR THE DRIVE ?
1195 045234 001442                          BEQ     6$              ;BR IF NONE
1196 045236 013704 040154                   MOV     RMADR,R4
1197 045242 111264 000010                   MOVB    (R2),RMCS2(R4)  ;SELECT DRIVE
1198 045246 016203 000014                   MOV     14(R2),R3       ;GET THE ERROR TABLE POINTER
1199 045252 001433                          BEQ     6$              ;EXIT IF NO ADDRESS
1200 045254 005037 045310                   CLR     3$              ;COUNTER & POINTER
1201 045260 023727 045310 000022    1$:     CMP     3$,#RMDB        ;REACHED THE BUFFER REGISTER ?
1202 045266 001006                          BNE     2$              ;BR IF NOT
1203 045270 032764 000200 000010            BIT     #BIT07,RMCS2(R4)        ;'OR' SET ?
1204 045276 001002                          BNE     2$              ;BR IF SET
1205 045300 005023                          CLR     (R3)+           ;STORE RMDB AS ZEROES
1206 045302 000405                          BR      4$              ;CONTINUE
1207 045304 004037 044666           2$:     JSR     R0,RD.RM        ;READ THE SELECTED REGISTER
1208 045310 000000                  3$:     .WORD   0               ;REGISTER INDEX
1209 045312 045336                          5$                      ;ERROR RETURN ADDRESS
1210 045314 012623                          MOV     (SP)+,(R3)+     ;STORE THE REGISTER CONTENTS
1211 045316 023727 045310 000046    4$:     CMP     3$,#RMEC2       ;REACHED THE END ?
1212 045324 001406                          BEQ     6$              ;BR IF YES
1213 045326 062737 000002 045310            ADD     #2,3$           ;INCREMENT THE REGISTER INDEX
1214 045334 000751                          BR      1$              ;CONTINUE READING THE REGISTERS
1215 045336 004737 042264           5$:     JSR     PC,CI7          ;PROCESS THE UNCORRECTABLE PARITY ERROR
1217 045342 013746 001234           6$:     MOV     $CPUOP,-(SP)    ;CHECK THE CPU (RH) TYPE
1218 045346 042716 003777                   BIC     #^C174000,(SP)  ;LEAVE THE CPU TYPE BITS
1219 045352 022726 030000                   CMP     #30000,(SP)+    ;SEE IF RH70
1220 045356 001004                          BNE     7$              ;IF NE, NO
1221 045360 063704 040162                   ADD     RHEXT,R4        ;POINT TO RMBAE
1222 045364 012423                          MOV     (R4)+,(R3)+     ;STORE THE CONTENTS
1223 045366 011413                          MOV     (R4),(R3)       ;GET RMCS3
1224 045370 104413                  7$:     RESREG                  ;RESTORE R0 - R5
1228 045372 000207                          RTS     PC              ;RETURN
1229
1230                                ;ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A ''TRE''
1231                                ;CALL
1232                                ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
1233                                ;       JSR     PC,SET.IE       ;SET ''IE''
1234                                ;       RETURN
1235
```

```
1236 045374  010446                    SET.IE: MOV     R4,-(SP)        ;SAVE R4
1237 045376  013704  040154                    MOV     RMADR,R4        ;PICKUP ADDRESS OF RMCS1
1238 045402  010164  000010                    MOV     R1,RMCS2(R4)    ;SELECT DRIVE
1239 045406  011446                            MOV     (R4),-(SP)      ;READ RMCS1
1240 045410  052716  040000                    BIS     #BIT14,(SP)     ;SET THE ''TRE'' BIT OF THE WORD READ
1241 045414  000316                            SWAB    (SP)            ;ADJUST FOR DATO
1242 045416  112714  000100                    MOVB    #BIT06,(R4)     ;SET ''IE''
1243 045422  032764  010000  000010            BIT     #BIT12,RMCS2(R4)        ;IS 'NED'=1?
1244 045430  001002                            BNE     1$              ;YES--CLEAR ''TRE''
1245 045432  005726                            TST     (SP)+           ;CLEAN OFF THE STACK
1246 045434  000402                            BR      2$
1247 045436  112664  000001            1$:     MOVB    (SP)+,1(R4)     ;CLEAR ''TRE''
1248 045442  012604                    2$:     MOV     (SP)+,R4        ;RESTORE R4
1249 045444  000207                            RTS     PC              ;RETURN TO CALLER
1250
1251                                    ;QUEUE COUNT
1252
1253 045446    000                     QCNT:   .BYTE   0               ;DRIVE 0
1256 045447    000                             .BYTE   0               ;DRIVE 1
     045450    000                             .BYTE   0               ;DRIVE 2
     045451    000                             .BYTE   0               ;DRIVE 3
     045452    000                             .BYTE   0               ;DRIVE 4
     045453    000                             .BYTE   0               ;DRIVE 5
     045454    000                             .BYTE   0               ;DRIVE 6
     045455    000                             .BYTE   0               ;DRIVE 7
1257
1258                                    ;QUEUE INPUT POINTERS
1259
1260 045456  045540                    QINPT:  .WORD   QDRV0           ;DRIVE 0
1263 045460  045560                            .WORD   QDRV1           ;DRIVE 1
     045462  045600                            .WORD   QDRV2           ;DRIVE 2
     045464  045620                            .WORD   QDRV3           ;DRIVE 3
     045466  045640                            .WORD   QDRV4           ;DRIVE 4
     045470  045660                            .WORD   QDRV5           ;DRIVE 5
     045472  045700                            .WORD   QDRV6           ;DRIVE 6
     045474  045720                            .WORD   QDRV7           ;DRIVE 7
1264
1265                                    ;QUEUE OUTPUT POINTERS
1266
1267 045476  045540                    QOUTPT: .WORD   QDRV0           ;DRIVE 0
1270 045500  045560                            .WORD   QDRV1           ;DRIVE 1
     045502  045600                            .WORD   QDRV2           ;DRIVE 2
     045504  045620                            .WORD   QDRV3           ;DRIVE 3
     045506  045640                            .WORD   QDRV4           ;DRIVE 4
     045510  045660                            .WORD   QDRV5           ;DRIVE 5
     045512  045700                            .WORD   QDRV6           ;DRIVE 6
     045514  045720                            .WORD   QDRV7           ;DRIVE 7
1271
1272 045516  045540                    QSTART: .WORD   QDRV0           ;DRIVE 0 START ADDRESS
1273 045520  045560                    QSTOP:  .WORD   QDRV1           ;DRIVE 0 STOP ADDRESS & DRIVE 1 START ADDRESS
1274 045522  045600                            .WORD   QDRV2           ;STOP DRIVE 1--START DRIVE 2
1275 045524  045620                            .WORD   QDRV3           ;STOP DRIVE 2--START DRIVE 3
1276 045526  045640                            .WORD   QDRV4           ;STOP DRIVE 3--START DRIVE 4
1277 045530  045660                            .WORD   QDRV5           ;STOP DRIVE 4--START DRIVE 5
1278 045532  045700                            .WORD   QDRV6           ;STOP DRIVE 5--START DRIVE 6
1279 045534  045720                            .WORD   QDRV7           ;STOP DRIVE 6--START DRIVE 7
1280 045536  045740                            .WORD   QTERM           ;STOP DRIVE 7
```

```
1281
1282                                    ;DRIVE REQUEST QUEUES
1283
1286 045540                             QDRV0:  .BLKW   10
     045560                             QDRV1:  .BLKW   10
     045600                             QDRV2:  .BLKW   10
     045620                             QDRV3:  .BLKW   10
     045640                             QDRV4:  .BLKW   10
     045660                             QDRV5:  .BLKW   10
     045700                             QDRV6:  .BLKW   10
     045720                             QDRV7:  .BLKW   10
1287          045740                    QTERM=.
1288
1289                                    ;ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
1290                                    ;
1291                                    ;CALL
1292                                    ;       JSR     PC,CLRQUE
1293
1294 045740  104412                     CLRQUE: SAVREG                  ;SAVE R0 - R5
1295 045742  012702  045446                     MOV     #QCNT,R2        ;ZERO THE QUEUE COUNTS
1296 045746  005022                             CLR     (R2)+           ;DRIVES 0 & 1
1297 045750  005022                             CLR     (R2)+           ;DRIVES 2 & 3
1298 045752  005022                             CLR     (R2)+           ;DRIVES 4 & 5
1299 045754  005022                             CLR     (R2)+           ;DRIVES 6 & 7
1300 045756  012703  000010                     MOV     #8.,R3          ;MOVE THE STARTING
1301 045762  012701  045516                     MOV     #QSTART,R1      ;ADDRESS OF THE QUEUE INTO
1302 045766  012122                     1$:     MOV     (R1)+,(R2)+     ;THE QUEUE INPUT POINTER
1303 045770  005303                             DEC     R3
1304 045772  001375                             BNE     1$
1305 045774  012703  000010                     MOV     #8.,R3          ;MOVE THE STARTING ADDRESS
1306 046000  012701  045516                     MOV     #QSTART,R1      ;OF THE QUEUE INTO THE
1307 046004  012122                     2$:     MOV     (R1)+,(R2)+     ;QUEUE OUTPUT POINTER
1308 046006  005303                             DEC     R3
1309 046010  001375                             BNE     2$
1310 046012  104413                             RESREG                  ;RESTORE R0 - R5
1311 046014  000207                             RTS     PC
1312
1313                                    ;EMPTY THE QUEUE SPECIFIED BY R1
1314                                    ;
1315                                    ;CALL
1316                                    ;       MOV     DRVNUM,R1       ;DRIVE NUMBER TO R1
1317                                    ;       JSR     PC,EMPTYQ
1318
1319 046016  105061  045446             EMPTYQ: CLRB    QCNT(R1)        ;CLEAR NUMBER OF ITEMS IN QUEUE
1320 046022  006301                             ASL     R1
1321 046024  016161  045456  045476             MOV     QINPT(R1),QOUTPT(R1)    ;SET OUTPUT QUEUE POINTER INPUT POINTER
1322 046032  006201                             ASR     R1
1323 046034  000207                             RTS     PC
1324
1325                                    ;ROUTINE TO PUT A REQUEST IN QUEUE
1326                                    ;
1327                                    ;CALL
1328                                    ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER
1329                                    ;       MOV     #DPB,R2         ;ADDRESS OF PARAMETER BLOCK
1330                                    ;       JSR     R0,DRVQUE       ;GO PUT REQUEST IN QUEUE
1331                                    ;       RETURN1                 ;RETURN HERE IF QUEUE IS FULL
1332                                    ;       RETURN2                 ;RETURN HERE IF REQUEST IS IN QUEUE
```

```
1333
1334 046036  122761  000010  045446  DRVQUE: CMPB    #10,QCNT(R1)      ;IS QUEUE FULL?
1335 046044  001421                          BEQ     2$                ;BR IF YES-TAKE RETURN1
1336 046046  105261  045446                  INCB    QCNT(R1)          ;INCREMENT QUEUE COUNT
1337 046052  006301                          ASL     R1
1338 046054  010271  045456                  MOV     R2,@QINPT(R1)     ;PUT THIS REQUEST IN QUEUE
1339 046060  062761  000002  045456          ADD     #2,QINPT(R1)      ;UPDATE THE QUEUE POINTER
1340 046066  026161  045456  045520          CMP     QINPT(R1),QSTOP(R1)     ;TIME TO RESET THE POINTER
1341 046074  001003                          BNE     1$                ;BR IF NO
1342 046076  016161  045516  045456          MOV     QSTART(R1),QINPT(R1)    ;YES--RESET POINTER
1343 046104  006201                  1$:     ASR     R1
1344 046106  005720                          TST     (R0)+             ;TAKE RETURN 2
1345 046110  000200                  2$:     RTS     R0                ;RETURN TO USER
1346
1347                                 ;ROUTINE TO GET THE "DPB" ADDRESS OF NEXT REQUEST IN QUEUE
1348                                 ;
1349                                 ;CALL
1350                                 ;       MOV     #DRVNUM,R1        ;DRIVE NUMBER TO R1
1351                                 ;       JSR     PC,GETREQ         ;GO GET THE REQUEST
1352                                 ;       RETURN                    ;R2="DPB" ADDRESS OF THE REQUEST
1353                                 ;                                 ;R2=0 IF NO REQUEST IN QUEUE
1354
1355 046112  005002                  GETREQ: CLR     R2
1356 046114  105761  045446                  TSTB    QCNT(R1)          ;IS THERE ANY REQUEST IN QUEUE?
1357 046120  001404                          BEQ     2$                ;NO
1358 046122  006301                  1$:     ASL     R1
1359 046124  017102  045476                  MOV     @QOUTPT(R1),R2    ;PICKUP "DPB" POINTER FOR THIS DRIVE
1360 046130  006201                          ASR     R1
1361 046132  000207                  2$:     RTS     PC                ;RETURN TO USER
1362
1363                                 ;ROUTINE TO "POP" THE REQUEST FROM QUEUE
1364                                 ;
1365                                 ;CALL
1366                                 ;       MOV     #DRVNUM,R1        ;DRIVE NUMBER TO R1
1367                                 ;       JSR     PC,POPQUE         ;CALL TO REMOVE REQUEST
1368                                 ;       RETURN                    ;R2=ADDRESS OF DPB REMOVED
1369
1370 046134  105361  045446          POPQUE: DECB    QCNT(R1)          ;DECREMENT QUEUE COUNT
1371 046140  006301                          ASL     R1
1372 046142  017102  045476                  MOV     @QOUTPT(R1),R2    ;GET THE "DPB" POINTER
1373 046146  005071  045476                  CLR     @QOUTPT(R1)       ;REMOVE DPB ADDRESS FROM THE QUEUE
1374 046152  062761  000002  045476          ADD     #2,QOUTPT(R1)     ;UPDATE THE QUEUE POINTER
1375 046160  026161  045476  045520          CMP     QOUTPT(R1),QSTOP(R1)    ;TIME TO RESET THE POINTER?
1376 046166  001003                          BNE     1$                ;NO--BR TO EXIT
1377 046170  016161  045516  045476          MOV     QSTART(R1),QOUTPT(R1)   ;YES--RESET THE POINTER
1378 046176  006201                  1$:     ASR     R1
1379 046200  000207                          RTS     PC                ;RETURN TO USER
1380
```

```
                        .SBTTL  DATA, CONTROL, & STATUS BLOCKS

                        ;BLOCK LOCATION EQUATE STATEMENTS

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

                        ;DRIVE'S HISTORY AND CURRENT INDICATOR STORAGE EQUATES

000020                  $WRDL   = $FMT+17               ;WORD COUNT LENGTH (POSITIVE)
000022                  $SSEC   = $WRDL+2               ;SECTOR SIZE FOR CURRENT OPERATION (256. OR 258.)
000024                  $CODE   = $WRDL+4               ;PRESENT COMMAND SELECTION CODE
000026                  $PACK   = $WRDL+6               ;WRITE DATA PACK INDICATOR
000027                  $PREVO  = $WRDL+7               ;PREVIOUS COMMAND SELECTION CODE
000030                  $PATTC  = $WRDL+10              ;PATTERN CODE
000032                  $PREVA  = $WRDL+12              ;PREVIOUS ADDRESS- TRK, SEC, CYL (DOUBLE WORD)
000036                  $ENDAT  = $WRDL+16              ;END OF PASS DATA COUNT (DOUBLE WORD)
000042                  $ENDSK  = $WRDL+22              ;END OF PASS SEEK COUNT (DOUBLE WORD)
000046                  $OPERC  = $WRDL+26              ;OPERATION COUNT (DOUBLE WORD) PER PASS
000052                  $POSIT  = $WRDL+32              ;SEEK COUNT (DOUBLE WORD)
000056                  $WTOFL  = $WRDL+36              ;TOTAL WORDS WRITTEN OVERFLOW COUNT
000060                  $WRITN  = $WRDL+40              ;TOTAL WORDS WRITTEN COUNT (DOUBLE WORD)
000064                  $RDOFL  = $WRDL+44              ;TOTAL WORDS READ OVERFLOW COUNT
000066                  $READ   = $WRDL+46              ;TOTAL WORDS READ COUNT (DOUBLE WORD)
000072                  $TOTAL  = $WRDL+52              ;TOTAL ERRORS (ALL TYPES) COUNT
000074                  $SOFT   = $WRDL+54              ;'SOFT' ERROR COUNT
000076                  $HARD   = $WRDL+56              ;'HARD' ERROR COUNT
000100                  $SKI    = $WRDL+60              ;'SKI' ERROR COUNT
000102                  $MISPO  = $WRDL+62              ;PROG DETECTED MIS-POSITIONING ERROR COUNT
000104                  $PASSC  = $WRDL+64              ;PASS COUNTER
000106                  $FAIR   = $WRDL+66              ;OPERATION QUEUE 'FAIRNESS' COUNT
000110                  $HLDWC  = $WRDL+70              ;HOLD WORD FOR 'RELBUF' ROUTINE

                        ;INDEX EQUATES TO THE NEXT OPERATION PARAMETERS

000112                  $NCODE  = $WRDL+72              ;NEXT OPERATION CODE
000113                  $NPATC  = $NCODE+1              ;NEXT PATTERN
000114                  $NSEC   = $NCODE+2              ;NEXT SECTOR
000115                  $NTRK   = $NCODE+3              ;NEXT TRACK
000116                  $NCYL   = $NCODE+4              ;NEXT CYLINDER
000120                  $NEXT   = $NCODE+6              ;PARAMETER SELECTION INDICATOR
000122                  $FIRST  = $NCODE+10             ;FIRST OPERATION INDICATOR

                        ;INDEX EQUATES FOR MAXIMUM/MINIMUM ADDRESSES

000124                  MAXCYL  = $NCODE+12             ;MAXIMUM CYLINDER ADDRESS
000126                  MINCYL  = MAXCYL+2              ;MINIMUM CYLINDER ADDRESS
000130                  MAXTRK  = MAXCYL+4              ;MAXIMUM TRACK ADDRESS
000132                  MINTRK  = MAXCYL+6              ;MINIMUM TRACK ADDRESS
```

```
                        MAXSEC  = MAXCYL+10             ;MAXIMUM SECTOR ADDRESS
                        MINSEC  = MAXCYL+12             ;MINIMUM SECTOR ADDRESS

                        ;PACK SERIAL NUMBER CONTAINED IN DEC144 FILE

                        $PSNL   = MAXCYL+14             ;LSB'S OF SERIAL NUMBER (5 OCTAL DIGITS)
                        $PSNM   - $PSNL+2               ;MSB'S OF SERIAL NUMBER (5 OCTAL DIGITS)

                        ;DEC144 BAD SECTOR ADDRESS STORAGE AREA INDEX EQUATE

                        $BDSEC  - $PSNL+4               ;BAD SECTOR STORAGE TABLE PLUS TERMINATOR

                        ;DRIVE (MBA) SERIAL NUMBER AREA INDEX EQUATE

                        $MBASN  = $BDSEC+<126.*8.>+2    ;DRIVE (MBA) SERIAL NUMBER

                        ;RH/RM REGISTER EQUATES

                        $RMCS1  - $MBASN+10             ;RM REGISTER STORAGE
                        $RMWC   = $RMCS1+2
                        $RMBA   = $RMCS1+4
                        $RMDA   = $RMCS1+6
                        $RMCS2  = $RMCS1+10
                        $RMDS   = $RMCS1+12
                        $RMER1  = $RMCS1+14
                        $RMAS   = $RMCS1+16
                        $RMLA   = $RMCS1+20
                        $RMDB   = $RMCS1+22
                        $RMMR1  = $RMCS1+24
                        $RMDT   = $RMCS1+26
                        $RMSN   = $RMCS1+30
                        $RMOF   = $RMCS1+32
                        $RMDC   = $RMCS1+34
                        $RMHR   = $RMCS1+36
                        $RMMR2  = $RMCS1+40
                        $RMER2  = $RMCS1+42
                        $RMEC1  = $RMCS1+44
                        $RMEC2  = $RMCS1+46
                        $RMBAE  = $RMCS1+50
                        $RMCS3  = $RMCS1+52

                        ;BLOCK FOR DRIVE 0
046202      000   000   DRIVE0: .BYTE   0,0              ;DRIVE NUMBER 0
046204                           .BLKW   5
046216  050340                   .WORD   .+$RMCS1-$REG
046220                           .BLKB   $RMCS3-$REG

                        ;BLOCK FOR DRIVE 1
050414      001   000   DRIVE1: .BYTE   1,0              ;DRIVE NUMBER 1
050416                           .BLKW   5
050430  052552                   .WORD   .+$RMCS1-$REG
050432                           .BLKB   $RMCS3-$REG

                        ;BLOCK FOR DRIVE 2
052626      002   000   DRIVE2: .BYTE   2,0              ;DRIVE NUMBER 2
052630                           .BLKW   5
052642  054764                   .WORD   .+$RMCS1-$REG
```

```
                                        .BLKB   $RMCS3-$REG

                                ;BLOCK FOR DRIVE 3
055040      003     000         DRIVE3: .BYTE   3,0             ;DRIVE NUMBER 3
055042                                  .BLKW   5
055054  057176                          .WORD   .+$RMCS1-$REG
055056                                  .BLKB   $RMCS3-$REG

                                ;BLOCK FOR DRIVE 4
057252      004     000         DRIVE4: .BYTE   4,0             ;DRIVE NUMBER 4
057254                                  .BLKW   5
057266  061410                          .WORD   .+$RMCS1-$REG
057270                                  .BLKB   $RMCS3-$REG

                                ;BLOCK FOR DRIVE 5
061464      005     000         DRIVE5: .BYTE   5,0             ;DRIVE NUMBER 5
061466                                  .BLKW   5
061500  063622                          .WORD   .+$RMCS1-$REG
061502                                  .BLKB   $RMCS3-$REG

                                ;BLOCK FOR DRIVE 6
063676      006     000         DRIVE6: .BYTE   6,0             ;DRIVE NUMBER 6
063700                                  .BLKW   5
063712  066034                          .WORD   .+$RMCS1-$REG
063714                                  .BLKB   $RMCS3-$REG

                                ;BLOCK FOR DRIVE 7
066110      007     000         DRIVE7: .BYTE   7,0             ;DRIVE NUMBER 7
066112                                  .BLKW   5
066124  070246                          .WORD   .+$RMCS1-$REG
066126                                  .BLKB   $RMCS3-$REG

113
114                             ;GENERAL PURPOSE PARAMETER BLOCK
115
116 070322      000             GENDPB: .BYTE   0               ;DRIVER PARAMETER BLOCK, DRIVE #
117 070323      000                     .BYTE   0               ;OFFSET VALUE OR FMT16, HCI OR ECI
118 070324      000                     .BYTE   0               ;COMMAND CODE
119 070325      000                     .BYTE   0               ;PSEL, A16 AND A17
120 070326  177776                      .WORD   -2              ;WORD COUNT (NEG)
121 070330  101066                      .WORD   CYLNDR          ;BUFFER ADDRESS
122 070332      000                     .BYTE   0               ;SECTOR ADDRESS
123 070333      000                     .BYTE   0               ;TRACK ADDRESS
124 070334  000000                      .WORD   0               ;CYLINDER ADDRESS
125 070336  070342                      .WORD   GENREG          ;ADDRESS TO SAVE ALL RH/RM REG'S
126 070340  000000                      .WORD   0               ;STATUS WORD
127
128 070342                      GENREG: .BLKW   24              ;REGISTER STORAGE
```

```
                        .SBTTL  ERROR MESSAGES

 070412  122  110  040  EM1:    .ASCIZ  /RH CONTROLLER INTERRUPT OCCURRED (RMAS=0)/
 070464  125  116  105  EM2:    .ASCIZ  /UNEXPECTED ATTENTION OCCURRED/
 070522  115  101  123  EM3:    .ASCIZ  /MASSBUS PARITY ERROR (MCPE=1)/
 070560  115  101  123  EM4:    .ASCIZ  /MASSBUS PARITY ERROR (PAR=1)/
 070615  101  104  104  EM5:    .ASCIZ  /ADDRESS PLUG CHANGE BIT SET/
 070651  122  110  040  EM6:    .ASCIZ  /RH CONTROLLER DIDN'T RESPOND TO ADDRESSING/
 070724  125  116  103  EM10:   .ASCIZ  /UNCORRECTABLE MASSBUS PARITY ERROR/
 070767  106  101  124  EM11:   .ASCIZ  /FATAL MASSBUS PARITY ERROR/
 071022  120  105  122  EM12:   .ASCIZ  /PERSISTENT DEVICE UNSAFE/
 071053  117  120  105  EM13:   .ASCIZ  /OPERATION NOT COMPLETED WITHIN TIME LIMIT/
 071125  104  122  111  EM14:   .ASCIZ  /DRIVE WENT OFFLINE/
 071150  116  117  040  EM15:   .ASCIZ  /NO RESPONSE TO PORT REQUEST/
 071204  110  105  101  EM20:   .ASCIZ  /HEADER CRC ERROR/
 071225  104  101  124  EM21:   .ASCIZ  /DATA CHECK ('DCK') ERROR/
 071256  127  122  111  EM22:   .ASCIZ  /WRITE CHECK ERROR - DATA CHECK ('DCK') SET/
 071331  127  122  111  EM23:   .ASCIZ  /WRITE CHECK ERROR - DATA CHECK ('DCK') NOT SET/
 071410  110  105  101  EM24:   .ASCIZ  /HEADER READ ERROR - 'FMT' BIT DROPPED/
 071456  110  105  101  EM25:   .ASCIZ  /HEADER READ ERROR - HEADER COMPARE ('HCE') ERROR/
 071537  106  117  122  EM26:   .ASCIZ  /FORMAT ERROR ('FER')/
 071564  110  105  101  EM27:   .ASCIZ  /HEADER COMPARE ('HCE') ERROR/
 071621  115  111  123  EM30:   .ASCIZ  /MISCELLANEOUS DRIVE ERROR/
 071653  117  120  105  EM31:   .ASCIZ  /OPERATION INCOMPLETE ('OPI') ERROR/
 071716  104  122  111  EM32:   .ASCIZ  /DRIVE TIMING ('DTE') ERROR/
 071751  120  101  122  EM33:   .ASCIZ  /PARITY ('PAR') ERROR AFTER OPERATION STARTED/
 072026  127  122  111  EM34:   .ASCIZ  /WRITE CLOCK FAILURE ('WCF') ERROR/
 072070  111  116  126  EM35:   .ASCIZ  /INVALID ADDRESS ('IAE') ERROR/
 072126  127  122  111  EM36:   .ASCIZ  /WRITE LOCK ('WLE') ERROR/
 072157  104  101  124  EM37:   .ASCIZ  /DATA CHECK ('DCK') SET DURING WRITE CHECK COMMAND/
 072241  122  110  040  EM40:   .ASCIZ  /RH CONTROLLER OR UNIBUS TRANSFER ERROR/
 072310  102  125  123  EM41:   .ASCIZ  /BUS ADDRESS OR WORD COUNT INCORRECT/
 072354  104  101  124  EM42:   .ASCIZ  /DATA COMPARE ERRORS - NO OTHER ERROR(S) DETECTED/
 072435  103  101  116  EM43:   .ASCIZ  /CAN'T MATCH DATA READ WITH A PATTERN/
 072502  105  122  122  EM44:   .ASCIZ  /ERROR BIT(S) SET, BUT NO ERROR SIGNALED BY THE RH CONTROLLER/
 072577  105  103  103  EM45:   .ASCIZ  /ECC LOGIC FAILURE - POSITION REGISTER VALUE NOT VALID/
 072665  102  125  123  EM46:   .ASCIZ  /BUS ADDRESS AND WORD COUNT NOT CONSISTENT/
 072737  123  105  105  EM50:   .ASCIZ  /SEEK INCOMPLETE ('SKI') ERROR/
 072775  120  122  117  EM51:   .ASCIZ  /PROGRAM DETECTED POSITIONING ERROR/
 073040  104  122  111  EM60:   .ASCIZ  /DRIVE UNSAFE ERROR/
```

ERROR MESSAGES

```
      1 073063  122  115  101  DH1:   .ASCIZ  /RMAS/
      2 073070  104  122  111  DH2:   .ASCIZ  /DRIVE   RMDS    RMER1   RMER2  RMMR2   RMAS/
      3 073144  104  122  111  DH3:   .ASCIZ  /DRIVE   REG ADR  DATA/
      4 073172  104  122  111  DH4:   .ASCIZ  /DRIVE   REG ADR    GOOD    BAD/
      5 073231  044  122  115  DH6:   .ASCIZ  /$RMADR/
      6 073240  104  122  126  DH14:  .ASCII  /DRV RMCS1   RMCS2   RMDS    RMER1   /
      7 073304  122  115  115         .ASCIZ  /RMMR2  RMER2   RMEC1   RMEC2/<CRLF>
      8 073342  122  115  127  DH15:  .ASCII  /RMWC    RMBA    RMDA    RMAS    RMLA    /
      9 073412  122  115  104         .ASCIZ  /RMDB    RMMR1   RMDT/<CRLF>
     10 073440  122  115  123  DH16:  .ASCIZ  /RMSN    RMOF    RMDC    RMHR    STATUS/<CRLF>
     12 073510  122  115  102  DH17:  .ASCIZ  /RMBAE   RMCS3/<CRLF>
     14                      .EVEN
```

SEQ 0205

```
 1 073530  001322  000000          DT1:    .WORD   ATTN,0
 2 073534  001220  040016  040020  DT2:    .WORD   DRIVE,RMERRS,RMERRS+2,RMERRS+4,RMERRS+6,ATTN,0
 3 073552  001220  044712  044714  DT3:    .WORD   DRIVE,RD.ADR,RD.WRD,0
 4 073562  001220  045132  045130  DT4:    .WORD   DRIVE,WRT.AD,WRT.WD,RD.WRD,0
 5 073574  001272  000000          DT6:    .WORD   $RMADR,0
 6 073600  002136  002146  002150  DT14:   .WORD   $RMCS1,$RMCS2,$RMDS,$RMER1,$RMMR2,$RMER2,$RMEC1,$RMEC2,0
 7 073622  002140  002142  002144  DT15:   .WORD   $RMWC,$RMBA,$RMDA,$RMAS,$RMLA,$RMDB,$RMMR1,$RMDT,0
 8 073644  002166  002170  002172  DT16:   .WORD   $RMSN,$RMOF,$RMDC,$RMHR,$STATUS,0
10 073660  002206  002210  000000  DT17:   .WORD   $RMBAE,$RMCS3,0
```

```
 1 073666  120  122  123  LIN2C:  .ASCIZ  /PRSNT COMMAND- /
 2 073706  040  040  120  LIN2P:  .ASCIZ  /  PREV COMMAND= /
 3 073727  052  040  105  LIN2S:  .ASCIZ  @* ERROR AT BAD TRACK/SECTOR@
 4 073763  105  122  122  LINM3:  .ASCIZ  /ERROR AT C/
 5 073776  040  124  000  T:      .ASCIZ  / T/
 6 074001  120  122  123  LINN3.  .ASCIZ  /PRSNT ADDR= C/
 7 074017  040  123  000  S:      .ASCIZ  / S/
 8 074022  040  040  120  LINP3:  .ASCIZ  /  PREV ADDR= C/
 9 074041  123  124  101  LINS3:  .ASCIZ  /START CYL= /
10 074055  040  040  105  LINEN3: .ASCIZ  /  END CYL= /
11 074071  040  040  101  LINA3:  .ASCIZ  /  ACTUAL CYL= /
12 074110  040  040  124  LINT3:  .ASCIZ  /  TRK= /
13 074120  040  040  122  LINCA3: .ASCIZ  /  RMDC= /
14 074131  122  115  104  LINDA3: .ASCIZ  /RMDA= /
15 074140  122  115  102  LINB3:  .ASCIZ  /RMBA= /
16 074147  040  040  122  LINW3:  .ASCIZ  /  RMWC= /
17 074160  123  124  101  LINST3: .ASCIZ  /START TRK= /
18 074174  123  124  101  LINSS3: .ASCIZ  /START SEC= /
19 074210  102  125  106  LINM4:  .ASCIZ  /BUFFER ADDR= /
20 074226  040  040  127  LINS4:  .ASCIZ  /  WRD CNT= /
21 074242  040  040  101  LINX4:  .ASCIZ  /  ACTUAL NMBR WRDS XFRD= /
22 074274  107  117  117  LIND5:  .ASCIZ  /GOOD DATA= /
23 074310  040  040  102  LINB5:  .ASCIZ  /  BAD DATA= /
24 074325  040  040  127  LINP5:  .ASCIZ  /  WORD POS= /
25 074342  110  105  101  LINS5:  .ASCIZ  /HEADER FROM ERROR SECTOR= /
26 074375  122  115  105  LINEP5: .ASCIZ  /RMEC1= /
27 074405  040  040  122  LINEO5: .ASCIZ  /  RMEC2= /
28 074417  123  105  103  LINB6:  .ASCIZ  /SECTOR IS ECC CORRECTABLE /
29 074452  123  105  103  LINC6:  .ASCIZ  /SECTOR READ CORRECTLY /
30 074501  103  117  122  LING6:  .ASCIZ  /CORRECTED ON /
31 074517  040  040  122  LINR6:  .ASCIZ  /  RETRIES/
32 074531  125  116  103  LINU06: .ASCIZ  /UNCORRECTABLE AFTER /
33 074556  040  040  124  LIN7M:  .ASCIZ  /  TOTAL MISPOS ERR= /
37 074603  124  117  124  LIN7P:  .ASCIZ  /TOTAL SEEKS= /
38 074621  040  040  124  LIN7S:  .ASCIZ  /  TOTAL SKI ERR= /
39 074643  124  117  124  LIN7T:  .ASCIZ  /TOTAL ERRORS:/
40 074661  040  040  127  LIN7OX: .ASCIZ  /  WOFL:/
41 074671  040  127   22  LIN7X:  .ASCIZ  / WRDS WRITN:/
42 074706  040  040  122  LIN7OR: .ASCIZ  /  ROFL:/
43 074716  040  127  122  LIN7R:  .ASCIZ  / WRDS READ:/
44 074732  105  122  122  LIN8M:  .ASCIZ  /ERROR DURING RETRY/
45 074755  104  101  124  LIN9B:  .ASCIZ  /DATA COMPARISON ERRORS/
46 075004  040  040  040  LIN9H:  .ASCII  /          GOOD    BAD/<CRLF>
47 075031  114  117  103          .ASCIZ  /LOC       DATA    DATA/<CRLF>
48 075060  114  117  103  LIN9I:  .ASCIZ  /LOC       DATA/<CRLF>
49 075077  124  117  124  LIN9E:  .ASCIZ  /TOTAL COMPARE ERRORS= /
50 075126  124  110  105  LIN9G:  .ASCIZ  /THE DATA COMPARED OK/<CRLF>
51 075154  105  122  122  LIN10A: .ASCIZ  /ERROR BURST BEGINS AT WORD /
52 075210  040  111  116  LIN10B: .ASCIZ  / IN DATA FIELD OF ERROR SECTOR/<CRLF>
53 075250  105  122  122  LIN10C: .ASCII  /ERROR WAS NOT IN THE DATA READ - /<CRLF>
54 075312  105  103  103          .ASCIZ  /ECC CORRECTION CAN'T BE PERFORMED/
55 075354  105  103  103  LIN10H: .ASCII  /ECC CORRECTION RESULTS/<CRLF>
56 075403  101  104  104          .ASCIZ  /ADDR    BAD     CORRECTED /<CRLF>
57 075437  103  117  116  LIN11H: .ASCIZ  /CONTENTS OF ERROR SECTOR (REPORTED ABOVE)/<CRLF>
58 075512  101  104  104  LIN11:  .ASCIZ  /ADDR      HEADER/<CRLF>
59 075533  101  104  104  LIN11A: .ASCIZ  /ADDR      DATA/<CRLF>
60 075552  040            BLNKS4: .ASCII  / /
```

```
61 075553  040                  BLNKS3: .ASCII  / /
62 075554  040                  BLNKS2: .ASCII  / /
63 075555  040  000              BLNKS1: .ASCIZ  / /
64 075557  122  105  124  LINX5: .ASCIZ  /RETRIEVED BY A RDHD COMMAND/
```

```
                                        .SBTTL  TELETYPE MESSAGES
                                
075613  077     000             QUES:   .ASCIZ  /?/
075615  075     000             EQUAL:  .ASCIZ  /=/
075617  054     000             COMMA:  .ASCIZ  /,/
075621  055     000             DASH:   .ASCIZ  /-/
075623  011     000             TAB:    .ASCIZ  <11>
075625  104     122     111     UNTMSG: .ASCIZ  /DRIVE/
075633  040     117     106     UNTOFF: .ASCIZ  / OFFLINE/
075644  040     117     116     UNTON:  .ASCIZ  / ONLINE/
075654  040     116     117     UNTNOT: .ASCIZ  / NOT BEING TESTED/
075676  040     101     114     UNTASN: .ASCIZ  / ALREADY BEING TESTED/
075724  040     116     117     NOTRM:  .ASCIZ  @ NOT AN RM05/3/2@
075745  040     116     117     NOTPRS: .ASCIZ  / NOT PRESENT/
075762  040     116     117     NOTAVL: .ASCIZ  / NOT AVAILABLE/
076001  040     125     116     NOTSAF: .ASCIZ  / UNSAFE/
076011  040     114     117     LODEV:  .ASCIZ  / LOAD DEVICE/
076026  200     125     116     SYSTAT: .ASCIZ  <CRLF>/UNIT STATUS:/
076044  122     115     060     $RM02:  .ASCIZ  /RM02/
076051  122     115     060     $RM03:  .ASCIZ  /RM03/
076056  122     115     060     $RM05:  .ASCIZ  /RM05/
076063  200     012     052     REPHD:  .ASCIZ  <CRLF><LF>/***** PERFORMANCE REPORT *****/
076124  052     052     052     STAR30: .ASCII  /**************************/
076155  052     052     052     STAR5:  .ASCIZ  /*****/<CRLF>
076164  200     104     122     SUMHD:  .ASCIZ  <CRLF>/DRIVE SUMMARY,  (OFLOW= 2,147,483,647.)/<CRLF>
076236  007     077     106     DROPNG: .ASCIZ  <07>/?FATAL OR EXCESSIVE ERRORS/
076272  200     105     116     ENDPAS: .ASCIZ  <CRLF>/END OF PASS #/
076311  040     117     116     MSGON:  .ASCIZ  / ON /
076316  200     105     116     ENDTST: .ASCIZ  <CRLF>/END OF TEST /
076334  106     117     122     MSGFOR: .ASCIZ  /FOR /
076341  200     052     052     DEASSG: .ASCIZ  <CRLF>/***** DRIVE DEASSIGNED/<CRLF>
076372  200     052     052     DRNUM:  .ASCIZ  <CRLF>/***** DRIVE #/
076411  040     123     124     ASGND:  .ASCIZ  / STARTED/
076422  200     007     077     NEDCLK: .ASCIZ  <CRLF><07>/? 'L' OR 'P' CLOCK REQUIRED ON SYSTEM/<CRLF>
076473  116     000             N:      .ASCIZ  /N/
076475  131     000             Y:      .ASCIZ  /Y/
076477  056     000             PERIOD: .ASCIZ  /./
076501  040     077     103     MSWRO:  .ASCIZ  / ?CAN'T WRITE IN 'READ ONLY' MODE/<CRLF>
076544  040     077     111     INVLD:  .ASCIZ  / ?INVALID COMMAND/<CRLF>
076567  200     105     116     ENTCOM: .ASCIZ  <CRLF>/ENTER COMMAND: /
076610  200     105     116     ENTLMT: .ASCIZ  <CRLF>/ENTER ADDRESS LIMITS:/<CRLF>
076640  200     105     116     ENTADR: .ASCIZ  <CRLF>/ENTER BAD SECTR ADRS:/<CRLF>
076670  072     000             COLON:  .ASCIZ  /:/
076672  116     117     116     NONE:   .ASCIZ  /NONE/
076677  040     077     111     BADENT: .ASCIZ  / ?INVALID ENTRY/<CRLF>
076720  200     106     101     MERR1:  .ASCIZ  <CRLF>/FAILED TO RETRIEVE BAD SECTOR FILE(DEC144) ON /
077000  200     111     116     MERR2:  .ASCIZ  <CRLF>/INVALID FILE(DEC144) STRUCTURE ON /
077044  052     040     102     MSFULL: .ASCIZ  /* BAD SECTOR TABLE IS FULL */<CRLF>
077102  103     131     114     MSGCTS: .ASCIZ  /CYL,TRK,SEC= /
077120  200     012     116     NODRVS: .ASCIZ  <CRLF><LF>/NO DRIVES ASSIGNED/<CRLF>
077146  052     040     116     NOENTY: .ASCIZ  /* NO ENTRIES */<CRLF>
077166  200     104     105     LSTHDR: .ASCIZ  <CRLF>/DEC144 AND MANUAL BAD SECTOR LIST/<CRLF>
077232  052     040     101     ALOST:  .ASCIZ  /* ALL CURRENT ENTRIES LOST */<CRLF>
077270  120     101     103     PACKSN: .ASCIZ  @PACK S/N: @
077303  115     102     101     MBASN:  .ASCIZ  @MBA S/N: @
077315  200     103     110     MSPRM:  .ASCIZ  <CRLF>/CHANGE DRIVE PARAMETERS (L) N ? /
                                        .EVEN
```

```
                                        ;PARAMETER ENTRY TABLE
                                        
077360  077524  020000  001450  PARLST: .WORD   PAR1,8192.,WRDCNT
077366  077534  077777  001452          .WORD   PAR2,32767.,INTRVL
077374  077704  077777  001456          .WORD   PAR19,32767.,PASSES
077402  077544  000017  001460          .WORD   PAR3,15.,PATTERN
077410  077644  000001  001462          .WORD   PAR11,1,RANDWC
077416  077654  000007  001464          .WORD   PAR14,7,RATIO
077424  077674  000001  001466          .WORD   PAR16,1,ENDING
077432  077664  000001  001470          .WORD   PAR15,1,WRTCHK
077440  077714  000001  001472          .WORD   PAR20,1,MESSAGE
077446  077724  000001  001474          .WORD   PAR21,1,RANDOM
077454  077634  000001  001476          .WORD   PAR10,1,BADBLK
077462  000000                          .WORD   0               ;TABLE TERMINATOR

077464     040     057     040  SLASH:  .ASCIZ  @ / @
077470     200     103     110  ASKPAR: .ASCIZ  <CRLF>/CHANGE PARAMETERS (L) N ? /
077524     127     122     104  PAR1:   .ASCIZ  /WRDCNT /
077534     111     116     124  PAR2:   .ASCIZ  /INTRVL /
077544     120     101     124  PAR3:   .ASCIZ  /PATTERN/
077554     115     101     130  PAR4:   .ASCIZ  /MAXCYL /
077564     115     111     116  PAR5:   .ASCIZ  /MINCYL /
077574     115     101     130  PAR6:   .ASCIZ  /MAXTRK /
077604     115     111     116  PAR7:   .ASCIZ  /MINTRK /
077614     115     101     130  PAR8:   .ASCIZ  /MAXSEC /
077624     115     111     116  PAR9:   .ASCIZ  /MINSEC /
077634     102     101     104  PAR10:  .ASCIZ  /BADBLK /
077644     122     101     116  PAR11:  .ASCIZ  /RANDWC /
077654     122     101     124  PAR14:  .ASCIZ  /RATIO  /
077664     127     122     124  PAR15:  .ASCIZ  /WRTCHK /
077674     105     116     104  PAR16:  .ASCIZ  /ENDING /
077704     120     101     123  PAR19:  .ASCIZ  /PASSES /
077714     115     105     123  PAR20:  .ASCIZ  /MESSAGE/
077724     122     101     116  PAR21:  .ASCIZ  /RANDOM /

                                        .EVEN

                                        ;PARAMETER TABLE POINTERS FOR ADDRESS LIMITS
077734  077754                  TABLE:  .WORD   TABLE0          ;PARAMETER TABLE FOR DRIVE 0
077736  100022                          .WORD   TABLE1          ;PARAMETER TABLE FOR DRIVE 1
077740  100070                          .WORD   TABLE2          ;PARAMETER TABLE FOR DRIVE 2
077742  100136                          .WORD   TABLE3          ;PARAMETER TABLE FOR DRIVE 3
077744  100204                          .WORD   TABLE4          ;PARAMETER TABLE FOR DRIVE 4
077746  100252                          .WORD   TABLE5          ;PARAMETER TABLE FOR DRIVE 5
077750  100320                          .WORD   TABLE6          ;PARAMETER TABLE FOR DRIVE 6
077752  100366                          .WORD   TABLE7          ;PARAMETER TABLE FOR DRIVE 7

                                        ;PARAMETER TABLE FOR ADDRESS LIMITS
077754  077564  000000  046330  TABLE0: .WORD   PAR5,0,MINCYL+DRIVE0
077762  077554  000000  046326          .WORD   PAR4,0,MAXCYL+DRIVE0
077770  077604  000000  046334          .WORD   PAR7,0,MINTRK+DRIVE0
077776  077574  000000  046332          .WORD   PAR6,0,MAXTRK+DRIVE0
100004  077624  000036  046340          .WORD   PAR9,30.,MINSEC+DRIVE0
100012  077614  000036  046336          .WORD   PAR8,30.,MAXSEC+DRIVE0
100020  000000                          .WORD   0               ;TERMINATOR

100022  077564  000000  050542  TABLE1: .WORD   PAR5,0,MINCYL+DRIVE1
```

```
100030  077554  000000  050540            .WORD   PAR4,0,MAXCYL+DRIVE1
100036  077604  000000  050546            .WORD   PAR7,0,MINTRK+DRIVE1
100044  077574  000000  050544            .WORD   PAR6,0,MAXTRK+DRIVE1
100052  077624  000036  050552            .WORD   PAR9,30.,MINSEC+DRIVE1
100060  077614  000036  050550            .WORD   PAR8,30.,MAXSEC+DRIVE1
100066  000000                            .WORD   0                       ;TERMINATOR

100070  077564  000000  052754  TABLE2:   .WORD   PAR5,0,MINCYL+DRIVE2
100076  077554  000000  052752            .WORD   PAR4,0,MAXCYL+DRIVE2
100104  077604  000000  052760            .WORD   PAR7,0,MINTRK+DRIVE2
100112  077574  000000  052756            .WORD   PAR6,0,MAXTRK+DRIVE2
100120  077624  000036  052764            .WORD   PAR9,30.,MINSEC+DRIVE2
100126  077614  000036  052762            .WORD   PAR8,30.,MAXSEC+DRIVE2
100134  000000                            .WORD   0                       ;TERMINATOR

100136  077564  000000  055166  TABLE3:   .WORD   PAR5,0,MINCYL+DRIVE3
100144  077554  000000  055164            .WORD   PAR4,0,MAXCYL+DRIVE3
100152  077604  000000  055172            .WORD   PAR7,0,MINTRK+DRIVE3
100160  077574  000000  055170            .WORD   PAR6,0,MAXTRK+DRIVE3
100166  077624  000036  055176            .WORD   PAR9,30.,MINSEC+DRIVE3
100174  077614  000036  055174            .WORD   PAR8,30.,MAXSEC+DRIVE3
100202  000000                            .WORD   0                       ;TERMINATOR

100204  077564  000000  057400  TABLE4:   .WORD   PAR5,0,MINCYL+DRIVE4
100212  077554  000000  057376            .WORD   PAR4,0,MAXCYL+DRIVE4
100220  077604  000000  057404            .WORD   PAR7,0,MINTRK+DRIVE4
100226  077574  000000  057402            .WORD   PAR6,0,MAXTRK+DRIVE4
100234  077624  000036  057410            .WORD   PAR9,30.,MINSEC+DRIVE4
100242  077614  000036  057406            .WORD   PAR8,30.,MAXSEC+DRIVE4
100250  000000                            .WORD   0                       ;TERMINATOR

100252  077564  000000  061612  TABLE5:   .WORD   PAR5,0,MINCYL+DRIVE5
100260  077554  000000  061610            .WORD   PAR4,0,MAXCYL+DRIVE5
100266  077604  000000  061616            .WORD   PAR7,0,MINTRK+DRIVE5
100274  077574  000000  061614            .WORD   PAR6,0,MAXTRK+DRIVE5
100302  077624  000036  061622            .WORD   PAR9,30.,MINSEC+DRIVE5
100310  077614  000036  061620            .WORD   PAR8,30.,MAXSEC+DRIVE5
100316  000000                            .WORD   0                       ;TERMINATOR

100320  077564  000000  064024  TABLE6:   .WORD   PAR5,0,MINCYL+DRIVE6
100326  077554  000000  064022            .WORD   PAR4,0,MAXCYL+DRIVE6
100334  077604  000000  064030            .WORD   PAR7,0,MINTRK+DRIVE6
100342  077574  000000  064026            .WORD   PAR6,0,MAXTRK+DRIVE6
100350  077624  000036  064034            .WORD   PAR9,30.,MINSEC+DRIVE6
100356  077614  000036  064032            .WORD   PAR8,30.,MAXSEC+DRIVE6
100364  000000                            .WORD   0                       ;TERMINATOR

100366  077564  000000  066236  TABLE7:   .WORD   PAR5,0,MINCYL+DRIVE7
100374  077554  000000  066234            .WORD   PAR4,0,MAXCYL+DRIVE7
100402  077604  000000  066242            .WORD   PAR7,0,MINTRK+DRIVE7
100410  077574  000000  066240            .WORD   PAR6,0,MAXTRK+DRIVE7
100416  077624  000036  066246            .WORD   PAR9,30.,MINSEC+DRIVE7
100424  077614  000036  066244            .WORD   PAR8,30.,MAXSEC+DRIVE7
100432  000000                            .WORD   0                       ;TERMINATOR
```

```
                        .SBTTL  ROUTINE TO SIZE MEMORY

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

100434  010046          $SIZE:  MOV     R0,-(SP)        ;;SAVE R0 ON THE STACK
100436  010146                  MOV     R1,-(SP)        ;;SAVE R1 ON THE STACK
100440  013746  000114          MOV     @#114,-(SP)     ;;SAVE MEMORY ERROR VECTOR PS & PC
100444  013746  000116          MOV     @#116,-(SP)
100450  012737  000116  000114  MOV     #116,@#114      ;;IGNORE PARITY ERRORS WHILE SIZING
100456  012737  000002  000116  MOV     #RTI,@#116
100464  013746  000004          MOV     @#ERRVEC,-(SP)  ;;SAVE PRESENT ERROR VECTOR PS & PC
100470  013746  000006          MOV     @#ERRVEC+2,-(SP)
100474  010600                  MOV     SP,R0           ;;SAVE THE STACK POINTER
                        ;;SET THE ERRVEC PS TO THE PRESENT PS
100476  104400                  TRAP                    ;;PUSH OLD PSW AND PC ON STACK
100500  012637  000006          MOV     (SP)+,@#ERRVEC+2        ;;SAVE THE PSW IN @#ERRVEC+2
100504  012737  100524  000004  MOV     #2$,@#ERRVEC    ;;SET FOR TIMEOUT
100512  012701  020000          MOV     #20000,R1       ;;FIRST ADDRESS
100516  005711          1$:     TST     (R1)            ;;TEST THIS ADDRESS
100520  005721                  TST     (R1)+           ;;STEP TO NEXT ADDRESS
100522  000775                  BR      1$              ;;TRY ANOTHER
100524  162701  000002  2$:     SUB     #2,R1           ;;DROP BACK
100530  010006                  MOV     R0,SP           ;;RESTORE THE STACK
100532  012637  000006          MOV     (SP)+,@#ERRVEC+2 ;;RESTORE ERROR VECTOR
100536  012637  000004          MOV     (SP)+,@#ERRVEC
100542  012637  000116          MOV     (SP)+,@#116     ;;RESTORE MEMORY ERROR VECTOP
100546  012637  000114          MOV     (SP)+,@#114
100552  010137  100564          MOV     R1,$LSTAD       ;;LAST ADDRESS
100556  012601                  MOV     (SP)+,R1        ;;RESTORE R1
100560  012600                  MOV     (SP)+,R0        ;;RESTORE R0
100562  000207                  RTS     PC
100564  000000  $LSTAD: .WORD   0               ;;CONTAINS THE LAST ADDRESS
```

```
                                .SBTTL  BUSADR - GET BUS ADDRESS AND VECTOR ADDRESS
                                ;
                                ;THIS ROUTINE IS USED TO INSURE THE BUS ADDRESS OF THE RH/RM
                                ;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
                                ;
100566  005737  001336  BUSADR: TST     CHGADR          ;INPUT FROM TTY REQUESTED?
100572  002053                  BGE     3$              ;NO--BRANCH
100574  005037  001336          CLR     CHGADR          ;YES--CLEAR THE REQUEST FLAG
100600  104401  001203          TYPE    ,$CRLF          ;TYPE A CR-LF
100604  005227  177777          INC     #-1             ;FIRST TIME THRU ?
100610  001402                  BEQ     1$              ;BR IF YES
100612  104401  001203          TYPE    ,$CRLF          ;CR-LF
100616  005037  001340  1$:     CLR     CFLAG           ;CLEAR CONTROL C FLAG
100622  012700  001272          MOV     #$RMADR,R0      ;FIRST ADDRESS
100626  104401  100740          TYPE    ,MRMCS1         ;'RMCS1='
100632  011046                  MOV     (R0),-(SP)      ;PRESENT RMCS1 ADDRESS
100634  104402                  TYPOC                   ;TYPE IT
100636  104401  075554          TYPE    ,BLNKS2         ;TYPE 2 BLANKS
100642  104411                  RDLIN                   ;GET THE ENTRY
100644  012601                  MOV     (SP)+,R1        ;ADDRESS OF ASCII TEXT
100646  005737  001340          TST     CFLAG           ;WAS IT CONTROL C ?
100652  001361                  BNE     1$              ;BR IF YES
100654  004537  100756          JSR     R5,CK.NUM       ;ENTER AND STORE THE NEW ADDRESS
100660  000756                  BR      1$              ;ERROR EXIT
100662  012700  001274  2$:     MOV     #$RMVEC,R0      ;VECTOR ADDRESS
100666  104401  100747          TYPE    ,MRMVEC         ;'RMVEC='
100672  011046                  MOV     (R0),-(SP)      ;PRESENT RH/RM VECTOR ADDRESS ON THE STACK
100674  104402                  TYPOC                   ;TYPE IT
100676  104401  075554          TYPE    ,BLNKS2         ;TYPE 2 BLANKS
100702  104411                  RDLIN                   ;READ THE ENTRY
100704  012601                  MOV     (SP)+,R1        ;ASCII TEXT ADDRESS
100706  005737  001340          TST     CFLAG           ;WAS IT CONTROL C ?
100712  001341                  BNE     1$              ;BR IF YES
100714  004537  100756          JSR     R5,CK.NUM       ;ENTER AND STORE NEW ADDRESS
100720  000760                  BR      2$              ;ERROR EXIT
100722  012700  001272  3$:     MOV     #$RMADR,R0      ;FIRST ADDRESS OF NEW PARAMETERS
100726  012701  040154          MOV     #RMADR,R1       ;FIRST ADDRESS OF WHERE TO PUT THEM
100732  012021                  MOV     (R0)+,(R1)+     ;BUS ADDRESS
100734  012021                  MOV     (R0)+,(R1)+     ;VECTOR ADDRESS
100736  000207                  RTS     PC              ;RETURN

100740      122     115     103 MRMCS1: .ASCIZ  @RMCS1=@
100747      122     115     126 MRMVEC: .ASCIZ  @RMVEC=@
```

```
                                        .SBTTL  CK.NUM - CHECK NUMBER (OCTAL)

                                        ;THIS ROUTINE CHECKS AN ASCIZ STRING FOR LEGAL CHARACTERS
                                        ;AND FORMS AN OCTAL NUMBER IN R2
                                        ;CALL:
                                        ;       MOV     #ADR,R0         ;ADDRESS TO PLACE NEW NUMBER
                                        ;       MOV     #ADR,R1         ;ADDRESS OF ASCIZ STRING
                                        ;       JSR     R5,CK.NUM       ;R5 CHANGED
                                        ;       RET                     ;ERROR EXIT
                                        ;       RET                     ;NORMAL EXIT

100756  010246                CK.NUM: MOV     R2,-(SP)        ;SAVE R2
100760  010346                        MOV     R3,-(SP)        ;SAVE R3
100762  010446                        MOV     R4,-(SP)        ;SAVE R4
100764  012703  000006                MOV     #6,R3           ;MAX OCTAL DIGITS IN THE NUMBER
100770  005002                        CLR     R2              ;FINAL OCTAL VALUE
100772  112104                1$:     MOVB    (R1)+,R4        ;GET CURRENT POINTED BYTE
100774  001424                        BEQ     3$              ;BRANCH,IF TERMINATOR DETECTED
100776  120427  000060                CMPB    R4,#'0          ;SMALLER THAN ASCII-0 ?
101002  103425                        BLO     5$              ;YES,ERROR EXIT
101004  120427  000067                CMPB    R4,#'7          ;LARGER THAN ASCII-7 ?
101010  101022                        BHI     5$              ;YES,ERROR EXIT
101012  006302                        ASL     R2              ;SHIFT LEFT
101014  103420                        BCS     5$              ;
101016  006302                        ASL     R2              ;ONE
101020  103416                        BCS     5$              ;
101022  006302                        ASL     R2              ;OCTAL DIGIT
101024  103414                        BCS     5$              ;ERROR IF CARRY BIT SET
101026  042704  177770                BIC     #177770,R4      ;CHOP OFF HIGHER BITS
101032  060402                        ADD     R4,R2           ;APPENDING CURRENT DIGIT TO NUMBER
101034  005303                        DEC     R3              ;DECREMENT BYTE COUNT
101036  001401                        BEQ     2$              ;BRANCH,IF LAST BYTE
101040  000754                        BR      1$              ;LOOPING BACK
101042  112104                2$:     MOVB    (R1)+,R4        ;CHECK TERMINATOR
101044  001004                        BNE     5$              ;ERROR EXIT
101046  005702                3$:     TST     R2              ;FINAL VALUE= 0
101050  001401                        BEQ     4$              ;YES,THEN NOT REPLACE THE ORIGINAL VALUE
101052  010210                        MOV     R2,(R0)         ;REPLACE THE ORIGINAL VALUE
101054  005725                4$:     TST     (R5)+           ;ADJUST FOR NORMAL RETURN
101056  012604                5$:     MOV     (SP)+,R4        ;RESTORE R4
101060  012603                        MOV     (SP)+,R3        ;RESTORE R3
101062  012602                        MOV     (SP)+,R2        ;RESTORE R2
101064  000205                        RTS     R5              ;EXIT

101066                          CYLNDR: .BLKW   258.            ;ONE SECTOR WORD CTR MAX SIZE

                        102072  ENDPGM=.

                        000200  .END    200
```

SYMBOL TABLE

```
ABASE = 176700        ASGN6   026254        BIT5  = 000040        CMSTR2  013602        DRIVE3  055040
ABNRML  031134        ASGN7   026266        BIT6  = 000100        CMTRK   001375        DRIVE4  057252
ACDW1 = 000000        ASKPAR  077470        BIT7  = 000200        COLON   076670        DRIVE5  061464
ACDW2 = 000000        ASNERR  031024        BIT8  = 000400        COMMA   075617        DRIVE6  063676
ACK   = 000123        ASNLST  001530        BIT9  = 001000        COMTBL  002064        DRIVE7  066110
ACPUOP= 000000        ASNMSG  031050        BLKADR  002044        CPSAVE  035124        DRNUM   076372
ACTDRV  040102        ASSIGN  025624        BLNKS1  075555        CR    = 000015        DROP    031054
ACTSTR  040103        ASWREG= 000000        BLNKS2  075554        CRLF  = 000200        DROPNG  076236
ADDW0 = 000000        ATA   = 100000        BLNKS3  075553        CTRAP   034474        DRQ   = 004000
ADDW1 = 000000        ATABIT  040142        BLNKS4  075552        CYLIMT  001422        DRVACT  040026
ADDW10= 000000        ATESTN= 000000        BPTVEC= 000014        CYLNDR  101066        DRVCLR= 000111
ADDW11= 000000        ATTN    001322        BSE   = 100000        DASH    075621        DRVER   011256
ADDW12= 000000        AT0   = 000001        BUFTBL  001642        DATAPK  026552        DRVINT  040400
ADDW13= 000000        AT1   = 000002        BUSADR  100566        DATA0   002472        DRVPAR  001430
ADDW14= 000000        AT2   = 000004        CFLAG   001340        DATA1   002532        DRVPRM  027020
ADDW15= 000000        AT3   = 000010        CHGADR  001336        DATA10  003172        DRVQUE  046036
ADDW2 = 000000        AT4   = 000020        CHKWC   020316        DATA11  003232        DRVSTA  040036
ADDW3 = 000000        AT5   = 000040        CI1     041434        DATA12  003272        DRVTYP  040046
ADDW4 = 000000        AT6   = 000100        CI3     041542        DATA13  003332        DRY   = 000200
ADDW5 = 000000        AT7   = 000200        CI4     041650        DATA14  003372        DSWR  = 177570
ADDW6 = 000000        AUNIT = 000000        CI5     042226        DATA15  003432        DTE   = 010000
ADDW7 = 000000        AUSWR = 000000        CI6     042250        DATA2   002572        DTEER   012050
ADDW8 = 000000        AUTLST  031662        CI7     042264        DATA3   002632        DTUW    040140
ADDW9   000000        AVAIL   001576        CI7B    042306        DATA4   002672        DT00  = 000001
ADEVCT= 000100        AVECT1= 000254        CI8     042364        DATA5   002732        DT01  = 000002
ADEVM = 000000        AVECT2= 000000        CKBUS   013270        DATA6   002772        DT02  = 000004
AENV  = 000000        A16   = 000400        CKCLK   023344        DATA7   003032        DT03  = 000010
AENVM = 000000        A17   = 001000        CKCLK1  023440        DATA8   003072        DT04  = 000020
AFATAL= 000000        BADBLK  001476        CKCLK2  023510        DATA9   003132        DT05  = 000040
ALOST   077232        BADENT  076677        CKCLK3  023542        DCK   = 100000        DT06  = 000100
AMADR1= 000000        BADSEC  001342        CKERR   013176        DCKER   010156        DT07  = 000200
AMADR2= 000000        BADTMO  003552        CKFMT   011310        DCKER1  010274        DT08  = 000400
AMADR3= 000000        BAI   = 000010        CKHCE   011464        DDISP = 177570        DT1     073530
AMADR4= 000000        BEGCOD  001502        CKLMTS  020164        DEASGN  026330        DT14    073600
AMAMS1= 000000        BEGPAT  001500        CKSWR = 104407        DEASSG  076341        DT15    073622
AMAMS2= 000000        BEGWC   001504        CK.CHR  032720        DH1     073063        DT16    073644
AMAMS3= 000000        BIT0  = 000001        CK.DEC  032672        DH14    073240        DT17    073660
AMAMS4= 000000        BIT00 = 000001        CK.DIG  032772        DH15    073342        DT2     073534
AMSGAD= 000000        BIT01 = 000002        CK.NUM  100756        DH16    073440        DT3     073552
AMSGLG= 000000        BIT02 = 000004        CK.OCT  032644        DH17    073510        DT4     073562
AMSGTY= 000000        BIT03 = 000010        CLKFLG  001312        DH2     073070        DT6     073574
AMTYP1= 000000        BIT04 = 000020        CLOCK   025074        DH3     073144        DUNIT   001532
AMTYP2= 000000        BIT05 = 000040        CLR   = 000040        DH4     073172        DVA   = 004000
AMTYP3= 000000        BIT06 = 000100        CLRDPB  026602        DH6     073231        DVC   = 000200
AMTYP4= 000000        BIT07 = 000200        CLRQUE  045740        DISPLA  001156        ECBAD0  001412
AOE   = 001000        BIT08 = 000400        CMCNT   001370        DISPLY= 104414        ECBAD1  001420
APASS = 000000        BIT09 = 001000        CMCYL   001372        DISPRE  000174        ECBIT   001376
APRIOR= 000000        BIT1  = 000002        CMDAT   013614        DLT   = 100000        ECC     014606
APTCSU= 000040        BIT10 = 002000        CMHED   013540        DONE    007634        ECCX    015344
APTENV= 000001        BIT11 = 004000        CMPAR   013354        DPE   = 000010        ECC1    015210
APTSIZ= 000200        BIT12 = 010000        CMPARD  013372        DPINT   040056        ECC2    015340
APTSPO= 000100        BIT13 = 020000        CMPLMT  001446        DPR   = 000400        ECGD    001410
ASGND   076411        BIT14 = 040000        CMPRES  020700        DPRQS   040066        ECGD1   001416
ASGN1   025722        BIT15 = 100000        CMPRT   014174        DRIVE = 001220        ECH   = 000100
ASGN2   026004        BIT2  = 000004        CMPRX   014166        DRIVE0  046202        ECI   = 004000
ASGN3   026102        BIT3  = 000010        CMSEC   001374        DRIVE1  050414        ECMSK0  001402
ASGN4   026252        BIT4  = 000020        CMSTR   013456        DRIVE2  052626        ECMSK1  001404
```

SYMBOL TABLE

```
ECSEC   001400   ERCTR   001364   INVLD   076544   LINS4   074226   MERR1   076720
ECWRD   001406   ERPRC1  007254   IOTVEC= 000020   LINS5   074342   MERR2   077000
ECWRD1  001414   ERPROC  007240   IR     = 000100   LINT3   074110   MESSAG  001472
EMPTYQ  046016   ERR    = 040000   ISR      042614   LINU06  074531   MINCYL= 000126
EMTVEC= 000030   ERROR  = 104000   IVC    = 010000   LINW3   074147   MINSEC= 000136
EM1     070412   ERRVEC= 000004   KSR      025240   LINX4   074242   MINTRK= 000132
EM10    070724   FACTOR  016400   KSR1     025246   LINX5   075557   MINUTE  001346
EM11    070767   FAIRNS  001332   LBC    = 002000   LIN10A  075154   MNTBL   002112
EM12    071022   FALPAR  007420   LBT    = 002000   LIN10B  075210   MOH    = 020000
EM13    071053   FALPR1  007434   LF     = 000012   LIN10C  075250   MOL    = 010000
EM14    071125   FER    = 000020   LIMIT    001366   LIN10H  075354   MRMCS1  100740
EM15    071150   FILBUF  016772   LINA3    074071   LIN11   075512   MRMVEC  100747
EM2     070464   FILLZ   032334   LINB3    074140   LIN11A  075533   MSFULL  077044
EM20    071204   FILLO   032442   LINB5    074310   LIN11H  075437   MSGCTS  077102
EM21    071225   FMTER   012244   LINB6    074417   LIN2C   073666   MSGFOR  076534
EM22    071256   FMT16  = 010000   LINCA3   074120   LIN2P   073706   MSGON   076311
EM23    071331   FRSTER  001356   LINC6    074452   LIN2S   073727   MSPRM   077315
EM24    071410   F0     = 000002   LINDA3   074131   LIN3.1  022032   MSWRO   076501
EM25    071456   F1     = 000004   LINDEC   023176   LIN3.3  022140   MXF    = 001000
EM26    071537   F2     = 000010   LIND5    074274   LIN3.4  022172   MXWNDW  040164
EM27    071564   F3     = 000020   LINEN3   074055   LIN6.1  022642   M.DPID  031756
EM3     070522   F4     = 000040   LINEO5   074405   LIN6.2  022664   M.DP40  032014
EM30    071621   GENDPB  070322   LINEP5   074375   LIN7M   074556   M.DP41  032050
EM31    071653   GENPAR  017470   LINE1    021046   LIN7OR  074706   M.DP42  032060
EM32    071716   GENREG  070342   LINE2    021126   LIN7OX  074661   M.DP44  032112
EM33    071751   GETADR  027520   LINE2A   021276   LIN7P   074603   M.DP50  032124
EM34    072026   GETBUF  016402   LINE2B   021314   LIN7R   074716   N       076473
EM35    072070   GETID   027420   LINE3    021566   LIN7S   074621   NED    = 010000
EM36    072126   GETLMT  027346   LINE3A   021574   LIN7T   074643   NEDCLK  076422
EM37    072157   GETPAT  020122   LINE3B   021602   LIN7X   074671   NEM    = 004000
EM4     070560   GETREG= 000141   LINE3C   021614   LIN8M   074732   NEWASN  026530
EM40    072241   GETREM  031664   LINE3D   021624   LIN9B   074755   NEWUNT  001554
EM41    072310   GETREQ  046112   LINE3E   021672   LIN9E   075077   NODRVS  077120
EM42    072354   GO     = 000001   LINE3F   021760   LIN9G   075126   NOENTY  077146
EM43    072435   GODRIV  017050   LINE4    022236   LIN9H   075004   NOMTCH  012724
EM44    072502   GTSWR  = 104406   LINE5    022326   LIN9I   075060   NONE    076672
EM45    072577   HCE    = 000200   LINE5A   022470   LKPAR   005320   NOTAVL  075762
EM46    072665   HCEER   012306   LINE5B   022536   LODEV   076011   NOTPRS  075745
EM5     070615   HCI    = 002000   LINE6    022600   LODPAR  020430   NOTRM   075724
EM50    072737   HCRC   = 000400   LINE6A   022612   LSC    = 004000   NOTSAF  076001
EM51    072775   HCRCER  011132   LINE6B   022620   LSTAD   001334   NSA    = 100000
EM6     070651   HOUR    001344   LINE6C   022626   LSTHDR  077166   OFFCOD  002414
EM60    073040   HT     = 000011   LINE6D   022634   MAIN    006142   OFFDIR= 000001
ENDCMP  014450   HZ      001314   LINE7    022710   MAINDA  006170   OFFON  = 000001
ENDCON  001434   IAE    = 002000   LINE7A   023040   MAIN1   006306   OFFSET= 000115
ENDING  001466   IAEER   012164   LINE8    023132   MAIN2   006424   OFFST   015766
ENDPAS  076272   IBSAVE  035126   LING6    074501   MANTER  030124   OFLIN   007532
ENDPGM= 102072   IDLE    007006   LINM3    073763   MASK    001326   OFMSG0  002312
ENDSEK  001440   IE     = 000100   LINM4    074210   MATCH   014516   OFMSG1  002345
ENDTST  076316   ILF    = 000001   LINN3    074001   MAXCYL= 000124   OFMSG2  002370
ENTADR  076640   ILR    = 000002   LINOCT   023144   MAXER   001444   OFMTBL  002420
ENTCOM  076567   INCHRD . 024630   LINP3    074022   MAXSEC= 000134   ONES    003374
ENTLMT  076610   INCMIS  024700   LINP5    074325   MAXTRK= 000130   ONESEC  001352
ENTPR   005412   INCSKI  024654   LINR6    074517   MBASN   077303   OPE    = 020000
EOP1    031214   INCSOF  024604   LINSS3   074174   MCPE   = 020000   OPI    = 020000
EOP2    031240   INCTOT  024724   LINST3   074160   MCPEMX  040152   OPIER   011744
EQUAL   075615   INTRVL  001452   LINS3    074041   MDPE   = 000400   OPIER1  012010
```

SYMBOL TABLE

```
OPT     041206
OPTBL   002072
OR    = 000200
ORDERQ  001506
PACK    001320
PACKSN  077270
PAR   = 000010
PARENT  030700
PARER   012072
PARLST  077360
PAR1    077524
PAR10   077634
PAR11   077644
PAR14   077654
PAR15   077664
PAR16   077674
PAR19   077704
PAR2    077534
PAR20   077714
PAR21   077724
PAR3    077544
PAR4    077554
PAR5    077564
PAR6    077574
PAR7    077604
PAR8    077614
PAR9    077624
PASSES  001456
PAT     000020
PATTER  001460
PCLOCK  001310
PERIOD  076477
PFECH   035306
PFECH1  035316
PFECH2  035400
PFECH3  035432
PFECH4  035442
PFTSTN  035446
PGE   = 002000
PGM     001000
PIP     020000
PIRQ  - 177772
PIRQVE= 000240
POPQUE  046134
POSER   011672
PROCES  007150
PRTBAD  015352
PRTIM   007576
PR0   - 000000
PR1   = 000040
PR2   = 000100
PR3   - 000140
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSEL  = 002000
PSW   = 177776
PUNSAF  007336
PWRFLG  037616
PWRVEC= 000024
QCNT    045446
QDRV0   045540
QDRV1   045560
QDRV2   045600
QDRV3   045620
QDRV4   045640
QDRV5   045660
QDRV6   045700
QDRV7   045720
QINPT   045456
QOUTPT  045476
QSTART  045516
QSTOP   045520
QTERM = 045740
QUES    075613
RANCYL  017654
RANDOM  001474
RANDWC  001462
RANPAT  020072
RANSEC  017744
RANSIZ  020000
RANTRK  017710
RANXIT  020112
RATIO   001464
RDCHR = 104410
RDDAT = 000171
RDHD  = 000173
RDLIN = 104411
RDY   = 000200
RD.ADR  044712
RD.RM   044666
RD.RM1  044710
RD.RM2  045030
RD.RM3  045034
RD.RM4  045040
RD.WRD  044714
READDR  023216
READHD  016012
READIN= 000121
RECAL = 000107
RECALT  015650
RECALO  015740
REDAPK  026540
RELBUF  016536
RELSE = 000113
REPHD   076063
REPLZ   032340
RESREG= 104413
RESVEC= 000010
RETRY   001330
RHEXT   040162
RMADR   040154
RMAS  = 000016
RMBA  = 000004
RMBAE = 000050
RMCS1 = 000000
RMCS2 = 000010
RMCS3 = 000052
RMDA  = 000006
RMDB  = 000022
RMDC  = 000034
RMDS  = 000012
RMDT  = 000026
RMEC1 = 000044
RMEC2 = 000046
RMERRS  040016
RMER1 = 000014
RMER2 = 000042
RMHR  = 000036
RMINIT  040166
RMLA  = 000020
RMMR1 = 000024
RMMR2 = 000040
RMOF  = 000032
RMR   = 000004
RMSN  = 000030
RMTMR   044254
RMVEC   040156
RMWC  = 000002
RM05    040714
RNOP  = 000101
RTC   = 000117
RTNCTR  015624
RTURN   031636
R6    =%000006
R7    =%000007
S       074017
SATPOW  037636
SAVEFG  040114
SAVER1  001360
SAVER5  001362
SAVREG= 104412
SC      043120
SCMND   026436
SCOPE = 000004
SC04  = 000400
SC1   = 000100
SC10  = 001000
SC11    043736
SC12    044046
SC13    044116
SC2   = 000200
SC20  = 002000
SC3     043164
SC4     043170
SC5     043202
SC6     043366
SC6A    043502
SC7     043630
SC8     043706
SDETAL  023674
SEARCH= 000131
SECLMT  001424
SECOND  001350
SEEK  = 000105
SEEKFG  040116
SELDRV= 000145
SETFMT= 000143
SETVEC  005440
SET.IE  045374
SHDTYP  023664
SIZE70  004744
SIZMEM  005112
SKI   = 040000
SKIER   012450
SLASH   077464
SPOTCK  020714
SRCHWT  040100
STA     005730
STACK = 001100
START   003634
START1  003644
START2  003662
STAR30  076124
STAR5   076155
STATIN  001316
STATIS  016232
STATPR  023550
STKLMT= 177774
STNDAT  002426
STO     044344
STO1    044400
STO2    044474
STO3    044524
STO5    044550
STO6    044556
STO7    044614
STO8    044644
STO9    044654
SUMARY  023636
SUMHD   076164
SUPRS   032150
SUPRSL  032134
SUPR1   032162
SUPR2   032224
SVRH70  045230
SWR     001154
SWREG   000176
SWTIM   007474
SW0   = 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
SYSTAT  076026
T       073776
TAB     075623
TABLE   077734
TABLE0  077754
TABLE1  100022
TABLE2  100070
TABLE3  100136
TABLE4  100204
TABLE5  100252
TABLE6  100320
TABLE7  100366
TAB.XY= 001114
TAP   = 040000
TBITVE= 000014
TD      042656
THEAD   017572
THEAD1  017614
TIMER   040120
TKVEC = 000060
TPVEC = 000064
TRAPVE= 000034
TRE   = 040000
TRFER   012350
TRKLMT  001426
TRNSWT  040076
TRTVEC= 000014
TST1    003652
TYLIST  030520
TYMBA   032472
TYPACK  032522
TYPCK   032526
TYPDS = 104405
TYPE  = 104401
TYPMBA  032476
TYPOC = 104402
TYPON = 104404
TYPOS = 104403
```

SYMBOL TABLE

```
TYPRI4  032444   $BUF  = 000006   $HINUM 037022   $POWER 037620   $SOFT = 000074
UCPAR   007402   $CDW1   001266   $HLDWC= 000110   $PREVA= 000032   $SSEC = 000022
ULDFLG  040104   $CDW2   001270   $ICNT   001120   $PREVO= 000027   $STUP = 177777
UNIT    001324   $CHARC  036000   $ILLUP  037610   $PSEL = 000003   $SUPRL  032234
UNS   = 040000   $CKSWR  033526   $INTAG  001151   $PSNL = 000140   $SUPRS  032250
UNSAF   012610   $CMTAG  001114   $ITEMB  001130   $PSNM = 000142   $SUPR1  032262
UNTASN  075676   $CM3  = 000C00   $LF     001204   $PWRAD  037604   $SUPR2  032324
UNTMSG  075625   $CM4  = 000001   $LFLG   036721   $PWRDN  037436   $SVPC = 000210
UNTNOT  075654   $CNTLC  034433   $LKCSB  001300   $PWRMG  037600   $SWR  = 122000
UNTOFF  075633   $CNTLG  034445   $LKCSR  001276   $PWRUP  037510   $SWREG  001230
UNTON   075644   $CNTLU  034440   $LKS    001304   $QUES   001202   $TATUS= 000016
UPE   = 020000   $CODE = 000024   $LLVEC  001306   $RAND   036724   $TERM = 000032
US1   = 000001   $COMND= 000002   $LONUM  037024   $RDCHR  034070   $TESTN  001212
US2   = 000002   $CPUOP  001234   $LPADR  001122   $RDLIN  034160   $TIME   024750
US4   = 000004   $CRLF   001203   $LPERR  001124   $RDOFL= 000064   $TKB    001162
VV    = 000100   $CYL  = 000012   $LPVEC  001302   $RDSZ = 000017   $TKCNT  033210
WAIT    001620   $DBLK   036446   $LSTAD  100564   $READ = 000066   $TKINT  033226
WATPAK  026564   $DB2D   037122   $MADR1  001240   $REG  = 000014   $TKQEN= 033225
WCE   = 040000   $DB2O   037316   $MADR2  001244   $RESRE  037064   $TKQIN  033212
WCF   = 000040   $DECVL  037302   $MADR3  001250   $RETRY  016104   $TKQOU  033214
WCFER   012512   $DEVCT  001216   $MADR4  001254   $RHEXT= 000001   $TKQSR  033216
WCHKX = 000000   $DEVM   001264   $MAIL   001206   $RMADR  001272   $TKS    001160
WCKD  = 000151   $DIV    031710   $MAMS1  001236   $RMAS = 002154   $TKSRV  033276
WCKER   010624   $DOAGN  031632   $MAMS2  001242   $RMBA = 002142   $TMPO   001174
WCKHD = 000153   $DSPLY  032616   $MAMS3  001246   $RMBAE= 002206   $TN   = 000002
WC.HK   043032   $DTBL   036436   $MAMS4  001252   $RMCS1= 002136   $TNPWR  037232
WLE   = 004000   $ENDAD  031622   $MBADR  001102   $RMCS2= 002146   $TOTAL= 000072
WLEER   012216   $ENDAT= 000036   $MBASN= 002126   $RMCS3= 002210   $TPB    001166
WRDCNT  001450   $ENDCT  031606   $MFLG   036720   $RMDA = 002144   $TPFLG  001173
WRL     004000   $ENDSK= 000042   $MISPO= 000102   $RMDB = 002160   $TPS    001164
WRTCHK  001470   $ENV    001226   $MNEW   034463   $RMDC = 002172   $TRAP   037730
WRTDAT= 000161   $ENVM   001227   $MSGAD  001222   $RMDS = 002150   $TRAP2  037752
WRTHD   000163   $EOP    031162   $MSGLG  001224   $RMDT = 002164   $TRK  = 000011
WRTPK   017140   $EOPCT  031600   $MSGTY  001206   $RMEC1= 002202   $TRP  = 000015
WRT.AD  045132   $ERFLG  001117   $MSWR   034452   $RMEC2= 002204   $TRPAD  037764
WRT.RM  045042   $ERMAX  001131   $MTYP1  001237   $RMER1= 002152   $TSTM   001104
WRT.R1  045126   $ERROR  034540   $MTYP2  001243   $RMER2= 002200   $TSTNM  001116
WRT.R2  045214   $ERRPC  001132   $MTYP3  001247   $RMHR = 002174   $TTYIN  034414
WRT.R3  045220   $ERRTB  003472   $MTYP4  001253   $RMLA = 002156   $TYPDS  036232
WRT.R4  045224   $ERRTY  035130   $NCODE= 000112   $RMMR1= 002162   $TYPE   035450
WRT.R5  045226   $ERTTL  001126   $NCYL = 000116   $RMMR2= 002176   $TYPEC  035662
WRT.WD  045130   $ETABL  001226   $NEXT = 000120   $RMOF = 002170   $TYPEX  036002
XXDP    001432   $ETEND  001272   $NPATC= 000113   $RMSN = 002166   $TYPOC  036030
Y       076475   $FAIR = 000106   $NSEC = 000114   $RMVEC  001274   $TYPON  036044
ZEROS   002472   $FATAL  001210   $NTRK = 000115   $RMWC = 002140   $TYPOS  036004
ZROIND  001354   $FFLG   036722   $NULL   001170   $RM02   076044   $UNIT   001220
$APTHD  001100   $FILLC  001172   $NWTST= 000000   $RM03   076051   $UNITM  001110
$ATYC   036502   $FILLS  001171   $OCNT   036226   $RM05   076056   $USWR   001232
$ATY1   036456   $FIRST= 000122   $OCTVL  037420   $RTNAD  031634   $VECT1  001256
$ATY3   036464   $FMT  = 000001   $OMODE  036230   $SAVRE  037026   $VECT2  001260
$ATY4   036474   $GDADR  001134   $OPERC= 000046   $SAVR6  037614   $WCNT = 000004
$AUTOB  001150   $GDDAT  001140   $PACK = 000026   $SB2D   033130   $WRDL = 000020
$BASE   001262   $GET42  031612   $PASS   001214   $SB2O   033160   $WRITN= 000060
$BDADR  001136   $GTSWR  033616   $PASSC= 000104   $SEC  = 000010   $WTOFL= 000056
$BDDAT  001142   $HARD = 000076   $PASTM  001106   $SETUP= 000156   $XOFF = 000023
$BDSEC= 000144   $HD   = 000000   $PATTC= 000030   $SIZE   100434   $XON  = 000021
$BELL   001176   $HIBTS  001100   $POSIT= 000052   $SKI  = 000100   $$GET4= 000000
```

SYMBOL TABLE

$OFILL 036227   .$X   = 001100

. ABS. 102072   000
        000000   001
ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 62464 WORDS (244 PAGES)
DYNAMIC MEMORY AVAILABLE FOR 70 PAGES
CZRMUB.BIC,CZRMUB/C-CZRMUB.DOC,CZRMUB,SYSMAC/M

```
$$GET4  35-1    35-1#
$OFILL  45-1    45-1#   45-1*   45-1*
$40CAT  42-1
$APTHD   5-8     5-8#
$ASTAT  47-1    47-1
$ATY1   47-1#
$ATY3   44-1    47-1#
$ATY4   42-1    47-1#
$ATYC   47-1    47-1#
$AUTOB   -0#     9-30*   9-76    9-163   9-178   10-12   11-82   23-66   29-58   29-76   32-11   35-1    35-1    41-1
        41-1    41-1
$BASE    6-0#    9-71    9-73
$BDADR   6-0#
$BDDAT   6-0#
$BDSEC  21-22   31-36   32-18   32-95   55-68#  55-72
$BELL    6-0#   22-10   41-1    41-1    41-1    42-1    42-1    42-1
$BUF    11-98*  11-134*  12-704  13-10   14-10   14-96   14-102  14-107  14-111  14-153  22-102  22-252  22-259  55-9#
$CDW1    6-0#
$CDW2    6-0#   31-44*  31-53   31-85*
$CHARC  44-1    44-1#   44-1*   44-1*   44-1*
$CKSWR  41-1#   53-1    53-1
$CM3     6-0     6-0#
$CM4     6-0     6-0     6-0#    6-0#
$CMTAG   6-0#    9-25    9-25    9-25
$CNTLC  41-1    41-1    41-1    41-1    41-1#
$CNTLG  41-1    41-1#
$CNTLU  41-1    41-1    41-1#
$CODE   12-669  13-3    13-30   13-90   13-254  13-350  13-356  14-14   14-16   14-24   14-151  15-32*  15-53*  15-56*
        20-25*  20-33*  20-39   29-33*  29-43   54-758*  55-20#
$COMND  13-394*  13-408*  13-428*  13-444*  13-463*  15-54*  15-57*  20-27*  20-28   20-32*  29-13   29-35*  31-43*  54-747
        54-759*  55-6#
$CPUOP   6-0#    9-110*   9-137*  22-89   22-271  54-<17
$CRLF    6-0#    9-50    9-219    9-263   11-39   11-61   12-47   12-59   12-60   12-66   12-67   12-92   12-657  12-666
        12-666  12-666  12-666  13-159  13-181  13-193  13-319  13-328  13-354  13-367  13-381  13-383  13-384  22-13
        22-14   22-16   22-17   22-36   22-70   22-72   22-106  22-119  22-138  22-158  22-177  22-192  22-220  22-243
        22-262  22-295  22-309  22-322  22-329  22-362  22-371  22-403  22-430  24-21   24-72   24-159  27-15   27-157
        27-161  31-97   32-119  32-126  34-7    34-26   35-1    35-1    41-1    41-1    41-1    41-1    42-1    42-1
        42-1    42-1    42-1    43-1    43-1    43-1    43-1    44-1    44-1    44-1    63-16   63-19
$CYL    13-15   13-418*  13-461*  14-181  15-22   15-24*  15-31*  20-15   20-37*  22-165  29-13   29-110*  31-39*  54-756
        55-12#
$DB2D   22-393  22-401  22-422  24-107  24-118  24-126  40-126  50-1#
$DB2O   40-143  51-1#
$DBLK   46-1    46-1    46-1#
$DECVL  50-1    50-1#
$DEVCT   6-0#   35-1*   35-1*
$DEVM    6-0#
$DIV    13-249  13-343  22-291  36-11   36-39#
$DOAGN  35-1    35-1    35-1#
$DSPLY  39-103#  53-2
$DTBL   46-1    46-1#
$ENDAD   5-5    35-1#
$ENDAT  14-26*  14-27*  35-1    35-1    35-1    55-25#
$ENDCT  35-1#
$ENDSK  14-183*  14-184*  35-1    35-1    55-26#
$ENV     6-0#    9-30     9-66   10-6    42-1    44-1    47-1    47-1
$ENVM    6-0#    9-25    44-1    44-1    47-1
```

CROSS REFERENCE TABLE (CREF V01-05)

```
ACDW1    6-0       6-0
ACDW2    6-0       6-0
ACK      4-246#
ACPUOP   6-0       6-0
ACTDRV   54-93#    54-347*  54-398*  54-691*  54-699*  54-986
ACTSTR   54-99#    54-98P*  54-:02*
ADDW0    6-0
ADDW1    6-0
ADDW10   6-0
ADDW11   6-0
ADDW12   6-0
ADDW13   6-0
ADDW14   6-0
ADDW15   6-0
ADDW2    6-0
ADDW3    6-0
ADDW4    6-0
ADDW5    6-0
ADDW6    6-0
ADDW7    6-0
ADDW8    6-0
ADDW9    6-0
ADEVCT   6-0       6-0
ADEVM    6-0       6-0
AENV     6-0       6-0
AENVM    6-0       6-0
AFATAL   6-0       6-0
ALOST    32-50     60-53#
AMADR1   6-0       6-0
AMADR2   6-0       6-0
AMADR3   6-0       6-0
AMADR4   6-0       6-0
AMAMS1   6-0       6-0
AMAMS2   6-0       6-0
AMAMS3   6-0       6-0
AMAMS4   6-0       6-0
AMSGAD   6-0       6-0
AMSGLG   6-0       6-0
AMSGTY   6-0       6-0
AMTYP1   6-0       6-0
AMTYP2   6-0       6-0
AMTYP3   6-0       6-0
AMTYP4   6-0       6-0
AOE      4-154#
APASS    6-0       6-0
APRIOR   6-0
APTCSU   44-1      47-1#
APTENV   42-1      44-1      47-1      47-1#
APTSIZ    9-25     47-1#
APTSPO   44-1      47-1      47-1#
ASGN1    27-107#
ASGN2    27-105    27-121#
ASGN3    27-117    27-132    27-140#
ASGN4    27-141    27-168#
ASGN6    27-149    27-170#
ASGN7    27-148    27-173#
ASGND    11-64     60-33#
```

CROSS REFERENCE TABLE (CREF V01-05)

```
BIT3      4-74#   12-151
BIT4      4-74#   12-401
BIT5      4-74#   12-155
BIT6      4-74#   12-241  22-97
BIT7      4-74#   12-234  12-445  13-490
BIT8      4-74#   12-353  12-376  12-382  12-414  12-420  12-580
BIT9      4-74#   12-167  12-580
BLKADR    7-0#     9-90   10-26   24-25   27-144  27-165  28-15   52-22
BLNKS1   13-364   13-376  22-19   22-74   24-75   24-77   24-95   24-97   59-63#
BLNKS2    9-248   12-666  12-666  12-666  12-666  13-175  13-178  13-315  13-315  13-315  13-324  13-324  13-324  13-373
         22-77    22-116  22-167  22-172  22-188  22-307  22-307  22-318  59-62#  63-25   63-36
BLNKS3   13-361   59-61#
BLNKS4    9-221   59-60#
BPTVEC    4-74#
BSE       4-228#
BUFTBL    7-0#     9-153*  9-154*  9-155*  9-158*  9-159*  9-161*  9-162*  9-165*  9-166   9-168   14-48   14-50   14-64*
         14-91    14-92   14-104* 14-109* 14-114  14-115  14-128*
BUSADR    9-78    63-13#
CFLAG     7-0#     9-100*  9-177*  9-183   10-43*  11-3    11-84*  27-14*  27-22   27-86*  27-91   32-15*  32-36   32-121
         32-127*  33-9*   33-21   33-31*  41-5*   63-20*  63-28   63-39
CHGADR    7-0#     9-15*   9-18*   10-14   10-40*  29-60   29-78   52-19*  63-13   63-15*
CHKWC    15-41    17-47   19-16#
CI1      54-444   54-472#
CI3      54-446   54-496#
CI4      54-438   54-518#
CI5      54-494   54-516  54-596#  54-606
CI6      54-534   54-544  54-546   54-548  54-602#
CI7      54-377   54-453  54-484   54-488  54-492  54-501  54-510  54-514  54-526  54-533  54-539  54-543  54-557  54-563
         54-567   54-577  54-588   54-605  54-608#  54-715  54-910  54-<15
CI7B     54-623#  54-916
CI8      54-626   54-640#  54-769   54-:66
CK.CHR   40-50#   40-87   40-95
CK.DEC   40-28#   40-56
CK.DIG   32-53    32-60   32-67    33-23   40-81#
CK.NUM   63-30    63-41   64-12#
CK.OCT   40-10#   40-58
CKBUS    12-16    12-700#
CKCLK     9-205   23-38#  35-6     52-16
CKCLK1   23-41    23-52#
CKCLK2   23-54    23-63#
CKCLK3   23-50    23-61   23-71#
CKERR    12-15    12-681#
CKFMT    12-141   12-376#
CKHCE    12-145   12-414#
CKLMTS   15-27    16-36   18-14#
CKSWR    42-1     42-1    53-1#
CLKFLG    7-0#    23-38*  23-43*  23-56*  26-4
CLOCK    23-46    23-58   26-31#
CLR       4-102#   9-209  54-233   54-680
CLRDPB   10-27    27-150  29-10#
CLROUE   54-215   54-679  54-<94#
CMCNT     7-0#    12-668* 12-672* 13-11*  13-12*  13-23   13-25   13-26*  13-29*  13-88   13-92   13-95*  13-122*
CMCYL     7-0#    13-15*  13-16*  13-35   13-107* 13-108  13-114
CMDAT    13-31    13-44   13-53#  13-96   13-98
CMHED    13-35#
CMPAR    12-19    13-3#
CMPARD   12-271   13-8#
```

	12-657	12-658	12-666	12-666	12-666	12-666	12-666	12-666	12-666	12-666	12-690	12-708	13-153	13-158
	13-159	13-160	13-175	13-178	13-181	13-190	13-193	13-268	13-273	13-310	13-315	13-315	13-315	13-319
	13-324	13-324	13-324	13-327	13-328	13-354	13-355	13-358	13-361	13-364	13-367	13-369	13-373	13-376
	13-381	13-383	13-384	13-507	22-13	22-14	22-16	22-17	22-19	22-36	22-62	22-64	22-70	22-71
	22-72	22-73	22-74	22-77	22-82	22-85	22-93	22-105	22-106	22-116	22-119	22-125	22-131	22-138
	22-152	22-155	22-158	22-164	22-167	22-168	22-172	22-173	22-177	22-185	22-188	22-189	22-192	22-202
	22-207	22-209	22-212	22-216	22-220	22-225	22-228	22-235	22-239	22-243	22-251	22-254	22-257	22-262
	22-268	22-284	22-293	22-295	22-302	22-307	22-307	22-308	22-309	22-315	22-318	22-319	22-322	22-328
	22-329	22-335	22-341	22-347	22-353	22-360	22-362	22-370	22-371	22-384	22-387	22-390	22-395	22-398
	22-403	22-419	22-424	22-427	22-430	22-439	22-453	38-36	53-2#					
DISPRE	5-1#	9-25												
DLT	4-112#													
DONE	12-26	12-119#												
DPE	4-221#													
DPINT	54-61#	54-244	54-247	54-357	54-789	54-816	54-955	54-957*	54-:30	54-:53	54-:59	54-:69*		
DPR	4-134#													
DPRQS	54-70#	54-363	54-411*	54-448*	54-628*	54-645	54-663*	54-792	54-:32	54-:55	54-:61	54-:79*		
DRIVE	7-0#	42-1*	58-2	58-3	58-4									
DRIVE0	7-0	9-103*	9-150*	55-112#	61-54	61-54	61-54	61-54	61-54	61-54				
DRIVE1	7-0	9-103*	9-150*	55-112#	61-54	61-54	61-54	61-54	61-54	61-54				
DRIVE2	7-0	9-103*	9-150*	55-112#	61-54	61-54	61-54	61-54	61-54	61-54				
DRIVE3	7-0	9-103*	9-150*	55-112#	61-54	61-54	61-54	61-54	61-54	61-54				
DRIVE4	7-0	9-103*	9-150*	55-112#	61-54	61-54	61-54	61-54	61-54	61-54				
DRIVE5	7-0	9-103*	9-150*	55-112#	61-54	61-54	61-54	61-54	61-54	61-54				
DRIVE6	7-0	9-103*	9-150*	55-112#	61-54	61-54	61-54	61-54	61-54	61-54				
DRIVE7	7-0	9-103*	9-150*	55-112#	61-54	61-54	61-54	61-54	61-54	61-54				
DRNUM	27-155	60-32#												
DROP	12-55	12-100	34-20#	34-40										
DROPNG	34-27	60-26#												
DRQ	4-190#													
DRVACT	54-30#	54-371	54-551*	54-599*	54-630*	54-643	54-662*	54-704*	54-812	54-847	54-879*	54-887*	54-907	54-922*
	54-:44*													
DRVCLR	4-241#													
DRVER	12-185	12-366#												
DRVINT	54-235	54-269#	54-359	54-944	54-958									
DRVPAR	7-0#	27-87*	27-103*	27-153	29-62	29-80	32-13							
DRVPRM	10-31	27-163	29-56#											
DRVQUE	54-366	54-379	54-752	54-=34#										
DRVSTA	9-222	10-24	27-147	54-40#	54-225*	54-226*	54-227*	54-228*	54-238*	54-270*	54-280*	54-324*	54-330*	54-353
	54-361	54-385	54-419	54-423	54-552*	54-683*	54-795	54-803	54-818	54-873*	54-880*	54-896	54-960	54-:70*
DRVTYP	9-226	9-250	9-253	24-79	24-82	27-173	29-128	54-52#	54-271*	54-287*	54-292*	54-297*	54-302*	54-388
	54-684*													
DRY	4-133#													
DSWR	4-74#	6-0	9-25											
DT00	4-181#													
DT01	4-182#													
DT02	4-183#													
DT03	4-184#													
DT04	4-185#													
DT05	4-186#													
DT06	4-187#													
DT07	4-188#													
DT08	4-189#													
DT1	8-6	58-1#												
DT14	22-78	58-6#												
DT15	22-83	58-7#												
DT16	22-86	58-8#												

```
DT17    22-94    58-10#
DT2      8-13     8-34    58-2#
DT3      8-20    58-3#
DT4      8-27    58-4#
DT6      8-41    58-5#
DTE      4-157#
DTEER   12-175   12-493#
DTUW    27-10    54-139#  54-221   54-440   54-493*  54-637*  54-648   54-664   54-666*  54-677*  54-694   54-705*  54-:35   54-:47*
        54-:19
DUNIT    7-0#    11-14    28-15*   34-25*   35-1*
DVA      4-127#
DVC      4-222#
ECBAD0   7-0#    13-287*  13-315
ECBAD1   7-0#    13-304*  13-324
ECBIT    7-0#    13-260*  13-261*  13-262   13-263*  13-281*
ECC     12-263   13-241#
ECC1    13-302   13-310#
ECC2    13-277   13-300   13-327#
ECCX    13-318   13-325   13-328#
ECGD     7-0#
ECGD1    7-0#
ECH      4-151#
ECI      4-210#
ECMSK0   7-0#    13-279*  13-283*  13-288   13-290   13-299
ECMSK1   7-0#    13-280*  13-284*  13-293   13-305   13-307
ECSEC    7-0#    13-241*  13-253*  13-256*  13-258*  13-274
ECWRD    7-0#    13-262*  13-264*  13-265*  13-266*  13-267*  13-269   13-274*  13-275   13-287   13-289   13-290*  13-291*  13-295
        13-315   13-315
ECWRD1   7-0#    13-295*  13-296*  13-297   13-301*  13-304   13-306   13-307*  13-308*  13-317   13-324   13-324
EM1      8-4     56-3#
EM10    12-61    56-9#
EM11    12-68    56-10#
EM12    12-49    56-11#
EM13    12-82    56-12#
EM14    12-94    56-13#
EM15   12-105    56-14#
EM2      8-11    56-4#
EM20   12-341    56-15#
EM21   12-216    56-16#
EM22   12-300    56-17#
EM23   12-284    56-18#
EM24   12-389    56-19#
EM25   12-433    56-20#
EM26   12-539    56-21#
EM27   12-557    56-22#
EM3      8-18    56-5#
EM30   12-367    56-23#
EM31   12-475    56-24#
EM32   12-496    56-25#
EM33   12-502    56-26#
EM34   12-607    56-27#
EM35   12-520    56-28#
EM36   12-530    56-29#
EM37   12-303    56-30#
EM4      8-25    56-6#
EM40   12-575    56-31#
EM41   12-708    56-32#
```

```
EM42    13-153  56-33#
EM43    12-651  12-656  56-34#
EM44    12-690  56-35#
EM45    12-201  56-36#
EM46    22-105  56-37#
EM5      8-32   56-7#
EM50    12-595  56-38#
EM51    12-459  56-39#
EM6      8-39   56-8#
EM60    12-626  56-40#
EMPTYQ  54-627  54-682  54-735  54-:84  54-=19#
EMTVEC   4-74#   9-25*   9-25*
ENDCMP  13-126  13-186#
ENDCON   7-0#   10-4*   10-5*   10-9*   10-10*  35-1    35-1
ENDING   7-0#   35-1    61-9
ENDPAS  35-1    60-27#
ENDPGM   9-154   9-159  64-47#
ENDSEK   7-0#   35-1    35-1
ENDTST  35-1    60-29#
ENTADR  32-16   60-42#
ENTCOM  27-18   60-40#
ENTLMT  29-64   60-41#
ENTPR    9-190   9-196#
EOP1    35-1    35-1#
EOP2    15-35   35-1    35-1    35-1    35-1    35-1#
EQUAL   60-4#
ERCTR    7-0#   13-9*   13-47*  13-62   13-75*  13-188  13-191
ERPRC1  12-25   12-30#  12-238  13-508
ERPROC  12-5    12-9    12-11   12-24#
ERR      4-140#
ERROR    4-74#
ERRVEC   4-74#   9-25    9-25*   9-25*   9-27*   9-28*   9-111   9-112*  9-138*  23-40   23-41*  23-54*  23-71*  42-1
        42-1*   42-1*   62-1    62-1    62-1*   62-1*   62-1*   62-1*
F0       4-122#
F1       4-123#
F2       4-124#
F3       4-125#
F4       4-126#
FACTOR  14-9*   14-10*  14-12*  14-18   14-26   14-28   14-35#
FAIRNS   7-0#
FALPAR  12-36   12-66#
FALPR1  12-62   12-70#
FER      4-149#
FILBUF  11-102  11-136  14-150#
FILLO   39-11*  39-15   39-36*  39-39#
FILLZ   26-15   26-20   26-25   39-11#  39-89
FMT16    4-211#
FMTER   12-384  12-538#
FRSTER   7-0#   12-270*  12-272  12-648  12-674*  12-675*  13-6*   13-78   13-133  13-149  13-161*  13-186
GENDPB  12-247*  12-251  13-393*  13-394*  13-396  13-428*  13-430  13-432  13-443*  13-444*  13-446  13-459*  13-460*  13-461*
        13-462*  13-463*  13-464*  13-466  13-468  27-142*  31-38*  31-39*  31-40*  31-41*  31-42*  31-43*  31-47  31-49
        31-52*  31-53   31-66   31-82   31-84*  55-116#
GENPAR  11-130  16-7#
GENREG  12-382  12-420  55-125  55-128#
GETADR  10-29   27-152  31-33#
GETBUF  11-97   11-133  14-45#
GETID   10-28   27-151  30-10#
```

```
        13-504  22-384#
LINE7A  12-466  12-600  22-419#
LINE8   12-254  12-329  13-503  22-439#
LINEN3  22-228  59-10#
LINEO5  22-319  59-27#
LINEP5  22-315  59-26#
LING6   22-347  59-30#
LINM3   22-125  59-4#
LINM4   22-251  59-19#
LINN3   22-131  59-6#
LINOCT  12-666  12-666  12-666  12-666  12-666  12-666  12-666  13-174  13-177  13-180  13-315  13-315  13-315
        13-324  13-324  13-324  13-360  13-363  13-366  13-372  13-378  22-114  22-154  22-157  22-187  22-191  22-253
        22-283  22-286  22-307  22-307  22-317  22-321  22-449#
LINP3   22-209  59-8#
LINP5   22-293  59-24#
LINR6   22-370  59-31#
LINS3   22-164  22-225  59-9#
LINS4   22-254  59-20#
LINS5   22-302  59-25#
LINSS3  22-173  59-18#
LINST3  22-168  59-17#
LINT3   22-239  59-12#
LINUO6  22-353  59-32#
LINW3   22-155  59-16#
LINX4   22-257  59-21#
LINX5   22-308  59-64#
LKPAR    9-174#
LODEV    9-245  27-113  27-128  60-17#
LODPAR  11-131  20-8#
LSC      4-224#
LSTAD    7-0#    9-152*  9-155   9-156
LSTHDR  32-94   60-52#
M.DP40  36-77#  36-98
M.DP41  36-82   36-89#
M.DP42  36-88   36-93#
M.DP44  36-100  36-105#
M.DP50  36-75   36-109#
M.DPID  36-46   36-66#
MAIN    11-3#   11-194  15-67   35-7
MAIN1   10-41   11-20   11-49#
MAIN2   11-59   11-79#
MAINDA  11-13#
MANTER  27-164  32-10#
MASK     7-0#   12-222* 12-239  12-285* 12-353* 12-401* 12-445* 12-480* 12-507* 12-583* 12-614* 12-632* 13-492  13-497
MATCH   13-54   13-214#
MAXCYL  17-3    18-36   19-28   29-69*  29-92   29-95*  55-54#  55-55   55-56   55-57   55-58   55-59   55-63   61-54
        61-54   61-54   61-54   61-54   61-54   61-54   61-54
MAXER    7-0#   34-38
MAXSEC  17-25   18-27   19-19   19-33   29-71*  29-104  29-107* 55-58#  61-54   61-54   61-54   61-54   61-54   61-54
        61-54   61-54
MAXTRK  17-14   18-31   19-23   19-30   29-70*  29-98   29-101* 55-56#  61-54   61-54   61-54   61-54   61-54   61-54
        61-54   61-54
MBASN   39-60   60-55#
MCPE     4-85#
MCPEMX  54-157# 54-;01  54-;47
MDPE     4-105#
MERR1   31-94   60-46#
```

CROSS REFERENCE TABLE (CREF V01-05)

PACK	7-0#	9-99*	15-63*	27-166	28-49*	28-54*	28-59*	28-65*					
PACKSN	39-78	60-54#											
PAR	4-148#												
PAR1	61-3	61-18#											
PAR10	61-13	61-27#											
PAR11	61-7	61-28#											
PAR14	61-8	61-29#											
PAR15	61-10	61-30#											
PAR16	61-9	61-31#											
PAR19	61-5	61-32#											
PAR2	61-4	61-19#											
PAR20	61-11	61-33#											
PAR21	61-12	61-34#											
PAR3	61-6	61-20#											
PAR4	61-21#	61-54	61-54	61-54	61-54	61-54	61-54	61-54	61-54				
PAR5	61-22#	61-54	61-54	61-54	61-54	61-54	61-54	61-54	61-54				
PAR6	61-23#	61-54	61-54	61-54	61-54	61-54	61-54	61-54	61-54				
PAR7	61-24#	61-54	61-54	61-54	61-54	61-54	61-54	61-54	61-54				
PAR8	61-25#	61-54	61-54	61-54	61-54	61-54	61-54	61-54	61-54				
PAR9	61-26#	61-54	61-54	61-54	61-54	61-54	61-54	61-54	61-54				
PARENT	9-197	29-89	33-8#										
PARER	12-153	12-501#											
PARLST	9-169*	9-196	61-3#										
PASSES	7-0#	35-1	61-5										
PAT	4-101#												
PATTER	7-0#	13-219	17-69	17-71	61-6								
PCLOCK	7-0#	23-39*	23-44*										
PERIOD	24-94	24-103	24-109	24-114	24-120	24-128	24-142	24-145	24-148	24-151	24-158	39-65	60-37#
PFECH	43-1	43-1#											
PFECH1	43-1	43-1#											
PFECH2	43-1	43-1#											
PFECH3	43-1	43-1#											
PFECH4	43-1	43-1#											
PFTSTN	43-1	43-1#	43-1*										
PGE	4-107#												
PGM	4-135#												
PIP	4-139#												
PIRQ	4-74#												
PIRQVE	4-74#												
POPQUE	54-421	54-472	54-589	54-891	54-926	54-970	54-=70#						
POSER	12-426	12-458#											
PR0	4-74#												
PR1	4-74#												
PR2	4-74#												
PR3	4-74#												
PR4	4-74#	27-13											
PR5	4-74#												
PR6	4-74#	9-28											
PR7	4-74#	35-1											
PROCES	11-172	12-3#											
PRTBAD	12-221	12-258	12-612	13-333#									
PRTIM	12-42	12-104#											
PS	4-74	4-74#	9-93*	27-13*	27-80*	35-1*	39-43	39-44*	39-105*	52-15*	54-213	54-214*	54-243*	54-251*
	54-345	54-346*	54-399*	54-409	54-457*								
PSEL	4-84#												
PSW	4-74#												
PUNSAF	12-32	12-47#											

```
PWRFLG   9-174    9-212    9-216*   52-1*    52-2#
PWRVEC   4-74#    9-25*    9-25*    52-1*    52-1*    52-1*    52-1*    52-1*    52-1*
QCNT    54-<53#  54-<95   54-=19*  54-=34   54-=36*  54-=56   54-=70*
QDRV0   54-<60   54-<67   54-<72   54-<86#
QDRV1   54-<63   54-<70   54-<73   54-<86#
QDRV2   54-<63   54-<70   54-<74   54-<86#
QDRV3   54-<63   54-<70   54-<75   54-<86#
QDRV4   54-<63   54-<70   54-<76   54-<86#
QDRV5   54-<63   54-<70   54-<77   54-<86#
QDRV6   54-<63   54-<70   54-<78   54-<86#
QDRV7   54-<63   54-<70   54-<79   54-<86#
QINPT   54-<60#  54-=21   54-=38*  54-=39*  54-=40   54-=42*
QOUTPT  54-<67#  54-=21*  54-=59   54-=72   54-=73*  54-=74*  54-=75   54-=77*
QSTART  54-<72#  54-=01   54-=06   54-=42   54-=77
QSTOP   54-<73#  54-=40   54-=75
QTERM   54-<80   54-<87#
QUES    34-8     60-3#
R6       4-74#    9-25     9-25*    9-25*
R7       4-74#
RANCYL  16-29    17-3#
RANDOM   7-0#    16-28    61-12
RANDWC   7-0#    17-37    61-7
RANPAT  17-59#
RANSEC  17-25#
RANSIZ  16-41    17-36#   17-44
RANTRK  17-14#
RANXIT  17-60    17-63#
RATIO    7-0#    10-8*    16-13    61-8
RD.ADR  54-;03*  54-;04*  54-;06#  58-3
RD.RM   54-283   54-313   54-319   54-537   54-561   54-575   54-713   54-767   54-841   54-851   54-867   54-:22   54-;01#  54-;54
        54-;68   54-;74   54-<07
RD.RM1  54-;05#  54-;33
RD.RM2  54-;01*  54-;32#
RD.RM3  54-;12   54-;24   54-;34#
RD.RM4  54-;17   54-;36#
RD.WRD  54-;07#  54-;08   58-3     58-4
RDCHR   41-1     53-1#
RDDAT    4-255#   7-0      15-54    20-32    31-43
RDHD     4-256#   7-0      13-463
RDLIN    9-181    27-20    27-89    32-34    33-19    53-1#    63-26    63-37
RDY      4-81#
READDR  12-338   12-380   12-418   13-414   20-18    21-24    22-200   23-14#
READHD  12-339   12-381   12-419   13-459#
READIN   4-245#
RECAL    4-240#  13-408   13-428
RECAL0  13-428#  27-145
RECALT  12-467   12-601   12-643   13-406#
REDAPK  27-58    28-54#
RELBUF  11-182   14-86#
RELSE    4-242#
REPHD   11-191   60-22#
REPLZ   24-92    39-13#
RESREG  14-164   27-77    36-49    50-1     51-1     53-1#    54-252   54-393   54-396   54-458   54-686   54-698   54-:01   54-<24
        54-=10
RESVEC   4-74#
RETRY    7-0#    12-204*  12-227*  12-243*  12-244   12-244   12-249*  12-290*  12-312*  12-317*  12-320*  12-321   12-321   12-354*
        12-402*  12-446*  12-481*  12-508*  12-582*  12-613*  12-633*  13-487*  13-499*  13-500   13-500   22-368
```

```
RHEXT    9-109*   9-118*   9-122*   9-128   54-165#  54-<21
RM05    13-395   13-409   13-429   13-445   13-465   14-175   31-46    54-345#
RMADR    9-85*    9-208   54-162#  54-232   54-352   54-475   54-496   54-518   54-693   54-:20   54-;03   54-;09   54-;13   54-;27
        54-;60   54-;64   54-;96   54-<37   63-44
RMAS    54-180#  54-318*  54-765   54-895*  54-931*  54-954*  54-:50
RMBA    54-175#
RMBAE   54-194#
RMCS1   54-173#  54-274*  54-281   54-307   54-416*  54-417   54-491   54-513   54-556   54-576   54-587   54-604   54-625   54-714
        54-768   54-827   54-858   54-903   54-972   54-;55
RMCS2    9-209*  54-177#  54-233*  54-273*  54-275   54-365*  54-415*  54-476*  54-497*  54-519*  54-608   54-653   54-669   54-680*
        54-712*  54-839*  54-939*  54-953*  54-:21*  54-,10   54-;65   54-;97*  54-<03   54-<38*  54-<43
RMCS3   54-195#
RMDA    54-176#  54-483   54-509   54-525
RMDB    54-182#  54-<01
RMDC    54-187#  54-487   54-500   54-532
RMDS    54-178#  54-314   54-842   54-868   54-940   54-:23
RMDT    54-184#  54-284
RMEC1   54-191#
RMEC2   54-192#  54-<11
RMER1   12-382   12-420   54-179#  54-320   54-852   54-941   54-;69
RMER2   54-190#  54-942
RMERRS  54-23#   54-216   54-805   54-810   54-820   54-940*  54-941*  54-942*  54-943*  58-2     58-2     58-2     58-2
RMHR    54-188#
RMINIT   9-206   54-212#
RMLA    54-181#
RMMR1   54-183#
RMMR2   54-189#  54-943
RMOF    54-186#  54-311   54-538   54-542   54-562   54-566
RMR      4-147#
RMSN    54-185#
RMTMR   26-49    54-986#
RMVEC    9-86*   54-163#  54-229   54-231   54-346
RMWC    54-174#
RNOP     4-238#
RTC      4-244#  13-394
RTNCTR  13-393#
RTURN   35-1     35-3#
S       22-207   22-216   59-7#
SATPOW  52-1     52-9#
SAVEFG   9-207*  54-117#  54-591   54-718   54-928
SAVER1   7-0#    13-21*   13-81*   13-131   13-139   13-143*
SAVER5   7-0#    13-22*   13-82*   13-140   13-144*
SAVREG  14-150   27-12    36-39    50-1     51-1     53-1#    54-212   54-348   54-408   54-640   54-692   54-989   54-:93   54-<94
SC      54-697   54-765#
SC04     4-199#
SC1      4-197#
SC10     4-200#
SC11    54-840   54-848   54-919#
SC12    54-801   54-814   54-939#
SC13    54-791   54-794   54-950#
SC2      4-198#
SC20     4-201#
SC3     54-780#  54-785
SC4     54-783#  54-817   54-831   54-833   54-884   54-905   54-917   54-937   54-979
SC5     54-781   54-789#
SC6     54-813   54-835#
SC6A    54-811   54-861#
```

```
SW06      4-74      4-74#     20-23
SW07      4-74      4-74#     13-168
SW08      4-74      4-74#     35-1      35-1 *
SW09      4-74      4-74#
SW1       4-74#
SW10      4-74#     22-8
SW11      4-74#
SW12      4-74#
SW13      4-74#     22-11     42-1
SW14      4-74#
SW15      4-74#     12-74     22-404    22-431
SW2       4-74#
SW3       4-74#     13-64     13-333
SW4       4-74#
SW5       4-74#     22-183
SW6       4-74#
SW7       4-74#
SW8       4-74#
SW9       4-74#
SWR       6-0#      9-25      9-25      9-25*     9-25*     9-25*     9-30      9-214     11-184    12-17     12-74     13-64     13-168    13-333
          15-36     16-9      20-23     22-8      22-11     22-80     22-150    22-183    22-404    22-431    27-52     27-62     34-36     35-1
          35-1      35-1      39-103    41-1      41-1      41-1*     42-1      42-1      42-1      42-1      42-1      52-1      52-1*
SWREG     5-1#      9-25      9-30      41-1      41-1      41-1
SWTIM     12-38     12-81#
SYSTAT    9-218     60-18#
T         22-202    22-212    59-5#
TAB       27-159    60-7#
TAB.XY    5-9#      6-0
TABLE     29-82     61-39#
TABLE0    61-39     61-54#
TABLE1    61-42     61-54#
TABLE2    61-42     61-54#
TABLE3    61-42     61-54#
TABLE4    61-42     61-54#
TABLE5    61-42     61-54#
TABLE6    61-42     61-54#
TABLE7    61-42     61-54#
TAP       4-192#
TBITVE    4-74#
TD        54-696    54-704#
THEAD     16-28#
THEAD1    16-31     16-35#
TIMER     54-130#   54-451*   54-597*   54-661*   54-707*   54-836*   54-882*   54-934*   54-951*   54-975*   54-992    54-994*   54-:63*   54-:71*
          54-:78*
TKVEC     4-74#     41-1*     41-1*
TPVEC     4-74#
TRAPVE    4-74#     9-25*     9-25*
TRE       4-86#
TRFER     12-133    12-574#
TRKLMT    7-0#      13-104    19-23     23-32     29-70     29-85     29-86     29-127*   29-130*   31-40     32-60
TRNSWT    54-79#    54-473*   54-631    54-636*   54-647    54-667*   54-678*   54-709    54-710*   54-:34    54-:46*
TRTVEC    4-74#
TST1      9-19#
TYLIST    32-42     32-93#
TYMBA     24-96     27-158    39-60#
TYPACK    24-98     27-160    39-78#
TYPCK     39-79#
```

```
$$CMRE   5-579#
$$CMTM   5-579#   6-0
$$ESCA   4-74#
$$NEWT   4-74#    9-19
$$SET   53-1     53-1     53-1     53-1     53-1     53-1     53-1     53-1     53-1     53-1     53-1     53-1#    53-2
$$SETM   9-25     9-25#
$$SKIP   4-74#
.$ACT1   4-49#    5-5
.$APTB   4-52#    6-0      6-0#
.$APTH   4-52#    5-8
.$APTY   4-52#   47-1
.$CATC   4-50#    5-1
.$CMTA   4-50#    5-579
.$DB2D   4-51#   50-1
.$DB2O   4-51#   51-1
.$EOP    4-52#   35-1
.$ERRO   4-50#   42-1
.$ERRT   4-50#   43-1
.$POWE   4-52#   52-1
.$RAND   4-52#   48-1
.$READ   4-49#   41-1
.$SAVE   4-51#   49-1
.$SIZE   4-51#   62-1
.$TRAP   4-51#   53-1
.$TYPD   4-50#   46-1
.$TYPE   4-49#   44-1
.$TYPO   4-50#   45-1
.EQUAT   4-49#    4-74
.HEADE   4-49#    4-56
.SETUP   4-49#    4-261
.SWRHI   4-49#    4-57
.SWRLO   4-49#    4-57#    4-58     4-59     4-60     4-62     4-64     4-69     4-71     4-72
A       25-2#    25-16    25-17    25-18    25-19    25-20
CKLHR    4-10#   40-87    40-95
CKDIG    4-20#   32-53    32-60    32-67    33-23
CKNUM    4-33#
COMMEN   4-74#
ENDCOM   4-74#
ERENTR  41-13#   42-1
ERRCAL  53-7#    54-772   54-830   54-832   54-;18   54-;77
ERROR    4-74#
ESCAPE   4-74#
GETPRI   4-74#   62-1
GETSWR   4-74#    9-30     9-30#
MORETA   5-11#    6-0
MULT     4-74#
NEWTST   4-74#    9-19
POP      4-74#   12-427   12-462   13-145   21-50    22-109   24-33    28-44    29-46    29-113   29-131   30-29    31-98    31-107
        32-80    32-128   39-34    46-1     47-1     47-1     48-1     49-1     52-1     52-1
PUSH     4-74#   12-422   13-138   21-21    22-35    22-388   22-396   23-40    24-7     28-28    29-10    29-122   30-10    31-33
        31-37    32-10    32-93    46-1     47-1     47-1     47-1     48-1     49-1     52-1     52-1
REPORT   4-74#
SETPRI   4-74#   41-1
SETTRA  53-1     53-1     53-1     53-1     53-1     53-1     53-1     53-1     53-1     53-1     53-1     53-1#    53-2
SETUP    4-74#    9-25
SKIP     4-74#
```

```
SLASH    4-74#
STARS    4-74#   5-5     5-8     5-8     5-8     6-0     6-0     6-0     9-19    22-1    22-3    35-1    37-4    38-4
        41-1    41-1    41-1    41-1    41-1    42-1    43-1    44-1    45-1    46-1    47-1    48-1    49-1    50-1
        51-1    52-1    52-1    53-1    54-15   62-1
SWRSU    4-74#   9-25    9-25#
TRMTRP  53-1#   53-3
TYPBIN   4-74#
TYPDEC   4-74#  24-102  24-113  24-141  24-144  24-147  24-150  35-1
TYPNAM   4-74#   9-30
TYPNUM   4-74#
TYPOCS   4-74#   9-220   11-63   12-72   24-74   27-156  31-96   34-10   34-31   35-1    35-1
TYPOCT   4-74#  41-1    43-1    43-1
TYPTXT   4-74#   9-6     9-44    9-55    9-61    24-68   24-89   24-101  24-104  24-112  24-115  24-123  24-139  24-140
        24-143  24-146  24-149  24-152
UPCODE  51-2#   52-1
XXEP    34-53#  35-1
```