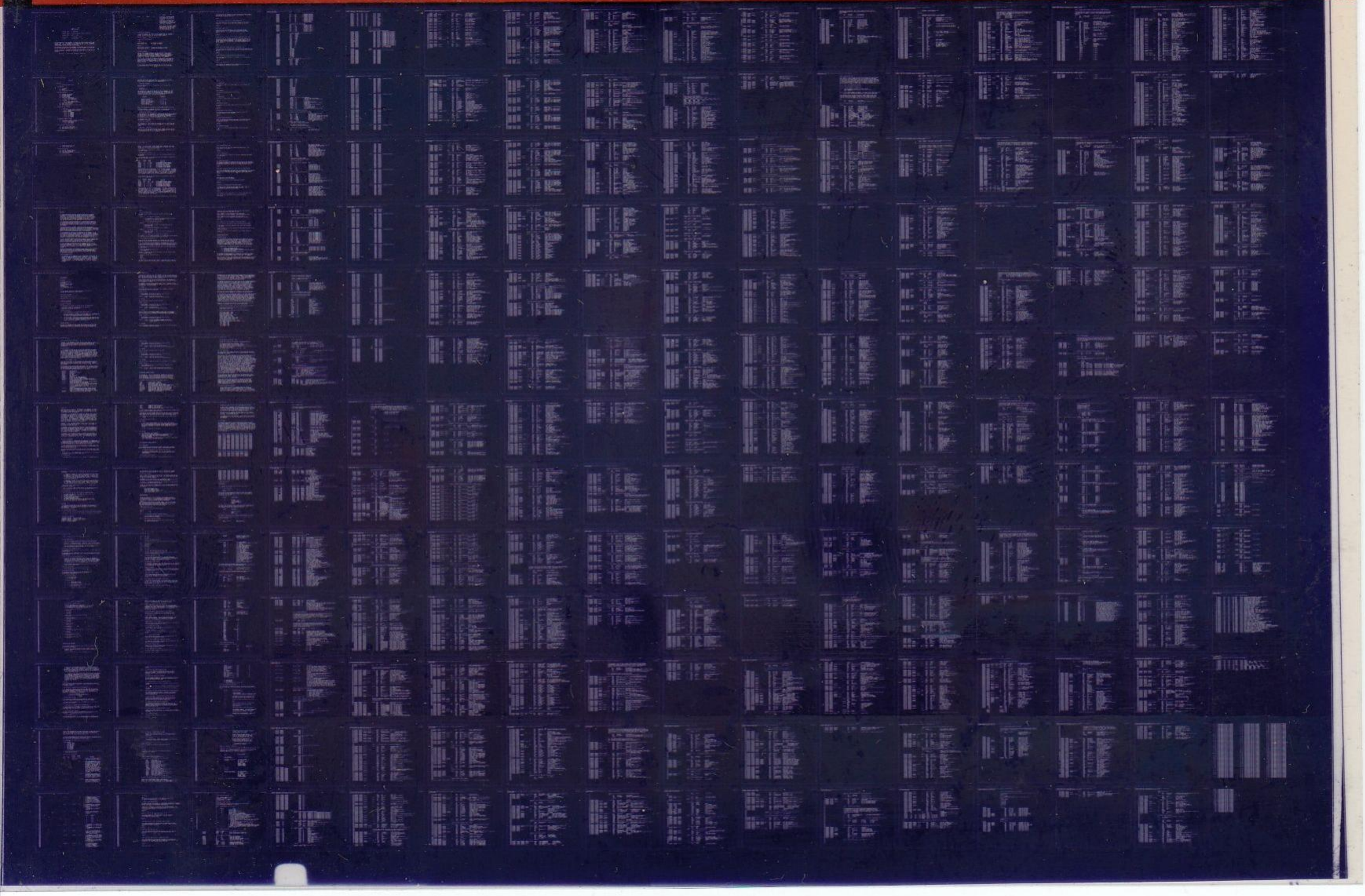
RM80

RM80 PERF EXER
CZRNAA0

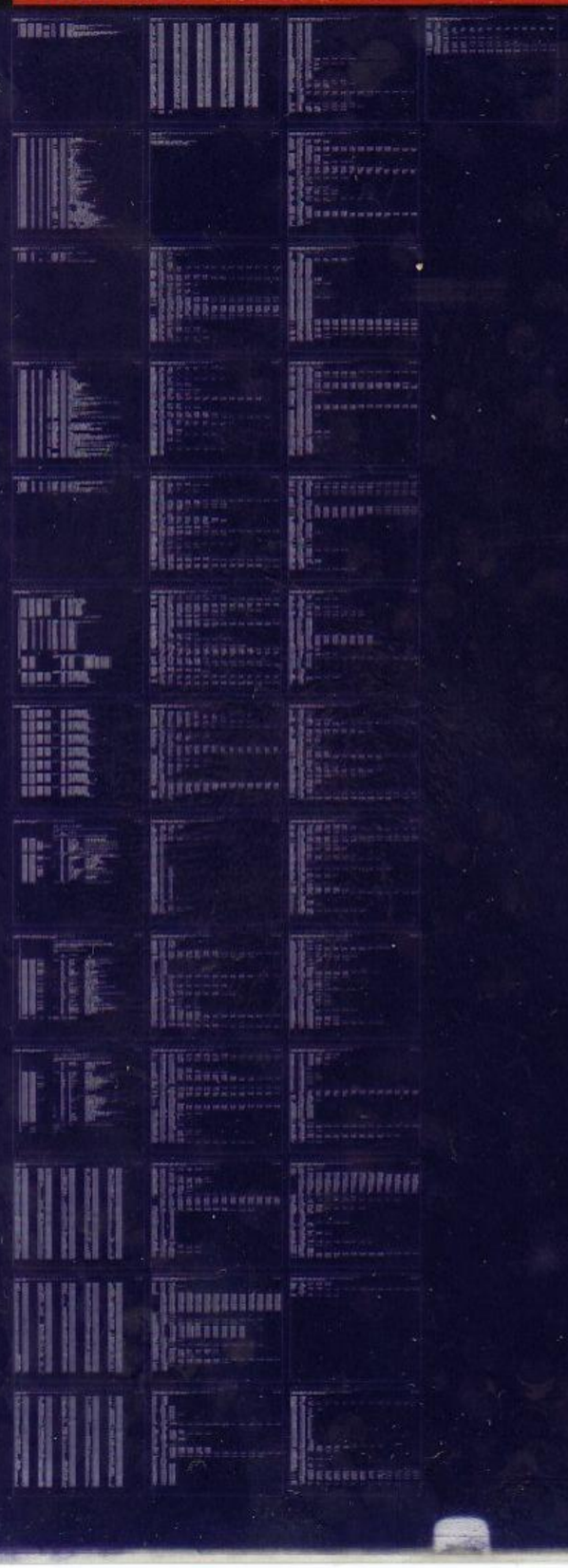
AH-T105A-MC
FICHE 1 OF 2



RM80

RM80 PERF EXER
CZRNAAO

AH-T105A-MC
FICHE 2 OF 2



```
                        .REM    @
```

IDENTIFICATION

PRODUCT CODE:	AC-T104A-MC
PRODUCT NAME:	CZRNAA0 RM80 PERFORMANCE EXERCISER
PRODUCT DATE:	APRIL 1, 1982
MAINTAINER:	CX DIAGNOSTIC GROUP
AUTHOR:	MIKE LEAVITT

CONTENTS

1. ABSTRACT

THE RM80 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 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 THRCUGH THE KEYBOARD, DYNAMIC PROGRAM OPTIONS ARE SELECTED VIA SWITCH REGISTER SETTINGS AND ERRORS ARE REPORTED ON THE CONSOLE TERMINAL.

1.1 GENERAL DOCUMENT NOTES

A. IN REFERENCE TO ALL NUMBERS IN THIS 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-11/70 PROCESSOR
16K MEMORY
KW11-L OR KW11-P CLOCK
PROGRAM LOADING DEVICE
TERMINAL
RH70 CONTROLLER
1 TO 8 RM80 DISK DRIVES

2.2 MEDIA

THE PERFORMANCE EXERCISER PROGRAM REQUIRES FORMATTED DISK AND MUST BE FORMATTED IN (16 BIT) MODE.

2.3 PRELIMINARY PROGRAMS

RM80 DISKLESS TEST, PART 1 & 2

RM80 FUNCTIONAL TEST, PART 1, 2 & 3 (OPTIONAL PART 4)

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), CHANGE 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 PROMPT FE CYLINDER MESSAGE DURING AUTO TEST MODE
SW<01>=1	INHIBIT DATA COMPARE AFTER READ COMMAND, W/O ERROR
SW<00>=1	"READ ONLY" OR LOCKED "READ ONLY" MODE

SW<00>

WHEN THIS 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.

HOWEVER, IF THE OPERATOR WISHES TO MAKE SURE THAT THERE IS ABSOLUTELY NO WRITTING ON THE DISK AT ANYTIME, THE PROGRAM MAY BE LOCKED IN "READ ONLY" MODE.

THE PROGRAM CAN BE LOCKED INTO "READ ONLY" MODE BY STARTING OR RESTARTING THE PROGRAM WITH SW0 SET(1). AFTER THE PROGRAM HAS BEEN LOCKED IN "READ ONLY" MODE, SW0 WILL HAVE NO FURTHER EFFECT UNTIL THE LOCKED MODE IS RELEASED. TO RELEASE THE PROGRAM FROM THE LOCKED "READ ONLY" CONDITION, THE PROGRAM MUST BE STARTED OR RESTARTED WITH SW0 CLEAR(0).

FOR EXAMPLE, THE PROGRAM IS STARTED AT ADDRESS 200 AND LOCKED IN "READ ONLY" MODE. A SEQUENTIAL READ WILL OCCUR TO ALL ON-LINE DRIVES. UPON COMPLETION OF THE READ, THE PROGRAM GOES INTO A "READ ONLY" TESTING MODE AND WILL STAY THAT WAY UNTIL RELEASED.

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: 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. 1.75 HRS. (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 RM80 IS THE XXDP LOADING DEVICE.

AUTOMATIC MODE OR CHAIN MODE (MONITOR)

1. IF SW02 OF THE SWITCH REGISTER IS SET(1) WHEN THE PROGRAM IS STARTED AT 200 OR 204, THE OPERATOR IS ALLOWED TO CHOOSE BETWEEN EXERCISING THE USER PORTION OF THE DISK OR JUST THE FE CYLINDERS (SEE SECTION 4.1). IF SW02 IS CLEAR(0), ALL THE INPUT DIALOGUE IS BYPASSED AND THE TEST IS PERFORMED ON THE FE CYLINDERS ONLY.

2. 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

```
343                                     = 0  ALLOW TTY CONSOLE OUTPUT
344
345                                 BIT 4 TO BIT 0 ARE NOT USED
346
347                             3.  SWITCH 1 (SOFTWARE SWITCH REGISTER)
348                                 IF ENVIRONMENT MODE BIT 7 (SIZING BIT ) IS SET TO 1,
349                                 THE SOFTWARE SWITCH REGISTER WILL BE USED, INSTEAD
350                                 OF THE HARDWARE TTY CONSOLE SWITCH REGISTER.
351
352                             4.  SWITCH 2 (USER SWITCH REGISTER)
353                                    NOT USED
354
355                             5.  CPU OPTIONS
356                                    NOT USED
357
358                             6.  MEMORY TYPES 1-4 AND MAX MEMORY ADDRESSES
359                                    NOT USED
360
361                             7.  INTERRUPT VECTOR 1:
362                                    USED WHEN ENVIRONMENT MODE BIT 7 = 1;DEFAULT = 254
363
364                             8.  BUS PRIORITY 1:
365                                    NOT USED.
366
367                             9.  INTERRUPT VECTOR 2:
368                                    NOT USED
369
370                             10. BUS PRIORITY 2:
371                                    NOT USED
372
373                             11. BASE ADDRESS:
374                                    USED WHEN ENVIRONMENT MODE BIT 7 = 1;DEFAULT = 176700
375
376                             12. DEVICE MAP:
377                                    NOT USED
378
379                             13. CONTROLLER DESCRIPTOR WORDS:
380                                    NOT USED
381
382                             14. CONTROLLER DESCRIPTOR WORDS:
383                                    NOT USED
384
385
386
387
388
389                     4.      CONTROLLING THE PROGRAM
390                             ----------------------
391
392                             THE FOLLOWING KEYBOARD CONVENTIONS ARE USED BY THE KEYBOARD ENTRY
393                             ROUTINES IN THE PROGRAM:
394
395                             A.  TO DELETE AN INCORRECT CHARACTER FROM AN ENTRY STRING, TYPE A
396                                 'RUBOUT'. TYPING A RUBOUT WILL DELETE SUCESSIVE CHARACTERS
397                                 FROM THE INPUT.
398
399                             B.  TO DELETE AN ENTIRE LINE, TYPE A 'CONTROL U' (^U).
```

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

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

4.1 PARAMETERS

THE FOLLOWING QUESTIONS ARE ASKED TO DETERMINE HOW THE RM80 WILL BE TESTED. THE DEFAULT ANSWERS TO THESE QUESTIONS ARE ALWAYS AS DOCUMENTED HERE.

'DO YOU WISH TO EXERCISE ONLY FE CYLINDERS (L) Y ?'

A 'Y' ANSWER WILL PROCEED WITH EXERCISING ONLY THE FE CYLINDERS AND SKIP THE FOLLOWING QUESTION. A 'N' ANSWER WILL PROCEED TO NEXT WARNING MESSAGE AND QUESTION (UNLESS THE EXERCISER IS IN 'READ ONLY'' MODE(SW0=1), IN WHICH CASE THE WARNING WILL BE OMITTED BUT THE QUESTION WILL BE ASKED).

'! CUSTOMER DATA WILL BE OVERWRITTEN !'

'ARE YOU SURE (L) N ?'

A 'Y' ANSWER WILL PROCEED WITH EXERCISING THE WHOLE DISK. A 'N' ANSWER WILL PROCEED WITH EXERCISING ONLY THE FE CYLINDERS.

IF ONLY THE FE CYLINDERS ARE TO BE EXERCISED, THE FOLLOWING MESSAGE WILL BE PRINTED.

'* EXERCISER WILL OPERATE ON FE CYLINDERS ONLY *'

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

'LOCKED IN READ ONLY MODE'

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

'CHANGE PARAMETERS (L) N ?'

THE OPERATOR MUST ENTER A 'Y' IF PARAMETER ENTRIES ARE TO BE MADE. ANY OTHER CHARACTER IS ACCEPTED AS A 'N' ENTRY. THE PROGRAM WILL

IDENTIFY THE PARAMETER BY THE NAME GIVEN BELOW, DISPLAY THE CURRENT VALUE OF THE PARAMETER AND WAIT FOR THE ENTRY. THE PROGRAM WILL TYPE 'INVALID ENTRY' IF THE ENTRY IS NOT CORRECT AND WAIT FOR A CORRECT ENTRY TO BE TYPED. (SEE SECTION 4.1.1)

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

THE FOLLOWING IS AN EXAMPLE DRIVE STATUS PRINTOUT:

```
 'DRIVE STATUS:
   0    ONLINE   RM80
   1    LOAD DEVICE
   2    OFFLINE  RM80
   3    NOT PRESENT
   4    NOT PRESENT
   5    NOT AN RM80
   6    NOT PRESENT
   7    NOT PRESENT'
```

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

4.1.1 KEYBOARD ENTRY PARAMETERS

NAME	BASE	DEFAULT VALUE	VALUE RANGE	FUNCTION
WRDCNT	10.	7936. (SEE NOTE)	6 - 7936.	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 7936.(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 7936. 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 DISK LOCATION. IF PARAMETER = 0, PRINT ERROR MESSAGES ASSOCIATED WITH BAD DISK 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 HDA SERIAL NUMBER HAS CHANGED SINCE THE LAST TIME IT WAS READ. NOTE: IF THE HDA SERIAL NUMBER HAS CHANGED, THIS MOST LIKELY MEANS THAT THE HDA OR DRIVE HAD BEEN REPLACED WHILE THE DRIVE WAS DEASSIGNED.

4.1.2 CHANGE DEVICE ADDRESSES

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 RM80	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 HDA SERIAL NUMBER IS IN THE FOLLOWING FORMAT:

'MBA S/N: X HDA 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 DECIMAL SERIAL NUMBER READ FROM THE DEC144 BAD SECTOR FILE ON THE HDA. 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.

THE FOLLOWING TABLE VALUES ARE USED WHEN THE PROGRAM IS EXERCISING THE USER PORTION OF THE DISK.

NAME	DEFAULT VALUE	VALUE RANGE	FUNCTION
MINCYL	0	0 - 558.	THE MINIMUM CYLINDER ADDRESS
MAXCYL	558.	0 - 558.	THE MAXIMUM CYLINDER ADDRESS
MINTRK	0	0 - 13.	THE MINIMUM TRACK ADDRESS
MAXTRK	13.	0 - 13.	THE MAXIMUM TRACK ADDRESS
MINSEC	0	0 - 30.	THE MINIMUM SECTOR ADDRESS
MAXSEC	30.	0 - 30.	THE MAXIMUM SECTOR ADDRESS

WHEN OPERATING WITH CYLINDER 558. AS THE MAXIMUM CYLINDER, THE TRACK ADDRESS MAY BE SELECTABLE FROM 0 - 13.. IF YOU REMEMBER, CYLINDER 558. TRACK 13. IS THE DEC144 TRACK. DURING THE TEST, IF THE PROGRAM TRIES TO ACCESS CYLINDER 558. TRACK 13., THE PROGRAM AUTOMATICALLY SELECTS THE NEXT USABLE TRACK ADDRESS. THIS ALLOWS ALL THE REST OF THE CYLINDERS, TRACK 0 AND 13. TO BE ADDRESSED BY THE OPERATOR.

THE FOLLOWING TABLE VALUES ARE USED WHEN THE PROGRAM IS EXERCISING ON THE 'FE' CYLINDERS ONLY.

NAME	DEFAULT VALUE	VALUE RANGE	FUNCTION
MINCYL	559.	559. - 560.	THE MINIMUM CYLINDER ADDRESS
MAXCYL	560.	559. - 560.	THE MAXIMUM CYLINDER ADDRESS
MINTRK	0	0 - 13.	THE MINIMUM TRACK ADDRESS
MAXTRK	13.	0 - 13.	THE MAXIMUM TRACK ADDRESS
MINSEC	0	0 - 30.	THE MINIMUM SECTOR ADDRESS
MAXSEC	30.	0 - 30.	THE MAXIMUM SECTOR ADDRESS

WHEN OPERATING ON THE FE CYLINDERS ONLY, THE TRACK ADDRESS MAY BE SELECTABLE FROM 0 - 13. . IF YOU REMEMBER, CYLINDER 559. TRACK 0 IS THE SKIP SECTOR FILE AND CYLINDER 559. TRACK 1 IS USED FOR THE ALTERNATE DEC144 TRACK. DURING THE TEST, IF THE PROGRAM TRIES TO ACCESS CYLINDER 559. TRACK 0 OR 1, THE PROGRAM AUTOMATICALLY SELECTS THE NEXT USABLE TRACK ADDRESS. THIS ALLOWS CYLINDER 560. TRACK 0 AND 1 TO BE ADDRESSED BY THE OPERATOR.

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

```
742                                     TYPEOUT:
743
744                                       'ENTER BAD SECTR ADRS:'
745
746                                     THE FORMATS USED TO ENTER BAD SECTOR ADDRESS LOCATIONS ARE AS
747                                     FOLLOWS:
748
749                                       EXAMPLE 1:    CYL,TRK,SEC= C,T,S<CR>
750
751                                             A.  THE PROGRAM WILL INHIBIT DATA ERROR MESSAGES FOR ERRORS
752                                                 OCCURRING AT THE SPECIFIED ADDRESS.
753
754                                             B.  LEADING ZEROS ARE NOT REQUIRED.
755
756                                       EXAMPLE 2:    CYL,TRK,SEC= C,T<CR>
757
758                                             A. WHEN THIS FORMAT IS USED, THE ENTIRE TRACK WILL BE
759                                                CONSIDERED BAD.
760
761                                             B. DATA ERRORS WILL BE HANDLED AS IN 'EXAMPLE 1'.
762
763                                       EXAMPLE 3:    CYL,TRK,SEC= C<CR>
764
765                                             A. WHEN THIS FORMAT IS USED, THE ENTIRE CYLINDER WILL BE
766                                                CONSIDERED BAD
767
768                                             B. DATA ERRORS WILL BE HANDLED AS IN 'EXAMPLE 1'.
769
770                                     IF CONTROL C (^C) IS TYPED AS AN ENTRY, ALL CURRENT BAD SECTOR ENTRIES
771                                     WILL BE LOST AND THE FOLLOWING MESSAGE WILL BE TYPED.
772
773                                       '* ALL CURRENT ENTRIES LOST *'
774
775                                     AFTER TYPING THE PREVIOUS MESSAGE, THE PROGRAM WILL WAIT FOR THE
776                                     OPERATOR TO ENTER ANOTHER BAD SECTOR AS IN THE PREVIOUS EXAMPLES.
777
778                                     IF 'L' IS TYPED FOR AN INPUT CHARACTER, THE PROGRAM WILL TYPE A LIST
779                                     OF DEC144 BAD SECTORS, AND THE MANUALLY ENTERED BAD SECTORS, WHICH ARE
780                                     STORED IN THE DRIVE PARAMETER TABLE(DPB) FOR THAT PARTICULAR DRIVE.
781
782                                     IF THERE ARE NO BAD SECTORS IN THE DPB TABLE, THE FOLLOWING MESSAGE
783                                     WILL BE TYPED:
784
785                                       'DEC144 AND MANUAL BAD SECTOR LIST
786                                        * NO ENTRIES *'
787
788                                     HOWEVER, IF THERE ARE ENTRIES IN DPB TABLE, THE LIST WILL BE TYPED
789                                     IN THE FOLLOWING FORMAT:
790
791                                       'DEC144 AND MANUAL BAD SECTOR LIST
792                                        8,8,3
793                                        16,13
794                                        256
795                                        500,1,29'
796
797                                     THE ABOVE LIST OF BAD SECTORS, ENTRY 1 INDICATES THAT CYLINDER 8.,
798                                     TRACK 8., SECTOR 3 IS THE BAD SECTOR. ENTRY 2 INDICATES THAT ON
```

CYLINDER 16., TRACK 13., ALL THE SECTORS ARE BAD (ENTIRE TRACK IS BAD). ENTRY 3 INDICATES THAT ON CYLINDER 256., ALL TRACKS AND SECTORS ARE BAD (ENTIRE CYLINDER IS BAD). ENTRY 4 INDICATES THAT CYLINDER 500., TRACK 1, SECTOR 29. IS THE BAD SECTOR.

AFTER TYPING EITHER OF THE TWO PREVIOUS MESSAGES, THE PROGRAM WILL RETURN TO WAIT FOR MORE ENTRIES TO BE MADE INTO THE BAD SECTOR TABLE, AS IN EXAMPLES 1, 2 AND 3.

TO TERMINATE THE BAD SECTOR ADDRESS ENTRY, YPE A 'CARRIAGE RETURN' IN RESPONSE TO THE ENTRY REQUEST OR TERMINATE THE ENTRY WITH A 'PERIOD' FOLLOWED BY A 'CARRIAGE RETURN'.

4.2.1 'T' COMMAND

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

FORMAT: TN<CR>

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

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

4.2.2 'D' COMMAND

USED TO DEASSIGN A DRIVE(S) BEING EXERCISED.

FORMAT: DN<CR>

N = DRIVE NUMBER. MAY BE 0 TO 7 OR 'A'. ENTRY MUST BE 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:  W0<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:  R0<CR> - READ THE DATA ON DRIVE 0.
                                              RA<CR> - READ THE DATA ON ALL AVAILABLE DRIVES.

                            4.2.6   'WT' COMMAND

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

                                    FORMAT: WTN<CR>

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

                                    EXAMPLE: WT0<CR> - WRITE A DATA PATTERN AND TEST DRIVE 0
                                             WTA<CR> - WRITE A DATA PATTERN AND TEST ALL DRIVES


                    5.      PERFORMANCE SUMMARY TYPEOUT
                            ---------------------------

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

                            THE SUMMARY TYPEOUT CONTAINS THE FOLLOWING FIELDS:

                            'TIME'          ELAPSED TIME OF PROGRAM
                            'DRIVE'         DRIVE NUMBER - DRIVE TYPE
                            'PASS'          PRESENT PASS COUNT FOR THE DRIVE
                            'MBA S/N'       HARD WIRED MASSBUS ADAPTER SERIAL NUMBER(RMSN)
                            'HDA S/N'       SERIAL NUMBER READ FROM THE DEC144 FILE
                            'WT OFLOW'      NUMBER OF TIMES 'WRDS WRITN' HAS OVERFLOWED
                            'WRDS WRITN'    TOTAL NUMBER OF WORDS WRITTEN BY THE DRIVE
                            'RD OFLOW'      NUMBER OF TIMES 'WRDS READ' HAS OVERFLOWED
                            'WRDS READ'     TOTAL NUMBER OF WORDS READ BY THE DRIVE
                            'SEEKS'         NUMBER OF SEEK OPERATIONS THE DRIVE PERFORMED
                            'SOFT'          NUMBER OF SOFT DATA ERRORS
```

'HARD' NUMBER OF HARD DATA ERRORS
'SKI' NUMBER OF 'SKI' ERRORS
'MISP' NUMBER OF PROGRAM DETECTED POSITIONING ERRORS
'OTHER' TOTAL ERRORS OF OTHER TYPES

ALL DATA TRANSFER COUNTS, SEEK COUNTS AND ERROR COUNTS ARE ACCUMULATIVE AND WILL NOT BE CLEARED AFTER EACH PASS.

TO CALCULATE THE TOTAL NUMBER WORDS READ OR WRITTEN, TAKE THE OVERFLOW COUNT (RD OFLOW OR WT OFLOW), MULTIPLY IT BY 2,147,483,647., THEN ADD THAT NUMBER TO THE WORDS READ OR WRITTEN (WRDS READ OR WRDS WRITN).

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

5.2 SOFT/HARD ERROR DEFINITIONS

5.2.1 HARD ERRORS

A. A 'DTE' (DRIVE TIMING ERROR) OR A 'DCK' (DATA CHECK ERROR) WHICH OCCURS DURING A READ DATA OR A READ HEADER & DATA OPERATION AND IS NOT CORRECTABLE OR DOES NOT BECOME CORRECTABLE AFTER THE PROGRAM HAS PERFORMED THE COMPLETE RETRY SEQUENCE ON THE BAD SECTOR.

THE RETRY SEQUENCE IS 16. RE-READS.

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 BY THE BAD SECTOR TABLE, THE PROGRAM WILL INHIBIT THE ERROR REPORT FOR THAT ERROR.

DATA CHECK ERRORS ( )
WRITE CHECK ERROR ('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.

```
1027                                    EM3     NOT USED
1028
1029                                    EM4     NOT USED
1030
1031                                    EM5     ADDRESS PLUG CHANGE BIT SET
1032
1033                                            THE 'OPE' BIT WAS SET WHEN THE INDICATED DRIVE INTERRUPTED.
1034
1035                                    EM6     NOT USED
1036
1037                                    EM10    NOT USED
1038
1039                                    EM11    NOT USED
1040
1041                                    EM12    NOT USED
1042
1043                                    EM13    OPERATION NOT COMPLETED WITHIN TIME LIMIT
1044
1045                                            THE DRIVE DID NOT COMPLETE THE OPERATION WITHIN 10. SECONDS
1046                                            AFTER THE OPERATION WAS INITIATED.
1047
1048                                    EM14    DRIVE WENT OFFLINE
1049
1050                                            THE DRIVE WENT OFFLINE DURING THE INDICATED OPERATION.
1051                                            (THE 'MOL' BIT BECAME ZERO.)  THE PROGRAM WILL AUTOMATICALLY
1052                                            DEASSIGN THE DRIVE.  THE OPERATOR MUST REASSIGN THE DRIVE
1053                                            WITH THE 'T' COMMAND TO RE-INITIATE TESTING.
1054
1055                                    EM15    NO RESPONSE TO PORT REQUEST
1056
1057                                            THE PROGRAM IS TESTING A DUAL PORT DRIVE WHICH HAS NOT SWITCHED
1058                                            TO THE REQUESTING PORT WITHIN 15. SECONDS AFTER PORT REQUEST
1059                                            TO THE DRIVE FROM THE REPORTING PORT.
1060
1061                                    EM20    HEADER CRC ERROR
1062
1063                                            A HEADER CRC ERROR WAS DETECTED AT THE INDICATED DISK ADDRESS.
1064                                            THE CONTENTS OF THE HEADER ARE DISPLAYED.  THE OPERATION WILL
1065                                            BE RETRIED 3 TIMES.
1066
1067                                    EM21    DATA CHECK ('DCK') ERROR
1068
1069                                            A DATA CHECK ERROR WAS DETECTED AT THE INDICATED SECTOR.
1070                                            THE FULL RETRY SEQUENCE WILL BE INITIATED
1071                                            FOR THE SECTOR IN ERROR IF THE ECC HARD ERROR ('ECH) BIT
1072                                            IS SET.
1073
1074                                    EM22    WRITE CHECK ERROR - DATA CHECK ('DCK') SET
1075
1076                                            A WRITE CHECK ERROR OCCURRED AND THE DATA CHECK ('DCK') BIT
1077                                            WAS SET.  IF 'ECH' IS NOT SET, THE OPERATION WILL BE RETRIED
1078                                            UP TO 3 TIMES; IF THE 'ECH' BIT IS SET, THE OPERATION WILL
1079                                            BE RETRIED UP TO 16. TIMES.
1080
1081                                    EM23    WRITE CHECK ERROR - DATA CHECK ('DCK') NOT SET
1082
1083                                            A WRITE CHECK ERROR OCCURRED AND 'DCK' WAS NOT SET.  THE
```

```
1084                    WORDS WHICH CAUSED THE ERROR ARE DISPLAYED IN THE ERROR
1085                    MESSAGE.  THE OPERATION WILL BE RETRIED 3 TIMES.
1086
1087            EM24    HEADER READ ERROR - 'FMT' BIT DROPPED
1088
1089                    A WRITE DATA, WRITE CHECK DATA, OR A READ DATA WAS BEING
1090                    PERFORMED AND A 'FMT' ERROR OCCURRED.  THE PROGRAM RE-READ THE
1091                    HEADER OF THE ERROR SECTOR AND THE 'HCRC' BIT WAS SET.  THE
1092                    CONTENTS OF THE HEADER ARE DISPLAYED.  THE OPERATION WILL
1093                    BE RETRIED 3 TIMES.
1094
1095            EM25    HEADER READ ERROR - HEADER COMPARE ('HCE') ERROR
1096
1097                    SIMILAR TO EM24, EXCEPT THAT THE 'HCE' ERROR BIT WAS
1098                    SET INITIALLY.  THE OPERATION WILL BE RETRIED 3 TIMES.
1099
1100            EM26    FORMAT ERROR ('FER')
1101
1102                    FORMAT ERROR OCCURRED.  WHEN THE HEADER WAS RE-READ, THE
1103                    'HCRC' BIT WAS NOT SET.  THE CONTENTS OF THE HEADER ARE
1104                    DISPLAYED.  THE OPERATION WILL BE RETRIED 3 TIMES.
1105
1106            EM27    HEADER COMPARE ('HCE') ERROR
1107
1108                    SIMILAR TO EM26 EXCEPT THAT THE 'HCE' BIT WAS SET INITIALLY.
1109                    THE OPERATION WILL BE RETRIED 3 TIMES.
1110
1111            EM30    MISCELLANEOUS DRIVE ERROR
1112
1113                    THIS MESSAGE IS GIVEN FOR THE FOLLOWING ERROR BITS:
1114                     'AOE', 'RMR', 'ILF', OR 'ILR'
1115
1116            EM31    OPERATION INCOMPLETE ('OPI') ERROR
1117
1118                    AN OPERATION INCOMPLETE ERROR OCCURRED AT THE INDICATED
1119                    SECTOR.
1120
1121            EM32    DRIVE TIMING ('DTE') ERROR
1122
1123                    DRIVE TIMING ERROR OCCURRED ON THE INDICATED SECTOR.  THE
1124                    OPERATION WILL BE RETRIED 3 TIMES.
1125
1126            EM33    PARITY ('PAR') ERROR AFTER OPERATION STARTED
1127
1128                    THE 'PAR' BIT WAS SET WHEN THE OPERATION WAS COMPLETED.  THE
1129                    OPERATION WILL BE RETRIED 3 TIMES.
1130
1131            EM34    WRITE CLOCK FAILURE ('WCF')
1132
1133                    A WRITE CLOCK FAILURE OCCURRED DURING THE OPERATION.  THE
1134                    OPERATION WILL BE RETRIED 3 TIMES.
1135
1136            EM35    INVALID ADDRESS ('IAE') ERROR
1137
1138                    AN INVALID ADDRESS ERROR OCCURRED DURING THE OPERATION.
1139
1140            EM36    WRITE LOCK ('WLE') ERROR
```

```
1141
1142                                    A WRITE OPERATION WAS ATTEMPTED BUT THE DRIVE WAS WRITE
1143                                    LOCKED.
1144
1145                            EM40    RH CONTROLLER OR UNIBUS TRANSFER ERROR
1146
1147                                    'TRE' IS SET IN THE RH CONTROL REGISTER AND NO DRIVE
1148                                    ERROR HAS OCCURRED.  THE OPERATION WILL BE RETRIED 3
1149                                    TIMES IF THE ERROR WAS CAUSED BY 'DLT', 'UPE', 'MXF',
1150                                    OR 'MDPE'.
1151
1152                            EM41    BUS ADDRESS OR WORD COUNT INCORRECT
1153
1154                                    NO DRIVE ERROR OCCURRED BUT EITHER THE BUS ADDRESS INDICATES
1155                                    THAT AN INCORRECT NUMBER OF WORDS WERE TRANSFERED OR THE
1156                                    WORD COUNT REGISTER IS NOT ZERO.
1157
1158                            EM42    DATA COMPARE ERRORS - NO DRIVE ERROR DETECTED
1159
1160                                    NO SUBSYSTEM ERROR WAS SIGNALED; HOWEVER, THE DATA DOES NOT
1161                                    COMPARE.
1162
1163                            EM43    CAN'T MATCH DATA READ WITH A PATTERN - UNDEFINED DATA PATTERN
1164
1165                                    THE DATA IN THE BUFFER DOES NOT MATCH ANY OF THE STANDARD
1166                                    PATTERNS.
1167
1168                            EM44    ERROR BIT(S) SET, BUT NO ERROR SIGNALED BY THE RH CONTROLLER
1169
1170                                    THE OPERATION COMPLETED NORMALLY; HOWEVER, THE PROGRAM
1171                                    FOUND EITHER ERROR BITS IN THE RM SET OR ERROR BITS IN
1172                                    THE RH CONTROLLER SET.
1173
1174                            EM45    ECC LOGIC FAILURE - POSITION REGISTER VALUE NOT VALID
1175
1176                                    DURING 'DCK' ERROR PROCESSING, THE CONTENTS OF THE ECC
1177                                    POSITION REGISTER (RMEC1) WAS NOT VALID. THE POSITION
1178                                    REGISTER WAS EITHER 0 OR GREATER THAN 010040.
1179
1180                            EM46    BUS ADDRESS OR WORD COUNT NOT CONSISTENT
1181
1182                                    THE PROGRAM WAS PROCESSING AN ERROR AND FOUND THAT THE
1183                                    NUMBER OF WORDS TRANSFERED AS INDICATED BY THE BUS ADDRESS
1184                                    REGISTER DOES NOT AGREE WITH THE TRANSFER COUNT FROM THE
1185                                    WORD COUNT REGISTER.
1186
1187                            EM47    ECC LOGIC FAILURE - PATTERN REGISTER IS ZERO
1188
1189                                    DURING 'DCK' ERROR PROCESSING, THE CONTENTS OF ECC PATTERN
1190                                    REGISTER (RMEC2) WAS NOT VALID. THE PATTERN REGISTER CONTAINED
1191                                    ALL ZEROS.
1192
1193                            EM50    SEEK INCOMPLETE ERROR
1194
1195                                    THE DRIVE SIGNALED EITHER 'SKI' ERROR.
1196
1197                            EM51    NOT USED
```

```
1198
1199                            EM52    ECH ERROR - UNCORRECTABLE ECC ERROR
1200
1201                                    DURING 'DCK' ERROR PROCESSING, THE 'ECH' BIT WAS SET IN
1202                                    RMER1, WHICH INDICATES THAT DATA CHECK CANNOT BE ECC
1203                                    CORRECTED.
1204
1205                            EM60    DEVICE UNSAFE
1206
1207                                    THE INDICATED DRIVE UNSAFE ERROR OCCURRED; THE ERROR WAS
1208                                    CLEARED BY A 'DRIVE CLEAR' INSTRUCTION.
1209
1210
1211                    7.2     DETAIL ERROR LINES
1212
1213                            THE LINE NUMBERS GIVEN BELOW ARE FOR REFERENCE ONLY.
1214
1215                            LINE 1
1216                            ------
1217
1218                            HH:MM:SS
1219
1220                            'HH:MM:SS' IS THE TIME SINCE THE PROGRAM WAS STARTED.
1221                            (HOURS, MINUTES, SECONDS)
1222
1223                            LINE 2
1224                            ------
1225
1226                            'PRSNT COMMAND= XXXX  PREV COMMAND= YYYY'
1227
1228                            MNEMONICS USED FOR THE COMMANDS ARE DEFINED BELOW:
1229
1230                                    SEEK    - SEEK (OCTAL 5)
1231                                    RECAL   - RECALIBRATE (OCTAL 7)
1232                                    DRVCLR  - DRIVE CLEAR (OCTAL 11)
1233                                    RELSE   - RELEASE (OCTAL 13)
1234                                    OFFSET  - OFFSET (OCTAL 15)
1235                                    RTC     - RETURN TO CENTERLINE (OCTAL 17)
1236                                    READIN  - READIN PRESET (OCTAL 21)
1237                                    PACK    - PACK ACKNOWLEDGE (OCTAL 23)
1238                                    SEARCH  - SEARCH (OCTAL 31)
1239                                    *GETREG - GET REGISTERS (OCTAL 41)
1240                                    *SETFMT - SET FORMAT (ECI OR HCI) (OCTAL 43)
1241                                    *SELDRV - SELECT DRIVE (OCTAL 45)
1242                                    WCKD    - WRITE CHECK DATA (OCTAL 51)
1243                                    WCKHD   - WRITE CHECK HEADER & DATA (OCTAL 53)
1244                                    WRTDAT  - WRITE DATA (OCTAL 61)
1245                                    WRTHD   - WRITE CHECK HEADER & DATA (OCTAL 63)
1246                                    RDDAT   - READ DATA (OCTAL 71)
1247                                    RDHD    - READ HEADER & DATA (OCTAL 73)
1248
1249                            * SPECIAL RM DRIVER COMMAND (NOT A CONTROLLER COMMAND)
1250
1251                            (DISPLAY OF THE RH/RM REGISTERS IN TWO GROUPS:
1252                            RMCS1, RMCS2, RMDS, RMER1, RMER2, RMEC1 AND RMEC2 FORM THE FIRST
1253                            GROUP:  ALL THE OTHER REGISTERS ARE IN THE SECOND GROUP.
1254                            IF SW<05> IS SET, ONLY THE REGISTERS IN THE FIRST GROUP WILL BE
```

DISPLAYED.)

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

'* ERROR AT BAD TRACK/SECTOR'

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

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

LINE 3

ERROR AT CXXX TYY SZZ PREV ADDR= CUUU TVV SWW

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

LINE 4

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

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

LINE 5

START CYL= XXX END CYL= YYY

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

LINE 6

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

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

LINE 7

RMBA= XXXX RMWC= YYYY

```
1312                    THIS LINE GIVES THE CONTENTS OF THE RH CONTROLLER BUFFER ADDRESS
1313                    REGISTER AND THE RH CONTROLLER WORD COUNT REGISTER.  THIS LINE IS
1314                    NOT PRINTED IF SW<05> IS NOT SET.
1315
1316                    LINE 8
1317                    -------
1318
1319                    START CYL= XXX  START TRK= YY  START SECTOR= ZZ
1320
1321                    THIS LINE IDENTIFIES THE STARTING DISK ADDRESS OF THE PRESENT
1322                    OPERATION.  CYLINDER, TRACK, AND SECTOR VALUES ARE DECIMAL.
1323
1324                    LINE 9
1325                    -------
1326
1327                    RMDA= XXXX  RMCA= YYYY
1328
1329                    THIS LINE GIVES THE CONTENTS OF THE RM TRACK AND SECTOR
1330                    ADDRESS REGISTER AND THE CONTENTS OF THE DESIRED CYLINDER
1331                    ADDRESS REGISTER.  THIS LINE IS NOT PRINTED IF SW<05> IS NOT
1332                    SET.
1333
1334                    LINE 10
1335                    -------
1336
1337                    BUFFER ADDR= XXXX  WRD CNT= YYYY  ACTUAL NUMBR WRDS XFRD= ZZZZ
1338
1339                    THIS LINE GIVES THE STARTING ADDRESS OF THE BUFFER USED FOR THE
1340                    CURRENT DATA TRANSFER OPERATION, ITS SIZE(WORD COUNT), AND THE
1341                    ACTUAL NUMBER OF WORD TRANSFERED.  THE STARTING ADDRESS OF THE
1342                    BUFFER IS IN OCTAL, THE WORD COUNT AND WORDS TRANSFERED VALUE
1343                    ARE IN DECIMAL.
1344
1345                    LINE 11
1346                    -------
1347
1348                    EXPCTD DATA= XXXX  RECEVD DATA= YYYY  WORD POS= ZZZ
1349
1350                    THIS LINE GIVES THE EXPECTED DATA, THE RECIEVED DATA FROM THE DISK,
1351                    AND THE LOCATION OF THE WORD IN THE SECTOR. THE WORD POSITION IS IN
1352                    DECIMAL.
1353
1354                    LINE 12
1355                    -------
1356
1357                    HEADER CONTENTS OF ERROR SECTOR= XXXX XXXX XXXX XXXX
1358
1359                    THIS LINE GIVES THE CONTENTS OF THE HEADER OF THE SECTOR WHICH
1360                    GAVE THE ERROR.
1361
1362                    LINE 13
1363                    -------
1364
1365                    RMEC1= XXXX  RMEC2= YYYY
1366
1367                    THIS LINE WILL BE PRINTED AFTER A SUCESSFUL RETRY OF A SECTOR
1368                    WHICH BECAME ECC CORRECTABLE DURING RETRY.
```

LINE 14

ECC CORRECTABLE

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

LINE 15

READ CORRECTLY

THE SECTOR IN ERROR WAS READ WITHOUT ERROR.

LINE 16

ECC CORRECTABLE

THE SECTOR IN ERROR BECAME ECC CORRECTABLE

LINE 17

CORRECTED ON X RETRY

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

LINE 18

UNCORRECTABLE AFTER X RETRIES

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

LINE 19

DIFFERENT ERROR DURING RETRY

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

LINE 20

DATA COMPARISON ERRORS

A PRINT JT OF THE DATA COMPARISON ERRORS FOLLOW THIS LINE.

LINE 21

TOTAL COMPARE ERRORS= XXXX

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

LINE 22

THE DATA COMPARED OK

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

LINE 23

ECC CORRECTION RESULTS

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

LINE 24

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

THIS IS AN INFORMATIONAL LINE WHICH WILL BE PRINTED FOR 'DCK' ERRORS WHICH ARE ECC CORRECTABLE OR WHICH BECOME ECC CORRECTABLE DURING RETRY. 'XXX' IS THE WORD OFFSET VALUE FROM 'RMEC1' AND IS IN DECIMAL.

LINE 25

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

THE DATA ERROR WAS NOT IN DATA TRANSFERED TO MEMORY.

LINE 26

CONTENTS OF THE ERROR SECTOR (REPORTED ABOVE)

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

LINE 27

TOTAL ERRORS:X WOFL:N WRDS WRITN: YYYY ROFL:N WRDS READ: ZZZZ

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

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

'WOFL' NUMBER OF TIMES 'WRDS WRITN' HAS OVERFLOWED
'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 RM80, AND IS ONLINE. THE ASSIGNMENT ROUTINE
THEN ISSUES A 'READIN PRESET' INSTRUCTION, SETS 'FMT16', AND ISSUES
A 'RECALIBRATE' INSTRUCTION.

```
1540                    PARAMETERS FOR THE OPERATION ARE SELECTED AND A BUFFER IS ASSIGNED. IF
1541                    THE OPERATION IS A WRITE OR WRITE CHECK COMMAND, THE ASSIGNED BUFFER
1542                    WILL BE FILLED WITH THE SELECTED PATTERN.  (WRITE CHECK COMMANDS ARE
1543                    ISSUED AFTER EACH WRITE COMMAND.  THE WRITE CHECK COMMAND USES THE
1544                    PARAMETERS SELECTED FOR THE PRECEDING WRITE COMMAND.)  CONTROL
1545                    IS THEN PASSED TO THE COMMAND INITIATION ROUTINE.
1546
1547                    THE COMMAND INITIATION ROUTINE FIRST LOOKS AT THE CYLINDER ADDRESS OF
1548                    THE REQUESTED OPERATION. IF THE DRIVE MUST SEEK TO ANOTHER CYLINDER TO
1549                    PERFORM THE OPERATION, THE PROGRAM ISSUES A SEARCH INSTRUCTION
1550                    TO THE DRIVE WITH A 'TARGET' SECTOR WHICH IS 5 SECTORS EARLIER
1551                    THAN THE 'TRANSFER' SECTOR.  (THIS ALLOWS THE PROGRAM TO INITIATE
1552                    OPERATIONS ON ANOTHER DRIVE WHILE THE PRESENT DRIVE, OR OTHER
1553                    DRIVES, ARE SEARCHING FOR 'TARGET' SECTORS.  ALL SEEKS ISSUED BY
1554                    THE PROGRAM ARE IMPLIED SEEK SEARCH OPERATIONS.)  WHEN A SEARCHING
1555                    DRIVE FINDS THE 'TARGET' SECTOR AND INTERRUPTS, THE PROGRAM THEN
1556                    ISSUES THE REQUESTED COMMAND TO THE DRIVE THAT INTERRUPTED.
1557
1558                    WHEN THE DATA TRANSFER OPERATION IS COMPLETE, THE DRIVE REGISTERS
1559                    ARE STORED AND A DATA TRANSFER IS INITIATED FOR A WAITING DRIVE.
1560
1561                    IF THE OPERATION HAS BEEN COMPLETED NORMALLY, THE SAVED DRIVE REGISTERS
1562                    ARE CHECKED TO VERIFY THAT NO ERROR BITS ARE SET; THE RH CONTROLLER
1563                    BUS ADDRESS AND WORD COUNT ADDRESS REGISTERS ARE CHECKED TO VERIFY
1564                    THAT THE CORRECT NUMBER OF WORDS HAVE BEEN TRANSFERED AND THAT THE
1565                    TWO REGISTERS ARE CONSISTENT WITH EACH OTHER; AND IF THE COMMAND WAS
1566                    A READ COMMAND, THE DATA BUFFER IS COMPARED.  WHEN THIS SEQUENCE IS
1567                    COMPLETED, THE DRIVE IS RETURNED TO THE ASSIGNED, INACTIVE LIST. THE
1568                    PROGRAM THEN INITIATES A DATA TRANSFER ON A WAITING DRIVE AND
1569                    RESELECTS AND REINITIATES ANOTHER OPERATION ON THE RELEASED DRIVE.
1570
1571                    ERRORS WHICH OCCUR ARE PROCESSED IN THE FOLLOWING ORDER.  MULTIPLE
1572                    ERRORS WILL BE REPORTED AS THE FIRST ERROR TYPE CHECKED.
1573
1574                    A.  ERRORS REPORTED FOR OPERATIONS WHICH HAVE NOT COMPLETED NORMALLY.
1575
1576                        OPERATION NOT COMPLETED WITHIN TIME LIMIT - EM13
1577                        DRIVE WENT OFFLINE - EM14
1578
1579                    B.  ERRORS REPORTED FOR OPERATIONS WHICH COMPLETE NORMALLY.
1580
1581                        CORRECTABLE UNSAFE - EM60
1582                        DRIVE TIMING ERROR - EM32
1583                        DATA CHECK ERROR - EM21
1584                        WRITE CHECK WITH DCK SET - EM22
1585                        HEADER CRC ERRORS - EM20
1586                        FORMAT ERRORS - EM24, EM26
1587                        HEADER COMPARE ERRORS - EM25, EM27
1588                        PROGRAM DETECTED POSITIONING ERROR - EM51
1589                        SEEK INCOMPLETE ERROR - EM50
1590                        WRITE CHECK WITHOUT 'DCK' SET - EM23
1591                        RH CONTROLLER OR UNIBUS TRANSFER ERROR - EM40
1592                        'OPI' ERROR - EM31
1593                        'PAR' ERROR - EM33
1594                        'WCF' ERROR - EM34
1595                        'IAE' ERROR - EM35
1596                        'WLE' ERROR - EM36
```

MISCELLANEOUS DRIVE ERROR - EM30

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

BUS ADDRESS OR WORD COUNT INCORRECT - EM41
DATA COMPARE ERRORS - NO DRIVE ERROR DETECTED - EM42
CAN'T MATCH DATA READ WITH A PATTERN - EM43
ERROR BIT(S) SET, BUT NO ERROR SIGNALED BY THE RH CONTROLLER - EM44
ECC LOGIC FAILURE - EM45
BUS ADDRESS OR WORD COUNT NOT CONSISTENT - EM46

8.2 DUAL PORT OPERATION

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

WHEN THE DUAL PORT HANDLER ROUTINE IN THE EXERCISER PROGRAM RECEIVES A REQUEST FOR A DRIVE, THE PROGRAM VERIFIES THAT THE DRIVE IS ONLINE. THE DRIVE IS SELECTED AND THE RMCS1 REGISTER IS READ TO TEST THE 'DVA" BIT. IF THE DRIVE IS IN NEUTRAL, THIS WILL SEIZE THE DRIVE. IF THE DRIVE IS SEIZED BY THE OTHER PORT, A DRIVE CLEAR COMMAND IS ISSUED TO THE DRIVE TO SET 'PORT REQUEST'. THE PROGRAM THEN CHECKS 'DVA' IN 'RMCS1'. IF THE DRIVE IS AVAILABLE AS INDICATED BY THE 'DVA' BIT, THE COMMAND SEQUENCE WILL BE INITIATED IN THE NORMAL MANNER (SEE SECTION 8.1 ABOVE). IF 'DVA' WAS NOT SET, THE PROGRAM MAKES AN ENTRY FOR THE DRIVE IN AN INTERNAL 'PORT REQUEST PENDING' TABLE AND STARTS A 15. SECOND TIMER FOR THE DRIVE. IF THE DRIVE HAS NOT SWITCHED TO THE REQUESTING SYSTEM WITHIN THE 15. SECOND INTERVAL, THE PROGRAM REPORTS A 'NO RESPONSE TO PORT REQUEST' ERROR. NORMALLY THIS ERROR MESSAGE INDICATES A FAILURE IN THE DUAL PORT CONTROL LOGIC IN THE DRIVE BEING TESTED; HOWEVER, UNDER CERTAIN CONDITIONS (E.G. MASSBUS PARITY ERRORS BEING REPORTED ON THE OTHER SYSTEM ON A TTY), THE OTHER PROCESSOR WAS UNABLE TO PROCESS THE DRIVE AFTER IT HAD REQUESTED THE DRIVE. THE OPERATOR MUST BE AWARE OF WHAT THE OTHER SYSTEM IS DOING AT ALL TIMES TO INTERPRET THE PORT RELATED ERROR MESSAGES PROPERLY.

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

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

8.3 SELECTION OF OPERATION VARIABLES

A. SECTOR ADDRESS SELECTION IS RANDOM BETWEEN THE VALUES IN 'MINSEC' AND 'MAXSEC'. TRACK ADDRESS SELECTION IS RANDOM BETWEEN THE VALUES IN 'MINTRK' AND 'MAXTRK'. CYLINDER ADDRESS SELECTION IS RANDOM BETWEEN 'MINCYL' AND 'MAXCYL'. IF A MINIMUM ADDRESS IS GREATER THAN THE CORRESPONDING MAXIMUM ADDRESS, THE

PROGRAM WILL SWAP 'MAX' AND 'MIN' ADDRESSES AND CONTINUE.

B. THE WORD COUNT IS RANDOMLY SELECTED BETWEEN 6 AND THE VALUE 'WRDCNT'. THIS IS NECESSARY AS THE PROGRAM REQUIRES 4 LOCATIONS IN THE DATA PORTION OF THE SECTOR TO BE ABLE TO MATCH THE DATA TO A PATTERN FOR DATA COMPARISON PURPOSES AND NEEDS 2 MORE LOCATIONS IF A READ HEADER & DATA COMMAND IS ISSUED.

C. THE DATA WRITTEN IS RANDOMLY SELECTED AMONG THE 15. STANDARD PATTERNS. THE PARAMETER 'PATTERN' ENABLES THE RANDOM PATTERN SELECTION, IF THIS PARAMETER IS 0.

D. THE COMMANDS ARE SELECTED RANDOMLY. WRITE CHECK DATA COMMAND IS PERFORMED ONLY IF THE PREVIOUS COMMAND WAS THE APPROPRIATE WRITE DATA COMMAND.

8.4 DATA PATTERNS

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

PAT 1	PAT 2	PAT 3	PAT 4	PAT 5	PAT 6	PAT 7	*PAT 8
000001	177776	000000	133331	052525	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 RM80, 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
                                  1             RM80
                                 -1             NOT AN RM80

                        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,RM80         ;MAKE THE CALL
                                PNTDPB                  ;ADDRESS OF DPB*
```

```
                        RETURN1                 ;RETURN IF QUEUE IS FULL
                        RETURN2                 ;RETURN IF REQUEST IS IN
                                                ;QUEUE OR THERE IS AN
                                                ;ERROR CONDITION


                    *DPB (DATA PARAMETER BLOCK)

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

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

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

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


9.4.1   EXAMPLE - WRITE 1000. WORDS

        1$:     JSR     R0,RM80                 ;CALL THE DRIVER
                WRTDPB                          ;DPB ADDRESS
                BR      1$                      ;WAIT FOR QUEUE IF FULL
        2$:     TST     WRTDPB+16               ;WAIT FOR COMMAND TO COMPLETE
                BEQ     2$
                BMI     ERROR1                  ;ERROR OCCURRED
                .
                .
                .

        WRTDPB: .BYTE   5                       ;DRIVE #5
                .BYTE   0                       ;
```

```
                        .BYTE   161             ;WRITE COMMAND
                        .BYTE   0               ;
                        .WORD   -1000.          ;WORD COUNT
                        .WORD   WRTBUF          ;BUFFER ADDRESS
                        .BYTE   3               ;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
9(3)(4)	SOFTWARE TIMEOUT OCCURRED ON THIS DRIVE
7	DONE
6(2)	ERROR OCCURRED DURING AN I/O OPERATION
5(2)	ERROR OCCURRED DURING AN OPERATION OTHER THAN I/O.
4(2)	CORRECTABLE UNSAFE CONDITION OCCURRED
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       RH70 INTERRUPT          *R4= RMCS1'S ADDRESS
                            OCCURRED (RHAS=0)

                    2       UNEXPECTED ATTENTION     R1= DRIVE NUMBER
                            OCCURRED                 R3= ATA BIT
                                                    *R4= RMCS1'S ADDRESS
                                                     R5= (RMAS)
                                                     RMERRS  =RMDS
                                                     RMERRS+2=RMER1
                                                     RMERRS+4=RMER2
                                                     RMERRS+6=RMMR2

                    3       NOT USED

                    4       NOT USED

                    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

                    * THIS IS THE ACTUAL UNIBUS ADDRESS (176700)

          @
```

```
 1                              ;*LAST REVISION 23-DEC-81
56
57                              .TITLE  CZRNAA0 RM80 PERF EXER
                                ;*COPYRIGHT (C) 1982
                                ;*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
                                ;*
58                              .SBTTL  OPERATIONAL SWITCH SETTINGS
                                ;*
                                ;*      SWITCH                  USE
                                ;*      ------          --------------------
                                ;*        15            HALT ON ERROR
                                ;*        13            INHIBIT ERROR TYPEOUTS
                                ;*        10            BELL ON ERROR
                                ;*         8            INHIBIT END OF PASS MESSAGES
59                              ;*         7            DISPLAY ALL DATA COMPARE ERRORS
60                              ;*         6            DON'T CHANGE PARAMETERS (LOOP ON PRESENT VALUES)
61                              ;*         5            A. PARTIAL REGISTER DISPLAY IF ERROR
62                              ;*                      B. NO ECC CORRECTION RESULTS DISPLAYED IF ERROR
63                              ;*         4            A. DO NOT CHECK FOR MAXIMUM ERROR COUNTS
64                              ;*                      B. DO NOT DROP DRIVE AT END OF TEST
65                              ;*         3            A. DISPLAY ERROR SECTOR IF 'DCK', 'DTE', OR 'WCF' ERROR
66                              ;*                      B. DISPLAY SECTOR IF 'DCK' ERR UNCORRECTABLE AFTER
57                              ;*                         28TH RETRY
68                              ;*                      C. IF DATA COMPARE ERROR & SW07 SET, DISPLAY
69                              ;*                         REMAINDER OF BUFFER
70                              ;*         2            A. DO NOT TYPE DRIVE STATUS AT PROGRAM START
71                              ;*                      B. DO NOT TYPE PERFORMANCE REPORT AFTER SPECIFIED TIME
72                              ;*                      C. PROMPT 'FE CYLINDER' MESSAGE IN AUTO(CHAIN) RUN MODE
73                              ;*         1            INHIBIT DATA COMPARE AFTER READ COMMAND, W/O ERROR
74                              ;*         0            READ ONLY MODE
75
76                              .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
```

BASIC 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                  PP0     = 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
```

BASIC DEFINITIONS

```
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
 77
 78                     .SBTTL  RH CONTROLLER REGISTERS
 79
 80                     ;CONTROL AND STATUS REGISTER 1 (RMCS1)
 81
 82      000100         IE      = 100                   ;INTERRUPT ENABLE (BIT #6)
 83      000200         RDY     = 200                   ;READY (BIT #7)
 84      000400         A16     = 400                   ;HIGH ORDER BUS ADDRESS BIT (BIT #8)
 85      001000         A17     = 1000                  ;HIGH ORDER BUS ADDRESS BIT (BIT #9)
 86      002000         PSEL    = 2000                  ;PORT SELECT (BIT #10)
 87      020000         MCPE    = 20000                 ;MASSBUSS PARITY ERROR (BIT #13)
 88      040000         TRE     = 40000                 ;TRANSFER ERROR (BIT #14)
 89                     ;SC     = 100000                ;SPECIAL CONDITION (BIT #15)
 90
 91                     ;WORD COUNT REGISTER (RMWC)
 92                     ;(EACH BIT IS CALLED BY BIT NUMBER)
 93
 94                     ;BUS ADDRESS REGISTER (RMBA)
 95                     ;(EACH BIT IS CALLED BY BIT NUMBER)
 96
 97                     ;CONTROL AND STATUS REGISTER 2 (RMCS2)
 98
 99      000001         US1     = 1                     ;UNIT SELECT (BIT #0)
100      000002         US2     = 2                     ;UNIT SELECT (BIT #1)
```

```
                        US4     = 4                     ;UNIT SELECT (BIT #2)
                        BAI     = 10                    ;BUS ADDRESS INCREMENT INHIBIT (BIT #3)
                        PAT     = 20                    ;MASSBUS PARITY TEST (BIT #4)
                        CLR     = 40                    ;CLEAR (BIT #5)
                        IR      = 100                   ;INPUT READY (BIT #6)
                        OR      = 200                   ;OUTPUT READY (BIT #7)
                        MDPE    = 400                   ;MASS BUS PARITY ERROR (BIT #8)
                        MXF     = 1000                  ;MISSED TRANSFER ERROR (BIT #9)
                        PGE     = 2000                  ;PROGRAM ERROR (BIT #10)
                        NEM     = 4000                  ;NON EXISTENT MEMORY (BIT #11)
                        NED     = 10000                 ;NON EXISTENT DRIVE (BIT #12)
                        UPE     = 20000                 ;UNIBUS PARITY ERROR (BIT #13)
                        WCE     = 40000                 ;WRITE CHECK ERROR (BIT #14)
                        DLT     = 100000                ;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      = 1                     ;GO BIT (BIT #0)
                        F0      = 2                     ;FUNCTION CODE BIT #1
                        F1      = 4                     ;FUNCTION CODE BIT #2
                        F2      = 10                    ;FUNCTION CODE BIT #3
                        F3      = 20                    ;FUNCTION CODE BIT #4
                        F4      = 40                    ;FUNCTION CODE BIT #5
                        DVA     = 4000                  ;DEVICE AVAILABLE (BIT #11)

                        ;DRIVE STATUS REGISTER (RMDS1) (#01)

                        OFFON   = 1                     ;OFFSET ON (BIT #0)
                        VV      = 100                   ;VOLUME VALID (BIT #6)
                        DRY     = 200                   ;DRIVE READY (BIT #7)
                        DPR     = 400                   ;DRIVE PRESENT (BIT #8)
                        PGM     = 1000                  ;PROGRAMABLE (BIT #9)
                        LBT     = 2000                  ;LAST BLOCK TRANSFERRED (BIT #10)
                        WRL     = 4000                  ;WRITE LOCK (BIT #11)
                        MOL     = 10000                 ;MEDIUM ON-LINE (BIT #12)
                        PIP     = 20000                 ;POSITIONING OPERATION IN PROGRESS (BIT #13)
                        ERR     = 40000                 ;COMPOSITE ERROR (BIT #14)
                        ATA     = 100000                ;ATTENTION ACTIVE (BIT #15)

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

                        ILF     = 1                     ;ILLEGAL FUNCTION (BIT #0)
                        ILR     = 2                     ;ILLEGAL REGISTER (BIT #1)
                        RMR     = 4                     ;REGISTER MODIFICATION REFUSED (BIT #2)
                        PAR     = 10                    ;PARITY ERROR (BIT #3)
                        FER     = 20                    ;FORMAT ERROR (BIT #4)
                        WCF     = 40                    ;WRITE CLOCK FAIL (BIT #5)
                        ECH     = 100                   ;ECC HARD ERROR (BIT #6)
                        HCE     = 200                   ;HEADER COMPARE ERROR (BIT #7)
                        HCRC    = 400                   ;HEADER CRC ERROR (BIT #8)
                        AOE     = 1000                  ;ADDRESS OVERFLOW ERROR (BIT #9)
                        IAE     = 2000                  ;INVALID ADDRESS ERROR (BIT #10)
```

```
RM REGISTERS

    158         004000          WLE     = 4000                  ;WRITE LOCK ERROR (BIT #11)
    159         010000          DTE     = 10000                 ;DRIVE TIMING ERROR (BIT #12)
    160         020000          OPI     = 20000                 ;OPERATION INCOMPLETE (BIT #13)
    161         040000          UNS     = 40000                 ;DRIVE UNSAFE (BIT #14)
    162         100000          DCK     = 100000                ;DATA CHECK ERROR (BIT 15)
    163
    164                         ;MAINTAINABILITY REGISTER (RMMR1)(#03)
    165
    166
    167                         ;ATTENTION SUMMARY PSEUDO-REGISTER (RMAS) (#04)
    168
    169         000001          AT0     = 1                     ;DEVICE 0 (BIT #0)
    170         000002          AT1     = 2                     ;DEVICE 1 (BIT #1)
    171         000004          AT2     = 4                     ;DEVICE 2 (BIT #2)
    172         000010          AT3     = 10                    ;DEVICE 3 (BIT #3)
    173         000020          AT4     = 20                    ;DEVICE 4 (BIT #4)
    174         000040          AT5     = 40                    ;DEVICE 5 (BIT #5)
    175         000100          AT6     = 100                   ;DEVICE 6 (BIT #6)
    176         000200          AT7     = 200                   ;DEVICE 7 (BIT #7)
    177
    178                         ;DESIRED SECTOR/TRACK ADDRESS REGISTER (RMDA) (#05)
    179
    180
    181                         ;DRIVE TYPE REGISTER (RMDT) (#06)
    182
    183         000001          DT00    = 1                     ;DRIVE TYPE NUMBER BIT 1
    184         000002          DT01    = 2                     ;DRIVE TYPE NUMBER BIT 2
    185         000004          DT02    = 4                     ;DRIVE TYPE NUMBER BIT 3
    186         000010          DT03    = 10                    ;DRIVE TYPE NUMBER BIT 4
    187         000020          DT04    = 20                    ;DRIVE TYPE NUMBER BIT 5
    188         000040          DT05    = 40                    ;DRIVE TYPE NUMBER BIT 6
    189         000100          DT06    = 100                   ;DRIVE TYPE NUMBER BIT 7
    190         000200          DT07    = 200                   ;DRIVE TYPE NUMBER BIT 8
    191         000400          DT08    = 400                   ;DRIVE TYPE NUMBER BIT 9
    192         004000          DRQ     = 4000                  ;DRIVE REQUEST REQUIRED (BIT #11)
    193         020000          MOH     = 20000                 ;MOVING HEAD (BIT #13)
    194         040000          TAP     = 40000                 ;TAPE DRIVE (BIT #14)
    195         100000          NSA     = 100000                ;NOT SECTOR ADDRESSED (BIT #15)
    196
    197                         ;LOOK-AHEAD REGISTER (RMLA) (#07)
    198
    199         000100          SC1     = 100                   ;SECTOR COUNT FIELD 0 (BIT #6)
    200         000200          SC2     = 200                   ;SECTOR COUNT FIELD 1 (BIT #7)
    201         000400          SC04    = 400                   ;SECTOR COUNT FIELD 2 (BIT #8)
    202         001000          SC10    = 1000                  ;SECTOR COUNT FIELD 3 (BIT #9)
    203         002000          SC20    = 2000                  ;SECTOR COUNT FIELD 4 (BIT #10)
    204
    205                         ;SERIAL NUMBER REGISTER (RMSN) (#10)
    206                         ;(EACH IS CALLED BY BIT NUMBER)
    207
    208                         ;OFFSET REGISTER (RMOF) (#11)
    209
    210         000001          OFFDIR  = 1                     ;OFFSET DIRECTION
    211         001000          SSEI    = 1000                  ;SKIP SECTOR ERROR INHIBIT (BIT #9)
    212         002000          HCI     = 2000                  ;HEADER COMPARE INHIBIT (BIT #10)
    213         004000          ECI     = 4000                  ;ERROR CORRECTION CODE INHIBIT (BIT #11)
    214         010000          FMT16   = 10000                 ;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)
000040                          SSE     = 40                    ;SKIP SECTOR ERROR (BIT #5)
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
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  003542            JMP     @#START1              ;;JUMP TO STARTING ADDRESS OF PROGRAM
2
3 000204    000137  003532            JMP     @#START         ;CHANGE THE RH ADDRESS
4
5                               .SBTTL  ACT11 HOOKS

                                ;;**************************************************************
                                ;HOOKS REQUIRED BY ACT11
            000210                      $SVPC=.                 ;SAVE PC
            000046                      .=46
  000046    031752                      $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
```

```
      0                                 .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
```

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

USER DEFINED TAGS

```
                                .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          DRVNO:  .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       ;'CONTROL C' FLAG
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  000000          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 ERROR 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  001056          CYLIMT: .WORD   558.    ;CYLINDER ADDRESS LIMIT
001424  000036          SECLMT: .WORD   30.     ;SECTOR ADDRESS LIMIT
```

USER DEFINED TAGS

```
001426  000015                  TRKLMT: .WORD   13.     ;TRACK ADDRESS LIMIT
001430  001057                  FE1:    .WORD   559.   ;1ST FE CYLINDER
001432  001060                  FE2:    .WORD   560.   ;2ND FE CYLINDER
001434  000000                  FEFLAG: .WORD   0      ;EXERCISE FE CYLINDERS ONLY=0, EXERCISE ENTIRE DISK=1
001436  000000                  DEC2:   .WORD   0      ;DECREMENT TRK/SEC ONCE=0, DECREMENT TRK/SEC TWICE=1
001440  000000                  RDONLY: .WORD   0      ;NOT READ ONLY=0, READ ONLY=1
001442  000000                  DRVPAR: .WORD   0      ;WHEN DRIVES ARE BEING ASSIGNED,
                                                       ;0=CHANGE DRIVE PARAMETERS
                                                       ;1=DO NOT CHANGE DRIVE PARAMETERS
001444  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 RM80.

                        .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

001446  002740                  ENDCON: .WORD   002740 ;(1.875 X 10^8 WORDS) OR (3 X 10^9 BITS) READ
001450  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

001452  015200                  ENDSEK: .WORD   015200 ;(1 X 10^6 SEEKS)
001454  000006                          .WORD   000006

001456  000031                  MAXER:  .WORD   25.    ;MAXIMUM ERRORS ALLOWED PER DRIVE
001460  000004                  CMPLMT: .WORD   4      ;NUMBER OF COMPARE ERRORS TYPED OUT
001462  017400                  WRDCNT: .WORD   7936.  ;MAXIMUM WORD COUNT (31. SECTORS)
001464  000000  000000          INTRVL: .WORD   0,0    ;FIRST WORD IS THE PERFORMANCE TYPEOUT INTERVAL
                                                       ;(IN MINUTES). SECOND WORD IS THE INTERVAL COUNTER.
001470  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].
001472  000000                  PATTERN:.WORD   0      ;IF EQ 0,RANDOMLY SELECT DATA PATTERN
                                                       ;IF NOT EQ 0,SELECT ONE SET OF PATTERN
                                                       ;POINTED BY THE 'PATTERN'.
001474  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
001476  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
```

COMMON PARAMETERS

```
                                                    ;7 -  1/7
001500  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)
001502  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.
001504  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.
001506  000000          RANDOM: .WORD   0           ;IF EQ TO 0, RANDOMLY SELECT DATA BLOCK
                                                    ;ADDRESS. IF NOT EQU 0,SEQUENTIALLY
                                                    ;SELECT DATA BLOCK ADDRESS
001510  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 HDA SERIAL NUMBER HAS CHANGED SINCE THE
                                                    ;LAST TIME IT WAS READ. (NOTE: IF THE SERIAL NO. HAS
                                                    ;CHANGED, THIS MOST LIKELY MEANS THAT THE HDA OR DRIVE
                                                    ;HAD BEEN REPLACED WHILE THE DRIVE WAS DEASSIGNED)

                        .SBTTL  VALUES FOR FIRST OPERATION

001512  000010          BEGPAT: .WORD   8.          ;STARTING PATTERN CODE [RANGE 1 - 15.]
001514  000004          BEGCOD: .WORD   4           ;STARTING COMMAND CODE [RANGE 0 - 5]
                                                    ;0 = WRITE CHECK DATA ('WCKD')
                                                    ;1 = WRITE CHECK HEADER & DATA ('WCHKHD' - NOT USED)
                                                    ;2 = WRITE DATA ('WRTDAT')
                                                    ;3 = WRITE HEADER & DATA ('WRTHD' - NOT USED)
                                                    ;4 = READ DATA ('RDDAT')
                                                    ;5 = READ HEADER & DATA ('RDHD')
001516  000400          BEGWC:  .WORD   256.        ;STARTING  WRD CNT [RANGE 6 - WRDCNT]

                        .SBTTL  TABLES, CONSTANTS, AND VARIABLE LOCATIONS

                        ;LIST OF DRIVES PERFORMING COMMANDS
001520  000000          ORDERQ: .WORD   0
001522  000000                  .WORD   0
001524  000000                  .WORD   0
001526  000000                  .WORD   0
001530  000000                  .WORD   0
001532  000000                  .WORD   0
001534  000000                  .WORD   0
001536  000000                  .WORD   0
001540  000000                  .WORD   0

001542  000000          ASNLST: .WORD   0                   ;A BIT SET IS AN ASSIGNED DRIVE

                        ;ADDRESSES OF DRIVES TO BE DEASSIGNED
001544  000000          DDRVS:  .WORD   0
001546  000000                  .WORD   0
001550  000000                  .WORD   0
001552  000000                  .WORD   0
```

```
001554  000000                          .WORD   0
001556  000000                          .WORD   0
001560  000000                          .WORD   0
001562  000000                          .WORD   0
001564  000000                          .WORD   0

                                ;ADDRESSES OF NEWLY ASSIGNED DRIVES
001566  000000                  NEWUNT: .WORD   0
001570  000000                          .WORD   0
001572  000000                          .WORD   0
001574  000000                          .WORD   0
001576  000000                          .WORD   0
001600  000000                          .WORD   0
001602  000000                          .WORD   0
001604  000000                          .WORD   0
001606  000000                          .WORD   0

                                ;LIST OF DRIVES WAITING FOR BUFFERS/PARAMETERS
001610  000000                  AVAIL:  .WORD   0
001612  000000                          .WORD   0
001614  000000                          .WORD   0
001616  000000                          .WORD   0
001620  000000                          .WORD   0
001622  000000                          .WORD   0
001624  000000                          .WORD   0
001626  000000                          .WORD   0
001630  000000                          .WORD   0

                                ;LIST OF DRIVES WAITING FOR BUFFERS
001632  000000                  WAIT:   .WORD   0
001634  000000                          .WORD   0
001636  000000                          .WORD   0
001640  000000                          .WORD   0
001642  000000                          .WORD   0
001644  000000                          .WORD   0
001646  000000                          .WORD   0
001650  000000                          .WORD   0
001652  000000                          .WORD   0

                                ;BUFFER ALLOCATION TABLE ENTRY COUNT
001654  000000                  BUFTBL: .WORD   0
001656  000000  000000                  .WORD   0,0
001662  000000  000000                  .WORD   0,0
001666  000000  000000                  .WORD   0,0
001672  000000  000000                  .WORD   0,0
001676  000000  000000                  .WORD   0,0
001702  000000  000000                  .WORD   0,0
001706  000000  000000                  .WORD   0,0
001712  000000  000000                  .WORD   0,0
001716  000000  000000                  .WORD   0,0
001722  000000  000000                  .WORD   0,0
001726  000000  000000                  .WORD   0,0
001732  000000  000000                  .WORD   0,0
001736  000000  000000                  .WORD   0,0
001742  000000  000000                  .WORD   0,0
001746  000000  000000                  .WORD   0,0
```

```
001752  000000  000000                  .WORD   0,0
001756  000000  000000                  .WORD   0,0
001762  000000  000000                  .WORD   0,0
001766  000000  000000                  .WORD   0,0
001772  000000  000000                  .WORD   0,0
001776  000000  000000                  .WORD   0,0
002002  000000  000000                  .WORD   0,0
002006  000000  000000                  .WORD   0,0
002012  000000  000000                  .WORD   0,0
002016  000000  000000                  .WORD   0,0
002022  000000  000000                  .WORD   0,0
002026  000000  000000                  .WORD   0,0
002032  000000  000000                  .WORD   0,0
002036  000000  000000                  .WORD   0,0
002042  000000  000000                  .WORD   0,0
002046  000000  000000                  .WORD   0,0
002052  000000  000000                  .WORD   0,0

                                ;DRIVE PARAMETER BLOCK(DPB) POINTER TABLE
002056  045416                  BLKADR: .WORD   DRIVE0  ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 0
002060  047632                          .WORD   DRIVE1  ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 1
002062  052046                          .WORD   DRIVE2  ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 2
002064  054262                          .WORD   DRIVE3  ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 3
002066  056476                          .WORD   DRIVE4  ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 4
002070  060712                          .WORD   DRIVE5  ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 5
002072  063126                          .WORD   DRIVE6  ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 6
002074  065342                          .WORD   DRIVE7  ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 7

                                ;DRIVER COMMAND CONTROL TABLE (USED IN RM DRIVER)
002076      151                 COMTBL: .BYTE   WCKD    ;WRITE CHECK DATA
002077      377                         .BYTE   -1      ;WRITE CHECK HEADER AND DATA (NOT USED)
002100      161                         .BYTE   WRTDAT  ;WRITE DATA
002101      377                         .BYTE   -1      ;WRITE HEADER AND DATA (NOT USED)
002102      171                         .BYTE   RDDAT   ;READ DATA
002103      173                         .BYTE   RDHD    ;READ HEADER AND DATA

                                ;FUNCTION(COMMAND) CODE CONTROL TABLE
002104      004                 OPTBL:  .BYTE   4       ;SEEK
002105      006                         .BYTE   6       ;RECAL
002106      010                         .BYTE   10      ;DRIVE CLEAR
002107      012                         .BYTE   12      ;RELEASE
002110      014                         .BYTE   14      ;OFFSET
002111      016                         .BYTE   16      ;RETURN TO CENTERLINE
002112      020                         .BYTE   20      ;READIN PRESET
002113      022                         .BYTE   22      ;PACK ACKNOWLEDGE
002114      030                         .BYTE   30      ;SEARCH
002115      050                         .BYTE   50      ;WRITE CHECK DATA
002116      052                         .BYTE   52      ;WRITE CHECK HEADER AND DATA
002117      060                         .BYTE   60      ;WRITE DATA
002120      062                         .BYTE   62      ;WRITE HEADER AND DATA
002121      070                         .BYTE   70      ;READ DATA
002122      072                         .BYTE   72      ;READ HEADER AND DATA
002123      377                         .BYTE   -1      ;TERMINATOR
                                .EVEN

                                ;MESSAGE CONTROL TABLE FOR 'OPTBL' TABLE
002124      123     105     105 MNTBL:  .ASCIZ  /SEEK   /
```

```
002134  122  105  103          .ASCIZ  /RECAL  /
002144  104  122  126          .ASCIZ  /DRVCLR /
002154  122  105  114          .ASCIZ  /RELSE  /
002164  117  106  106          .ASCIZ  /OFFSET /
002174  122  124  103          .ASCIZ  /RTC    /
002204  122  105  101          .ASCIZ  /READIN /
002214  120  101  103          .ASCIZ  /PACK   /
002224  123  105  101          .ASCIZ  /SEARCH /
002234  127  103  113          .ASCIZ  /WCKD   /
002244  127  103  113          .ASCIZ  /WCKHD  /
002254  127  122  124          .ASCIZ  /WRTDAT /
002264  127  122  124          .ASCIZ  /WRTHD  /
002274  122  104  104          .ASCIZ  /RDDAT  /
002304  122  104  110          .ASCIZ  /RDHD   /
002314  116  117  116          .ASCIZ  /NONE   /
                        .EVEN

                        .SBTTL  DATA PATTERNS

                        ;STANDARD DATA PATTERN POINTER TABLE
002324  002370          STNDAT: .WORD   DATA0   ;STANDARD DA.A PATTERN 0
002326  002430                  .WORD   DATA1   ;STANDARD DATA PATTERN 1
002330  002470                  .WORD   DATA2   ;STANDARD DATA PATTERN 2
002332  002530                  .WORD   DATA3   ;STANDARD DATA PATTERN 3
002334  002570                  .WORD   DATA4   ;STANDARD DATA PATTERN 4
002336  002630                  .WORD   DATA5   ;STANDARD DATA PATTERN 5
002340  002670                  .WORD   DATA6   ;STANDARD DATA PATTERN 6
002342  002730                  .WORD   DATA7   ;STANDARD DATA PATTERN 7
002344  002770                  .WORD   DATA8   ;STANDARD DATA PATTERN 8
002346  003030                  .WORD   DATA9   ;STANDARD DATA PATTERN 9
002350  003070                  .WORD   DATA10  ;STANDARD DATA PATTERN 10
002352  003130                  .WORD   DATA11  ;STANDARD DATA PATTERN 11
002354  003170                  .WORD   DATA12  ;STANDARD DATA PATTERN 12
002356  003230                  .WORD   DATA13  ;STANDARD DATA PATTERN 13
002360  003270                  .WORD   DATA14  ;STANDARD DATA PATTERN 14
002362  003330                  .WORD   DATA15  ;STANDARD DATA PATTERN 15
002364  002370                  .WORD   ZEROS   ;ALL 0'S PATTERN
002366  003272                  .WORD   ONES    ;ALL 1'S PATTERN

002370                  ZEROS:
002370  000000          DATA0:  .WORD   000000  ;ALL 0'S DATA PATTERN
002372  000000                  .WORD   000000
002374  000000                  .WORD   000000
002376  000000                  .WORD   000000
002400  000000                  .WORD   000000
002402  000000                  .WORD   000000
002404  000000                  .WORD   000000
002406  000000                  .WORD   000000
002410  000000                  .WORD   000000
002412  000000                  .WORD   000000
002414  000000                  .WORD   000000
002416  000000                  .WORD   000000
002420  000000                  .WORD   000000
002422  000000                  .WORD   000000
002424  000000                  .WORD   000000
002426  000000                  .WORD   000000
```

DATA PATTERNS

```
002430  000001                  DATA1:  .WORD   000001  ;STANDARD PATTERN 1
002432  000003                          .WORD   000003
002434  000007                          .WORD   000007
002436  000017                          .WORD   000017
002440  000037                          .WORD   000037
002442  000077                          .WORD   000077
002444  000177                          .WORD   000177
002446  000377                          .WORD   000377
002450  000777                          .WORD   000777
002452  001777                          .WORD   001777
002454  003777                          .WORD   003777
002456  007777                          .WORD   007777
002460  017777                          .WORD   017777
002462  037777                          .WORD   037777
002464  077777                          .WORD   077777
002466  177777                          .WORD   177777

002470  177776                  DATA2:  .WORD   177776  ;STANDARD PATTERN 2
002472  177774                          .WORD   177774
002474  177770                          .WORD   177770
002476  177760                          .WORD   177760
002500  177740                          .WORD   177740
002502  177700                          .WORD   177700
002504  177600                          .WORD   177600
002506  177400                          .WORD   177400
002510  177000                          .WORD   177000
002512  176000                          .WORD   176000
002514  174000                          .WORD   174000
002516  170000                          .WORD   170000
002520  160000                          .WORD   160000
002522  140000                          .WORD   140000
002524  100000                          .WORD   100000
002526  000000                          .WORD   000000

002530  000000                  DATA3:  .WORD   000000  ;STANDARD PATTERN 3
002532  000000                          .WORD   000000
002534  000000                          .WORD   000000
002536  177777                          .WORD   177777
002540  177777                          .WORD   177777
002542  177777                          .WORD   177777
002544  000000                          .WORD   000000
002546  000000                          .WORD   000000
002550  177777                          .WORD   177777
002552  177777                          .WORD   177777
002554  000000                          .WORD   000000
002556  177777                          .WORD   177777
002560  000000                          .WORD   000000
002562  177777                          .WORD   177777
002564  000000                          .WORD   000000
002566  177777                          .WORD   177777

002570  133331                  DATA4:  .WORD   133331  ;STANDARD PATTERN 4
002572  133331                          .WORD   133331
002574  133331                          .WORD   133331
002576  133331                          .WORD   133331
002600  133331                          .WORD   133331
002602  133331                          .WORD   133331
```

DATA PATTERNS

```
002604  133331                          .WORD   133331
002606  133331                          .WORD   133331
002610  133331                          .WORD   133331
002612  133331                          .WORD   133331
002614  133331                          .WORD   133331
002616  133331                          .WORD   133331
002620  133331                          .WORD   133331
002622  133331                          .WORD   133331
002624  133331                          .WORD   133331
002626  133331                          .WORD   133331

002630  052525                  DATA5:  .WORD   052525  ;STANDARD PATTERN 5
002632  052525                          .WORD   052525
002634  052525                          .WORD   052525
002636  125252                          .WORD   125252
002640  125252                          .WORD   125252
002642  125252                          .WORD   125252
002644  052525                          .WORD   052525
002646  052525                          .WORD   052525
002650  125252                          .WORD   125252
002652  125252                          .WORD   125252
002654  052525                          .WORD   052525
002656  125252                          .WORD   125252
002660  052525                          .WORD   052525
002662  125252                          .WORD   125252
002664  052525                          .WORD   052525
002656  125252                          .WORD   125252

002670  155555                  DATA6:  .WORD   155555  ;STANDARD PATTERN 6
002672  155555                          .WORD   155555
002674  155555                          .WORD   155555
002676  155555                          .WORD   155555
002700  155555                          .WORD   155555
002702  155555                          .WORD   155555
002704  155555                          .WORD   155555
002706  155555                          .WORD   155555
002710  155555                          .WORD   155555
002712  155555                          .WORD   155555
002714  155555                          .WORD   155555
002716  155555                          .WORD   155555
002720  155555                          .WORD   155555
002722  155555                          .WORD   155555
002724  155555                          .WORD   155555
002726  155555                          .WORD   155555

002730  026455                  DATA7:  .WORD   026455  ;STANDARD PATTERN 7
002732  026455                          .WORD   026455
002734  026455                          .WORD   026455
002736  151322                          .WORD   151322
002740  151322                          .WORD   151322
002742  151322                          .WORD   151322
002744  026455                          .WORD   026455
002746  026455                          .WORD   026455
002750  151322                          .WORD   151322
002752  151322                          .WORD   151322
002754  026455                          .WORD   026455
002756  151322                          .WORD   151322
```

```
002760  026455                          .WORD   026455
002762  151322                          .WORD   151322
002764  026455                          .WORD   026455
002766  151322                          .WORD   151322

002770  155555          DATA8:          .WORD   155555  ;STANDARD PATTERN 8
002772  133333                          .WORD   133333
002774  155555                          .WORD   155555
002776  133333                          .WORD   133333
003000  155555                          .WORD   155555
003002  133333                          .WORD   133333
003004  155555                          .WORD   155555
003006  133333                          .WORD   133333
003010  155555                          .WORD   155555
003012  133333                          .WORD   133333
003014  155555                          .WORD   155555
003016  133333                          .WORD   133333
003020  155555                          .WORD   155555
003022  133333                          .WORD   133333
003024  155555                          .WORD   155555
003026  133333                          .WORD   133333

003030  000001          DATA9:          .WORD   000001  ;STANDARD PATTERN 9
003032  000002                          .WORD   000002
003034  000004                          .WORD   000004
003036  000010                          .WORD   000010
003040  000020                          .WORD   000020
003042  000040                          .WORD   000040
003044  000100                          .WORD   000100
003046  000200                          .WORD   000200
003050  000400                          .WORD   000400
003052  001000                          .WORD   001000
003054  002000                          .WORD   002000
003056  004000                          .WORD   004000
003060  010000                          .WORD   010000
003062  020000                          .WORD   020000
003064  040000                          .WORD   040000
003066  100000                          .WORD   100000

003070  177776          DATA10:         .WORD   177776  ;STANDARD PATTERN 10
003072  177775                          .WORD   177775
003074  177773                          .WORD   177773
003076  177767                          .WORD   177767
003100  177757                          .WORD   177757
003102  177737                          .WORD   177737
003104  177677                          .WORD   177677
003106  177577                          .WORD   177577
003110  177377                          .WORD   177377
003112  176777                          .WORD   176777
003114  175777                          .WORD   175777
003116  173777                          .WORD   173777
003120  167777                          .WORD   167777
003122  157777                          .WORD   157777
003124  137777                          .WORD   137777
003126  077777                          .WORD   077777

003130  172666          DATA11:         .WORD   172666  ;STANDARD PATTERN 11
```

DATA PATTERNS

```
003132  155555                          .WORD   155555
003134  172666                          .WORD   172666
003136  155555                          .WORD   155555
003140  172666                          .WORD   172666
003142  155555                          .WORD   155555
003144  172666                          .WORD   172666
003146  155555                          .WORD   155555
003150  172666                          .WORD   172666
003152  155555                          .WORD   155555
003154  172666                          .WORD   172666
003156  155555                          .WORD   155555
003160  172666                          .WORD   172666
003162  155555                          .WORD   155555
003164  172666                          .WORD   172666
003166  155555                          .WORD   155555

003170  077777                  DATA12: .WORD   077777  ;STANDARD PATTERN 12
003172  137777                          .WORD   137777
003174  157777                          .WORD   157777
003176  167777                          .WORD   167777
003200  173777                          .WORD   173777
003202  175777                          .WORD   175777
003204  176777                          .WORD   176777
003206  177377                          .WORD   177377
003210  177577                          .WORD   177577
003212  177677                          .WORD   177677
003214  177737                          .WORD   177737
003216  177757                          .WORD   177757
003220  177767                          .WORD   177767
003222  177773                          .WORD   177773
003224  177775                          .WORD   177775
003226  177776                          .WORD   177776

003230  153333                  DATA13: .WORD   153333  ;STANDARD PATTERN 13
003232  066667                          .WORD   066667
003234  153333                          .WORD   153333
003236  066667                          .WORD   066667
003240  153333                          .WORD   153333
003242  066667                          .WORD   066667
003244  153333                          .WORD   153333
003246  066667                          .WORD   066667
003250  153333                          .WORD   153333
003252  066667                          .WORD   066667
003254  153333                          .WORD   153333
003256  066667                          .WORD   066667
003260  153333                          .WORD   153333
003262  066667                          .WORD   066667
003264  153333                          .WORD   153333
003266  066667                          .WORD   066667

003270  000000                  DATA14: .WORD   000000  ;STANDARD PATTERN 14
003272  177777                  ONES:   .WORD   177777  ;ALL 1'S DATA PATTERN
003274  177777                          .WORD   177777
003276  177777                          .WORD   177777
003300  177777                          .WORD   177777
003302  177777                          .WORD   177777
003304  177777                          .WORD   177777
```

DATA PATTERNS

```
003306  177777                          .WORD   177777
003310  177777                          .WORD   177777
003312  177777                          .WORD   177777
003314  177777                          .WORD   177777
003316  177777                          .WORD   177777
003320  177777                          .WORD   177777
003322  177777                          .WORD   177777
003324  177777                          .WORD   177777
003326  177777                          .WORD   177777

003330  177777                  DAT415: .WORD   177777  ;STANDARD PATTERN 15
003332  000000                          WORD    000000
003334  000000                          .WORD   000000
003336  000000                          .WORD   000000
003340  000000                          .WORD   000000
003342  000000                          .WORD   000000
003344  000000                          .WORD   000000
003346  000000                          .WORD   000000
003350  000000                          .WORD   000000
003352  000000                          .WORD   000000
003354  000000                          .WORD   000000
003356  000000                          .WORD   000000
003360  000000                          .WORD   000000
003362  000000                          .WORD   000000
003364  000000                          .WORD   000000
003366  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


  003370                        $ERRTB:
    1
    2                           ;ERROR 1
    3
    4 003370  067646                    EM1             ;RH CONTROLLER INTERRUPT OCCURRED (RMAS = 0)
    5 003372  072467                    DH1
    6 003374  073134                    DT1
    7 003376  000000                    0
    8
    9                           ;ERROR 2
   10
   11 003400  067720                    EM2             ;UNEXPECTED ATTENTION OCCURRED
   12 003402  072474                    DH2
   13 003404  073140                    DT2
   14 003406  000000                    0
   15
   16                           ;ERROR 3
   17
   18 003410  067756                    EM3             ;NOT USED
   19 003412  072550                    DH3
   20 003414  073156                    DT3
   21 003416  000000                    0
   22
   23                           ;ERROR 4
   24
   25 003420  070014                    EM4             ;NOT USED
   26 003422  072576                    DH4
   27 003424  073166                    DT4
   28 003426  000000                    0
   29
   30                           ;ERROR 5
   31
   32 003430  070051                    EM5             ;ADDRESS PLUG BIT CHANGED
   33 003432  072474                    DH2
   34 003434  073140                    DT2
   35 003436  000000                    0
   36
   37                           ;ERROR 6
   38
   39 003440  070105                    EM6             ;NOT USED
   40 003442  072635                    DH6
   41 003444  073200                    DT6
   42 003446  000000                    0
```

ERROR POINTER TABLE

```
                                        ;THIS ROUTINE HANDLES UNEXPECTED TIMEOUTS
                                        ;
003450  011600                BADTMO: MOV     (SP),R0         ;SAVE PC WHERE THE TIME OUT OCCURED
003452  005740                        TST     -(R0)           ;ADJUST PC -2
003454  022626                        CMP     (SP)+,(SP)+     ;RESTORE STACK POINTER
003456  104401  003464                TYPE    ,65$            ;;TYPE ASCIZ STRING
003462  000417                        BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <CRLF>/UNEXPECTED BUS TIMEOUT, PC=/
003522                          64$:
003522  010046                        MOV     R0,-(SP)        ;SETUP FOR TYPING OUT PC
003524  104402                        TYPOC
003526  000240                        NOP                     ;PUT 'HALT(0)' INSTRUCTION HERE IF YOU WISH
                                                                ;TO STOP ON UNEXPECTED TIMEOUT.
003530  000404                        BR      START1          ;BRANCH TO START1
                                        ;
                                        .SBTTL  START OF PROGRAM
                                        ;
003532  012737  177777  001336 START:  MOV     #-1,CHGADR      ;SET RH/RM ADDRESS CHANGE FLAG
003540  000407                        BR      START2          ;START THE PROGRAM
                                        ;
003542  012737  000400  001336 START1: MOV     #400,CHGADR     ;CLEAR THE RH/RM ADDRESS CHANGE FLAG
                                        ;;**************************************************************
003550  000240                  TST1:   NOP
003552  012737  000001  001212        MOV     #1,$TESTN       ;;SET TEST NUMBER IN APT MAIL BOX
                                        ;
003560  005227  000000        START2: INC     #0              ;TTY LOOP, WAIT FOR INCREMENT
003564  001375                        BNE     .-4             ;OF WORD
003566  000005                        RESET                   ;CLEAR THE WORLD
                                        ;
                                        .SBTTL  INITIALIZE THE COMMON TAGS
                                        ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
003570  012706  001114                MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
003574  005026                        CLR     (R6)+           ;;CLEAR MEMORY LOCATION
003576  022706  001154                CMP     #SWR,R6         ;;DONE?
003602  001374                        BNE     .-6             ;;LOOP BACK IF NO
003604  012706  001100                MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                ;;INITIALIZE A FEW VECTORS
003610  012737  034640  000030        MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
003616  012737  000340  000032        MOV     #340,@#EMTVEC+2 ;;LEVEL 7
003624  012737  040060  000034        MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
003632  012737  000340  000036        MOV     #340,@#TRAPVEC+2;LEVEL 7
003640  012737  037536  000024        MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
003646  012737  000340  000026        MOV     #340,@#PWRVEC+2 ;;LEVEL 7
003654  012737  176543  037122        MOV     #176543,$HINUM  ;;PRIME THE RANDOM NUMBER GENERATOR
003662  012737  123456  037124        MOV     #123456,$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.
003670  013746  000004                MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
003674  012737  003730  000004        MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
003702  012737  177570  001154        MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
003710  012737  177570  001156        MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
003716  022777  177777  175230        CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
003724  001012                        BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                ;;AND  THE HARDWARE SWR IS NOT = -1
003726  000403                        BR      65$             ;;BRANCH IF NO TIMEOUT
003730  012716  003736        64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
003734  000002                        RTI
```

```
INITIALIZE THE COMMON TAGS

003736  012737  000176  001154  65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
003744  012737  000174  001156          MOV     #DISPREG,DISPLAY
003752  012637  000004          66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR

003756  005037  001214                  CLR     $PASS           ;;CLEAR PASS COUNT
003762  132737  000200  001227          BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
003770  001403                          BEQ     67$             ;;YES,USE NON-APT SWITCH
003772  012737  001230  001154          MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
004000                          67$:
                        ;SETUP ''TIMEOUT'' TRAP VECTOR FOR UNEXPECTED BUS TIMEOUTS
004000  012737  003450  000004          MOV     #BADTMO,ERRVEC  ;SETUP FOR UNEXPECTED TIMEOUT
004006  012737  000300  000006          MOV     #PR6,ERRVEC+2   ;LEVEL 6

                        .SBTTL  TYPE PROGRAM NAME
                        ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
004014  005227  177777                  INC     #-1             ;;FIRST TIME?
004020  001027                          BNE     68$             ;;BRANCH IF NO
004022  104401  004030                  TYPE    ,69$            ;;TYPE ASCIZ STRING
004026  000424                          BR      68$             ;;GET OVER THE ASCIZ
                        ;;69$:  .ASCIZ  <CRLF>@CZRNAAO - RM80 PERFORMANCE EXERCISER@<CRLF>
004100                          68$:
                        .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
004100  005737  000042                  TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
004104  001012                          BNE     70$             ;;BRANCH IF YES
004106  123727  001226  000001          CMPB    $ENV,#1         ;;ARE WE RUNNING UNDER APT?
004114  001406                          BEQ     70$             ;;BRANCH IF YES
004116  023727  001154  000176          CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
004124  001005                          BNE     71$             ;;BRANCH IF NO
004126  104406                          GTSWR                   ;;GET SOFT-SWR SETTINGS
004130  000403                          BR      71$
004132  112737  000001  001150  70$:    MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
004140                          71$:

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

004140  005037  001444          CLR     XXDP            ;CLEAR 'XXDP' LOAD DEVICE STORAGE
004144  122737  000016  000041  CMPB    #16,@#41        ;LOADED FROM AN RM80 ?
004152  001121                  BNE     3$              ;BR IF NOT
004154  013737  000040  001444  MOV     @#40,XXDP       ;GET DEVICE INDICATOR AND NUMBER
004162  122737  000007  001444  CMPB    #7,XXDP         ;IS IT A VALID NUMBER ?
004170  103002                  BHIS    1$              ;YES
004172  105037  001444          CLRB    XXDP            ;NO, DEFAULT TO DRIVE 0
004176  005737  000042  1$:     TST     @#42            ;CHAIN MODE OR ACT11 AUTO ACCEPT ?
004202  001425                  BEQ     2$              ;BR IF NEITHER
004204  104401  004212          TYPE    ,73$            ;;TYPE ASCIZ STRING
004210  000412                  BR      72$             ;;GET OVER THE ASCIZ
                        ;;73$:  .ASCIZ  <CRLF>/NOT TESTING DRIVE /
004236                          72$:
004236  005046                  CLR     -(SP)           ;CLEAR WORD ON STACK
004240  113716  001444          MOVB    XXDP,(SP)       ;GET DRIVE ADDRESS
004244  104403                  TYPOS                   ;TYPE THE ADDRESS
004246     001                  .BYTE   1               ;ONLY 1 CHARACTER
004247     000                  .BYTE   0               ;SUPRESS LEADING ZEROS
004250  104401  001203          TYPE    ,$CRLF          ;CR-LF
004254  000460                  BR      3$              ;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 $FIRST FLAG FOR DRIVE 7
                                                        CLR     RDONLY          ;ASSUME READ/WRITE CONDITION
                                                        BIT     #SW0,@SWR       ;IS EXERCISER IN 'READ ONLY' MODE ?
                                                        BEQ     8$              ;BR IF NO
```

```
                        INC     RDONLY          ;LOCK PROGRAM IN 'READ ONLY' MODE
                8$:
                ;AUTO SIZE FOR RH70 CONTROLLER AND DETERMINE IF IT IS
                ;JUMPERED FOR 22 OR 32 REGISTERS
                SIZE70: CLR     RHEXT           ;CLEAR RMBAE OFFSET
                        BIC     #174000,$CPUOP  ;CLEAR CPU TYPE REGISTER
                        MOV     ERRVEC,-(SP)    ;SAVE CONTENTS OF ERROR VECTOR
                        MOV     #2$,ERRVEC      ;SETUP 'TRAP' RETURN ADDRESS
                        MOV     $RMADR,R0       ;GET RMCS1 ADDRESS
                        ADD     #50,R0          ;GET REGISTER OFFSET FOR RH70
                        MOV     #10.,R1         ;GET NUMBER OF REGISTERS TO CHECK
                        TST     (R0)+           ;TRAP IF NOT A VALID RMBAE
                        TST     (R0)+           ;TRAP IF NOT A VALID RMCS3
                        MOV     #50,RHEXT       ;LOAD OFFSET FOR RMBAE (22 REGISTER RH)
                1$:     TST     (R0)+           ;TRAP IF NOT A VALID REGISTER
                        DEC     R1              ;DONE WITH ALL 32 REGISTERS ?
                        BNE     1$              ;BR IF NO
                        MOV     #74,RHEXT       ;LOAD OFFSET FOR RMBAE (32 REGISTER RH)
                        BR      3$
                2$:     MOV     #3$,(SP)        ;SETUP RETURN ADDRESS
                        RTI
                3$:     MOV     $RMADR,R0       ;GET RMCS1 REGISTER
                        MOV     RHEXT,R1        ;GET RMBAE REGISTER OFFSET
                        BEQ     4$              ;BR IF NONE
                        ADD     R0,R1           ;GET RMBAE REGISTER
                        BIS     #A17!A16,(R0)   ;SET EXTENDED ADDRESS BITS IN RMCS1
                        CMP     #3,(R1)         ;ARE THE EXTENDED BITS SET IN RMBAE ?
                        BNE     4$              ;BR IF NO
                        CLR     (R1)            ;CLEAR EXTENDED ADDRESS BITS IN RMBAE
                        BIT     A17!A16,(R0)    ;ARE THE EXTEND BITS CLEAR IN RMCS1 ?
                        BNE     4$              ;BR IF NO
                        BIS     #BIT13!BIT12,$CPUOP     ;SET THE 11/70 CPU TYPE CODE
                4$:     MOV     (SP)+,ERRVEC    ;RESTORE CONTENTS OF ERROR VECTOR

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

                SIZMEM: INC     #-1             ;FIRST TIME THRU HERE ?
                        BNE     1$              ;BR IF NO
                        MOV     #-1,R0          ;FUDGE MSB'S FOR INITIALIZING HDA S/N
                        MOV     R0,DRIVE0+$HSNM ;INIT. S/N FOR DRIVE 0
                        MOV     R0,DRIVE1+$HSNM ;INIT. S/N FOR DRIVE 1
                        MOV     R0,DRIVE2+$HSNM ;INIT. S/N FOR DRIVE 2
                        MOV     R0,DRIVE3+$HSNM ;INIT. S/N FOR DRIVE 3
                        MOV     R0,DRIVE4+$HSNM ;INIT. S/N FOR DRIVE 4
                        MOV     R0,DRIVE5+$HSNM ;INIT. S/N FOR DRIVE 5
                        MOV     R0,DRIVE6+$HSNM ;INIT. S/N FOR DRIVE 6
                        MOV     R0,DRIVE7+$HSNM ;INIT. S/N FOR DRIVE 7
                        JSR     PC,$SIZE        ;SEE HOW MUCH MEMORY ON SYSTEM
                        MOV     $LSTAD,LSTAD    ;SAVE THE LAST ADDRESS
                1$:     MOV     #1,BUFTBL       ;LOAD NUMBER OF BUFFERS
                        MOV     #ENDPGM,BUFTBL+2        ;STARTING ADDRESS OF BUFFER
                        MOV     LSTAD,BUFTBL+4  ;LAST ADDR TO BUFFER ALLOCATION TABLE
```

```
    161 005132  023727  001334  160000          CMP     LSTAD,#160000   ;OVER 28K ?
    162 005140  101403                          BLOS    2$              ;NO
    163 005142  012737  160000  001660          MOV     #160000,BUFTBL+4        ;XXDP MAX MEMORY 28K
    164 005150  162737  102200  001660  2$:     SUB     #ENDPGM,BUFTBL+4        ;SUBTRACT PROGRAM SPACE
    165 005156  000241                          CLC                             ;CLEAR THE 'C' BIT
    166 005160  006037  001660                  ROR     BUFTBL+4                ;CONVERT TO WORD COUNT
    167 005164  162737  000144  001660          SUB     #100.,BUFTBL+4  ;SAVE ROOM FOR THE 'ABS' LOADER
    168 005172  105737  001150                  TSTB    $AUTOB                  ;RUNNING IN AUTO MODE ?
    169 005176  001403                          BEQ     3$              ;BR IF NO
    170 005200  162737  003000  001660          SUB     #1536.,BUFTBL+4 ;SUBTRACT 'XXDP' LOADER SIZE
    171 005206  023737  001462  001660  3$:     CMP     WRDCNT,BUFTBL+4 ;IS MAX WORD COUNT TOO LARGE ?
    172 005214  003406                          BLE     4$              ;BR IF NO
    173 005216  013737  001660  001462          MOV     BUFTBL+4,WRDCNT ;USE MAX AVAIL MEMORY AS MAX WRD CNT
    174 005224  013737  001462  077470          MOV     WRDCNT,PARLST+2 ;VALUE FOR THE PARAMETER TABLE
    175 005232                          4$:
    176
    177                                 ;SEE IF THE OPERATOR WANTS TO CHANGE ANY PARAMETERS
    178
    179 005232  005737  037716          LKPAR:  TST     PWRFLG          ;RETURNING FROM POWER FAIL ?
    180 005236  001154                          BNE     SETVEC          ;BRANCH IF YES
    184 005244  105737  001150                  TSTB    $AUTOB          ;RUNNING IN AUTO MODE ?
    185 005250  001407                          BEQ     1$              ;BR IF NO
    186 005252  032777  000004  173674          BIT     #SW02,@SWR      ;DOES USER WANT MANUAL INTERVENTION ?
    187 005260  001003                          BNE     1$              ;BR IF YES
    188 005262  104401  076674                  TYPE    ,FEONLY         ;TYPE FE CYLINDERS ONLY MESSAGE
    189 005266  000466                          BR      8$
    190
    191 005270  005037  001340          1$:     CLR     CFLAG           ;CLEAR CONTROL C FLAG
    192 005274  104401  076562                  TYPE    ,MESFE          ;TYPE 'EXERCISER FE CYLINDERS ONLY ?'
    193 005300  104411                          RDLIN                   ;READ THE ENTRY
    194 005302  012600                          MOV     (SP)+,R0        ;SAVE ADDRESS OF RESPONSE
    195 005304  005737  001340                  TST     CFLAG           ;WAS IT CONTROL C ?
    196 005310  001350                          BNE     LKPAR           ;BR IF YES
    197 005312  105710                          TSTB    (R0)            ;WAS RESPONSE A CARRIAGE RETURN ?
    198 005314  001451                          BEQ     7$              ;BR IF YES
    199 005316  105760  000001                  TSTB    1(R0)           ;WAS IT TERMINATED WITH CARRIAGE RETURN ?
    200 005322  001006                          BNE     2$              ;BR IF NO
    201 005324  122710  000131                  CMPB    #'Y,(R0)        ;WAS IT A 'Y' RESPONSE ?
    202 005330  001443                          BEQ     7$              ;BR IF YES
    203 005332  122710  000116                  CMPB    #'N,(R0)        ;WAS IT A 'N' RESPONSE ?
    204 005336  001403                          BEQ     3$              ;BR IF YES
    205 005340  104401  076341          2$:     TYPE    ,BADENT         ;TYPE BAD ENTRY MESSAGE
    206 005344  000732                          BR      LKPAR           ;TRY AGAIN
    207 005346  005737  001440          3$:     TST     RDONLY          ;PROGRAM RUNNING IN READ ONLY MODE ?
    208 005352  001002                          BNE     4$              ;BR IF YES (DO NOT TYPE OVERWRITE MESSAGE)
    209 005354  104401  076756                  TYPE    ,OVRWRT         ;TYPE DATA OVERWRITE MESSAGE
    210
    211 005360  104401  076646          4$:     TYPE    ,SURE           ;TYPE 'ARE YOU SURE ?'
    212 005364  104411                          RDLIN                   ;READ THE ENTRY
    213 005366  012600                          MOV     (SP)+,R0        ;SAVE ADDRESS OF RESPONSE
    214 005370  005737  001340                  TST     CFLAG           ;WAS IT CONTROL C ?
    215 005374  001316                          BNE     LKPAR           ;BR IF YES
    216 005376  105710                          TSTB    (R0)            ;WAS RESPONSE A CARRIAGE RETURN ?
    217 005400  001417                          BEQ     7$              ;BR IF YES
    218 005402  105760  000001                  TSTB    1(R0)           ;WAS IT TERMINATED WITH CARRIAGE RETURN ?
    219 005406  001006                          BNE     5$              ;BR IF NO
    220 005410  122710  000131                  CMPB    #'Y,(R0)        ;WAS IT A 'Y' RESPONSE ?
```

```
                                        BEQ     6$              ;BR IF YES
                                        CMPB    #'N,(R0)        ;WAS IT A 'N' RESPONSE ?
                                        BEQ     7$              ;BR IF YES
                        5$:             TYPE    ,BADENT         ;TYPE BAD ENTRY MESSAGE
                                        BR      3$              ;TRY AGAIN
                        6$:             INC     FEFLAG          ;EXERCISE THE ENTIRE DISK
                                        BR      8$
                        7$:             TYPE    ,FEONLY         ;TYPE FE CYLINDER ONLY MESSAGE

                        8$:             TST     RDONLY          ;IS PROGRAM LOCKED IN 'READ MODE' ?
                                        BEQ     9$              ;BR IF NO
                                        TYPE    ,MREAD          ;TYPE READ ONLY MESSAGE

                        9$:             CLR     CFLAG           ;CLEAR CONTROL C FLAG
                                        TSTB    $AUTOB          ;RUNNING IN AUTO MODE ?
                                        BNE     SETVEC          ;BR IF YES
                                        TYPE    ,ASKPAR         ;TYPE 'CHANGE PARAMETERS ?'
                                        RDLIN                   ;READ THE ENTRY
                                        MOV     (SP)+,R0        ;SAVE ADDRESS OF RESPONSE
                                        TST     CFLAG           ;WAS IT CONTROL C ?
                                        BNE     9$              ;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     10$             ;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
                        10$:            TYPE    ,BADENT         ;TYPE BAD ENTRY MESSAGE
                                        BR      9$              ;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
                                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$:     CLR     R4              ;DRIVE TABLE POINTER
                                TYPE    ,SYSTAT         ;TYPE STATUS HEADING
                        2$:     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)
```

```
                005651   000
                                                        .BYTE   0               ;;SUPPRESS LEADING ZEROS
    275 005652  104401  075231                          TYPE    ,BLNKS4         ;TYPE 4 BLANKS
    276 005656  105764  040164                          TSTB    DRVSTA(R4)      ;CHECK DRIVE'S STATUS
    277 005662  100416                                  BMI     5$              ;BR IF UNSAFE
    278 005664  001020                                  BNE     6$              ;BR IF ONLINE
    279
    280 005666  105764  040174                          TSTB    DRVTYP(R4)      ;SEE IF OFFLINE OR NONEXISTENT
    281 005672  001404                                  BEQ     3$              ;BR IF NONEXISTENT
    282 005674  100006                                  BPL     4$              ;BR IF OFFLINE
    283 005676  104401  075403                          TYPE    ,NOTRM          ;DRIVE NOT AN RM80
    284 005702  000433                                  BR      11$             ;CHECK NEXT DRIVE
    285
    286 005704  104401  075420          3$:             TYPE    ,NOTPRS         ;DRIVE NOT PRESENT
    287 005710  000430                                  BR      11$             ;CHECK NEXT DRIVE
    288
    289 005712  104401  075312          4$:             TYPE    ,UNTOFF         ;DRIVE OFFLINE
    290 005716  000416                                  BR      8$              ;PRINT DRIVE TYPE
    291
    292 005720  104401  075454          5$:             TYPE    ,NOTSAF         ;DRIVE UNSAFE
    293 005724  000413                                  BR      8$              ;PRINT DRIVE TYPE
    294
    295 005726  005737  001444          6$:             TST     XXDP            ;LOADED FROM THIS DEVICE ?
    296 005732  001406                                  BEQ     7$              ;BR IF NO
    297 005734  123704  001444                          CMPB    XXDP,R4         ;LOADED FROM THIS DRIVE ?
    298 005740  001003                                  BNE     7$              ;BR IF NO
    299 005742  104401  075464                          TYPE    ,LODEV          ;TYPE 'LOAD DEVICE'
    300 005746  000411                                  BR      11$
    301 005750  104401  075323          7$:             TYPE    ,UNTON          ;DRIVE ONLINE
    302 005754  104401  075233          8$:             TYPE    ,BLNKS2         ;TYPE 2 BLANKS
    307 005760  012737  075520  005770                  MOV     #$RM80,10$      ;ASSUME ADDRESS OF RM80 MESSAGE
    312
    313 005766  104401                  9$:             TYPE                    ;TYPE THE DRIVE TYPE MESSAGE
    314 005770  000000                  10$:            .WORD   0               ;MESSAGE ADDRESS HERE
    315
    316 005772  005204                  11$:            INC     R4              ;INCREMENT DRIVE NUMBER/TABLE POINTER
    317 005774  020427  000010                          CMP     R4,#8.          ;FINISHED ?
    318 006000  001317                                  BNE     2$              ;BR IF NOT
    319 006002  104401  001203                          TYPE    ,$CRLF          ;CR-LF
    320 006006                          12$:
    321
```

GET VALUE FOR SOFTWARE SWITCH REGISTER

```
                                        ;INITIALIZE PROGRAM PARAMETERS FOR STARTUP
                                        
006006  004737  033326          STA:    JSR     PC,$TKINT       ;INITIALIZE THE KEYBOARD INTERRUPT HANDLER
006012  012737  002740  001446          MOV     #002740,ENDCON  ;INITIALIZE XFER COUNT(LSB)
006020  012737  005455  001450          MOV     #005455,ENDCON+2
006026  105737  001226                  TSTB    $ENV            ;APT SCRIPT MODE, THEN MAKE IT RUN 2 MIN.
006032  001411                          BEQ     1$              ;NO
006034  012737  000001  001476          MOV     #1,RATIO        ;SPEED UP TEST
006042  012737  077777  001446          MOV     #77777,ENDCON   ;INITIALIZE QUICK XFER COUNT(LSB)
006050  012737  000027  001450          MOV     #27,ENDCON+2
                                        
006056  105737  001150          1$:     TSTB    $AUTOB          ;RUNNING IN AUTO MODE ?
006062  001003                          BNE     2$              ;BR IF YES
006064  005737  001336                  TST     CHGADR          ;START AT 200 ?
006070  003456                          BLE     8$              ;NO
                                        
006072  005001                  2$:     CLR     R1              ;DRIVE #
006074  005002                          CLR     R2              ;AVAIL TABLE INDEX
006076  005003                          CLR     R3              ;DRIVE# * 2
006100  005737  001444          3$:     TST     XXDP            ;LOADED FROM THIS DEVICE ?
006104  001403                          BEQ     4$              ;BR IF NO
006106  123701  001444                  CMPB    XXDP,R1         ;LOADED FROM THIS DRIVE ?
006112  001435                          BEQ     7$              ;BR IF YES
006114  105761  040164          4$:     TSTB    DRVSTA(R1)      ;DRIVE ON LINE ?
006120  003432                          BLE     7$              ;NO
006122  110137  067556                  MOVB    R1,GENDPB       ;GET DRIVE NUMBER
006126  016300  002056                  MOV     BLKADR(R3),R0   ;LOAD DPB ADDRESS
006132  004737  015624                  JSR     PC,RECAL0       ;RECALIBRATE DRIVE
006136  004737  026630                  JSR     PC,CLRDPB       ;CLEAR DPB BLOCK
006142  004737  027550                  JSR     PC,GETID        ;GET DRIVE (MBA) SERIAL NUMBER
006146  004537  027650                  JSR     R5,GETADR       ;RETRIEVE BAD SECTOR FILE
006152  010062  001566                  MOV     R0,NEWUNT(R2)   ;LOAD DPB ADDRESS TO ABAIL QUEUE
006156  004737  027054                  JSR     PC,DRVPRM       ;SETUP DRIVE PARAMETER LIMITS
006162  005060  000124                  CLR     $FIRST(R0)      ;RESET $FIRST FLAG FOR FIRST 204 START
006166  005737  037716                  TST     PWRFLG          ;RETURNING FROM POWER FAIL ?
006172  001005                          BNE     7$              ;BRANCH IF YES
006174  112760  177776  000026          MOVB    #-2,$PACK(R0)   ;SETUP COMMAND 'WT' (WRITE DATA AND TEST)
006202  004737  017000                  JSR     PC,WRTPK        ;SETUP INITIAL PARAMETERS
                                        
006206  022322                  7$:     CMP     (R3)+,(R2)+     ;INCREMENT INDEX
006210  005201                          INC     R1              ;NEXT DRIVE
006212  020127  000007                  CMP     R1,#7           ;ALL DRIVES ASSIGNNED ?
006216  003730                          BLE     3$              ;NO
006220  005037  001336                  CLR     CHGADR          ;CLEAR START FLAG
006224  000403                          BR      9$
                                        
006226  012737  000001  001340  8$:     MOV     #1,CFLAG        ;DUMMY 'CONTROL C' FLAG
006234  005037  037716          9$:     CLR     PWRFLG          ;CLEAR POWER FAIL FLAG
```

```
                                        .SBTTL  MAIN PROGRAM
                                
006240  005737  001340      MAIN:     TST     CFLAG           ;KEYBOARD INTERRUPTED ?
006244  001407                        BEQ     3$              ;BR IF NOT
006246  005737  001520      1$:       TST     ORDERQ          ;ANY DRIVES IN ORDER QUE ?
006252  001402                        BEQ     2$              ;BR IF NO, ELSE
006254  000137  007104                JMP     IDLE            ;LET ALL DRIVES FINISH ORDER
006260  004737  025254      2$:       JSR     PC,KSR          ;SERVICE THE KEYBOARD
006264  000240              3$:       NOP                     ; !! FOR DEBUGGING !!
                                
                                        ;CHECK FOR DRIVES TO BE DROPPED
                                
006266  012703  000010      MAINDA:   MOV     #8.,R3          ;DRIVE COUNTER
006272  012705  001544                MOV     #DDRVS,R5       ;ADDRESS OF 'DROP DRIVE' TABLE
006276  005715              1$:       TST     (R5)            ;SEE IF ENTRY AT PRESENT POSITION
006300  001004                        BNE     3$              ;BR IF THERE IS ONE
006302  005725              2$:       TST     (R5)+           ;INCREMENT TO NEXT TABLE POSITION
006304  005303                        DEC     R3              ;DECREMENT DRIVE COUNTER
006306  001373                        BNE     1$              ;BR IF MORE TO CHECK
006310  000435                        BR      MAIN1           ;GO CHECK FOR NEW ASSIGNED DRIVES
                                
006312  012701  001610      3$:       MOV     #AVAIL,R1       ;ADDRESS OF 'AVAILABLE DRIVES' TABLE
006316  005711              4$:       TST     (R1)            ;IF AT END OF 'AVAIL' TABLE ?
006320  001404                        BEQ     5$              ;BR IF YES
006322  021115                        CMP     (R1),(R5)       ;IS DRIVE IN 'AVAIL' THE TABLE ?
006324  001412                        BEQ     7$              ;BR IF YES
006326  005721                        TST     (R1)+           ;NO, INCREMENT 'AVAIL' TABLE ADDRESS
006330  000772                        BR      4$              ;AND CONTINUE LOOKING
                                
006332  012701  001632      5$:       MOV     #WAIT,R1        ;ADDRESS OF THE 'WAIT' BUFFER TABLE
006336  005711              6$:       TST     (R1)            ;AT THE END OF 'WAIT' TABLE ?
006340  001760                        BEQ     2$              ;BR IF YES
006342  021115                        CMP     (R1),(R5)       ;IS DRIVE IN THE 'WAIT' TABLE ?
006344  001402                        BEQ     7$              ;BR IF YES
006346  005721                        TST     (R1)+           ;NO, INCREMENT 'WAIT' TABLE ADDRESS
006350  000772                        BR      6$              ;AND CONTINUE LOOKING
                                
006352  011100              7$:       MOV     (R1),R0         ;PUT THE DRIVE'S BLOCK ADDRESS IN R0
006354  104401  001203                TYPE    ,$CRLF          ;CR-LF
006360  104401  076003                TYPE    ,DEASSG         ;TYPE 'DRIVE DEASSIGNED'
006364  004737  023714                JSR     PC,SUMARY       ;TYPE THE DRIVE'S PERFORMANCE SUMMARY
006370  104401  075617                TYPE    ,STAR5          ;TYPE '****...ETC'
006374  005015                        CLR     (R5)            ;CLEAR THE 'DROP DRIVE' TABLE ENTRY
006376  004737  020750                JSR     PC,CMPRES       ;COMPRESS THE RESPECTIVE TABLE
006402  000737                        BR      2$              ;SEE IF ANY MORE DRIVES
                                
                                        ;LOOK FOR DRIVES TO BE ASSIGNED
                                
006404  012703  000010      MAIN1:    MOV     #8.,R3          ;DRIVE COUNT
006410  005001                        CLR     R1              ;ASSIGN LIST INDEX
006412  005002                        CLR     R2              ;'AVAIL' INDEX
006414  005005                        CLR     R5              ;NEW DRIVE INDEX
006416  005765  001566      1$:       TST     NEWUNT(R5)      ;NEW DRIVE IN THIS POSITION
006422  001005                        BNE     3$              ;BR IF THERE IS
006424  005725              2$:       TST     (R5)+           ;INCREMENT R5
006426  005201                        INC     R1              ;INCREMENT ASSIGN INDEX
006430  005303                        DEC     R3              ;DECREMENT DRIVE COUNT
```

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

                        001203  3$:     TYPE    ,$CRLF          ;CR-LF
                        075304          TYPE    ,UNTMSG         ;'DRIVE'
                                        MOV     R1,-(SP)        ;;SAVE R1 FOR TYPEOUT
                                                                ;;TYPE DRIVE NUMBER
006450  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
006452     002                          .BYTE   2               ;;TYPE 2 DIGIT(S)
006453     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
                        076053          TYPE    ,ASGND          ;'STARTED'
006454  005762  001610  4$:     TST     AVAIL(R2)               ;AT END OF AVAILABLE TABLE
006460  001402                          BEQ     5$              ;BR IF YES
006464  005722                          TST     (R2)+           ;INCREMENT AVAILABLE TABLE INDEX
006466  000773                          BR      4$              ;CONTINUE LOOKING FOR END OF TABLE
006470  016562  001566  001610  5$:     MOV     NEWUNT(R5),AVAIL(R2)    ;MOVE ADDR OF DRIVE INTO AVAIL LST
006500  005065  001566                  CLR     NEWUNT(R5)      ;TAKE DRIVE OUT OF NEW DRIVE TABLE
006504  156137  040300  001542          BISB    ATABIT(R1),ASNLST       ;SET DRIVE ASSIGNED INDICATOR
006512  005037  032012                  CLR     AUTLST          ;CLEAR AUTO ASSIGN
006516  005722                          TST     (R2)+           ;INCREMENT AVAILABLE TABLE POINTER
006520  000741                          BR      2$              ;LOOK FOR MORE DRIVES

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

006522  005002                  MAIN2:  CLR     R2              ;START FROM THE FIRST LOCATION
006524  105737  001542                  TSTB    ASNLST          ;ANY DRIVES ACTIVE ?
006530  001025                          BNE     2$              ;BR IF YES
006532  105737  001150                  TSTB    $AUTOB          ;RUNNING IN AUTO MODE ?
006536  001020                          BNE     1$              ;BR IF YES
006540  012737  000001  001340          MOV     #1,CFLAG        ;DUMMY 'CONTROL C' FLAG
006546  013737  001314  001352          MOV     HZ,ONESEC       ;RESTORE ONE SECOND COUNTER VALUE
006554  005037  001344                  CLR     HOUR            ;CLEAR THE HOUR'S COUNTER
006560  005037  001346                  CLR     MINUTE          ;CLEAR THE MINUTE'S COUNTER
006564  005037  001350                  CLR     SECOND          ;CLEAR THE SECOND'S COUNTER
006570  005037  001466                  CLR     INTRVL+2        ;CLEAR INTERVAL COUNTER
006574  104401  077103                  TYPE    ,NODRVS         ;TYPE 'NO DRIVES ASSIGNED'
006600  000137  031742          1$:     JMP     $GET42          ;GIVE CONTROL TO MONITOR

006604  005762  001632          2$:     TST     WAIT(R2)        ;ANY DRIVES WAITING FOR THE BUFFER ?
006610  001435                          BEQ     5$              ;BR IF NO
006612  016200  001632                  MOV     WAIT(R2),R0     ;LOAD R0 WITH THE DPB ADDRESS
006616  005046                          CLR     -(SP)           ;CLEAR THE STACK FOR BUFFER REQ
006620  004737  016242                  JSR     PC,GETBUF       ;CALL TO GET THE BUFFER RT.
006624  012660  000006                  MOV     (SP)+,$BUF(R0)  ;IF 0,BUFFER IS STILL NOT AVAILABLE
006630  001423                          BEQ     4$              ;BRANCH IF NO BUFFER AVAILABLE
006632  005060  000122                  CLR     $NEXT(R0)       ;CLEAR PARAMETER SELECT FLAG
006636  005060  000106                  CLR     $FAIR(R0)       ;CLEAR THE FAIR FLAG
006642  004737  016632                  JSR     PC,FILBUF       ;FILL THE BUFFER
006646  004737  016710                  JSR     PC,GODRIV       ;SET COMMAND AND GO
006652  012705  001520                  MOV     #ORDERQ,R5      ;PUT THE WAIT QUE INTO ORDER QUE
006656  005725                  3$:     TST     (R5)+           ;QUE AVAILABLE ?
006660  001376                          BNE     3$              ;BR IF NO
006662  010045                          MOV     R0,-(R5)        ;LOAD THE DPB ADDRESS INTO THE ORDER QUE
006664  012701  001632                  MOV     #WAIT,R1        ;REMOVE THE DRIVE FROM THE 'WAIT' QUE
006670  060201                          ADD     R2,R1           ;OFFSET THE QUE POSITION
006672  004737  020750                  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 007120  001775                          BEQ     2$              ;BR IF DRIVE NOT FINISHED
169 007122  005741                          TST     -(R1)           ;BACKUP THE QUE POINTER
170 007124  010146                          MOV     R1,-(SP)        ;SAVE THE QUE ADDRESS
171 007126  004737  016072                  JSR     PC,STATIS       ;ACCUMULATE STATISTICS FOR DRIVE IN R0
172 007132  004737  007252                  JSR     PC,PROCES       ;PROCESS END OF COMMAND
173 007136  005060  000112                  CLR     $SSENB(R0)      ;CLEAR SKIP SECTORING ENABLED
174 007142  005037  001342                  CLR     BADSEC          ;CLEAR THE BAD TRK/SEC ERROR INDICATOR
175 007146  004737  031264                  JSR     PC,ABNRML       ;SEE IF ANY DRIVES HAVE TOO MANY ERRORS
176 007152  004737  031312                  JSR     PC,$EOP         ;IS IT END OF PASS ?
177 007156  012601                          MOV     (SP)+,R1        ;RESTORE THE ORDER TABLE INDEX
178 007160  012705  001610                  MOV     #AVAIL,R5       ;ADDRESS OF THE 'AVAIL' TABLE
179 007164  005725                  3$:     TST     (R5)+           ;IS IT THE END OF THE 'AVAIL' TABLE ?
180 007166  001376                          BNE     3$              ;BR IF NO
181 007170  011145                          MOV     (R1),-(R5)      ;MOVE THE DPB INTO THE 'AVAIL' TABLE
182 007172  004737  020750                  JSR     PC,CMPRES       ;COMPRESS THE ORDER QUE
183 007176  004737  016376                  JSR     PC,RELBUF       ;RESTORE BUFFER
184
185 007202  032777  000004  171744  4$:     BIT     #SW02,@SWR      ;TYPE PERFORMANCE SUMMARY
186 007210  001016                          BNE     5$              ;BR IF NOT
187 007212  005737  001316                  TST     STATIN          ;TIME TO TYPE THE PERFORMANCE SUMMARY ?
188 007216  001413                          BEQ     5$              ;BR IF NOT
189 007220  005037  001316                  CLR     STATIN          ;CLEAR THE INDICATOR
190 007224  005737  001542                  TST     ASNLST          ;ANY DRIVES ASSIGNED ?
191 007230  001406                          BEQ     5$              ;BR IF NO
192 007232  104401  075525                  TYPE    ,REPHD          ;TYPE PERFORMANCE REPORT HEADING
193 007236  004737  023626                  JSR     PC,STATPR       ;TYPE THE SUMMARY
194 007242  104401  075566                  TYPE    ,STAR30         ;TYPE ''****...ETC''
195 007246  000137  006240          5$:     JMP     MAIN            ;CONTINUE THE LOOP
```

```
                                        ;PROCESS THE COMMAND TERMINATION
                                        ;
007252  111037  001324          PROCES: MOVB    (R0),DRVNO      ;DRIVE NUMBER FOR ANY ERROR MESSAGES
007256  005760  000016                  TST     $STATUS(R0)     ;SEE IF DRIVER SIGNALED AN ERROR
007262  100427                          BMI     ERPROC          ;BR IF ERROR
007264  032760  100000  002140          BIT     #BIT15,$RMCS1(R0)       ;SEE IF 'SC' SET
007272  001410                          BEQ     1$              ;BR IF NOT SET
007274  032760  040000  002140          BIT     #BIT14,$RMCS1(R0)       ;SEE IF 'TRE' SET
007302  001017                          BNE     ERPROC          ;BR IF SET
007304  032760  040000  002152          BIT     #BIT14,$RMDS(R0)        ;SEE IF 'ERR' SET
007312  001013                          BNE     ERPROC          ;BR IF SET
                                        ;
                                        ;NO ERRORS DETECTED IN REGISTERS, DO SOME CHECKING ANYWAY
                                        ;
007314  004737  013062          1$:     JSR     PC,CKERR        ;CHECK ERROR BITS
007320  004737  013154                  JSR     PC,CKBUS        ;CHECK BUS ADDRESS & WORD COUNT
007324  032777  000002  171622          BIT     #SW01,@SWR      ;DO DATA COMPARE ?
007332  001002                          BNE     2$              ;BR IF NO
007334  004737  013240                  JSR     PC,CMPAR        ;COMPARE DATA W/O ERROR
007340  000207                  2$:     RTS     PC              ;RETURN
                                        ;
                                        ;COMMAND TERMINATED WITH AN ERROR - PROCESS THE ERROR
                                        ;
007342  032760  000200  000016  ERPROC: BIT     #BIT07,$STATUS(R0)      ;DONE BIT SET ?
007350  001402                          BEQ     ERPRC1          ;BR IF COMMAND DIDN'T COMPLETE NORMALLY
007352  000137  007550                  JMP     DONE            ;PROCESS ERROR WITH 'DONE' BIT SET
                                        ;
                                        ;PROCESS COMMAND COMPLETION WITH 'ERROR' & 'DONE NOT' BITS
                                        ;
007356                          ERPRC1:
007356  032760  001000  000016          BIT     #BIT09,$STATUS(R0)      ;TIMEOUT?
007364  001011                          BNE     SWTIM           ;BR IF YES
007366  032760  040002  000016          BIT     #BIT14!BIT01,$STATUS(R0) ;DRIVE WENT OFFLINE ?
007374  001024                          BNE     OFLIN           ;BR IF IT DID
007376  032760  000004  000016          BIT     #BIT2,$STATUS(R0)       ;PORT REQUEST TIME OUT ?
007404  001042                          BNE     PRTIM           ;BR IF IT DID
007406  000207                          RTS     PC              ;ERROR. RETURN
                                        ;
                                        ;
                                        ;SOFTWARE TIMEOUT OCCURRED
                                        ;
007410  004737  021134          SWTIM:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
007414  104414  070307                  DISPLY  ,EM13           ;PRINT THE TIME OUT MESSAGE
007420  004737  021214                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
007424  004737  021654                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
007430  004737  022324                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
007434  004737  024740                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
007440  004737  022746                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
007444  000207                          RTS     PC              ;RETURN
                                        ;
                                        ;DRIVE WENT OFFLINE
                                        ;
007446  104401  001203          OFLIN:  TYPE    ,$CRLF          ;CR-LF
007452  004737  021134                  JSR     PC,LINE1        ;PRINT LINE 1 OF THE ERROR MESSAGE
007456  104414  070361                  DISPLY  ,EM14           ;PRINT OFFLINE MESSAGE
007462  004737  021214                  JSR     PC,LINE2        ;PRINT LINE 2 OF THE ERROR MESSAGE
007466  004737  021654                  JSR     PC,LINE3        ;PRINT LINE 3 OF THE ERROR MESSAGE
```

```
                                        JSR     PC,LINE4        ;PRINT LINE 4 OF THE ERROR MESSAGE
                                        JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
                                        JSR     PC,LINE7        ;PRINT LINE 7 OF THE ERROR MESSAGE
                                        JMP     DROP            ;DROP THE DRIVE

                                ;PORT REQUEST TIMEOUT ERROR

                                PRTIM:  JSR     PC,LINE1        ;TYPE LINE 1 OF THE ERROR MESSAGE
                                        DISPLY  ,EM15           ;PRINT PORT TIME OUT MESSAGE
                                        JSR     PC,LINE2        ;TYPE LINE 2 OF THE ERROR MESSAGE
                                        JSR     PC,LINE3        ;TYPE LINE 3 OF THE ERROR MESSAGE
                                        JSR     PC,LINE4        ;TYPE LINE 4 OF THE ERROR MESSAGE
                                        JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
                                        JSR     PC,LINE7        ;TYPE LINE 7 OF THE ERROR MESSAGE
                                        RTS     PC              ;RETURN

                                ;PROCESS COMMAND COMPLETION WITH 'ERROR' & 'DONE' BITS SET

                        000016  DONE:   BIT     #BIT04,$TATUS(R0)       ;UNSAFE OCCURRED
                                        BEQ     1$              ;BR IF NOT
                                        JMP     UNSAF           ;REPORT UNSAFE

                        002154  1$:     BIT     #BIT14,$RMER1(R0)       ;UNSAFE OCCURRED
                                        BEQ     2$              ;BR IF NOT
                                        JMP     UNSAF           ;REPORT UNSAFE

                        002150  2$:     BIT     #BIT14,$RMCS2(R0)       ;IS 'WCE' SET ?
                                        BEQ     3$              ;BRANCH IF NOT SET
                                        JMP     WCKER           ;WRITE CHECK ERROR

                        002152  3$:     BIT     #BIT14,$RMDS(R0)        ;CHECK 'ERR'
                                        BNE     4$              ;BR IF SET
                                        JMP     TRFER           ;PROCESS 'TRE'

                        002154  4$:     BIT     #BIT08,$RMER1(R0)       ;'HCRC' SET?
                                        BEQ     5$              ;BR IF NOT
                                        JMP     HCRCER          ;PROCESS 'HCRC'

                        002154  5$:     BIT     #BIT04,$RMER1(R0)       ;'FMT' SET?
                                        BEQ     6$              ;BR IF NOT SET
                                        JMP     CKFMT           ;CHECK FORMAT ERROR

                        002154  6$:     BIT     #BIT07,$RMER1(R0)       ;'HCE' SET?
                                        BEQ     7$              ;BR IF NOT SET
                                        JMP     CKHCE           ;CHECK 'HCE' ERROR

                        002154  7$:     BIT     #BIT13,$RMER1(R0)       ;'OPI' SET?
                                        BEQ     8$              ;BR IF NOT SET
                                        JMP     OPIER           ;REPORT 'OPI'

                        002154  8$:     BIT     #BIT3,$RMER1(R0)        ;'PAR' SET?
                                        BEQ     9$              ;BR IF NOT SET
                                        JMP     PARER           ;REPORT 'PAR'

                        002154  9$:     BIT     #BIT5,$RMER1(R0)        ;'WCF' SET?
                                        BEQ     10$             ;BR IF NOT SET
                                        JMP     WCFER           ;REPORT 'WCF'
```

```
163
164 007740  032760  002000  002154  10$:    BIT     #BIT10,$RMER1(R0)       ;'IAE' SET?
165 007746  001402                          BEQ     11$                     ;BR IF NOT SET
166 007750  000137  012106                  JMP     IAEER                   ;REPORT 'IAE'
167
168 007754  032760  004000  002154  11$:    BIT     #BIT11,$RMER1(R0)       ;'WLE' SET?
169 007762  001402                          BEQ     12$                     ;BR IF NOT SET
170 007764  000137  012140                  JMP     WLEER                   ;REPORT 'WLE'
171
172 007770  032760  001000  002154  12$:    BIT     #BIT9,$RMER1(R0)        ;'AOE' SET?
173 007776  001405                          BEQ     13$                     ;BR IF NOT SET
174 010000  032760  002000  002152          BIT     #BIT10,$RMDS(R0)        ;'LBT' SET?
175 010006  001401                          BEQ     13$                     ;BR IF NOT SET
176 010010  000207                          RTS     PC                      ;'AOE' & 'LBT' SET, EXIT
177
178 010012  032760  010000  002154  13$:    BIT     #BIT12,$RMER1(R0)       ;SEE IF 'DTE' SET
179 010020  001402                          BEQ     14$                     ;BR IF NOT
180 010022  000137  011772                  JMP     DTEER                   ;REPORT 'DTE' ERROR
181
182 010026  005760  002154          14$:    TST     $RMER1(R0)              ;SEE IF 'DCK' SET
183 010032  100002                          BPL     15$                     ;BR IF NOT
184 010034  000137  010072                  JMP     DCKER                   ;PROCESS 'DCK'
185
186 010040  032760  040000  002202  15$:    BIT     #BIT14,$RMER2(R0)       ;'SKI' SET
187 010046  001006                          BNE     16$                     ;BRANCH IF SKI SET
188 010050  032760  100000  002202          BIT     #BIT15,$RMER2(R0)       ;'BSE' SET ?
189 010056  001004                          BNE     17$                     ;BRANCH IF SO (NO, OTHER ERROR)
190 010060  000137  011200                  JMP     DRVER                   ;REPORT ERROR
191
192 010064  000137  012372          16$:    JMP     SKIER                   ;REPORT SKI ERROR
193 010070  000207                  17$:    RTS     PC                      ;EXIT FROM ERROR ANALYSIS  ROUT.
194
195                                 ;PROCESS DATA ('DCK') CHECK ERROR
196
197 010072  004737  020764          DCKER:  JSR     PC,SPOTCK               ;SEE IF ERROR AT A BAD SECTOR ON THE DISK
198 010076  000207                          RTS     PC                      ;IT IS, DON'T REPORT IT
199
200 010100  032760  000100  002154          BIT     #ECH,$RMER1(R0) ;ECH ERROR SET ?
201 010106  001411                          BEQ     1$                      ;BR IF NO
202 010110  022760  010040  002204          CMP     #10040,$RMEC1(R0)       ;OTHERWISE RPEC1=10040
203 010116  001024                          BNE     2$                      ;REPORT ECC LOGICAL FAILURE
204 010120  004737  021134                  JSR     PC,LINE1                ;FIRST LINE OF ERROR MESSAGE
205 010124  104414  072400                  DISPLY  ,EM52                   ;ECH ERROR - ECC UNCORRECTABLE
206 010130  000451                          BR      7$
207
208 010132  026027  002204  010040  1$:     CMP     $RMEC1(R0),#10040       ;IS POSITION COUNT OVER MAXIMUM ?
209 010140  101013                          BHI     2$                      ;BR IF YES
210 010142  005760  002204                  TST     $RMEC1(R0)              ;POSITION COUNT 0 ?
211 010146  001410                          BEQ     2$                      ;BR IF YES
212 010150  005760  002206                  TST     $RMEC2(R0)              ;VALUE IN PATTERN REGISTER ?
213 010154  001033                          BNE     6$                      ;BR IF YES
214 010156  004737  021134                  JSR     PC,LINE1                ;TYPE FIRST LINE OF ERROR MESSAGE
215 010162  104414  072222                  DISPLY  ,EM47                   ;TYPE 'ECC LOGIC ERROR'
216 010166  000404                          BR      3$
217 010170  004737  021134          2$:     JSR     PC,LINE1                ;TYPE FIRST LINE OF ERROR MESSAGE
218 010174  104414  072062                  DISPLY  ,EM45                   ;TYPE 'ECC LOGIC ERROR'
219 010200  004737  021214          3$:     JSR     PC,LINE2                ;TYPE LINE 2 OF ERROR MESSAGE
```

```
                                                        ;INCREMENT TOTAL ERROR COUNT
                                                        ;RETRY COUNT
                                                        ;RETRY THE COMMAND
                                                        ;RETRY WAS NOT SUCCESSFUL
                                                        ;PRINT 'CORRECTED ON N RETRIES'
                                                        ;FINISH THE ERROR REPORT

                                                        ;PRINT 'UNCORRECTABLE AFTER N RETRIES'
                                                        ;TYPE LINE 7 OF ERROR MESSAGE
                                                        ;RETURN

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

                                                        ;PRINT LINE 1 OF ERROR MESSAGE
                                                        ;DATA CHECK ERROR

                                                        ;PRINT LINE 2 OF ERROR MESSAGE
                                                        ;PRINT LINE 3 OF ERROR MESSAGE
                                                        ;PRINT LINE 4 OF ERROR MESSAGE
                                                        ;SEE IF BAD SECTOR TO BE PRINTED
                                                        ;LOAD ERROR MASK
                                                        ;CHECK 'DTE' & 'ECH'
                                                        ;BR IF SET
                                                        ;PRINT 'SECTOR IS ECC CORRECTABLE'
                                                        ;FINISH THE ERROR REPORT
                                                        ;RETRY COUNT

                                                        ;RETRY
                                                        ;TEST FOR DONE
                                                        ;BR IF NOT DONE
                                                        ;BR IF NOT ERROR
                                                                ;SEE IF COMMAND TERMINIATED NORMALLY
                                                        ;BR IF NOT
                                                        ;INCREMENT TOTAL ERROR COUNT
                                                        ;'DIFFERENT ERROR DURING RETRY'
                                                        ;SEE WHICH ERROR

                                                        ;LOOK AT CURRENT ERROR
                                                        ;BR IF DIFFERENT ERROR
                                                          ;'ECH' OR 'DTE' STILL SET ?
                                                        ;BR IF NEITHER SET
                                                        ;INCREMENT RETRY COUNT
                                                        ;DONE ?
                                                        ;BR IF NOT
                                                        ;PRINT LINE 8 OF ERROR MESSAGE
                                                        ;INCREMENT 'HARD' ERROR COUNT
                                                        ;INCREMENT TOTAL ERROR COUNT
                                                        ;PRINT LINE 7 OF ERROR MESSAGE
                                                        ;PRINT THE BAD SECTOR
                                                        ;CLEAN UP AND RETURN

                                                        ;PRINT 'SECTOR IS ECC CORRECTABLE'
                                                        ;PRINT LINE 5B OF THE ERROR MESSAGE
                                                        ;INCREMENT 'SOFT' ERROR COUNT
                                                        ;CORRECT THE ERROR USING ECC AND CHECK IT
                                                        ;COMPARE THE BUFFER
```

```
                                                ;PRINT 'UNCORRECTABLE AFTER N RETRIES'
                                                ;INCREMENT ERROR COUNT
                                                ;PRINT LINE 6A OF ERROR MESSAGE
                                                ;INCREMENT 'SOFT' ERROR COUNT
                                                ;SET PROCESSING 'DCKER' INDICATOR
                                                ;COMPARE THE BUFFER
                                                ;ERROR IN COMPARE ?
                                                ;BRANCH IF ERROR
                                                ;INCREMENT TOTAL ERROR COUNT
                                                ;'DATA COMPARE OK' MESSAGE
                                                ;PRINT LINE 7 OF ERROR MESSAGE
                                                ;RETURN

                                                ;WRITE CHECK ERROR PROCESSING

                                                ;SEE IF 'DCK' SET ALSO
                                                ;BR IF IT IS
                                                ;PRINT LINE 1 OF ERROR MESSAGE
                                                ;PRINT WCE & DCK NOT
                                                ;CLEAR ERROR MASK
                                                ;PRINT LINE 2 OF ERROR MESSAGE
                                                ;PRINT LINE 3 OF ERROR MESSAGE
                                                ;PRINT LINE 4 OF ERROR MESSAGE
                                                ;PRINT LINE 5 OF ERROR MESSAGE
                                                ;INCREMENT TOTAL ERROR COUNT
                                                ;RETRY LIMIT
                                                ;RETRY THE OPERATION
                                                ;RETRY UNSUCESSFUL
                                                ;PRINT 'CORRECTED ON N RETRIES'
                                                ;FINISH PROCESSING THE ERROR

                                                ;PRINT 'UNCORRECTABLE AFTER N RETRIES'
                                                ;FINISH PROCESSING THE ERROR
                                                ;SEE IF ERROR AT BAD SECTOR ON THE DISK
                                                ;EXIT IF AT BAD SECTOR ON DISK
                                                ;PRINT LINE 1 OF ERROR MESSAGE
                                                ;ASSUME THAT EM22 WILL BE PRINTED
                                                ;DID 'WCK' ALSO SET ?
                                                ;BR IF IT DID
                                                ;MESSAGE FOR 'DCK' AND 'WCK' NOT DURING
                                                ;WRITE CHECK
                                                ;TYPE THE ERROR MESSAGE
                                                ;MESSAGE ADDRESS GOES HERE
                                                ;PRINT LINE 2 OF ERROR MESSAGE
                                                ;PRINT LINE 3 OF ERROR MESSAGE
                                                ;PRINT LINE 4 OF ERROR MESSAGE
                                                ;PRINT LINE 5 OF ERROR MESSAGE
                                                ;ECH SET ALSO ?
                                                ;FINISH PROCESSING THE ERROR
                                                ;RETRY LIMIT - 16 (10)
                                                ;RETRY THE COMMAND
                                                ;COMMAND FINISHED ?
                                                ;BR IF NOT
                                                ;BR IF ERROR ON COMMAND
                                                ;INCREMENT RETRY COUNT
```

```
    349 010766  004737  022706                  JSR     PC,LINE6C       ;PRINT 'CORRECTED ON N RETRIES'
    350 010772  000423                          BR      10$             ;FINISH ERROR PROCESSING
    351
    352 010774  105237  001331          8$:     INCB    RETRY+1         ;INCREMENT RETRY COUNT
    353 011000  123737  001330  001331          CMPB    RETRY,RETRY+1   ;DONE ?
    354 011006  001714                          BEQ     1$              ;BR IF AT RETRY LIMIT
    355 011010  032760  100000  002154          BIT     #BIT15,$RMER1(R0)       ;'DCK' SET
    356 011016  001407                          BEQ     9$              ;BR IF NOT - DIFFERENT ERROR
    357 011020  032760  000100  002154          BIT     #BIT06,$RMER1(R0)       ;'ECH' ALSO SET ?
    358 011026  001347                          BNE     6$              ;BR IF IT IS, RETRY COMMAND
    359 011030  004737  022706                  JSR     PC,LINE6C       ;PRINT 'CORRECTED ON N RETRIES'
    360 011034  000402                          BR      10$             ;FINISH PROCESSING ERROR
    361
    362 011036  004737  023170          9$:     JSR     PC,LINE8        ;PRINT LINE 8 - 'DIFFERENT ERROR '
    363 011042  004737  024740          10$:    JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
    364 011046  004737  022746                  JSR     PC,LINE7        ;FINISH THE ERROR MESSAGE
    365 011052  000207                  11$:    RTS     PC              ;RETURN
    366
    367                                 ;REPORT 'HCRC' ERROR
    368
    369 011054  004737  020764          HCRCER: JSR     PC,SPOTCK       ;SEE IF ERROR AT BAD SECTOR
    370 011060  000446                          BR      3$              ;EXIT IF IT IS
    371 011062  004737  023254                  JSR     PC,READDR       ;READ ERROR SECTOR HEADER
    372 011066  004737  015652                  JSR     PC,READHD       ;GET THE HEAD INFORMATION
    373 011072  004737  021134                  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
    374 011076  104414  070440                  DISPLY  ,EM20           ;REPORT 'HCRC'
    375 011102  004737  021214                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
    376 011106  004737  021654                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
    377 011112  004737  022324                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
    378 011116  004737  022556                  JSR     PC,LINE5A       ;PRINT THE HEADER INFORMATION
    384 011122  004737  024620          1$:     JSR     PC,INCSOF       ;INCREMENT 'SOFT' ERROR COUNT
    385 011126  004737  024740                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
    386 011132  012737  000400  001326          MOV     #BIT8,MASK      ;SET ERROR MASK
    387 011140  012737  000003  001330          MOV     #3,RETRY        ;RETRY LIMIT
    388 011146  004737  015744                  JSR     PC,$RETRY       ;RETRY COMMAND
    389 011152  000405                          BR      2$              ;RETRY NOT SUCESSFUL
    390 011154  004737  022706                  JSR     PC,LINE6C       ;PRINT 'CORRECTED ON N RETRIES'
    391 011160  004737  022746                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
    392 011164  000404                          BR      3$              ;EXIT
    393
    394 011166  004737  022714          2$:     JSR     PC,LINE6D       ;PRINT 'UNCORRECTABLE AFTER N RETRIES'
    395 011172  004737  022746                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
    396 011176  000207                  3$:     RTS     PC              ;RETURN
    397
    398                                 ;REPORT DRIVE ERROR
    399
    400 011200  004737  021134          DRVER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
    401 011204  104414  071055                  DISPLY  ,EM30           ;REPORT DRIVE ERROR
    402 011210  004737  021214                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
    403 011214  004737  021654                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
    404 011220  004737  024740                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
    405 011224  004737  022746                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
    406 011230  000207                          RTS     PC              ;RETURN
    407
    408                                 ;PROCESS FORMAT ('FER') ERROR
    409
    410 011232  032760  000400  002154  CKFMT:  BIT     #BIT8,$RMER1(R0)        ;'HCRC' SET ON ORIGINAL ERROR ?
```

```
011240  001402                          BEQ     1$              ;BR IF NOT SET
011242  000137  011054                   JMP     HCRCER          ;REPORT HCRC ERROR

011246  004737  023254          1$:     JSR     PC,READDR       ;GET CORRECTED TRACK & SECTOR ADDRSSES
011252  004737  015652                  JSR     PC,READHD       ;READ HEADER
011256  032737  000400  067612          BIT     #BIT8,GENREG+RMER1      ;'HCRC' SET WHEN HEADER READ?
011264  001002                          BNE     2$              ;BR IF 'HCRC' SET
011266  000137  012166                   JMP     FMTER           ;NO, ERROR IS 'FMT' ONLY

011272  004737  020764          2$:     JSR     PC,SPOTCK       ;SEE IF ERROR AT BAD SECTOR ON THE DISK
011276  000442                          BR      5$              ;EXIT IF IT IS
011300  004737  021134                  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
011304  104414  070644                  DISPLY  ,EM24           ;HEADER READ ERROR - FMT BIT DROPPED UP
011310  004737  021214                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
011314  004737  021654                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
011320  004737  022324                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
011324  004737  022556          3$:     JSR     PC,LINE5A       ;DISPLAY HEADER
011330  004737  024620                  JSR     PC,INCSOF       ;INCREMENT SOFT ERROR COUNT
011334  004737  024740                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
011340  012737  000020  001326          MOV     #BIT4,MASK      ;SET ERROR MASK
011346  012737  000003  001330          MOV     #3,RETRY        ;RETRY LIMIT
011354  004737  015744                  JSR     PC,$RETRY       ;RETRY THE COMMAND
011360  000405                          BR      4$              ;RETRY NOT SUCESSFUL
011362  004737  022706                  JSR     PC,LINE6C       ;PRINT 'CORRECTED ON N RETRIES'
011366  004737  022746                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
011372  000404                          BR      5$              ;EXIT

011374  004737  022714          4$:     JSR     PC,LINE6D       ;PRINT 'UNCORRECTABLE AFTER N RETRIES'
011400  004737  022746                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
011404  000207                  5$:     RTS     PC              ;RETURN

                                ;PROCESS HEADER COMPARE ('HCE') ERROR

011406  032760  000400  002154  CKHCE:  BIT     #BIT8,$RMER1(R0)        ;HCRC SET ON ORIGINAL ERROR ?
011414  001402                          BEQ     1$              ;BR IF NOT SET
011416  000137  011054                   JMP     HCRCER          ;REPORT HEADER CRC ERROR
011422  004737  023254          1$:     JSR     PC,READDR       ;GET CURRENT SECTOR & TRACK ADDRS
011426  004737  015652                  JSR     PC,READHD       ;READ HEADER OF CURRENT SECTOR
011432  032737  000400  067612          BIT     #BIT8,GENREG+RMER1      ;'HCRC' SET ?
011440  001017                          BNE     3$              ;BR IF SET
011442  013746  101174                  MOV     CYLNDR,-(SP)    ;;PUSH CYLNDR ON STACK
011446  042737  170000  101174          BIC     #170000,CYLNDR  ;CLEAR FORMAT,MFG,USER AND SSF BITS FROM HEADER
011454  026037  002174  101174          CMP     $RMDC(R0),CYLNDR        ;CORRECT CYLINDER ?
011462  001402                          BEQ     2$              ;BR IF IT IS
011464  000137  011614                   JMP     POSER           ;REPORT POSITIONING ERROR
011470                          2$:
011470  012637  101174                  MOV     (SP)+,CYLNDR    ;;POP STACK INTO CYLNDR
011474  000137  012230                   JMP     HCEER           ;REPORT 'HCE' ERROR

011500  004737  020764          3$:     JSR     PC,SPOTCK       ;SEE IF ERROR AT BAD SECTOR
011504  000442                          BR      6$              ;EXIT IF IT IS
011506  004737  021134                  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
011512  104414  070712                  DISPLY  ,EM25           ;HEADER READ ERROR - 'HCE' SET
011516  004737  021214                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
011522  004737  021654                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
011526  004737  022324                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
011532  004737  022556          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 'CORRECTED ON N RETRIES'
                                                JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
                                                BR      6$              ;EXIT

                                        5$:     JSR     PC,LINE6D       ;PRINT 'UNCORRECTABLE AFTER N RETRIES'
                                                JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
                                        6$:     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

                                        OPIER:  JSR     PC,SPOTCK       ;SEE IF ERROR AT BAD SECTOR
                                                RTS     PC              ;RETURN IF IT IS
                                                JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
                                                DISPLY  ,EM31           ;'OPI' ERROR
                                                JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
                                                JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
                                                JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
                                                JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
                                                MOV     #BIT13,MASK     ;ERROR MASK
                                        OPIER1: MOV     #3,RETRY        ;RETRY LIMIT
                                                JSR     PC,$RETRY       ;RETRY THE COMMAND
                                                BR      1$              ;RETRY UNSUCESSFUL
                                                JSR     PC,LINE6C       ;PRINT 'CORRECTED ON N RETRIES'
                                                JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
                                                RTS     PC              ;EXIT

                                        1$:     JSR     PC,LINE6D       ;PRINT 'UNCORRECTABLE AFTER N RETRIES'
                                                JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
                                                RTS     PC              ;RETURN

                                        ;REPORT 'DTE' ERROR

                                        DTEER:  JSR     PC,SPOTCK       ;SEE IF ERROR AT BAD SECTOR
                                                RTS     PC              ;RETURN IF IT IS
                                                JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
                                                DISPLY  ,EM32           ;'DTE' ERROR
                                                JMP     DCKER1          ;FINISH PROCESSING THE 'DTE' ERROR
```

```
                                        ;REPORT 'PAR' ERROR
                                        
012014  004737  021134          PARER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
012020  104414  071205                  DISPLY  ,EM33           ;REPORT 'PAR'
012024  004737  021214                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
012030  004737  021760                  JSR     PC,LINE3E       ;PRINT LINE 3E OF ERROR MESSAGE
012034  004737  022324                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
012040  004737  024740                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
012044  012737  000010  001326          MOV     #BIT03,MASK     ;ERROR MASK
012052  012737  000003  001330          MOV     #3,RETRY        ;RETRY LIMIT
012060  004737  015744                  JSR     PC,$RETRY       ;RETRY COMMAND
012064  000405                          BR      2$              ;RETRY UNSUCESSFUL
012066  004737  022706                  JSR     PC,LINE6C       ;PRINT 'CORRECTED ON N RETRIES'
012072  004737  022746          1$:     JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
012076  000207                          RTS     PC              ;EXIT
                                        
012100  004737  022714          2$:     JSR     PC,LINE6D       ;PRINT 'UNCORRECTABLE AFTER N RETRIES'
012104  000772                          BR      1$              ;FINISH ERROR MESSAGE
                                        
                                        ;REPORT 'IAE' ERROR
                                        
012106  004737  021134          IAEER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
012112  104414  071324                  DISPLY  ,EM35           ;REPORT 'IAE'
012116  004737  021214                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
012122  004737  022046                  JSR     PC,LINE3F       ;PRINT LINE 3F OF ERROR MESSAGE
012126  004737  024740                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
012132  004737  022746                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
012136  000207                          RTS     PC              ;RETURN
                                        
                                        ;REPORT 'WLE' ERROR
                                        
012140  004737  021134          WLEER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
012144  104414  071362                  DISPLY  ,EM36           ;REPORT 'WLE'
012150  004737  021214                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
012154  004737  024740                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
012160  004737  022746                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
012164  000207                          RTS     PC              ;RETURN
                                        
                                        ;REPORT FORMAT ERROR
                                        
012166  004737  021134          FMTER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
012172  104414  070773                  DISPLY  ,EM26           ;FORMAT ERROR
012176  004737  021214                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
012202  004737  021654                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
012206  004737  022324                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
012212  004737  022556                  JSR     PC,LINE5A       ;PRINT LINE 5A OF ERROR MESSAGE
012216  004737  024740                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
012222  004737  022746                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
012226  000207                          RTS     PC
                                        
                                        ;REPORT HEADER COMPARE ERROR
                                        
012230  004737  021134          HCEER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
012234  104414  071020                  DISPLY  ,EM27           ;HEADER COMPARE ERROR
012240  004737  021214                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
012244  004737  021654                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
```

```
                                        JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
                                        JSR     PC,LINE5A       ;PRINT LINE 5A OF ERROR MESSAGE
                                        JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
                                        JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
                                        RTS     PC              ;RETURN

                                ;PROCESS CONTROL/INTERFACE TRANSFER ERROR

                                TRFER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
                                        DISPLY  ,EM40           ;RH CONTROLLER OR UNIBUS TRANSFER ERROR
                                        JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
                                        JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
                                        JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
                                        JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
012322  032760  121400  002150        BIT     #BIT15!BIT13!BIT9!BIT8,$RMCS2(R0)       ;'DLT','UPE','MXF', 'MDPE' SET ?
012330  001415                        BEQ     2$              ;BR IF NONE SET
012332  012737  000003  001330        MOV     #3,RETRY        ;RETRY LIMIT
012340  005037  001326                CLR     MASK            ;CLEAR ERROR MASK
012344  004737  015744                JSR     PC,$RETRY       ;RETRY THE OPERATION
012350  000403                        BR      1$              ;RETURN HERE IF RETRY UNSUCESSFUL
012352  004737  022706                JSR     PC,LINE6C       ;PRINT 'CORRECTED ON N RETRIES'
012356  000402                        BR      2$              ;FINISH THE ERROR REPORT

012360  004737  022714        1$:     JSR     PC,LINE6D       ;PRINT 'UNCORRECTABLE AFTER N RETRIES'
012364  004737  022746        2$:     JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
012370  000207                        RTS     PC

                                ;PROCESS 'SKI' ERRORS

012372  004737  021134        SKIER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
012376  104414  072277                DISPLY  ,EM50           ;'SKI' ERROR
012402  004737  021214                JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
012406  004737  021670                JSR     PC,LINE3B       ;PRINT LINE 3B OF ERROR MESSAGE
012412  004737  024740                JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
012416  004737  024670                JSR     PC,INCSKI       ;INCREMENT 'SKI' ERROR COUNT
012422  004737  023076                JSR     PC,LINE7A       ;PRINT LINE 7A OF ERROR MESSAGE
012426  004737  015534                JSR     PC,RECALT       ;RECALIBRATE
012432  000207                        RTS     PC

                                ;REPORT WRITE CLOCK FAILURE ('WCF')

012434  004737  021134        WCFER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
012440  104414  071262                DISPLY  ,EM34           ;REPORT WRITE CLOCK FAILURE
012444  004737  021214                JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
012450  004737  021662                JSR     PC,LINE3A       ;PRINT LINE 3A OF ERROR MESSAGE
012454  004737  022324                JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
012460  004737  024740                JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
012464  004737  015236                JSR     PC,PRTBAD       ;SEE IF BAD SECTOR TO BE PRINTED
012470  012737  000003  001330        MOV     #3,RETRY        ;RETRY COUNT
012476  012737  000040  001326        MOV     #BIT05,MASK     ;ERROR MASK
012504  004737  015744                JSR     PC,$RETRY       ;RETRY THE COMMAND
012510  000405                        BR      2$              ;RETURN HERE IF RETRY UNSUCESSFUL
012512  004737  022706                JSR     PC,LINE6C       ;PRINT 'CORRECTED ON N RETRIES'
012516  004737  022746        1$:     JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
012522  000207                        RTS     PC

012524  004737  022714        2$:     JSR     PC,LINE6D       ;PRINT 'UNCORRECTABLE AFTER N RETRIES'
```

```
                                        BR      1$

                                ;PROCESS DRIVE UNSAFE ERROR

012532  004737  021134        UNSAF:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
012536  104414  072444                DISPLY  ,EM60           ;REPORT DRIVE UNSAFE
012542  004737  021214                JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
012546  004737  021654                JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
012552  004737  024740                JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
012556  032760  040000  002202        BIT     #BIT14,$RMER2(R0)       ;IS 'SKI' ALSO SET ?
012564  001016                        BNE     2$              ;BR IF YES
012566  012737  040000  001326        MOV     #BIT14,MASK     ;LOAD THE ERROR MASK
012574  012737  000003  001330        MOV     #3,RETRY        ;RETRY COUNT
012602  004737  015744                JSR     PC,$RETRY       ;RETRY THE COMMAND
012606  000403                        BR      1$              ;RETRY WAS UNSUCESSFUL
012610  004737  022706                JSR     PC,LINE6C       ;PRINT 'CORRECTED ON N RETRIES'
012614  000402                        BR      2$              ;CONTINUE WITH ERROR REPORT

012616  004737  022714        1$:     JSR     PC,LINE6D       ;PRINT 'UNCORRECTABLE AFTER N RETRIES'
012622  004737  022746        2$:     JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
012626  032760  040000  002202        BIT     #BIT14,$RMER2(R0)       ;CHECK 'SKI' AGAIN
012634  001001                        BNE     3$              ;BR IF SET
012636  000207                        RTS     PC              ;RETURN
012640  004737  015534        3$:     JSR     PC,RECALT       ;RECALIBRATE
012644  000207                        RTS     PC              ;RETURN

                                ;REPORT AN 'UNKNOWN' DATA PATTERN

012646  105737  001356        NOMTCH: TSTB    FRSTER          ;FIRST ERROR IN THE SECTOR ?
012652  001013                        BNE     1$              ;BR IF NOT OR IF PROCESSING 'DCKER'
012654  004737  021134                JSR     PC,LINE1        ;TYPE LINE 1 OF ERROR MESSAGE
012660  104414  071671                DISPLY  ,EM43           ;'CAN'T MATCH DATA WITH PATTERN'
012664  004737  021214                JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
012670  004737  021662                JSR     PC,LINE3A       ;PRINT LINE 3A OF ERROR MESSAGE
012674  004737  022324                JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
012700  000404                        BR      2$              ;CONTINUE PROCESSING ERROR
012702  104414  071671        1$:     DISPLY  ,EM43           ;'CAN'T MATCH DATA WITH PATTERN'
012706  104414  001203                DISPLY  ,$CRLF          ;CR-LF
012712  104414  074473        2$:     DISPLY  ,LIN9I          ;HEADER FOR DATA PRINTOUT
012716  010146                        MOV     R1,-(SP)        ;ADDRESS OF WORD 1
012720  004737  023202                JSR     PC,LINOCT       ;TYPE WORD 1
012724  104414  075233                DISPLY  ,BLNKS2         ;TYPE 2 BLANKS
012730  012146                        MOV     (R1)+,-(SP)     ;ADDRESS OF WORD 1
012732  004737  023202                JSR     PC,LINOCT       ;TYPE WORD 1
012736  104414  001203                DISPLY  ,$CRLF          ;CR-LF
012742  010146                        MOV     R1,-(SP)        ;ADDRESS OF WORD 2
012744  004737  023202                JSR     PC,LINOCT       ;TYPE WORD 2
012750  104414  075233                DISPLY  ,BLNKS2         ;TYPE 2 BLANKS
012754  012146                        MOV     (R1)+,-(SP)     ;ADDRESS OF WORD 2
012756  004737  023202                JSR     PC,LINOCT       ;TYPE WORD 2
012762  104414  001203                DISPLY  ,$CRLF          ;CR-LF
012766  010146                        MOV     R1,-(SP)        ;ADDRESS OF WORD 3
012770  004737  023202                JSR     PC,LINOCT       ;TYPE WORD 3
012774  104414  075233                DISPLY  ,BLNKS2         ;TYPE 2 BLANKS
013000  012146                        MOV     (R1)+,-(SP)     ;ADDRESS OF WORD 3
013002  004737  023202                JSR     PC,LINOCT       ;TYPE WORD 3
013006  104414  001203                DISPLY  ,$CRLF          ;CR-LF
```

```
        013012  010146                          MOV     R1,-(SP)        ;ADDRESS OF WORD 4
        013014  004737  023202                  JSR     PC,LINOCT       ;TYPE WORD 4
        013020  104414  075233                  DISPLY  ,BLNKS2         ;TYPE 2 BLANKS
        013024  012146                          MOV     (R1)+,-(SP)     ;ADDRESS OF WORD 4
        013026  004737  023202                  JSR     PC,LINOCT       ;TYPE WORD 4
        013032  104414  001203                  DISPLY  ,$CRLF          ;CR-LF
    707 013036  062701  000770                  ADD     #<252.*2.>,R1   ;INCREMENT BUFFER POINTER
    708 013042  005002                          CLR     R2              ;CLEAR 'WORDS TO COMPARE' COUNT IN R2
    709 013044  012737  177777  001356          MOV     #-1,FRSTER      ;SET ERROR FOUND INDICATOR
    710 013052  013737  001460  001366          MOV     CMPLMT,LIMIT    ;RESET THE COMPARE ERROR TYPEOUT LIMIT
    711 013060  000207                          RTS     PC              ;RETURN
    712
    713                                 ;CHECK ERROR BITS IN THE RH/RM REGISTERS
    714
    715 013062  032760  060000  002140  CKERR:  BIT     #60000,$RMCS1(R0)       ;SEE IF 'TRE' OR 'MCPE' SET
    716 013070  001012                          BNE     1$              ;BR IT EITHER SET
    717 013072  032760  177400  002150          BIT     #177400,$RMCS2(R0)      ;SEE IF ERROR BITS IN CS2 SET
    718 013100  001006                          BNE     1$              ;BR IF ANY SET
    719 013102  005760  002154                  TST     $RMER1(R0)      ;ANY BITS SET IN ER1
    720 013106  001003                          BNE     1$              ;BR IF ANY SET
    721 013110  005760  002202                  TST     $RMER2(R0)      ;ANY BITS SET IN ER2 ?
    722 013114  001416                          BEQ     2$              ;BR IF NONE SET
    723 013116  004737  021134          1$:     JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
    724 013122  104414  071765                  DISPLY  ,EM44           ;ERROR BITS SET, BUT 'SC' OR 'TRE' NOT SET
    725 013126  004737  021214                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
    726 013132  004737  021654                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
    727 013136  004737  022324                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
    728 013142  004737  024740                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
    729 013146  004737  022746                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
    730 013152  000207                  2$:     RTS     PC              ;RETURN
    731
    732                                 ;CHECK BUS ADDRESS REGISTER & WORD COUNT REGISTER
    733
    734 013154  005760  002142          CKBUS:  TST     $RMWC(R0)       ;CHECK WORD COUNT
    735 013160  001010                          BNE     1$              ;BR IF NOT ZERO
    736 013162  016046  000020                  MOV     $WRDL(R0),-(SP) ;WORD LENGTH
    737 013166  006316                          ASL     (SP)            ;CHANGE INTO BYTE COUNT
    738 013170  066016  000006                  ADD     $BUF(R0),(SP)   ;ADD THE STARTING LOCATION
    739 013174  022660  002144                  CMP     (SP)+,$RMBA(R0) ;BUFFER ADDRESS PROPER ?
    740 013200  001416                          BEQ     2$              ;BR IF OK
    741 013202  004737  021134          1$:     JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
    742 013206  104414  071544                  DISPLY  ,EM41           ;BUS ADDRESS OR WORD COUNT INCORRECT
    743 013212  004737  021214                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
    744 013216  004737  021712                  JSR     PC,LINE3D       ;PRINT LINE 3D OF ERROR MESSAGE
    745 013222  004737  022324                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
    746 013226  004737  024740                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
    747 013232  004737  022746                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
    748 013236  000207                  2$:     RTS     PC
```

```
                                        ;COMPARE THE BUFFER
                                        
013240  005037  001356          CMPAR:  CLR     FRSTER          ;CLEAR 'FIRST ERROR' INDICATOR
                                        
013244  132760  000004  000024  CMPARD: BITB    #4,$CODE(R0)    ;SEE IF READ COMMAND
013252  001001                          BNE     1$              ;BR IF IT IS
013254  000207                          RTS     PC              ;RETURN
                                        
013256  005037  001364          1$:     CLR     ERCTR           ;CLEAR THE ERROR COUNTER
013262  016001  000006                  MOV     $BUF(R0),R1     ;BUFFER ADDRESS
013266  016037  000020  001370          MOV     $WRDL(R0),CMCNT ;WORD COUNT TO WORKING LOCATION
013274  066037  002142  001370          ADD     $RMWC(R0),CMCNT ;CALCULATE ACTUAL WORDS TRANSFERED
013302  001001                          BNE     2$
013304  000207                          RTS     PC              ;EXIT--NO WORDS XFERED
                                        
013306  016037  000012  001372  2$:     MOV     $CYL(R0),CMCYL  ;CYLINDER ADDRESS WORKING LOCATION
013314  052737  170000  001372          BIS     #170000,CMCYL   ;SET MFG, USER, SSF  AND FMT BITS
013322  016037  000010  001374          MOV     $SEC(R0),CMSEC  ;SECTOR & TRACK ADDRESSES TO WORKING LOCNS
013330  013737  001460  001366          MOV     CMPLMT,LIMIT    ;DISPLAY LIMIT
013336  005237  001366                  INC     LIMIT           ;CONVERT PARAMETER INTO LIMIT VALUE
013342  012737  177777  001354  CMSTR:  MOV     #-1,ZROIND      ;CLEAR THE 'ZERO'S' INDICATOR
013350  005037  001360                  CLR     SAVER1          ;CLEAR THE R1 SAVE WORD
013354  005037  001362                  CLR     SAVER5          ;CLEAR THE R5 SAVE WORD
013360  023760  001370  000022          CMP     CMCNT,$SSEC(R0) ;IS BUFFER SIZE GREATER THAN ONE SECTOR ?
013366  101005                          BHI     1$              ;BR IF IT IS
013370  013702  001370                  MOV     CMCNT,R2        ;LESS THAN, USE REMAINING BUFFER
013374  005037  001370                  CLR     CMCNT           ;SET COUNTER TO 0
013400  000405                          BR      2$
013402  016002  000022          1$:     MOV     $SSEC(R0),R2    ;COMPARE SECTOR
013406  166037  000022  001370          SUB     $SSEC(R0),CMCNT ;DECREMENT WORD COUNT
013414  126027  000024  000005  2$:     CMPB    $CODE(R0),#5    ;READ HEADER & DATA?
013422  001026                          BNE     CMDAT           ;BR IF NOT
                                        
                                        ;COMPARE HEADER WORDS
                                        
013424  012705  001372          CMHED:  MOV     #CMCYL,R5       ;ADDRESS OF COMPARING CYLINDER
013430  052711  170000                  BIS     #170000,(R1)    ;SET BITS INCASE BAD SECTOR ENCOUNTER
013434  022521                          CMP     (R5)+,(R1)+     ;CHECK CYLINDER
013436  001402                          BEQ     1$              ;BR IF COMPARE OK
013440  004737  013466                  JSR     PC,CMSTR2       ;REPORT ERROR
013444  022521                  1$:     CMP     (R5)+,(R1)+     ;COMPARE SECTOR/TRACK
013446  001402                          BEQ     2$              ;BR IF EQ
013450  004737  013466                  JSR     PC,CMSTR2       ;REPORT ERROR
013454  162702  000002          2$:     SUB     #2,R2           ;SUBTRACT HEADER LENGTH FROM SIZE
013460  003007                          BGT     CMDAT           ;BR IF NOT FINISHED
013462  000137  014052                  JMP     CMPRX           ;COMPARE THE DATA PORTION
                                        
013466  005237  001364          CMSTR2: INC     ERCTR           ;INCREMENT THE ERROR COUNT
013472  004737  014060                  JSR     PC,CMPRT        ;REPORT THE COMPARISON ERROR
013476  000207                          RTS     PC              ;CHECK THE REST OF THE HEADER
                                        
                                        ;COMPARE DATA FIELD
                                        
013500  004737  027462          CMDAT:  JSR     PC,GETLMT       ;GET ADDRESS LIMITS
013504  004737  014402                  JSR     PC,MATCH        ;FIND THE PATTERN
013510  000403                          BR      1$              ;FOUND A PATTERN
013512  004737  012646                  JSR     PC,NOMTCH       ;RETURN HERE IF NO MATCH WITH PATTERN MADE
```

```
                                                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$:     CMPB    $CODE(R0),#5    ;READ HEAD AND DATA ?
                                                BEQ     9$              ;YES
                                        8$:     MOV     CMCNT,R2        ;SET COUNTER = REMAIN BUFFER LENGTH
                                                CMP     R2,#4           ;IS THERE AT LEAST 4 WORDS TO MATCH PATTERN ?
                                                BLT     CMPRX           ;BR IF NO
                                                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$:     CMP     CMCNT,#2        ;IS THERE AT LEAST 2 WORDS TO COMPARE HEADER ?
                                                BLT     CMPRX           ;BR IF NO
                                                INCB    CMSEC           ;INCREMENT COUNTER
                                                CMPB    CMSEC,SECLMT    ;MAX SECTOR # ?
                                                BLOS    10$             ;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     #170000,(SP)    ;SAVE ONLY THE CYLINDER BITS
```

```
                                                        ;LAST CYLINDER ?
                                                        ;NO
                                                        ;NORMAL RETURN,NOT WRAP AROUND

                                                        ;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
                                                        ;PRINT 'DATA COMPARISON ERRORS'
                                                        ;CR-LF
                                                        ;PRINT '         EXPCTD  RECEVD
                                                        ;            LOC      DATA    DATA'

                                                        ;SET FIRST ERROR FLAG
                                                        ;RETURN
```

```
                                                ;TYPEOUT LIMIT REACHED ?
014236  005737  001366          8$:   TST   LIMIT
014242  001403                        BEQ   9$            ;BR IF IT HAS
014244  005337  001366                DEC   LIMIT         ;DECREMENT LIMIT COUNTER
014250  001005                        BNE   10$           ;BR IF NOT AT LIMIT
014252  032777  000200  164674  9$:   BIT   #SW07,@SWR     ;PRINT ALL DATA COMPARE ERRORS ?
014260  001001                        BNE   10$           ;BR IF YES
014262  000207                        RTS   PC            ;RETURN

014264  010146                10$:   MOV   R1,-(SP)      ;BUFFER ADDRESS
014266  162716  000002                SUB   #2,(SP)       ;ADJUST ADDRESS
014272  004737  023202                JSR   PC,LINOCT     ;TYPE IT
014276  104414  075233                DISPLY .BLNKS2      ;TYPE 2 BLANKS
014302  016546  177776                MOV   -2(R5),-(SP)  ;PUT GOOD DATA ON THE STACK
014306  004737  023202                JSR   PC,LINOCT     ;TYPE IT
014312  104414  075233                DISPLY .BLNKS2      ;TYPE 2 BLANKS
014316  016146  177776                MOV   -2(R1),-(SP)  ;BAD DATA
014322  004737  023202                JSR   PC,LINOCT     ;TYPE IT
014326  104414  001203                DISPLY .$CRLF       ;CR-LF
014332  000207                        RTS   PC            ;RETURN

                                ;LAST LINE OF COMPARE ERROR REPORTING

014334  105737  001357          ENDCMP: TSTB  FRSTER+1    ;ANY COMPARE ERRORS FOUND ?
014340  001417                        BEQ   2$            ;BR IF NOT
014342  005737  001364                TST   ERCTR         ;SEE HOW MANY ERRORS
014346  001410                        BEQ   1$            ;BR IF ONLY CAN'T MATCH PATTERN
014350  104414  074531                DISPLY .LIN9E       ;'NUMBER OF ERRORS='
014354  013746  001364                MOV   ERPCTR,-(SP)  ;NUMBER OF ERRORS
014360  004737  023234                JSR   PC,LINDEC     ;TYPE IT
014364  104414  001203                DISPLY .$CRLF       ;CR-LF
014370  004737  024740          1$:   JSR   PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
014374  004737  022746                JSR   PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
014400  000207                  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

014402  010146                MATCH:  MOV   R1,-(SP)      ;SAVE R1 ON THE STACK
014404  013704  001472                MOV   PATTERN,R4    ;WAS RANDOM PATTERN ENABLED ?
014410  001402                        BEQ   1$            ;BR IF YES
014412  006304                        ASL   R4            ;* 2
014414  000416                        BR    4$            ;USE KNOWN PATTERN
014416  012704  000044          1$:   MOV   #44,R4        ;PATTERN TABLE INDEX
014422  011601                  2$:   MOV   (SP),R1       ;RELOAD R1
014424  162704  000002                SUB   #2,R4         ;DECREMENT INDEX
014430  001413                        BEQ   5$            ;BR IF PATTERN NOT MATCH
014432  016405  002324                MOV   SINDAT(R4),R5 ;ADDRESS OF PATTERN ADDRESS
014436  012703  000004                MOV   #4,R3         ;NUMBER OF LOCATIONS TO CHECK
014442  022125                  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

                                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
                                        CMPB    #5,$CODE(R0)    ;WAS OPERATION, READ HEAD & DATA
                                        BNE     3$              ;BR IF NOT
                                        ADD     #4,ECSEC        ;ADD HEADER SIZE (IN BYTES) BACK IN
                                        BR      3$              ;GO ADJUST THE STACK POINTER
                                2$:     SUB     #256.*2,ECSEC   ;SUBTRACT SECTOR DATA FIELD SIZE (IN BYTES)
                                3$:     ADD     #4,SP           ;ADJUST THE STACK POINTER
                                        MOV     $RMEC1(R0),ECBIT        ;ECC POSITION COUNT
                                        DEC     ECBIT           ;ADJUST BIT POSITION
                                        MOV     ECBIT,ECWRD     ;LOAD THE WORD COUNT LOCATION
                                        BIC     #^C17,ECBIT     ;SAVE THE BIT OFFSET COUNT
                                        BIC     #17,ECWRD       ;CLEAR THE BIT OFFSET
                                        ASR     ECWRD           ;CHANGE TO BYTE COUNT(DIVIDE BY 2)
                                        ASR     ECWRD           ;CHANGE TO BYTE COUNT(DIVIDE BY 4)
                                        ASR     ECWRD           ;CHANGE TO BYTE COUNT(DIVIDE BY 8.)
                                        DISPLY  ,LIN10A         ;'ERROR BURST BEGINS AT '
                                        MOV     ECWRD,-(SP)     ;PUT THE WORD COUNT ON THE STACK
                                        ASR     (SP)            ;GET STARTING WORD FOR MESSAGE(DIVIDE BY 16.)
                                        JSR     PC,$SB2D        ;CONVERT THE WORD COUNT TO DECIMAL
                                        JSR     PC,$SUPRL       ;AND PRINT IT
                                        DISPLY  ,LIN10B         ;' IN DATA FIELD OF ERROR SECTOR'
                                        ADD     ECSEC,ECWRD     ;FIND THE BEGINNING OF THE ERROR BURST
                                        CMP     $RMBA(R0),ECWRD ;SEE IF BURST WAS IN DATA READ
                                        BHI     4$              ;BR IF IN DATA READ
                                        JMP     ECC2            ;NOT IN DATA READ - REPORT IT

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

```
                                                                ;SAVE THE INCORRECT WORD
                                                                ;PUT LOWER MASK ON STACK
                                                                ;CLEAR ERRONEOUS ONE BITS FROM MASK
                                                                ;CLEAR ERRONEOUS ONE BITS FROM BAD WORD
                                                                ;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
```

```
342 015252  016046  000020                    MOV     $WRDL(R0),-(SP) ;FIND THE BEGINNING OF THE SECTOR
343 015256  066016  002142                    ADD     $RMWC(R0),(SP)  ;SUBTRACT THE WORDS NOT TRANSFERED
344 015262  001002                            BNE     1$
345 015264  005726                            TST     (SP)+           ;RESTORE STACK
346 015266  000207                            RTS     PC              ;EXIT--NO WORDS XFERRED
347 015270  005046                    1$:     CLR     -(SP)           ;MAKE THE UPPER DIVIDEND 0
348 015272  016046  000022                    MOV     $SSEC(R0),-(SP) ;DIVIDE THE WORDS XFERED BY THE SECTOR SIZE
349 015276  004737  032040                    JSR     PC,$DIV         ;DIVIDE
350 015302  005716                            TST     (SP)            ;REMANDER = 0 ?
351 015304  001403                            BEQ     2$              ;BR IF IT IS - COMPLETE SECTOR TRANSFERED
352 015306  006316                            ASL     (SP)            ;CONVERT THE RESIDUAL SECTOR INTO BYTE COUNT
353 015310  161601                            SUB     (SP),R1         ;SUBTRACT IT FROM THE END ADDRESS
354 015312  000410                            BR      3$              ;FINISH THE SIZING
355 015314  162701  001000            2$:     SUB     #256.*2,R1      ;SUBTRACT FULL SECTOR FROM END ADDR (IN BYTES)
356 015320  122760  000005  000024            CMPB    #5,$CODE(R0)    ;WAS OPERATION READ HEADER & DATA ?
357 015326  001002                            BNE     3$              ;BR IF NOT
358 015330  162701  000004                    SUB     #4,R1           ;SUBTRACT HEADER SIZE FROM ADDR
359 015334  062706  000004            3$:     ADD     #4,SP           ;RESTORE THE STACK POINTER
360 015340  104414  001203                    DISPLY  ,$CRLF          ;CR-LF
361 015344  104414  075116                    DISPLY  ,LIN11H         ;PRINT THE HEADER
362 015350  122760  000005  000024            CMPB    #5,$CODE(R0)    ;WAS OPERATION READ HEADER & DATA ?
363 015356  001021                            BNE     4$              ;BR IF NOT
364 015360  104414  075171                    DISPLY  ,LIN11          ;TYPE 'ADDR    HEADER'
365 015364  010146                            MOV     R1,-(SP)        ;PUT THE ADDRESS ON THE STACK
366 015366  004737  023202                    JSR     PC,LINOCT       ;TYPE THE ADDRESS
367 015372  104414  075232                    DISPLY  ,BLNKS3         ;TYPE 3 BLANKS
368 015376  012146                            MOV     (R1)+,-(SP)     ;PUT WORD ON STACK
369 015400  004737  023202                    JSR     PC,LINOCT       ;TYPE THE 1ST HEADER WORD
370 015404  104414  075234                    DISPLY  ,BLNKS1         ;TYPE 1 BLANK
371 015410  012146                            MOV     (R1)+,-(SP)     ;PUT WORD ON STACK
372 015412  004737  023202                    JSR     PC,LINOCT       ;TYPE THE 2ND HEADER WORD
373 015416  104414  001203                    DISPLY  ,$CRLF          ;CR-LF
374
375 015422  104414  075212            4$:     DISPLY  ,LIN11A         ;TYPE 'ADDR    DATA'
376 015426  012702  000010            5$:     MOV     #8.,R2          ;8. DATA WORDS PER LINE
377 015432  010146                            MOV     R1,-(SP)        ;PUT THE ADDRESS ON THE STACK
378 015434  004737  023202                    JSR     PC,LINOCT       ;TYPE THE ADDRESS
379 015440  104414  075233                    DISPLY  ,BLNKS2         ;TYPE 2 BLANKS
380 015444  020160  002144            6$:     CMP     R1,$RMBA(R0)    ;PRINTED ALL THE SECTOR ?
381 015450  001412                            BEQ     7$              ;BR IF ALL PRINTED
382 015452  104414  075234                    DISPLY  ,BLNKS1         ;TYPE 1 BLANK
383 015456  012146                            MOV     (R1)+,-(SP)     ;PUT THE DATA ON THE STACK
384 015460  004737  023202                    JSR     PC,LINOCT       ;TYPE THE DATA
385 015464  005302                            DEC     R2              ;DECREMENT THE HORIZONTAL COUNT
386 015466  001366                            BNE     6$              ;BR IF NOT AT THE END OF THE LINE
387 015470  104414  001203                    DISPLY  ,$CRLF          ;CR-LF
388 015474  000754                            BR      5$              ;RESTORE THE WORDS/LINE COUNT
389 015476  104414  001203            7$:     DISPLY  ,$CRLF          ;CR-LF
390 015502  104414  001203                    DISPLY  ,$CRLF          ;CR-LF
391 015506  000207                    8$:     RTS     PC              ;RETURN
392
393                                           ;ROUTINE TO DO AN RTC - DRIVE SELECTED IN R0
394                                           ;CALL:
395                                           ;       MOV     #DPB,R0         ;DPB ADDRESS
396                                           ;       JSR     PC,RTNCTR
397                                           ;       RETURN
398
```

```
    399 015510  111037  067556          RTNCTR: MOVB    (R0),GENDPB     ;MOVE THE DRIVE # TO THE GENERAL DPB
    400 015514  112737  000117  067560          MOVB    #RTC,GENDPB+$COMND      ;COMMAND CODE
    401 015522  004037  041000          1$:     JSR     R0,RM80         ;DRIVER ENTRANCE
    402 015526  067556                          GENDPB                  ;DPB ADDRESS FOR COMMAND
    403 015530  000774                          BR      1$              ;DRIVER DIDN'T ACCEPT COMMAND
    404 015532  000207                          RTS     PC              ;RETURN
    405
    406                                 ;ROUTINE TO DO A RECALIBRATE USING ACTIVE DPB
    407                                 ;CALL:
    408                                 ;       MOV     #DPB,R0         ;DPB ADDRESS
    409                                 ;       JSR     PC,RECALT
    410                                 ;       RETURN
    411
    412 015534  010037  015560          RECALT: MOV     R0,2$           ;LOAD THE DPB ADDRESS
    413 015540  116060  002140  000027          MOVB    $RMCS1(R0),$PREVO(R0)   ;SAVE THE PREVIOUS COMMAND
    414 015546  112760  000107  000002          MOVB    #RECAL,$COMND(R0)       ;LOAD THE NEW COMMAND
    415 015554  004037  041000          1$:     JSR     R0,RM80         ;START THE RECALIBRATE
    416 015560  000000                  2$:     .WORD   0               ;DPB ADDRESS
    417 015562  000774                          BR      1$              ;DRIVER DIDN'T ACCEPT THE COMMAND
    418 015564  005760  000016          3$:     TST     $TATUS(R0)      ;SEE IF FINISHED
    419 015570  001775                          BEQ     3$              ;IF EQ NO
    420 015572  004737  023254                  JSR     PC,READDR       ;DECREMENT THE ADDRESSES
    421 015576  012660  000034                  MOV     (SP)+,$PREVA+2(R0)      ;MOVE THE CYLINDER ADDRESS
    422 015602  112660  000033                  MOVB    (SP)+,$PREVA+1(R0)      ;MOVE THE TRACK ADDRESS
    423 015606  112660  000032                  MOVB    (SP)+,$PREVA(R0)        ;MOVE THE SECTOR ADDRESS
    424 015612  005060  000012                  CLR     $CYL(R0)        ;CLEAR THE CURRENT CYLINDER ADDRESS
    425 015616  005060  000010                  CLR     $SEC(R0)        ;CLEAR THE CURRENT TRK/SEC ADDRESS
    426 015622  000207                          RTS     PC              ;RETURN
    427
    428                                 ;ROUTINE TO A RECAL WITH NO DPB ACTIVE
    429                                 ;CALL:
    430                                 ;       MOVB    #DRIVE,GENDPB   ;DRIVE ADDRESS
    431                                 ;       JSR     PC,RECAL0
    432                                 ;       RETURN
    433
    434 015624  112737  000107  067560  RECAL0: MOVB    #RECAL,GENDPB+$COMND    ;RELCALIBRATE COMMAND
    435 015632  004037  041000          1$:     JSR     R0,RM80         ;DRIVER ENTRANCE
    436 015636  067556                          GENDPB                  ;DPB ADDRESS FOR COMMAND
    437 015640  000774                          BR      1$              ;DRIVER DIDN'T ACCEPT THE COMMAND
    438 015642  005737  067574          2$:     TST     GENDPB+$TATUS   ;SEE IF FINISHED
    439 015646  001775                          BEQ     2$              ;BR IF NOT FINISHED
    440 015650  000207                          RTS     PC
    441
    458                                 ;UTILITY READ HEADER ROUTINE
    459                                 ;CALL:
    460                                 ;       MOV     #DPB,R0         ;DPB ADDRESS
    461                                 ;       MOV     #SECTOR,-(SP)   ;SECTOR ADDRESS
    462                                 ;       MOV     #TRACK,-(SP)    ;TRACK ADDRESS
    463                                 ;       MOV     #CYLINDER,-(SP) ;CYLINDER ADDRESS
    464                                 ;       JSR     PC,READDR
    465                                 ;       RETURN
    466
    467 015652  116637  000004  067567  READHD: MOVB    4(SP),GENDPB+$TRK       ;TRACK ADDRESS
    468 015660  116637  000006  067566          MOVB    6(SP),GENDPB+$SEC       ;SECTOR ADDRESS
    469 015666  016637  000002  067570          MOV     2(SP),GENDPB+$CYL       ;CYLINDER ADDRESS
    470 015674  111037  067556                  MOVB    (R0),GENDPB     ;DRIVE NUMBER
    471 015700  112737  000173  067560          MOVB    #RDHD,GENDPB+$COMND     ;COMMAND
```

```
                                                MOV     #-2,GENDPB+$WCNT        ;WORD CTR = 2
                                1$:             JSR     R0,RM80                 ;DRIVER ENTRANCE
                                                GENDPB                          ;DPB ADDRESS FOR COMMAND
                                                BR      1$                      ;DRIVER DIDN'T ACCEPT COMMAND
                                2$:             TST     GENDPB+$TATUS           ;FINISHED?
                                                BEQ     2$                      ;BR IF NOT
                                                MOV     (SP),6(SP)              ;ADJUST STACK FOR RETURN
                                                ADD     #6,SP                   ;ADJUST RFTRUN POINTER
                                                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'

                                $RETRY: JSR     PC,GODRIV               ;RE-START COMMAND
                                1$:     TST     $TATUS(R0)              ;COMMAND FINISHED?
                                        BEQ     1$                      ;BR IF NOT
                                        BMI     2$                      ;BR IF ERROR
                                        INCB    RETRY+1                 ;INCREMENT RETRY COUNT
                                        ADD     #2,(SP)                 ;INCREMENT RETURN
                                        BR      5$                      ;GO TO EXIT
                                2$:     BIT     #BIT7,$TATUS(R0)                ;DID COMMAND TERMINATE NORMALLY ?
                                        BEQ     7$                      ;BR IF NOT
                                        TST     MASK                    ;IS ERROR MASK 0 ?
                                        BNE     3$                      ;BR IF NOT
                                        TST     $RMER1(R0)              ;MAKE SURE THAT THE DRIVE ERROR REG IS CLEAR
                                        BNE     6$                      ;BR IF NOT
                                        BR      4$                      ;CONTINUE RETRY
                                3$:     BIT     MASK,$RMER1(R0)         ;SAME ERROR?
                                        BEQ     6$                      ;BR IF NOT
                                4$:     INCB    RETRY+1                 ;INCREMENT RETRY COUNT
                                        CMPB    RETRY,RETRY+1           ;DONE ?
                                        BNE     $RETRY                  ;BR IF NOT DONE
                                5$:     RTS     PC                      ;RETURN
                                6$:     JSR     PC,LINE8                ;REPORT DIFFERENT ERROR
                                        JSR     PC,LINE7                ;PRINT LINE 7
                                        TST     (SP)+                   ;ADJUST STACK POINTER FOR DIRECT RETURN
                                        RTS     PC                      ;RETURN
                                7$:     DISPLY  ,LIN8M                  ;'DIFFERENT ERROR DURING RETRY'
                                        JMP     ERPRC1                  ;REPORT THE ERROR
```

```
                                        ;ROUTINE TO UPDATE THE PERFORMANCE SUMMARY STATISTICS
                                        ;CALL:
                                        ;       MOV     #DPB,R0         ;DPB ADDRESS
                                        ;       JSR     PC,STATIS
                                        ;       RETURN
                                        
 016072  032760  000300  000016 STATIS: BIT     #BIT07!BIT06,$STATUS(R0) ;CHECK FOR DATA TERMINATION
 016100  001456                         BEQ     3$              ;BR IF NOT DATA TERMINATION
 016102  016037  002144  016240         MOV     $RMBA(R0),FACTOR        ;STORE THE FINAL BUFFER ADDRESS
 016110  166037  000006  016240         SUB     $BUF(R0),FACTOR ;SUBTRACT THE INITIAL ADDRESS
 016116  001447                         BEQ     3$              ;BR IF NO DATA TRANSFER
 016120  006237  016240                 ASR     FACTOR          ;CONVERT TO A WORD COUNT
                                        
 016124  122760  000002  000024         CMPB    #2,$CODE(R0)    ;SEE IF COMMAND WAS A WRITE
 016132  001404                         BEQ     1$              ;BRANCH IF YES
 016134  122760  000000  000024         CMPB    #0,$CODE(R0)    ;PRESENT OPERATION AN AUTO WRITE CHECK ?
 016142  001012                         BNE     2$              ;BR IF NO
 016144  063760  016240  000060 1$:     ADD     FACTOR,$WRITN(R0)       ;ADD WORDS WRITTEN DURING WRITE DATA
 016152  005560  000062                 ADC     $WRITN+2(R0)    ;DID HIGH WORD OVFLO AFTER ADDING CARRY ?
 016156  102004                         BVC     2$              ;BR IF NO
 016160  005060  000062                 CLR     $WRITN+2(R0)    ;CLEAR HIGH WORD
 016164  005260  000056                 INC     $WTOFL(R0)      ;AND COUNT WRITE OVERFLOW
                                        
 016170  122760  000002  000024 2$:     CMPB    #2,$CODE(R0)    ;SEE IF COMMAND WAS A WRITE
 016176  001417                         BEQ     3$              ;BRANCH IF YES
 016200  063760  016240  000036         ADD     FACTOR,$ENDAT(R0)       ;END OF PASS DATA WORD COUNT
 016206  005560  000040                 ADC     $ENDAT+2(R0)    ;ADD ANY CARRY
 016212  063760  016240  000066         ADD     FACTOR,$READ(R0)        ;UPDATE THE READ WORD COUNT
 016220  005560  000070                 ADC     $READ+2(R0)     ;DID HIGH WORD OVFLO AFTER ADDING CARRY ?
 016224  102004                         BVC     3$              ;BR IF NO
 016226  005060  000070                 CLR     $READ+2(R0)     ;CLEAR HIGH WORD
 016232  005260  000064                 INC     $RDOFL(R0)      ;AND COUNT READ OVERFLOW
 016236  000207                 3$:     RTS     PC
                                        
 016240  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
                                        
 016242  010146                 GETBUF: MOV     R1,-(SP)        ;SAVE R1
 016244  010246                         MOV     R2,-(SP)        ;SAVE R2
 016246  010346                         MOV     R3,-(SP)        ;SAVE R3
 016250  013702  001654                 MOV     BUFTBL,R2       ;NUMBER OF SEPARATE BUFFERS
 016254  001444                         BEQ     5$              ;BR IF NONE AVAILABLE
 016256  012701  001656                 MOV     #BUFTBL+2,R1    ;FIRST ADDRESS OF ALLOCATION TABLE
 016262  026061  000020  000002 1$:     CMP     $WRDL(R0),2(R1) ;SEE IF THERE IS A BLOCK LARGE ENOUGH
 016270  101405                         BLOS    2$              ;BRANCH IF IT IS
 016272  005302                         DEC     R2              ;DECREMENT TABLE COUNT
 016274  001434                         BEQ     5$              ;BR IF THROUGH TABLE
 016276  062701  000004                 ADD     #4,R1           ;INCREMENT TABLE POINTER
 016302  000767                         BR      1$              ;CONTINUE LOOKING
 016304  011166  000010         2$:     MOV     (R1),10(SP)     ;BUFFER ADDRESS TO STACK
```

```
                                                SUB     $WRDL(R0),2(R1) ;ADJUST BUFFER WRD CNT
                                                BEQ     3$              ;BR IF DIFFERENCE IS ZERO
                                                ASL     $WRDL(R0)       ;CONVERT # WORDS TO BYTES
                                                ADD     $WRDL(R0),(R1)  ;MAKE NEW STARTING ADDRESS
                                                ASR     $WRDL(R0)       ;RETURN # BYTES TO WORDS
                                                BR      5$              ;RETURN
                                        3$:     DEC     BUFTBL          ;DECREMENT ENTRIES COUNT
                                                BEQ     5$              ;BR IF ALLOCATION TABLE EMPTY
                                                DEC     R2              ;DECREMENT TABLE COUNT
                                                BEQ     5$              ;BR IF ITEM WERE LAST ENTRY
                                                MOV     R1,R3           ;MOVE TABLE POINTER
                                                ADD     #4,R3           ;POINT TO NEXT ENTRY
                                        4$:     MOV     (R3)+,(R1)+     ;MOVE ITEMS
                                                MOV     (R3)+,(R1)+
                                                DEC     R2              ;DECREMENT TABLE COUNT
                                                BNE     4$              ;CONTINUE IF NOT AT END OF TABLE
                                        5$:     MOV     (SP)+,R3        ;RESTORE R3
                                                MOV     (SP)+,R2        ;RESTORE R2
                                                MOV     (SP)+,R1        ;RESTORE R1
                                                RTS     PC              ;RETURN

                                        ;ROUTINE TO PUT BUFFER BACK IN TABLE
                                        ;CALL:
                                        ;       MOV     #DPB,R0         ;DPB ADDRESS
                                        ;       JSR     PC,RELBUF
                                        ;       RETURN

                                        RELBUF: MOV     R1,-(SP)        ;SAVE R1
                                                MOV     R2,-(SP)        ;SAVE R2
                                                MOV     R3,-(SP)        ;SAVE R3
                                                MOV     R4,-(SP)        ;SAVE R4
                                                MOV     R5,-(SP)        ;SAVE R5
                                                MOV     #BUFTBL+2,R1    ;BEGINNING OF TABLE
                                                MOV     BUFTBL,R2       ;ENTRY COUNT
                                                BEQ     2$              ;BR IF EMPTY TABLE
                                                MOV     $HLDWC(R0),R3   ;TRIAL ADDRESS
                                                ASL     R3              ;CHANGE TO BYTE COUNT
                                                ADD     $BUF(R0),R3     ;ADDRESS OF HIGHER ADJACENT BLOCK
                                        1$:     CMP     (R1),R3         ;UPPER ADJACENT BLOCK
                                                BEQ     3$              ;BR IF YES
                                                ADD     #4,R1           ;INCREMENT POINTER
                                                DEC     R2              ;DECREMENT ENTRY COUNT
                                                BNE     1$              ;CONTINUE SEARCHING
                                                MOV     $BUF(R0),(R1)   ;PUT THE BUFFER BLOCK INTO THE TABLE
                                                MOV     $HLDWC(R0),2(R1)        ;BLOCK WRD CNT
                                                INC     BUFTBL          ;INCREMENT ENTRY COUNT
                                                INC     R2              ;INCREMENT R2 FOR USE LATER
                                                BR      4$              ;SEE IF A LOWER ADJACENT BLOCK IS IN THE TABLE
                                        2$:     MOV     $BUF(R0),(R1)+  ;BLOCK ADDRESS TO TABLE
                                                MOV     $HLDWC(R0),(R1)+        ;WRD CNT TO TABLE
                                                INC     BUFTBL          ;INCREMENT ENTRY COUNT
                                                BR      8$              ;EXIT
                                        3$:     MOV     $BUF(R0),(R1)   ;RELEASED BUFFER IS LOWER ADJACENT
                                                ADD     $HLDWC(R0),2(R1)        ;INCREMENTED WRD CNT
                                        4$:     MOV     R2,-(SP)        ;SAVE R2
                                                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
    173 016710  010046                GODRIV: MOV     R0,-(SP)        ;SAVE R0
    174 016712  010037  016722                MOV     R0,2$           ;CURRENT DPB ADDRESS
    175 016716  004037  041000          1$:   JSR     R0,RM80         ;CALL THE DRIVE HANDLER
    176 016722  000000                  2$:   .WORD   0               ;DRIVE BLOCK ADDRESS GOES HERE
    177 016724  000000                        HALT                    ;DRIVER REJECTED REQUEST
    178 016726  012600                        MOV     (SP)+,R0        ;RESTORE R0
    179 016730  062760  000001  000046       ADD     #1,$OPERC(R0)   ;INCREMENT THE OPERATION COUNT
    180 016736  005560  000050                ADC     $OPERC+2(R0)
    181 016742  026060  000034  000012       CMP     $PREVA+2(R0),$CYL(R0)   ;DID COMMAND REQUIRE A CYLINDER CHANGE ?
    182 016750  001412                       BEQ     3$              ;BR IF NO
    183 016752  062760  000001  000042       ADD     #1,$ENDSK(R0)   ;INCREMENT END OF PASS SEEK COUNT
    184 016760  005560  000044                ADC     $ENDSK+2(R0)    ;ADD ANY CARRY
    185 016764  062760  000001  000052       ADD     #1,$POSIT(R0)   ;INCREMENT SEEK COUNT
    186 016772  005560  000054                ADC     $POSIT+2(R0)    ;ADD ANY CARRY
    187 016776  000207                  3$:   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

017000  004737  027462          WRTPK:  JSR     PC,GETLMT       ;GET ADDRESS LIMITS
017004  005760  000050                  TST     $OPERC+2(R0)    ;IS THIS THE FIRST OPERATION ?
017010  001011                          BNE     1$              ;BR IF NO
017012  005760  000046                  TST     $OPERC(R0)      ;IS THIS THE FIRST OPERATION ?
017016  001006                          BNE     1$              ;BR IF NO
017020  012704  000010                  MOV     #$SEC,R4        ;GET INDEX TO SECTOR ADDR STORAGE IN DPB
017024  004737  020130                  JSR     PC,CKLMTS       ;GO CHECK DISK ADDRESS LIMITS
017030  000462                          BR      3$              ;BR IF NOT AT END OF SEQUENTIAL ADDRESSING
017032  000000                          HALT                    ;SHOULD NOT GET HERE

017034  116060  002140  000027  1$:     MOVB    $RMCS1(R0),$PREVO(R0)   ;SAVE CURRENT PARAMETERS
017042  016060  000010  000032          MOV     $SEC(R0),$PREVA(R0)     ;SAVE PREVIOUS TRACK/SECTOR ADDRESS
017050  016060  000012  000034          MOV     $CYL(R0),$PREVA+2(R0)   ;SAVE PREVIOUS CYLINDER ADDRESS
017056  016060  002146  000010          MOV     $RMDA(R0),$SEC(R0)      ;CURRENT SECTOR & TRACK ADDRESS
017064  016060  002174  000012          MOV     $RMDC(R0),$CYL(R0)      ;CURRENT CYLINDER ADDRESS

017072  032760  001000  002172          BIT     #SSEI,$RMOF(R0) ;IS SSEI STILL SET ?
017100  001402                          BEQ     2$              ;BR IF NOT
017102  005360  000010                  DEC     $SEC(R0)        ;IF SO, THEN BACKUP ONE SECTOR TO REFLECT THE
                                                                ;PROPER ADDRESS TO BE ACCESSED WHEN READING OR
                                                                ;WRITTING THE NEXT SEQUENTIAL SECTOR.

017106  012704  000010          2$:     MOV     #$SEC,R4        ;GET INDEX TO SECTOR ADDR STORAGE IN DPB
017112  004737  020130                  JSR     PC,CKLMTS       ;GO CHECK DISK ADDRESS LIMITS
017116  000427                          BR      3$              ;BR IF NOT AT END OF SEQUENTIAL ADDRESSING
017120  116060  000140  000010          MOVB    MINSEC(R0),$SEC(R0)     ;RESET SECTOR ADDRESS
017126  116060  000134  000011          MOVB    MINTRK(R0),$TRK(R0)     ;RESET TRACK ADDRESS
017134  016060  000130  000012          MOV     MINCYL(R0),$CYL(R0)     ;RESET CYLINDER ADDRESS
017142  112760  000004  000024          MOVB    #4,$CODE(R0)    ;SET CODE TO READ DATA
017150  122760  177776  000026          CMPB    #-2,$PACK(R0)   ;WAS WRITE DATA PACK IN PROGESS ?
017156  001473                          BEQ     8$              ;BR IF YES (START TESTING)
017160  004737  031370                  JSR     PC,EOP2         ;DROP THE DRIVE (NORMAL TERMINATION)
017164  032777  000020  161762          BIT     #SW04,@SWR      ;IS SWITCH 4 SET ?
017172  001471                          BEQ     9$              ;BR IF NO
017174  000744                          BR      2$              ;RE-CHECK FOR BSF & SSF TRACKS

017176  012704  000010          3$:     MOV     #$SEC,R4        ;GET INDEX TO SECTOR STORAGE
017202  013705  001462                  MOV     WRDCNT,R5       ;WORD COUNT IS MAXIMUM
017206  004737  020330                  JSR     PC,CHKWC        ;CHECK WORD COUNT FOR MAXCYL/MAXTRK
017212  010560  000020                  MOV     R5,$WRDL(R0)    ;GET WORD COUNT
017216  042760  000377  000020          BIC     #377,$WRDL(R0)  ;SECTOR BOUNDRY FOR WRITTING
017224  001002                          BNE     4$              ;NO
017226  105260  000021                  INCB    $WRDL+1(R0)     ;SET TO ONE SECTOR
017232  016060  000020  000004  4$:     MOV     $WRDL(R0),$WCNT(R0)     ;STORE FOR 2'S COMPLEMENT WORD
```

MAIN PROGRAM

```
     58 017240  016060  000020  000110          MOV     $WRDL(R0),$HLDWC(R0)    ;HOLD WORD FOR 'RELBUF' ROUTINE
     59 017246  005460  000004                  NEG     $WCNT(R0)               ;CHANGE WORD COUNT TO 2'S COMPLEMENT
     60 017252  012760  000400  000022          MOV     #256.,$SSEC(R0) ;SECTOR SIZE FOR READ
     61
     62 017260  105760  000026                  TSTB    $PACK(R0)               ;READ OR WRITE PACK ?
     63 017264  100407                          BMI     6$                      ;BR IF WRITE
     64 017266  112760  000004  000024  5$:     MOVB    #4,$CODE(R0)    ;CODE FOR READ DATA
     65 017274  112760  000171  000002          MOVB    #RDDAT,$COMND(R0)       ;DRIVE CODE FOR OPERATION
     66 017302  000415                          BR      7$                      ;SET UP FOR EXIT
     67 017304  005737  001440          6$:     TST     RDONLY                  ;LOCKED IN READ ONLY MODE ?
     68 017310  001366                          BNE     5$                      ;BR IF YES
     69 017312  112760  000002  000024          MOVB    #2,$CODE(R0)    ;CODE FOR WRTDAT
     70 017320  112760  000161  000002          MOVB    #WRTDAT,$COMND(R0)      ;OP CODE
     71 017326  004737  020070                  JSR     PC,GETPAT               ;GET PATTERN CODE
     72 017332  110560  000030                  MOVB    R5,$PATTC(R0)   ;PATTERN CODE
     73 017336  012760  177777  000122  7$:     MOV     #-1,$NEXT(R0)   ;SET PARAMETERS SELECTED INDICATOR
     74 017344  000207                          RTS     PC                      ;RETURN
     75
     76 017346  005037  001320          8$:     CLR     PACK                    ;SET 'TEST' FLAG
     77 017352  105060  000026                  CLRB    $PACK(R0)               ;SET DPB 'TEST' FLAG
     78 017356  005060  000122          9$:     CLR     $NEXT(R0)               ;CLEAR 'PARAMETER SELECTED' INDICATOR
     79 017362  005726                          TST     (SP)+                   ;CLEAR STACK LEVEL
     80 017364  000137  006240                  JMP     MAIN                    ;JUMP TO MAIN BACKGROUND LOOP
```

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

017370  004737  027462          GENPAR: JSR     PC,GETLMT       ;GET ADDRESS LIMITS
017374  004737  037024                  JSR     PC,$RAND        ;CYCLE THE RANDOM NUMBER GENERATOR
017400  005737  001440                  TST     RDONLY          ;LOCKED IN READ ONLY MODE ?
017404  001016                          BNE     1$              ;BR IF YES
017406  032777  000001  161540          BIT     #SW0,@SWR       ;SEE IF SW0 SET
017414  001012                          BNE     1$              ;BR IF SET - READ ONLY
017416  012705  000010                  MOV     #8.,R5          ;READ/WRITE SELECTION DIVISOR
017422  004737  032014                  JSR     PC,GETREM       ;GET SELECTION VALUE
017426  020537  001476                  CMP     R5,RATIO        ;DETERMINE IF READ OR WRITE
017432  103003                          BHIS    1$              ;BR IF READ
017434  012705  000002                  MOV     #2,R5           ;SELECT WRITE DATA COMMAND
017440  000407                          BR      2$              ;SELECT ADDRESS

017442  013705  037124          1$:     MOV     $LONUM,R5       ;SELECT READ OPERATION CODE
017446  000305                          SWAB    R5              ;SWAP BYTES IN R5
017450  042705  177776                  BIC     #^C1,R5         ;MASK OUT ALL BUT BIT 0
017454  062705  000004                  ADD     #4,R5           ;TABLE OFFSET FOR READ CODE
017460  110560  000114          2$:     MOVB    R5,$NCODE(R0)   ;COMMAND SELECTION CODE TO CONTROL BLOCK
017464  016060  002146  000116          MOV     $RMDA(R0),$NSEC(R0)     ;SECTOR AND TRACK
017472  016060  002174  000120          MOV     $RMDC(R0),$NCYL(R0)     ;CYLINDER NUMBER

017500  032760  001000  002172          BIT     #SSEI,$RMOF(R0) ;IS 'SSEI' STILL SET ?
017506  001402                          BEQ     3$              ;BR IF NOT
017510  005360  000010                  DEC     $SEC(R0)        ;IF SO, THEN BACKUP ONE SECTOR TO REFLECT THE
                                                                ;PROPER ADDRESS TO BE ACCESSED WHEN READING OR
                                                                ;WRITTING THE NEXT SEQUENTIAL SECTOR.

017514                          3$:

017514  005737  001506          THEAD:  TST     RANDOM          ;ENABLE RANDOM ADDRESS SELECT ?
017520  001427                          BEQ     RANCYL          ;YES
017522  005760  000050                  TST     $OPERC+2(R0)    ;IS THIS THE FIRST OPERATION ?
017526  001003                          BNE     1$              ;BR IF NO
017530  005760  000046                  TST     $OPERC(R0)      ;IS THIS THE FIRST OPERATION ?
017534  001405                          BEQ     2$              ;BR IF YES

017536  012704  000116          1$:     MOV     #$NSEC,R4       ;GET INDEX TO SECTOR ADDR STORAGE IN DPB
017542  004737  020130                  JSR     PC,CKLMTS       ;GO CHECK DISK ADDRESS LIMITS
017546  000412                          BR      3$              ;BR IF NOT AT END OF SEQUENTIAL ADDRESSING
017550  116060  000140  000116  2$:     MOVB    MINSEC(R0),$NSEC(R0)    ;RESET SECTOR ADDRESS
017556  116060  000134  000117          MOVB    MINTRK(R0),$NTRK(R0)    ;RESET TRACK ADDRESS
017564  016060  000130  000120          MOV     MINCYL(R0),$NCYL(R0)    ;RESET CYLINDER ADDRESS
017572  000761                          BR      1$              ;RE-CHECK FOR BSF & SSF TRACKS
017574  000137  017746          3$:     JMP     RANSIZ          ;GO CHECK FOR RANDOM WORD SIZE
```

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

                                        RANCYL: MOV     MAXCYL(R0),R5   ;GET MAXIMUM CYLINDER ADDRESS
                                                CMP     MINCYL(R0),R5   ;'MINCYL' AND 'MAXCYL' THE SAME ?
                                                BEQ     1$              ;BR IF THEY ARE
                                                SUB     MINCYL(R0),R5   ;GET NUMBER OF ALLOWABLE CYLINDERS
                                                INC     R5              ;INCREMENT DIFFERENCE TO USE AS DIVISOR
                                                JSR     PC,GETREM       ;GET THE RANDOM AUGMENT
                                                ADD     MINCYL(R0),R5   ;NEW CYLINDER ADDRESS
                                        1$:     MOV     R5,$NCYL(R0)    ;STORE CYLINDER ADDRESS IN DPB

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

                                        RANTRK: MOV     MAXTRK(R0),R5   ;GET MAXIMUM TRACK ADDRESS
                                                CMP     MINTRK(R0),R5   ;'MINTRK' AND 'MAXTRK' THE SAME ?
                                                BEQ     1$              ;BR IF THEY ARE
                                                SUB     MINTRK(R0),R5   ;GET NUMBER OF ALLOWABLE TRACKS
                                                INC     R5              ;INCREMENT DIFFERENCE TO USE AS DIVISOR
                                                JSR     PC,GETREM       ;GET THE RANDOM AUGMENT
                                                ADD     MINTRK(R0),R5   ;NEW TRACK ADDRESS
                                        1$:     MOVB    R5,$NTRK(R0)    ;STORE TRACK ADDRESS IN DPB

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

                                        RANSEC: MOV     MAXSEC(R0),R5   ;GET MAXIMUM SECTOR ADDRESS
                                                CMP     MINSEC(R0),R5   ;'MINSEC' AND 'MAXSEC' THE SAME ?
                                                BEQ     1$              ;BR IF THEY ARE
                                                SUB     MINSEC(R0),R5   ;GET NUMBER OF ALLOWABLE SECTORS
                                                INC     R5              ;INCREMENT DIFFERENCE TO USE AS DIVISOR
                                                JSR     PC,GETREM       ;GET THE RANDOM AUGMENT
                                                ADD     MINSEC(R0),R5   ;NEW SECTOR ADDRESS
                                        1$:     MOVB    R5,$NSEC(R0)    ;STORE SECTOR ADDRESS IN DPB

                                        ;MAKE SURE ADDRESS JUST GENERATED IS NOT 'BSF' OR 'SSF' TRACK

                                                MOV     #$NSEC,R4       ;GET INDEX TO SECTOR ADDR STORAGE IN DPB
                                                JSR     PC,CKLMTS       ;GO CHECK DISK ADDRESS LIMITS
                                                BR      2$              ;BR IF NOT AT END OF SEQUENTIAL ADDRESSING
                                                JSR     PC,$RAND        ;CYCLE THE RANDOM NUMBER GENERATOR
                                                JMP     RANCYL          ;GO GENERATE NEW ADDRESS
                                        2$:

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

                                        RANSIZ: MOV     WRDCNT,R5       ;GET MAX WORD COUNT
                                                TST     RANDWC          ;SELECT A RANDOM WORD COUNT ?
                                                BNE     2$              ;BR IF NOT
                                                INC     R5              ;INCREMENT THE MAXIMUM WRD CNT
                                                JSR     PC,GETREM       ;DIVIDE BY MAX VALUE
                                                CMP     R5,#6           ;WORD COUNT LESS THAN 6 ?
                                                BGE     2$              ;BR IF NO
                                        1$:     JSR     PC,$RAND        ;CYCLE THE RANDOM NUMBER GENERATOR
                                                BR      RANSIZ

                                        2$:     MOV     #$NSEC,R4       ;GET INDEX TO SECTOR STORAGE
                                                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.
020012  122760  000002  000114  3$:     CMPB    #2,$NCODE(R0)       ;WRITE OPERATION ?
020020  001005                          BNE     4$                  ;BR IF NO
020022  042705  000377                  BIC     #377,R5             ;WRITTING PARTIAL SECTOR ?
020026  001002                          BEQ     4$                  ;BR IF NO, ELSE,
020030  012705  000400                  MOV     #256.,R5            ;WRITE AT LEAST ONE SECTOR
020034  010560  000020          4$:     MOV     R5,$WRDL(R0)        ;WORD COUNT

                                ;GET A RANDOM PATTERN NUMBER

020040  122760  000002  000114  RANPAT: CMPB    #2,$NCODE(R0)       ;WRITE OPERATION ?
020046  001004                          BNE     RANXIT              ;BR IF NO
020050  004737  020070                  JSR     PC,GETPAT           ;GET PATTERN CODE
020054  110560  000115                  MOVB    R5,$NPATC(R0)       ;MOVE PATTERN CODE TO CONTROL BLOCK
020060  012760  177777  000122  RANXIT: MOV     #-1,$NEXT(R0)       ;SET PARAMETERS SELECTED INDICATOR
020066  000207                          RTS     PC                  ;RETURN

                                ;ROUTINE TO SELECT A PATTERN

020070  012705  000020          GETPAT: MOV     #16.,R5             ;SELECT PATTERN
020074  005737  001472                  TST     PATTERN             ;ENABLE RANDOM PATTERN SELECTION ?
020100  001403                          BEQ     1$                  ;YES
020102  013705  001472                  MOV     PATTERN,R5          ;USE INDEXED PATTERN
020106  000406                          BR      2$                  ;NO
020110  004737  037024          1$:     JSR     PC,$RAND            ;CYCLE THE RANDOM NUMBER GENERATOR
020114  004737  032014                  JSR     PC,GETREM           ;GET CODE
020120  005705                          TST     R5                  ;WAS PATTERN ZERO SELECTED ?
020122  001762                          BEQ     GETPAT              ;BR IF YES
020124  006305          2$:     ASL     R5                  ;MAKE CODE INTO TABLE INDEX
020126  000207                          RTS     PC
```

```
                                        ;THIS ROUTINE IS USED TO CHECK ADDRESS LIMITS BEFORE THE NEXT COMMAND
                                        ;IS PERFORMED. THIS WILL PROTECT AGAINST WRITTING ON CUSTOMER DATA BY
                                        ;CHECKING FOR MINIMUM ADDRESS VALUES. 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

020130  060004                 CKLMTS: ADD     R0,R4           ;POINT TO SECTOR STORAGE POINT IN DPB
020132  026460  000002  000130          CMP     2(R4),MINCYL(R0)        ;IS CYLINDER ADDRESS BELOW MIN. ?
020140  002003                         BGE     1$              ;BR IF NO
020142  016064  000130  000002          MOV     MINCYL(R0),2(R4)        ;RESET CYLINDER TO MIN.
020150  126460  000001  000134  1$:     CMPB    1(R4),MINTRK(R0)        ;IS TRACK ADDRESS BELOW MIN. ?
020156  002003                         BGE     2$              ;BR IF NO
020160  116064  000134  000001          MOVB    MINTRK(R0),1(R4)        ;RESET TRACK TO MIN.
020166  121460  000140          2$:     CMPB    (R4),MINSEC(R0) ;IS SECTOR ADDRESS BELOW MIN. ?
020172  002002                         BGE     3$              ;BR IF NO
020174  116014  000140                 MOVB    MINSEC(R0),(R4) ;RESET SECTOR TO MIN.

                                        ;LOOK FOR MAXIMUM LIMITS AND END OF SEQUENTIAL ADDRESSING

020200  121460  000136          3$:     CMPB    (R4),MAXSEC(R0) ;IS SECTOR ADDRESS AT MAXIMUM ?
020204  003404                         BLE     5$              ;BR IF NO
020206  116014  000140                 MOVB    MINSEC(R0),(R4) ;RESET SECTOR ADDRESS
020212  105264  000001          4$:     INCB    1(R4)           ;INCREMENT TO NEXT TRACK ADDRESS
020216  126460  000001  000132  5$:     CMPB    1(R4),MAXTRK(R0)        ;IS TRACK ADDRESS OVER MAXIMUM ?
020224  003407                         BLE     6$              ;BR IF NO
020226  116014  000140                 MOVB    MINSEC(R0),(R4) ;RESET SECTOR ADDRESS
020232  116064  000134  000001          MOVB    MINTRK(R0),1(R4)        ;RESET TRACK ADDRESS
020240  005264  000002                 INC     2(R4)           ;INCREMENT TO NEXT CYLINDER ADDRESS
020244  026460  000002  000126  6$:     CMP     2(R4),MAXCYL(R0)        ;IS CYLINDER ADDRESS OVER MAXIMUM ?
020252  003403                         BLE     7$              ;BR IF NO
020254  062716  000002                 ADD     #2,(SP)         ;ADJUST RETURN TO RESET DISK ADDRESS PARAMETERS
020260  000422                         BR      9$
                                                                ;CHECK NOT TO READ OR WRITE Bad SECTOR TRACK
020262  013746  001430          7$:     MOV     FE1,-(SP)       ;GET FIRST FE CYLINDER (LAST CYL+1)
020266  005316                         DEC     (SP)            ;LOOK AT LAST USER CYLINDER
020270  026426  000002                 CMP     2(R4),(SP)+     ;ARE WE ON LAST USER CYLINDER ?
020274  001004                         BNE     8$              ;BR IF NO
020276  126437  000001  001426         CMPB    1(R4),TRKLMT    ;IS THIS THE BAD SECTOR TRACK ?
020304  001742                         BEQ     4$              ;BR IF YES
                                                                ;CHECK NOT TO READ OR WRITE SKIP SECTOR FILE TRACKS
020306  026437  000002  001430  8$:     CMP     2(R4),FE1       ;ARE WE ON 1ST FE CYLINDER ?
020314  001004                         BNE     9$              ;BR IF NO
020316  126427  000001  000001         CMPB    1(R4),#1        ;ARE WE ON TRACK 0 OR 1 ?
020324  003732                         BLE     4$              ;BR IF YES
020326  000207                 9$:     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

020330  060004                  CHKWC:  ADD     R0,R4           ;POINT TO SECTOR STORAGE POINT IN DPB
020332  105760  000140                  TSTB    MINSEC(R0)      ;ALLOW SPIRAL RD/WRT ?
020336  001023                          BNE     2$              ;BR IF NO
020340  126037  000136  001424          CMPB    MAXSEC(R0),SECLMT       ;ALLOW SPIRAL RD/WRT ?
020346  001017                          BNE     2$              ;BR IF NO
020350  105760  000134                  TSTB    MINTRK(R0)      ;ALLOW SPIRAL RD/WRT ?
020354  001010                          BNE     1$              ;BR IF NO
020356  126037  000132  001426          CMPB    MAXTRK(R0),TRKLMT       ;ALLOW SPIRAL RD/WRT ?
020364  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
020366  026064  000126  000002          CMP     MAXCYL(R0),2(R4)        ;ON MAXIMUM CYLINDER ?
020374  001022                          BNE     4$              ;BR IF NO
020376  126064  000132  000001  1$:     CMPB    MAXTRK(R0),1(R4)        ;ON MAXIMUM TRACK ?
020404  001016                          BNE     4$              ;BR IF NO

020406  111404                  2$:     MOVB    (R4),R4         ;GET STARTING SECTOR ADDRESS
020410  016046  000136                  MOV     MAXSEC(R0),-(SP)        ;GET MAXIMUM SECTOR
020414  160416                          SUB     R4,(SP)         ;GET NUMBER SECTORS TO BE XFERD
020416  005004                          CLR     R4              ;CLEAR R4
020420  062704  000400          3$:     ADD     #256.,R4        ;ADD 1 SECTOR OF WORDS TO R4
020424  005316                          DEC     (SP)            ;DONE ALL SECTORS YET ?
020426  002374                          BGE     3$              ;BR IF NO
020430  005726                          TST     (SP)+           ;RESTORE STACK
020432  020504                          CMP     R5,R4           ;TOO MANY WORDS FOR TRACK ?
020434  003420                          BLE     5$              ;BR IF NO
020436  010405                          MOV     R4,R5           ;YES, CHANGE WORD COUNT
020440  000416                          BR      5$

020442  013746  001430          4$:     MOV     FE1,-(SP)       ;GET FIRST FE CYLINDER (LAST CYL+1)
020446  005316                          DEC     (SP)            ;LOOK AT LAST USER CYLINDER
020450  026426  000002                  CMP     2(R4),(SP)+     ;ARE WE ON LAST USER CYLINDER ?
020454  001010                          BNE     5$              ;BR IF NO
020456  013746  001426                  MOV     TRKLMT,-(SP)    ;GET LAST TRACK
020462  005316                          DEC     (SP)            ;LOOK AT NEXT TO LAST TRACK
020464  005046                          CLR     -(SP)           ;PUSH STACK
020466  116416  000001                  MOVB    1(R4),(SP)      ;GET CURRENT TRACK
020472  022626                          CMP     (SP)+,(SP)+     ;IS IT TRACK BEFORE BAD SECTOR TRACK ?
020474  001744                          BEQ     2$              ;BR IF YES (DON'T ALLOW SPIRAL TO BAD SEC TRK)
020476  000207                  5$:     RTS     PC
```

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

020500  010546                  LODPAR: MOV     R5,-(SP)        ;SAVE R5
020502  105760  000026                  TSTB    $PACK(R0)       ;'R' OR 'W' COMMAND FOR THE DRIVE ?
020506  001106                          BNE     4$              ;BR IF YES
020510  116060  002140  000027          MOVB    $RMCS1(R0),$PREVO(R0)   ;SAVE CURRENT PARAMETERS
020516  142760  177701  000027          BICB    #^C76,$PREVO(R0)        ;STRIP GO,AND IE BITS
020524  132760  000006  000114          BITB    #6,$NCODE(R0)   ;SEE IF NEXT OPERATION IS READ OR WRITE
020532  001007                          BNE     1$              ;BR IF EITHER
020534  016060  000012  000034          MOV     $CYL(R0),$PREVA+2(R0)   ;SAVE STARTING CYLINDER
020542  016060  000010  000032          MOV     $SEC(R0),$PREVA(R0)     ;SAVE STARTING SECTOR AND TRACK
020550  000410                          BR      2$
020552  004737  023254          1$:     JSR     PC,READDR       ;GET THE DECREMENTED SECTOR AND TRACK ADDRESSES
020556  012660  000034                  MOV     (SP)+,$PREVA+2(R0)      ;CYLINDER ADDRESS
020562  112660  000033                  MOVB    (SP)+,$PREVA+1(R0)      ;TRACK ADDRESS
020566  112660  000032                  MOVB    (SP)+,$PREVA(R0)        ;SECTOR ADDRESS

020572  032777  000100  160354  2$:     BIT     #SW06,@SWR      ;SWITCH 6 SET ?
020600  001051                          BNE     4$              ;BR IF SET
020602  116060  000114  000024          MOVB    $NCODE(R0),$CODE(R0)    ;LOGICAL CODE FOR OPERATION
020610  116005  000114                  MOVB    $NCODE(R0),R5   ;LOAD R5 FOR USE AS TABLE INDEX
020614  116560  002076  000002          MOVB    COMTBL(R5),$COMND(R0)   ;COMMAND CODE
020622  122760  000151  000002          CMPB    #WCKD $COMND(R0)        ;IS NEW COMMAND A WRITE CHECK DATA ?
020630  001012                          BNE     3$              ;BR IF NO
020632  122760  000060  000027          CMPB    #60,$PREVO(R0)  ;WAS PREVIOUS COMMAND A WRITE DATA ?
020640  001431                          BEQ     4$              ;BR IF YES
020642  112760  000171  000002          MOVB    #RDDAT,$COMND(R0)       ;CHANGE WRITE CHECK TO READ DATA COMMAND
020650  112760  000004  000024          MOVB    #4,$CODE(R0)    ;CODE NUMBER CHANGED TO READ DATA

020656  116060  000115  000030  3$:     MOVB    $NPATC(R0),$PATTC(R0)   ;PATTERN CODE
020664  016060  000116  000010          MOV     $NSEC(R0),$SEC(R0)      ;TRACK AND SECTOR ADDRESSES
020672  016060  000120  000012          MOV     $NCYL(R0),$CYL(R0)      ;CYLINDER ADDRESS
020700  012760  000400  000022          MOV     #256.,$SSEC(R0) ;INITIAL VALUE OF SECTOR SIZE
020706  132760  000001  000024          BITB    #1,$CODE(R0)    ;HEADER OPERATION ?
020714  001403                          BEQ     4$              ;BR IF NOT
020716  062760  000002  000022          ADD     #2,$SSEC(R0)    ;ADD HEADER SIZE
020724  016060  000020  000004  4$:     MOV     $WRDL(R0),$WCNT(R0)     ;GET WORD COUNT AND
020732  016060  000020  000110          MOV     $WRDL(R0),$HLDWC(R0)    ;HOLD WORD COUNT FOR 'RELBUF' ROUTINE
020740  005460  000004                  NEG     $WCNT(R0)       ;MAKE IT 2'S COMPLEMENT
020744  012605                          MOV     (SP)+,R5        ;RESTORE R5
020746  000207                          RTS     PC              ;RETURN
```

```
                                        ;ROUTINE TO COMPRESS A LIST
                                        ;CALL:
                                        ;       MOV     #ADDRS,R1       ;COMPRESS LIST STARTING AT THIS ADDRESS
                                        ;       JSR     PC,CMPRES
                                        ;       RETURN
                                        CMPRES: MOV     2(R1),(R1)      ;COMPRESS THE TABLE IN R1
                                                BEQ     1$              ;BR WHEN ZERO FOUND
                                                TST     (R1)+           ;INCREMENT R1
                                                BR      CMPRES          ;CONTINUE COMPRESSING TABLE
                                        1$:     RTS     PC              ;RETURN

                                        ;ROUTINE TO DETERMINE OF ERROR IS AT A LOCATION ON THE DISK DEFINED
                                        ;IN THE BAD TRACK/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

020764                                  SPOTCK:
020764  010146                                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
020766  012701  000146                          MOV     #$BDSEC,R1      ;INCREMENT FOR BAD SECTOR TABLE
020772  060001                                  ADD     R0,R1           ;ADD THE BLOCK'S STARTING ADDRESS
020774  005037  001436                1$:      CLR     DEC2            ;ASSUME DECREMENT SECTOR ONCE
021000  005760  000112                          TST     $SSENB(R0)      ;DID ERROR OCCUR DURING SKIP SECTORING ?
021004  001402                                  BEQ     2$              ;BR IF NO
021006  005237  001436                          INC     DEC2            ;DECREMENT SECTOR TWICE
021012  004737  023254                2$:      JSR     PC,READDR       ;DECREMENT THE SECTOR/TRACK ADDRESS
021016  021126                                  CMP     (R1),(SP)+      ;ON THE SAME CYLINDER ?
021020  001023                                  BNE     6$              ;BRANCH IF NOT
021022  122761  177777  000003                 CMPB    #-1,3(R1)       ;ALL BAD TRACKS ?
021030  001002                                  BNE     3$              ;BR IF NO
021032  005726                                  TST     (SP)+           ;ADJUST STACK AND
021034  000403                                  BR      4$              ;GO CHECK SECTORS
021036  122661  000003                3$:      CMPB    (SP)+,3(R1)     ;COMPARE THE TRACK ADDRESS
021042  001013                                  BNE     7$              ;BR IF IT IS NOT EQUAL
021044  122761  177777  000002        4$:      CMPB    #-1,2(R1)       ;ALL BAD SECTORS ?
021052  001002                                  BNE     5$              ;BR IF NO
021054  005726                                  TST     (SP)+           ;ADJUST STACK AND
021056  000413                                  BR      9$              ;CHECK 'MESSAGE'
021060  122661  000002                5$:      CMPB    (SP)+,2(R1)     ;COMPARE THE SECTOR ADDRESS
021064  001003                                  BNE     8$              ;BR IF NOT EQUAL
021066  000407                                  BR      9$              ;CHECK 'MESSAGE'
021070  005726                        6$:      TST     (SP)+           ;CLEAR OFF THE STACK
021072  005726                        7$:      TST     (SP)+           ;INCREMENT THE STACK POINTER
021074  062701  000004                8$:      ADD     #4,R1           ;GO TO THE NEXT LOCATION IN THE TABLE
021100  005711                                  TST     (R1)            ;EMPTY ENTRY OR TERMINATOR ?
021102  100407                                  BMI     10$             ;BR IF YES
021104  000733                                  BR      1$              ;TRY NEXT SECTOR
021106  005737  001504                9$:      TST     MESSAGE         ;PRINT THE ERROR ANYWAY ?
021112  001006                                  BNE     11$             ;BR IF NOT
021114  012737  177777  001342                 MOV     #-1,BADSEC      ;SET THE INDICATOR FOR THE IDENTIFICATION LINE
021122  062766  000002  000002        10$:     ADD     #2,2(SP)        ;INCREMENT THE RETURN
021130                                  11$:
021130  012601                                  MOV     (SP)+,R1        ;;POP STACK INTO R1
021132  000207                                  RTS     PC              ;RETURN
```

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

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

021134  032777  002000  160012  LINE1:  BIT     #SW10,@SWR      ;SWITCH 10 SET ?
021142  001402                          BEQ     1$              ;BR IF NOT
021144  104401  001176                  TYPE    ,$BELL          ;RING THE BELL
021150  032777  020000  157776  1$:     BIT     #SW13,@SWR      ;INHIBIT TYPEOUT ?
021156  001405                          BEQ     2$              ;BR IF NOT
021160  104414  001203                  DISPLY  ,$CRLF          ;CR-LF
021164  104414  001203                  DISPLY  ,$CRLF          ;CR-LF
021170  000410                          BR      3$              ;EXIT
021172  104414  001203          2$:     DISPLY  ,$CRLF          ;CR-LF
021176  104414  001203                  DISPLY  ,$CRLF          ;CR-LF
021202  004737  024764                  JSR     PC,$TIME        ;TYPE THE TIME
021206  104414  075234                  DISPLY  ,BLNKS1         ;TYPE 1 BLANK
021212  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'

021214                          LINE2:
021214  010346                          MOV     R3,-(SP)        ;:PUSH R3 ON STACK
021216  010446                          MOV     R4,-(SP)        ;:PUSH R4 ON STACK
021220  010546                          MOV     R5,-(SP)        ;:PUSH R5 ON STACK
021222  104414  001203                  DISPLY  ,$CRLF          ;CR-LF
021226  005037  021354                  CLR     4$              ;CLEAR MESSAGE ADDRESS STORAGE
021232  005004                          CLR     R4              ;WORKING REGISTER
021234  012737  073272  021354          MOV     #LIN2C,4$       ;ADDRESS OF 'PRSNT COMMAND = ' MSG
021242  116004  002140                  MOVB    $RMCS1(R0),R4   ;GET THE OPCODE
021246  042704  177701                  BIC     #^C76,R4        ;SAVE ONLY SIGNIFICANT BITS
021252  004737  021310                  JSR     PC,1$           ;TYPE THE FIRST MNEMONIC
021256  005737  021360                  TST     5$              ;SEE IF MNEMONIC ENTRY FOUND
021262  001440                          BEQ     LINE2A          ;BR IF NOT
021264  012737  073312  021354          MOV     #LIN2P,4$       ;ADDRESS OF 'PREVS COMMAND = ' MSG
021272  116004  000027                  MOVB    $PREVO(R0),R4   ;PREVIOUS OPERATION CODE
021276  042704  177701                  BIC     #^C76,R4        ;SAVE ONLY SIGNIFICANT BITS
021302  004737  021310                  JSR     PC,1$           ;TYPE THE PREVIOUS MNEMONIC
021306  000426                          BR      LINE2A          ;CONTINUE
021310  005005                  1$:     CLR     R5              ;CLEAR THE TABLE INDEX
021312  126504  002104          2$:     CMPB    OPTBL(R5),R4    ;LOOK FOR THE OPCODE
021316  001405                          BEQ     3$              ;BR WHEN OPCODE COUNT EQUALS OPCODE
021320  105765  002104                  TSTB    OPTBL(R5)       ;LOOK FOR END OF TABLE
021324  100402                          BMI     3$              ;BR IF END
021326  005205                          INC     R5              ;INCREMENT THE POINTER
021330  000770                          BR      2$              ;CONTINUE - NOT END OF TABLE
```

```
                                        3$:     ASL     R5              ;SHIFT INDEX
                                                ASL     R5              ;SHIFT THE INDEX
                                                ASL     R5              ;SHIFT THE INDEX
021340  012737  002124  021360                MOV     #MNTBL,5$       ;ADDRESS OF ASCII TEXT TABLE
021346  060537  021360                        ADD     R5,5$           ;ADD THE INDEX
021352  104414                                DISPLY                  ;TYPE IT
021354  000000                        4$:     .WORD   0               ;ADDRESS OF 'PRESENT' OR 'PREVIOUS' MESSAGE
021356  104414                                DISPLY                  ;TYPE THE OPERATION MNEMONIC
021360  000000                        5$:     .WORD   0               ;ADDRESS OF MESSAGE
021362  000207                                RTS     PC              ;RETURN TO MAIN ROUTINE

021364  005737  001342                LINE2A: TST     BADSEC          ;PRINT THE BAD SECTOR LINE ?
021370  001404                                BEQ     LINE2B          ;BR IF NOT
021372  104414  001203                        DISPLY  ,$CRLF          ;CR-LF
021376  104414  07333_                        DISPLY  ,LIN2S          ;ERROR ADDRESS DEFINED AS BAD AREA
021402  104414  001203                LINE2B: DISPLY  ,$CRLF          ;CR-LF
021406  104414  072644                        DISPLY  ,DH14           ;STANDARD RM REGISTER HEADER
021412  104414  075234                        DISPLY  ,BLNKS1         ;TYPE 1 BLANK
021416  013746  001324                        MOV     DRVNO,-(SP)     ;PUT THE DRIVE NUMBER ON THE STACK
021422  004737  023234                        JSR     PC,LINDEC       ;TYPE DRIVE NUMBER
021426  104414  075233                        DISPLY  ,BLNKS2         ;TYPE 2 BLANKS
021432  012705  073204                        MOV     #DT14,R5        ;REGISTER INDEXES
021436  004737  021620                        JSR     PC,3$           ;PRINT THE REGISTERS
021442  032777  000040  157504                BIT     #SW05,@SWR      ;PRINT THE OPTIONAL REGISTERS ?
021450  001031                                BNE     1$              ;BR IF NOT
021452  104414  072746                        DISPLY  ,DH15
021456  012705  073226                        MOV     #DT15,R5        ;SECOND DATA LINE
021462  004737  021620                        JSR     PC,3$           ;PRINT THEM
021466  104414  073044                        DISPLY  ,DH16
021472  012705  073250                        MOV     #DT16,R5        ;THIRD DATA LINE
021476  004737  021620                        JSR     PC,3$           ;PRINT THE REGISTERS
021502  013746  001234                        MOV     $CPUOP,-(SP)    ;CHECK THE CPU (RH) TYPE
021506  042716  003777                        BIC     #^C174000,(SP)  ;LEAVE THE CPU BITS
021512  022726  030000                        CMP     #30000,(SP)+    ;SEE IF RH70
021516  001006                                BNE     1$              ;BR IF NO
021520  104414  073114                        DISPLY  ,DH17
021524  012705  073264                        MOV     #DT17,R5        ;OPTIONAL FOURTH DATA LINE
021530  004737  021620                        JSR     PC,3$           ;PRINT THE REGISTERS
021534  032760  000100  000016        1$:     BIT     #BIT6,$STATUS(R0)       ;DATA ERROR ?
021542  001422                                BEQ     2$              ;BR IF NOT
021544  016046  000020                        MOV     $WRDL(R0),-(SP) ;TRANSFER WRD CNT
021550  066016  002142                        ADD     $RMWC(R0),(SP)  ;ADD REMAINING WORD COUNT
021554  006316                                ASL     (SP)            ;CONVERT TO AN BYTE INCREMENT
021556  066016  000006                        ADD     $BUF(R0),(SP)   ;BUFFER STARTING ADDRESS
021562  022660  002144                        CMP     (SP)+,$RMBA(R0) ;CORRECT BUFFER ADDRESS ?
021566  001410                                BEQ     2$              ;BR IF YES
021570  104414  072150                        DISPLY  ,EM46           ;'BUS ADDRESS AND WORD COUNT ARE NOT CONSISTENT'
021574  104414  001203                        DISPLY  ,$CRLF          ;CR-LF
021600  004737  021712                        JSR     PC,LINE3D       ;PRINT LINE 3D OF ERROR MESSAGE
021604  004737  022324                        JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
021610                                  2$:
021610  012605                                MOV     (SP)+,R5        ;;POP STACK INTO R5
021612  012604                                MOV     (SP)+,R4        ;;POP STACK INTO R4
021614  012603                                MOV     (SP)+,R3        ;;POP STACK INTO R3
021616  000207                                RTS     PC              ;RETURN TO ERROR PROCESSING ROUTINE
021620  012546                          3$:   MOV     (R5)+,-(SP)     ;PUT THE REGISTER INDEX ON THE STACK
021622  060016                                ADD     R0,(SP)         ;ADD DRIVE'S TABLE ADDRESS
```

ERROR MESSAGE GENERATION ROUTINES

```
                                                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  ,LIN#3          ;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'

                        157234  LINE3D: BIT     #SW05,@SWR      ;SWITCH 5 SET ?
                                        BEQ     1$              ;BR IF IT IS
                                        DISPLY  ,LIN#3          ;'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'

157100  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 '38 & 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'

022260  104414  073475        LIN3.4: DISPLY  ,LINA3          ;PRINT 'ACTUAL'
022264  013746  101174                MOV     CYLNDR,-(SP)    ;ACTUAL CYLINDER
022270  042716  010000                BIC     #BIT12,(SP)     ;CLEAR THE FORMAT BIT
022274  004737  023234                JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
022300  104414  073514                DISPLY  ,LINT3          ;PRINT TRACK
022304  005046                      CLR     -(SP)           ;CLEAR STACK WORD
022306  116016  002147                MOVB    $RMDA+1(R0),(SP)        ;PUT TRACK ON STACK
022312  004737  023234                JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
022316  104414  001203                DISPLY  ,$CRLF
022322  000207                      RTS     PC

                                        ;PRINT LINE 4 OF ERROR MESSAGE
                                        ;'BUFFER ADR = XXXXXX   WRD CNT = XXXX   ACTUAL NMBR WRDS XFRD = XXX'

022324  032760  000100  000016  LINE4:  BIT     #BIT06,$STATUS(R0)      ;DATA ERROR ?
022332  001427                      BEQ     1$              ;BR IF NOT
022334  104414  073614                DISPLY  ,LIN4           ;'PRINT BUFFER'
022340  016046  000006                MOV     $BUF(R0),-(SP)  ;BUFFER ADDR ON STACK
022344  004737  023202                JSR     PC,LINOCT       ;CONVERT TO OCTAL & PRINT
022350  104414  073632                DISPLY  ,LINS4          ;PRINT 'WRD CNT'
022354  016046  000020                MOV     $WRDL(R0),-(SP) ;WORD LENGTH SIZE(WORD COUNT)
022360  004737  023234                JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
022364  104414  073646                DISPLY  ,LINX4          ;'ACTUAL NMBR WRDS XFRD = '
022370  016046  002144                MOV     $RMBA(R0),-(SP) ;VALUE IN BUFFER ADDR REGISTER
022374  166016  000006                SUB     $BUF(R0),(SP)   ;SUBTRACT STARTING ADDRESS
022400  006216                      ASR     (SP)            ;CONVERT INTO A WORD COUNT
022402  004737  023234                JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
022406  104414  001203                DISPLY  ,$CRLF          ;CR-LF
022412  000207                1$:     RTS     PC              ;RETURN

                                        ;PRINT LINE 5 OF ERROR MESSAGE
                                        ;'EXPCTD DATA = XXXXXX   RECEVD DATA = XXXXXX   WORD POS = XXX'

022414  104414  073700        LINE5:  DISPLY  ,LIN5           ;PRINT 'EXPCTD DATA'
022420  162760  000002  002144        SUB     #2,$RMBA(R0)    ;BACK THE ADDRESS UP
022426  013746  001234                MOV     $CPUOP,-(SP)    ;CHECK THE CPU (RH) TYPE
022432  042716  003777                BIC     #^C174000,(SP)  ;LEAVE THE CPU BITS
022436  022726  030000                CMP     #30000,(SP)+    ;SEE IF RH70
022442  001012                      BNE     1$              ;BR IF NO
022444  162760  000004  002144        SUB     #4,$RMBA(R0)    ;BACKUP THE BUFFER POINTER
022452  032760  004000  002212        BIT     #BIT11,$RMCS3(R0)       ;SEE WHICH WORD HALF DIDN'T COMPARE
022460  001403                      BEQ     1$              ;IF EQ, EVEN HALF DIDN'T COMPARE
022462  162760  000002  002144        SUB     #2,$RMBA(R0)    ;BACKUP THE BUFFER POINTER AGAIN
022470  017046  002144        1$:     MOV     @$RMBA(R0),-(SP)        ;'EXPCTD' DATA - AT THE BUFFER LOCATION
022474  004737  023202                JSR     PC,LINOCT       ;TYPE IT
022500  104414  073716                DISPLY  ,LIN85          ;PRINT 'RECEVD DATA'
022504  016046  002162                MOV     $RMDB(R0),-(SP) ;RECEVD DATA FROM BUFFER
022510  004737  023202                JSR     PC,LINOCT       ;TYPE IT
022514  016046  002142                MOV     $RMWC(R0),-(SP) ;WORD LENGTH ON STACK
022520  066016  000020                ADD     $WRDL(R0),(SP)  ;MAKE INTO A POSITIVE NUMBER
022524  005046                      CLR     -(SP)           ;UPPER DIVIDEND TO ZERO
022526  016046  000022                MOV     $SSEC(R0),-(SP) ;SECTOR SIZE ON THE STACK
022532  004737  032040                JSR     PC,$DIV         ;DIVIDE WORDS XFERED BY SECTOR SIZE
022536  012616                      MOV     (SP)+,(SP)      ;MOVE REMAINDER UP THE STACK
022540  104414  073736                DISPLY  ,LINP5          ;PRINT 'WORD 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  ,LINB6          ;ECC CORRECTABLE
                                        DISPLY  ,$CRLF
                                        RTS     PC


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

                                LINE6A: DISPLY  ,LINC6          ;PRINT 'SECTOR READ CORRECTLY ON N RETRIES'
                                        BR      LIN6.2          ;TYPE THE REST OF THE LINE
                                ;PRINT LINE 6C OF THE ERROR MESSAGE
                                ;'CORRECTED ON NTH RETRY'

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

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

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

```
                                        ;RETRY COUNT TYPEOUT

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

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

                                LINE7:  DISPLY  ,LIN7T          ;TOTAL ERRORS
                                        MOV     $TOTAL(R0),-(SP)        ;TO STACK
                                        JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
                                        DISPLY  ,LIN7OX         ;PRINT 'WTOFL'
                                        MOV     $WTOFL(R0),-(SP)        ;;PUSH $WTOFL(R0) ON STACK
                                        JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
                                        DISPLY  ,LIN7X          ;PRINT 'WRDS WRITN'
                                        MOV     #$WRITN,-(SP)   ;ADDRESS OF LOW WORD ON STACK
                                        ADD     R0,(SP)
                                        JSR     PC,$DB2D        ;CONVERT
                                        JSR     PC,$SUPRL       ;PRINT
                                        DISPLY  ,LIN7OR         ;PRINT 'ROFL'
                                        MOV     $RDOFL(R0),-(SP)        ;;PUSH $RDOFL(R0) ON STACK
                                        JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
                                        DISPLY  ,LIN7R          ;'WRDS READ'
                                        MOV     #$READ,-(SP)    ;LOW WORD ADDRESS
                                        ADD     R0,(SP)
                                        JSR     PC,$DB2D        ;CONVERT
                                        JSR     PC,$SUPRL       ;PRINT IT
                                        DISPLY  ,$CRLF          ;CR-LF
                                        BIT     #SW15,@SWR      ;SEE IF 'HALT ON ERROR' - SWITCH 15
                                        BEQ     1$              ;BR IF NOT
                                        HALT                    ;SWITCH 15 HALT
                                1$:     RTS     PC

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

                                LINE7A: DISPLY  ,LIN7P          ;'TOTAL SEEKS = '
                                        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'
                                        
023170  104414  074343          LINE8:  DISPLY  ,LIN8M
023174  004737  021214                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
023200  000207                          RTS     PC
                                        
                                        ;OCTAL TYPEOUT ROUTINE
                                        ;CALL:
                                        ;       MOV     NUM,-(SP)       ;PUT THE NUMBER ON THE STACK
                                        ;       JSR     PC,LINOCT
                                        ;       RETURN
                                        
023202  016646  000002          LINOCT: MOV     2(SP),-(SP)     ;PUT NUMBER IN PROPER LOCATION ON STACK
023206  004737  033260                  JSR     PC,$SB2O        ;CONVERT THE NUMBER TO OCTAL
023212  012637  023226                  MOV     (SP)+,1$        ;GET THE ADDRESS OF THE ASCII STRING
023216  062737  000005  023226          ADD     #5.,1$          ;ADDRESS THE LAST 6 ASCII DIGITS
023224  104414                          DISPLY                  ;TYPE IT
023226  000000                  1$:     .WORD   0               ;ADDRESS
023230  012616                          MOV     (SP)+,(SP)      ;CORRECT THE STACK
023232  000207                          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
                                        
023234  016646  000002          LINDEC: MOV     2(SP),-(SP)     ;SET UP STACK FOR CONVERT
023240  004737  033230                  JSR     PC,$SB2D        ;CONVERT IT TO DECIMAL
023244  004737  032364                  JSR     PC,$SUPRL       ;TYPE IT (WITH LEADING ZEROS SUPRESSED)
023250  012616                          MOV     (SP)+,(SP)      ;RESTORE STACK POINTER
023252  000207                          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
                                ;
023254  004737  027462          READDR: JSR     PC,GETLMT       ;GET ADDRESS LIMITS
023260  162706  000006                  SUB     #6,SP           ;DECREMENT THE STACK POINTER
023264  016616  000006                  MOV     6(SP),(SP)      ;MOVE THE RETURN ADDR DOWN THE STACK
023270  005066  000006                  CLR     6(SP)           ;CLEAR STACK FOR SECTOR
023274  005066  000004                  CLR     4(SP)           ;CLEAR STACK FOR TRACK
023300  116066  002146  000006          MOVB    $RMDA(R0),6(SP) ;SECTOR ON STACK
023306  116066  002147  000004          MOVB    $RMDA+1(R0),4(SP)       ;TRACK ADDRESS
023314  016066  002174  000002          MOV     $RMDC(R0),2(SP) ;CYLINDER ADDRESS
023322  005766  000006          1$:     TST     6(SP)           ;SECTOR 0 ?
023326  001403                          BEQ     2$              ;BRANCH IF SO
023330  105366  000006                  DECB    6(SP)           ;DECREMENT ONE SECTOR
023334  000424                          BR      4$              ;BRANCH TO EXIT
023336  005766  000004          2$:     TST     4(SP)           ;ALSO ON TRACK 0 ?
023342  001406                          BEQ     3$              ;BRANCH IF SO
023344  113766  001424  000006          MOVB    SECLMT,6(SP)    ;LAST SECTOR
023352  105366  000004                  DECB    4(SP)           ;DECREMENT ONE TRACK
023356  000413                          BR      4$              ;EXIT
023360  005766  000002          3$:     TST     2(SP)           ;ALSO ON CYLINDER 0 ?
023364  001410                          BEQ     4$              ;BR IF YES
023366  113766  001424  000006          MOVB    SECLMT,6(SP)    ;LAST SECTOR
023374  113766  001426  000004          MOVB    TRKLMT,4(SP)    ;GET LAST TRACK
023402  005366  000002                  DEC     2(SP)           ;DECREMENT ONE CYLINDER COUNT
023406  005337  001436          4$:     DEC     DEC2            ;DECREMENT TWICE YET ?
023412  002343                          BGE     1$              ;BR IF NO
023414  005037  001436                  CLR     DEC2            ;DECREMENT TRK/SEC ONLY ONCE NEXT TIME
023420  000207                          RTS     PC              ;RETURN

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

023422  012737  177777  001312  CKCLK:  MOV     #-1,CLKFLG      ;CLEAR CLOCK AVAILABILITY FLAG
023430  012737  177777  001310          MOV     #-1,PCLOCK      ;CLEAR KW11-P CLOCK AVAILABILITY FLAG
023436  013746  000004                  MOV     ERRVEC,-(SP)    ;;PUSH ERRVEC ON STACK
023442  012737  023516  000004          MOV     #CKCLK1,ERRVEC  ;SET UP VECTOR FOR CLOCK CHECK
023450  005777  155622                  TST     @$LKCSR         ;CHECK FOR KW11-P
023454  005037  001312                  CLR     CLKFLG          ;SET CLOCK AVAILABILITY FLAG
023460  005037  001310                  CLR     PCLOCK          ;SET KW11-P CLOCK FLAG
023464  013701  001302                  MOV     $LPVEC,R1       ;KW11-P VECTOR ADDRESS
023470  012721  025110                  MOV     #CLOCK,(R1)+    ;SET UP KW11-P VECTOR
023474  012711  000300                  MOV     #300,(R1)       ;PSW - PRI 6
023500  012777  174575  155572          MOV     #-1667.,@$LKCSB ;LOAD COUNTER BUFFER WITH 16.67
023506  012777  000131  155562          MOV     #131,@$LKCSR    ;SET CLOCK - CNT UP, 10US, CONT INT
023514  000441                          BR      CKCLK3

023516  012716  023524          CKCLK1: MOV     #1$,(SP)        ;SETUP RETURN ADDRESS
```

GENERAL SUPPORT SUBROUTINES

```
     58 023522  000002                          RTI
     59 023524  012737  023566  000004  1$:     MOV     #CKCLK2,ERRVEC  ;CHANGE ERROR VECTOR TO CHECK FOR KW11-L
     60 023532  005777  155546                  TST     @$LKS           ;LOOK FOR KW11-L
     61 023536  005037  001312                  CLR     CLKFLG          ;SET CLOCK FLAG
     62 023542  013701  001306                  MOV     $LLVEC,R1       ;KW11-L VECTOR ADDRESS
     63 023546  012721  025110                  MOV     #CLOCK,(R1)+    ;SET UP KW11-L VECTOR
     64 023552  012711  000300                  MOV     #300,(R1)       ;PSW - PRI 6
     65 023556  012777  000100  155520          MOV     #100,@$LKS      ;SET KW11-L INTERRUPT
     66 023564  000415                          BR      CKCLK3
     67
     68 023566  012716  023574          CKCLK2: MOV     #1$,(SP)        ;SETUP RETURN ADDRESS
     69 023572  000002                          RTI
     70 023574  104401  076064          1$:     TYPE    ,NEDCLK         ;'P OR L CLOCK MUST BE ON SYSTEM'
     71 023600  105737  001150                  TSTB    $AUTOB          ;RUNNING IN AUTO MODE ?
     72 023604  001402                          BEQ     2$              ;BR IF NOT
     73 023606  000137  031742                  JMP     $GET42          ;ABORT PROGRAM
     74 023612  000000                  2$:     HALT                    ;HALT
     75 023614  000137  003532                  JMP     START           ;TRY AGAIN
     76 023620  012637  000004          CKCLK3: MOV     (SP)+,ERRVEC    ;RESTORE THE ERROR VECTOR
     77 023624  000207                          RTS     PC
```

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

023626                          STATPR:
023626  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
023630  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
023632  005737  001542                  TST     ASNLST          ;ANY DRIVES ASSIGNED ?
023636  001423                          BEQ     5$              ;BR IF NOT
023640  104401  001203          2$:     TYPE    ,$CRLF          ;CR-LF
023644  004737  023742                  JSR     PC,SHDTYP       ;TYPE THE HEADING
023650  005004                          CLR     R4              ;CLEAR THE DRIVE INDEX
023652  006304                  3$:     ASL     R4              ;CHANGE TO INDEX WORDS
023654  016400  002056                  MOV     BLKADR(R4),R0   ;GET THE DRIVE'S BLOCK ADDRESS
023660  006204                          ASR     R4              ;RESTORE R4
023662  136437  040300  001542          BITB    ATABIT(R4),ASNLST       ;IS THIS DRIVE ASSIGNED ?
023670  001402                          BEQ     4$              ;BR IF NOT
023672  004737  023752                  JSR     PC,SDETAL       ;TYPE THE PERFORMANCE SUMMARY
023676  005204                  4$:     INC     R4              ;INCREMENT THE INDEX
023700  020427  000010                  CMP     R4,#8.          ;FINISHED ?
023704  001362                          BNE     3$              ;BR IF NO
023706                          5$:
023706  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
023710  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
023712  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

023714  010046                  SUMARY: MOV     R0,-(SP)        ;SAVE R0
023716  010446                          MOV     R4,-(SP)        ;SAVE R4
023720  004737  023742                  JSR     PC,SHDTYP       ;TYPE THE HEADING
023724  005004                          CLR     R4              ;CLEAR R4 FOR DRIVE NUMBER
023726  111004                          MOVB    (R0),R4         ;DRIVE NUMBER
023730  004737  023752                  JSR     PC,SDETAL       ;TYPE THE STATISTICS
023734  012604                          MOV     (SP)+,R4        ;RESTORE R4
023736  012600                          MOV     (SP)+,R0        ;RESTORE R0
023740  000207                          RTS     PC              ;RETURN

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

023742  004537  032574          SHDTYP: JSR     R5,TYPRI4       ;TYPE SUMMARY HEADER
023746  075626                          SUMHD                   ;HEADER
023750  000207                          RTS     PC              ;RETURN

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

```
                                        ;       MOV     #DPB,R0         ;DPB ADDRESS
                                        ;       RETURN

                                        SDETAL:
023752                                          TYPE    ,65$            ;;TYPE ASCIZ STRING
023752  104401  023760                        BR      64$             ;;GET OVER THE ASCIZ
023756  000404                          ;;65$:  .ASCIZ  <CRLF>/TIME /
023770                                  64$:
023770  004737  024764                        JSR     PC,$TIME        ;TYPE ELAPSED TIME

                                        ;TYPE LINE 2 OF SUMMARY
023774  104401  001203                        TYPE    ,$CRLF          ;CR-LF
024000  104401  075304                        TYPE    ,UNTMSG         ;TYPE 'DRIVE'
024004  010446                              MOV     R4,-(SP)        ;;SAVE R4 FOR TYPEOUT
024006  104403                              TYPOS                   ;;GO TYPE--OCTAL ASCII
024010     002                              .BYTE   2               ;;TYPE 2 DIGIT(S)
024011     000                              .BYTE   0               ;;SUPPRESS LEADING ZEROS
024012  104401  075234                        TYPE    ,BLNKS1         ;TYPE 1 BLANK
024016  104401  075300                        TYPE    ,DASH           ;TYPE '-'
024022  104401  075234                        TYPE    ,BLNKS1         ;TYPE 1 BLANK
024026  104401  075520                        TYPE    ,$RM80          ;TYPE DRIVE TYPE
024032  104401  075276                        TYPE    ,COMMA          ;TYPE ','
024036  104401  024044                        TYPE    ,67$            ;;TYPE ASCIZ STRING
024042  000404                              BR      66$             ;;GET OVER THE ASCIZ
                                        ;;67$:  .ASCIZ  / PASS /
024054                                  66$:
024054  016046  000104                        MOV     $PASSC(R0),-(SP)        ;PUT THE PASS COUNT ON THE STACK
024060  004737  033230                        JSR     PC,$SB2D        ;CONVERT IT
024064  004537  032470                        JSR     R5,REPLZ        ;TYPE IT
024070  000003                              .WORD   3               ;TYPE 3 DIGITS
024072  104401  076141                        TYPE    ,PERIOD         ;TYPE '.'
024076  104401  075234                        TYPE    ,BLNKS1         ;TYPE 1 BLANK
024102  004737  032622                        JSR     PC,TYDRV        ;TYPE DRV SERIAL NUMBER
024106  104401  075234                        TYPE    ,BLNKS1         ;TYPE 1 BLANK
024112  004737  032652                        JSR     PC,TYHDA        ;TYPE HDA SERIAL NUMBER

                                        ;TYPE LINE 3 OF SUMMARY
024116  104401  024124                        TYPE    ,69$            ;;TYPE ASCIZ STRING
024122  000407                              BR      68$             ;;GET OVER THE ASCIZ
                                        ;;69$:  .ASCIZ  <CRLF><LF>/WT OFLOW  /
024142                                  68$:
024142  016046  000056                        MOV     $WTOFL(R0),-(SP)        ;;SAVE $WTOFL(R0) FOR TYPEOUT
024146  104405                              TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
024150  104401  076141                        TYPE    ,PERIOD         ;TYPE '.'
024154  104401  024162                        TYPE    ,71$            ;;TYPE ASCIZ STRING
024160  000407                              BR      70$             ;;GET OVER THE ASCIZ
                                        ;;71$:  .ASCIZ  / WRDS WRITN /
024200                                  70$:
024200  010046                              MOV     R0,-(SP)        ;GET ADDRESS OF DPB
024202  062716  000060                        ADD     #$WRITN,(SP)    ;POINT TO LOW NUMBER OF WRDS WRITTEN
024206  004737  037222                        JSR     PC,$DB2D        ;CONVERT DECIMAL NUMBER
024212  004737  032300                        JSR     PC,SUPRS        ;SUPPRESS LEADING ZEROS AND TYPE
024216  104401  076141                        TYPE    ,PERIOD         ;TYPE '.'

                                        ;TYPE LINE 4 OF SUMMARY
024222  104401  024230                        TYPE    ,73$            ;;TYPE ASCIZ STRING
024226  000406                              BR      72$             ;;GET OVER THE ASCIZ
```

```
                                        ;;73$:  .ASCIZ  <CRLF>/RD OFLOW  /
                                        ;;72$:
     024244
104  024244  016046  000064                   MOV     $RDOFL(R0),-(SP)        ;;SAVE $RDOFL(R0) FOR TYPEOUT
     024250  104405                             TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
105  024252  104401  076141                     TYPE    ,PERIOD         ;TYPE '.'
106  024256  104401  024264                     TYPE    ,75$            ;;TYPE ASCIZ STRING
     024262  000407                             BR      ;74$            ;;GET OVER THE ASCIZ
                                        ;;75$:  .ASCIZ  / WRDS READ  /
     024302                             ;;74$:
107  024302  010046                             MOV     R0,-(SP)        ;GET ADDRESS OF DPB
108  024304  062716  000066                     ADD     #$READ,(SP)     ;POINT TO LOW NUMBER OF WRDS READ
109  024310  004737  037222                     JSR     PC,$DB2D        ;CONVERT DECIMAL NUMBER
110  024314  004737  032300                     JSR     PC,SUPRS        ;SUPPRESS LEADING ZEROS AND TYPE
111  024320  104401  076141                     TYPE    ,PERIOD         ;TYPE '.'
112
113                                     ;TYPE LINE 5 OF SUMMARY
114  024324  104401  024332                     TYPE    ,77$            ;;TYPE ASCIZ STRING
     024330  000404                             BR      ;76$            ;;GET OVER THE ASCIZ
                                        ;;77$:  .ASCIZ  <CRLF>/SEEKS /
     024342                             ;;76$:
115  024342  010046                             MOV     R0,-(SP)        ;PUT $POSIT ON THE STACK
116  024344  062716  000052                     ADD     #$POSIT,(SP)    ;POINT TO LOW NUMBER OF SEEK COUNT
117  024350  004737  037222                     JSR     PC,$DB2D        ;CONVERT DECIMAL NUMBER
118  024354  004737  032300                     JSR     PC,SUPRS        ;SUPPRESS LEADING ZEROS AND TYPE
119  024360  104401  076141                     TYPE    ,PERIOD         ;TYPE '.'
128
129                                     ;TYPE LINES 6 AND 7 OF SUMMARY
130  024364  104401  024372                     TYPE    ,79$            ;;TYPE ASCIZ STRING
     024370  000405                             BR      ;78$            ;;GET OVER THE ASCIZ
                                        ;;79$:  .ASCIZ  <CRLF>/ERRORS:/
     024404                             ;;78$:
131  024404  104401  024412                     TYPE    ,81$            ;;TYPE ASCIZ STRING
     024410  000404                             BR      ;80$            ;;GET OVER THE ASCIZ
                                        ;;81$:  .ASCIZ  <CRLF>/SOFT /
     024422                             ;;80$:
132  024422  016046  000074                     MOV     $SOFT(R0),-(SP) ;;SAVE $SOFT(R0) FOR TYPEOUT
     024426  104405                             TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
133  024430  104401  076141                     TYPE    ,PERIOD         ;TYPE '.'
134  024434  104401  024442                     TYPE    ,83$            ;;TYPE ASCIZ STRING
     024440  000404                             BR      ;82$            ;;GET OVER THE ASCIZ
                                        ;;83$:  .ASCIZ  / HARD /
     024452                             ;;82$:
135  024452  016046  000076                     MOV     $HARD(R0),-(SP) ;;SAVE $HARD(R0) FOR TYPEOUT
     024456  104405                             TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
136  024460  104401  076141                     TYPE    ,PERIOD         ;TYPE '.'
137  024464  104401  024472                     TYPE    ,85$            ;;TYPE ASCIZ STRING
     024470  000403                             BR      ;84$            ;;GET OVER THE ASCIZ
                                        ;;85$:  .ASCIZ  / SKI /
     024500                             ;;84$:
138  024500  016046  000100                     MOV     $SKI(R0),-(SP)  ;;SAVE $SKI(R0) FOR TYPEOUT
     024504  104405                             TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
139  024506  104401  076141                     TYPE    ,PERIOD         ;TYPE '.'
140  024512  104401  024520                     TYPE    ,87$            ;;TYPE ASCIZ STRING
     024516  000404                             BR      ;86$            ;;GET OVER THE ASCIZ
                                        ;;87$:  .ASCIZ  / MISP /
     024530                             ;;86$:
141  024530  016046  000102                     MOV     $MISPO(R0),-(SP)        ;;SAVE $MISPO(R0) FOR TYPEOUT
```

```
                024534  104405                          TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
    142         024536  104401  076141                  TYPE    ,PERIOD         ;TYPE '.'
    143         024542  104401  024550                  TYPE    ,89$            ;;TYPE ASCIZ STRING
                024546  000404                          BR      88$             ;;GET OVER THE ASCIZ
                                                ;;89$:  .ASCIZ  / OTHER /
                024560                          88$:
    144         024560  016046  000072                  MOV     $TOTAL(R0),-(SP)        ;CALCULATE NUMBER OF OTHER ERRORS
    147         024564  166016  000074                  SUB     $SOFT(R0),(SP)  ;SUBTRACT $SOFT FROM $TOTAL
                024570  166016  000076                  SUB     $HARD(R0),(SP)  ;SUBTRACT $HARD FROM $TOTAL
                024574  166016  000100                  SUB     $SKI(R0),(SP)   ;SUBTRACT $SKI FROM $TOTAL
                024600  166016  000102                  SUB     $MISPO(R0),(SP) ;SUBTRACT $MISPO FROM $TOTAL
    148         024604  104405                          TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
    149         024606  104401  076141                  TYPE    ,PERIOD         ;TYPE '.'
    150         024612  104401  001203                  TYPE    ,$CRLF          ;CR-LF
    151         024616  000207                          RTS     PC
```

```
     1
    15
    16
                                        ;ROUTINE TO INCREMENT $SOFT
                                        ;
                                        ;NOTE:  $SOFT WILL NOT BE INCREMENTED BEYOND 77777 (32767.)

    024620  005737  001342      INCSOF: TST     BADSEC          ;SEE IF BAD TRK/SEC INDICATOR SET
    024624  001006                      BNE     1$              ;BR IF IT'S SET, DON'T INCREMENT COUNT
    024626  026027  000074  077777      CMP     $SOFT(R0),#77777        ;IS $SOFT ALREADY AT MAXIMUM ?
    024634  103002                      BHIS    1$              ;BR IF IT IS
    024636  005260  000074              INC     $SOFT(R0)               ;INCREMENT $SOFT
    024642  000207              1$:     RTS     PC              ;RETURN
    17
                                        ;ROUTINE TO INCREMENT $HARD
                                        ;
                                        ;NOTE:  $HARD WILL NOT BE INCREMENTED BEYOND 77777 (32767.)

    024644  005737  001342      INCHRD: TST     BADSEC          ;SEE IF BAD TRK/SEC INDICATOR SET
    024650  001006                      BNE     1$              ;BR IF IT'S SET, DON'T INCREMENT COUNT
    024652  026027  000076  077777      CMP     $HARD(R0),#77777        ;IS $HARD ALREADY AT MAXIMUM ?
    024660  103002                      BHIS    1$              ;BR IF IT IS
    024662  005260  000076              INC     $HARD(R0)               ;INCREMENT $HARD
    024666  000207              1$:     RTS     PC              ;RETURN
    18
                                        ;ROUTINE TO INCREMENT $SKI
                                        ;
                                        ;NOTE:  $SKI WILL NOT BE INCREMENTED BEYOND 77777 (32767.)

    024670  005737  001342      INCSKI: TST     BADSEC          ;SEE IF BAD TRK/SEC INDICATOR SET
    024674  001006                      BNE     1$              ;BR IF IT'S SET, DON'T INCREMENT COUNT
    024676  026027  000100  077777      CMP     $SKI(R0),#77777 ;IS $SKI ALREADY AT MAXIMUM ?
    024704  103002                      BHIS    1$              ;BR IF IT IS
    024706  005260  000100              INC     $SKI(R0)                ;INCREMENT $SKI
    024712  000207              1$:     RTS     PC              ;RETURN
19
                                        ;ROUTINE TO INCREMENT $MISPO
                                        ;
                                        ;NOTE:  $MISPO WILL NOT BE INCREMENTED BEYOND 77777 (32767.)

    024714  005737  001342      INCMIS: TST     BADSEC          ;SEE IF BAD TRK/SEC INDICATOR SET
    024720  001006                      BNE     1$              ;BR IF IT'S SET, DON'T INCREMENT COUNT
    024722  026027  000102  077777      CMP     $MISPO(R0),#77777       ;IS $MISPO ALREADY AT MAXIMUM ?
    024730  103002                      BHIS    1$              ;BR IF IT IS
    024732  005260  000102              INC     $MISPO(R0)              ;INCREMENT $MISPO
    024736  000207              1$:     RTS     PC              ;RETURN
20
                                        ;ROUTINE TO INCREMENT $TOTAL
                                        ;
                                        ;NOTE:  $TOTAL WILL NOT BE INCREMENTED BEYOND 77777 (32767.)

    024740  005737  001342      INCTOT: TST     BADSEC          ;SEE IF BAD TRK/SEC INDICATOR SET
    024744  001006                      BNE     1$              ;BR IF IT'S SET, DON'T INCREMENT COUNT
    024746  026027  000072  077777      CMP     $TOTAL(R0),#77777       ;IS $TOTAL ALREADY AT MAXIMUM ?
    024754  103002                      BHIS    1$              ;BR IF IT IS
    024756  005260  000072              INC     $TOTAL(R0)              ;INCREMENT $TOTAL
    024762  000207              1$:     RTS     PC              ;RETURN
```

```
                                        ;ROUTINE TO TYPE THE TIME
                                        
024764  005737  001312          $TIME:  TST     CLKFLG          ;CLOCK ON THE SYSTEM ?
024770  001046                          BNE     3$              ;BR IF NOT
024772  012737  000002  025040          MOV     #2,2$           ;ASSUME 2 DIGITS TO TYPE
025000  013746  001344                  MOV     HOUR,-(SP)      ;PUT 'HOURS' ON THE STACK
025004  021627  000144                  CMP     (SP),#100.      ;100. HOURS OR MORE ?
025010  002407                          BLT     1$              ;BR IF NO
025012  005237  025040                  INC     2$              ;TYPE 3 DIGITS
025016  021627  001750                  CMP     (SP),#1000.     ;1000. HOURS OR MORE ?
025022  002402                          BLT     1$              ;BR IF NO
025024  005237  025040                  INC     2$              ;TYPE 4 DIGITS
025030  004737  033230          1$:     JSR     PC,$SB2D        ;CONVERT TO DECIMAL
025034  004537  032464                  JSR     R5,FILLZ        ;TYPE IT
025040  000000          2$:     .WORD   0                       ;NUMBER OF HOUR DIGITS TO TYPE
025042  104401  076332                  TYPE    ,COLON          ;':'
025046  013746  001346                  MOV     MINUTE,-(SP)    ;PUT 'MINUTES' ON THE STACK
025052  004737  033230                  JSR     PC,$SB2D        ;CONVERT TO DECIMAL
025056  004537  032464                  JSR     R5,FILLZ        ;TYPE IT
025062  000002                          .WORD   2               ;TYPE 2 DIGITS
025064  104401  076332                  TYPE    ,COLON          ;':'
025070  013746  001350                  MOV     SECOND,-(SP)    ;PUT SECONDS ON THE STACK
025074  004737  033230                  JSR     PC,$SB2D        ;CONVERT TO DECIMAL
025100  004537  032464                  JSR     R5,FILLZ        ;TYPE IT
025104  000002                          .WORD   2               ;TYPE 2 DIGITS
025106  000207          3$:     RTS     PC
                                        
                                        ;CLOCK HANDLER ROUTINE
                                        
025110  005337  001352          CLOCK:  DEC     ONESEC          ;INCREMENT THE ONE SECOND COUNTER
025114  001027                          BNE     1$              ;BR IF A SECOND NOT COUNTED
025116  013737  001314  001352          MOV     HZ,ONESEC       ;RESTORE THE VALUE
025124  005237  001350                  INC     SECOND          ;COUNT THE SECOND
025130  022737  000074  001350          CMP     #60.,SECOND     ;AT MAXIMUM ?
025136  001016                          BNE     1$              ;BR IF NOT
025140  005037  001350                  CLR     SECOND          ;CLEAR THE SECOND'S COUNTER
025144  005237  001466                  INC     INTRVL+2        ;COUNT THE PERFORMANCE SUMMARY INTERVAL
025150  005237  001346                  INC     MINUTE          ;COUNT THE MINUTE
025154  022737  000074  001346          CMP     #60.,MINUTE     ;AT MAXIMUM ?
025162  001004                          BNE     1$              ;BR IF NOT
025164  005037  001346                  CLR     MINUTE          ;CLEAR THE MINUTE'S COUNTER
025170  005237  001344                  INC     HOUR            ;COUNT THE HOURS
025174  022737  000062  001314  1$:     CMP     #50.,HZ         ;CPU RUNNING @ 50HZ ?
025202  001403                          BEQ     2$              ;BR IF YES
025204  012746  000020                  MOV     #16.,-(SP)      ;16MS ON THE STACK @ 60HZ
025210  000402                          BR      3$
025212  012746  000024          2$:     MOV     #20.,-(SP)      ;20MS ON THE STACK @ 50HZ
025216  004737  044152          3$:     JSR     PC,RMTMR        ;DRIVER TIMER ROUTINE
025222  005737  001464                  TST     INTRVL          ;DISPLAY THE PERFORMANCE SUMMARY ?
025226  001411                          BEQ     4$              ;BR IF NOT
025230  023737  001464  001466          CMP     INTRVL,INTRVL+2 ;DISPLAY INTERVAL FINISHED ?
025236  001005                          BNE     4$              ;BR IF NOT
025240  012737  177777  001316          MOV     #-1,STATIN      ;SET PERFORMANCE SUMMARY DISPLAY FLAG
025246  005037  001466                  CLR     INTRVL+2        ;CLEAR THE PERFORMANCE INTERVAL COUNTER
025252  000002          4$:     RTI
```

GENERAL SUPPORT SUBROUTINES

```
                                        ;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

025254  005737  040276          KSR:  TST     DTUW            ;ANY DATA TRANSFERS UNDER WAY ?
025260  100375                  KSR1: BPL     KSR             ;BR IF YES
025262  104412                        SAVREG                  ;SAVE THE REGISTERS
025264  012737  000200  177776        MOV     #PR4,PS         ;SET PRIORITY TO 4
025272  013704  040310          1$:   MOV     RMADR,R4        ;GET RM/RH BASE ADDRESS
025276  012764  000040  000010        MOV     #CLR,RMCS2(R4)  ;CLEAR MASSBUS CONTROLLER
025304  005037  001340                CLR     CFLAG           ;CLEAR THE 'CONTROL C' FLAG
025310  104401  001203                TYPE    ,$CRLF          ;CR-LF
025314  004737  024764                JSR     PC,$TIME        ;TYPE THE TIME
025320  004737  033326                JSR     PC,$TKINT       ;INITIALIZE TTY KEYBOARD
025324  104401  076231                TYPE    ,ENTCOM         ;'ENTER COMMAND'

025330  104411                        RDLIN                   ;READ THE KEYBOARD
025332  012605                        MOV     (SP)+,R5        ;GET ADDRESS OF INPUT STRING
025334  005737  001340                TST     CFLAG           ;CHECK THE CONTROL C FLAG
025340  001405                        BEQ     2$              ;BR IF NO 'CONTROL C' ENTERED
025342  005737  001542                TST     ASNLST          ;ANY DRIVES ASSIGNED ?
025346  001136                        BNE     13$             ;BR IF YES
025350  000137  003532                JMP     START           ;JUMP TO START

025354  005205                2$:   INC     R5              ;POINT TO SECOND CHARACTER
025356  122715  000124              CMPB    #'T,(R5)        ;EQ TO A 'T' ?
025362  001465                      BEQ     9$              ;YES
025364  122715  000101              CMPB    #'A,(R5)        ;EQ TO AN 'A'
025370  001410                      BEQ     3$              ;BR IF IT IS
025372  121527  000067              CMPB    (R5),#'7        ;DRIVE NUMBER GREATER THAN AN ASCII 7 ?
025376  101117                      BHI     12$             ;BR IF IT IS
025400  121527  000060              CMPB    (R5),#'0        ;DRIVE NUMBER LESS THAN AN ASCII 0 ?
025404  103514                      BLO     12$             ;BR IF IT IS
025406  142715  177770              BICB    #^C7,(R5)       ;LEAVE ONLY LOWER 3 BITS IF CHAR NOT 'A'
025412  122765  000124  177777  3$: CMPB    #'T,-1(R5)      ;EQ TO 'T'
025420  001003                      BNE     4$              ;BR IF NOT EQ
025422  004737  026556              JSR     PC,NEWASN       ;ASSIGN DRIVE FOR TEST
025426  000506                      BR      13$             ;EXIT
025430  122765  000104  177777  4$: CMPB    #'D,-1(R5)      ;EQ TO 'D' ?
025436  001003                      BNE     5$              ;BR IF NOT EQ
025440  004737  026356              JSR     PC,DEASGN       ;DEASSIGN DRIVE
025444  000477                      BR      13$             ;EXIT
025446  122765  000123  177777  5$: CMPB    #'S,-1(R5)      ;EQ TO 'S'
025454  001003                      BNE     6$              ;BR IF NOT EQ
025456  004737  026464              JSR     PC,SCMND        ;TYPE STATISTICS
025462  000470                      BR      13$             ;EXIT
025464  122765  000127  177777  6$: CMPB    #'W,-1(R5)      ;EQ TO 'W'
025472  001012                      BNE     8$              ;BR IF NOT EQ
025474  005737  001440              TST     RDONLY          ;LOCKED IN "READ ONLY" MODE ?
025500  001053                      BNE     11$             ;BR IF YES
025502  032777  000001  153444      BIT     #SW0,@SWR       ;IS SWITCH 0 SET ?
025510  001047                      BNE     11$             ;BR IF SET, CAN'T DO 'W' COMMAND
```

```
                                7$:     JSR     PC,DATAPK       ;WRITE A DATA PACK
                                        BR      13$             ;EXIT
                                8$:     CMPB    #'R,-1(R5)      ;EQ TO 'R' ?
                                        BNE     12$             ;BR IF NOT EQ
                                        JSR     PC,REDAPK       ;READ A DATA PACK
                                        BR      13$             ;EXIT
                                9$:     CMPB    #'W,-1(R5)      ;WT COMMAND ?
                                        BNE     12$             ;NO
                                        TST     RDONLY          ;LOCKED IN "READ ONLY" MODE ?
                                        BNE     11$             ;BR IF YES
                                        BIT     #SW0,@SWR       ;IS SWITCH 0 SET ?
                                        BNE     11$             ;BR IF SET, CAN'T DO 'W' COMMAND
                                        CMPB    #'A,1(R5)       ;ALL DRIVES ?
                                        BEQ     10$             ;YES
                                        CMPB    1(R5),#'7       ;GREAT THAN 7
                                        BHI     12$             ;YES
                                        CMPB    1(R5),#'0       ;LESS THAN 0
                                        BLO     12$             ;YES
                                        BICB    #^C7,1(R5)      ;CHOP OFF THE HIGHER BITS
                                10$:    JSR     PC,WATPAK       ;ASSIGN DRIVES WITH WT COMMAND
                                        BR      13$
                                11$:    TYPE    ,MSWRO          ;TYPE 'CAN'T WRITE IN READ ONLY MODE'
                                        BR      1$              ;TRY AGAIN
                                12$:    TYPE    ,INVLD          ;TYPE 'INVALID COMMAND' MESSAGE
                                        BR      1$              ;TRY AGAIN
                                13$:    RESREG                  ;RESTORE R0 - R5
                                        TST     @#TKB           ;CLEAR THE TTY BUFFER
                                        BIS     #BIT06,@#TKS    ;SET TTY INTERRUPT ENABLE
                                        CLR     PS              ;SET PRIORITY BACK TO ZERO
                                        RTS     PC              ;RETURN

                        ;ROUTINE TO PROCESS THE ASSIGN REQUEST ('T', 'R', OR 'W' COMMANDS)

                        ASSIGN: MOVB    (R5),R4         ;PUT DRIVE # IN R4
                                1$:     CLR     CFLAG           ;CLEAR CONTROL C FLAG
                                        CLR     DRVPAR          ;ASSUME CHANGING DRIVE PARAMETERS
                                        TYPE    ,MSPRM          ;TYPE 'CHANGE DRIVE PARAMETERS ?'
                                        RDLIN                   ;READ THE ENTRY
                                        MOV     (SP)+,R0        ;SAVE ADDRESS OF RESPONSE
                                        TST     CFLAG           ;WAS IT CONTROL C ?
                                        BNE     1$              ;BR IF YES
                                        TSTB    (R0)            ;WAS RESPONSE A CARRIAGE RETURN (DEFAULT 'N')?
                                        BEQ     3$              ;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     4$              ;BR IF YES
                                        CMPB    #'N,(R0)        ;WAS IT A 'N' RESPONSE ?
                                        BEQ     3$              ;BR IF YES
                                2$:     TYPE    ,BADENT         ;TYPE BAD ENTRY MESSAGE
                                        BR      1$              ;TRY AGAIN
                                3$:     INC     DRVPAR          ;DO NOT CHANGE DRIVE PARAMETERS
                                4$:     CMPB    #'A,R4          ;ASSIGN ALL DRIVES ?
                                        BEQ     ASGN2           ;BR IF YES

                        ASGN1:  MOV     #UNTASN,ASNMSG  ;'DRIVE ASSIGNED' MESSAGE ADDRESS
                                TST     XXDP            ;LOADED FROM THIS DEVICE ?
```

```
 115 025776  001407                        BEQ     1$              ;BR IF NO
 116 026000  123704  001444                CMPB    XXDP,R4         ;LOADED FROM THIS DRIVE ?
 117 026004  001004                        BNE     1$              ;BR IF NO
 118 026006  012737  075464  031200        MOV     #LODEV,ASNMSG   ;'LOAD DEVICE' MESSAGE ADDRESS
 119 026014  000407                        BR      2$
 120 026016  136437  040300  001542  1$:   BITB    ATABIT(R4),ASNLST       ;DRIVE ALREADY ASSIGNED ?
 121 026024  001003                        BNE     2$              ;BR IF IT IS
 122 026026  004737  026130                JSR     PC,ASGN3        ;SEE IF DRIVE ON THE SYSTEM
 123 026032  000207                        RTS     PC              ;RETURN
 124 026034  000137  031154        2$:     JMP     ASNERR          ;EXIT ERROR
 125
 126 026040  005004                ASGN2:  CLR     R4              ;START WITH DRIVE 0
 127 026042  012737  075355  031200  1$:   MOV     #UNTASN,ASNMSG  ;ERROR MESSAGE
 128 026050  005737  001444                TST     XXDP            ;LOADED FROM THIS DEVICE ?
 129 026054  001407                        BEQ     2$              ;BR IF NO
 130 026056  123704  001444                CMPB    XXDP,R4         ;LOADED FROM THIS DRIVE ?
 131 026062  001004                        BNE     2$              ;BR IF NO
 132 026064  012737  075464  031200        MOV     #LODEV,ASNMSG   ;'LOAD DEVICE' MESSAGE ADDRESS
 133 026072  000413                        BR      4$
 134 026074  136437  040300  001542  2$:   BITB    ATABIT(R4),ASNLST       ;ALREADY ASSIGNED ?
 135 026102  001007                        BNE     4$              ;YES
 136 026104  004737  026130                JSR     PC,ASGN3        ;ASSIGN THE DRIVE
 137 026110  005204                3$:     INC     R4              ;INCREMENT DRIVE #
 138 026112  020427  000007                CMP     R4,#7           ;ALL DRIVE CHECKED ?
 139 026116  003751                        BLE     1$              ;NO
 140 026120  000207                        RTS     PC              ;YES
 141 026122  004737  031154        4$:     JSR     PC,ASNERR       ;ERROR MESSAGE
 142 026126  000770                        BR      3$              ;TO LOOP
 143
 144 026130  136437  040300  001542  ASGN3:  BITB    ATABIT(R4),ASNLST       ;DRIVE ALREADY ASSIGNED ?
 145 026136  001060                        BNE     ASGN4           ;BR IF IT IS
 146 026140  110437  067556                MOVB    R4,GENDPB       ;GET DRIVE NUMBER
 147 026144  006304                        ASL     R4              ;MAKE R4 WORD INDEX
 148 026146  016400  002056                MOV     BLKADR(R4),R0   ;PUT BLOCK'S ADDR INTO R0
 149 026152  004737  015624                JSR     PC,RECAL0       ;RECALIBRATE DRIVE
 150 026156  006204                        ASR     R4              ;MAKE R4 BYTE INDEX
 151 026160  105764  040164                TSTB    DRVSTA(R4)      ;DRIVE AVAILABLE?
 152 026164  001453                        BEQ     ASGN7           ;BR IF DRIVE OFFLINE OR NONEXISTENT
 153 026166  100445                        BMI     ASGN6           ;BR IF DRIVE UNSAFE
 154 026170  004737  026630                JSR     PC,CLRDPB       ;CLEAR BLOCK FOR DRIVE JUST ASSIGNED
 155 026174  004737  027550                JSR     PC,GETID        ;GET DRIVE (MBA) SERIAL NUMBER
 156 026200  004537  027650                JSR     R5,GETADR       ;RETRIEVE BAD SECTOR FILE
 157 026204  005737  001442                TST     DRVPAR          ;CHANGE DRIVE PARAMETERS ?
 158 026210  001017                        BNE     1$              ;BR IF NO
 159 026212  104401  076034                TYPE    ,DRNUM          ;TYPE DRIVE MESSAGE
 160 026216  010446                        MOV     R4,-(SP)        ;;SAVE R4 FOR TYPEOUT
     026220  104403                        TYPOS                   ;;GO TYPE--OCTAL ASCII
     026222     002                        .BYTE   2               ;;TYPE 2 DIGIT(S)
     026223     000                        .BYTE   0               ;;SUPPRESS LEADING ZEROS
 161 026224  104401  001203                TYPE    ,$CRLF          ;CR-LF
 162 026230  004737  032622                JSR     PC,TYDRV        ;TYPE DRV SERIAL NUMBER
 163 026234  104401  075302                TYPE    ,TAB            ;TYPE TAB CONTROL
 164 026240  004737  032652                JSR     PC,TYHDA        ;TYPE HDA SERIAL NUMBER
 165 026244  104401  001203                TYPE    ,$CRLF          ;CR-LF
 166 026250  006304                1$:     ASL     R4              ;MAKE R4 WORD INDEX
 167 026252  004737  027054                JSR     PC,DRVPRM       ;GET THE DRIVE'S ADDRESS LIMITS
 168 026256  004737  030254                JSR     PC,MANTER       ;MANUALLY ENTER BAD SECTOR INFORMATION
```

```
    169 026262  016464  002056  001566          MOV     BLKADR(R4),NEWUNT(R4)   ;DPB ADDRESS
    170 026270  113760  001320  000026          MOVB    PACK,$PACK(R0)  ;SET READ/WRITE DATA PACK INDICATOR
    171 026276  006204                          ASR     R4              ;MAKE R4 BYTE INDEX
    172 026300  000207                  ASGN4:  RTS     PC              ;RETURN
    173
    174 026302  012737  075454  031200  ASGN6:  MOV     #NOTSAF,ASNMSG  ;'UNSAFE' MESSAGE ADDRESS
    175 026310  000137  031154                  JMP     ASNERR          ;TO ERROR ROUTINE
    176
    177 026314  105764  040174          ASGN7:  TSTB    DRVTYP(R4)      ;DRIVE PRESENT?
    178 026320  001405                          BEQ     1$              ;BR IF NOT
    179 026322  100010                          BPL     2$              ;BR IF DRIVE OFFLINE
    180 026324  012737  075403  031200          MOV     #NOTRM,ASNMSG   ;ADDRESS OF 'NOT RM80' MSG
    181 026332  000407                          BR      3$              ;EXIT
    182 026334  012737  075420  031200  1$:     MOV     #NOTPRS,ASNMSG  ;ADDRESS OF 'NOT PRESENT' MSG
    183 026342  000403                          BR      3$              ;EXIT
    184 026344  012737  075312  031200  2$:     MOV     #UNTOFF,ASNMSG  ;ADDRESS OF 'DRIVE OFFLINE' MESSAGE
    185 026352  000137  031154          3$:     JMP     ASNERR          ;TO ERROR ROUTINE
```

```
                                        ;'D' COMMAND (ROUTINE TO DEASSIGN A DRIVE)

026356  005004                  DEASGN: CLR     R4              ;START WITH DRIVE 0
026360  012703  000010                  MOV     #8.,R3          ;COUNTER
026364  122715  000101                  CMPB    #'A,(R5)        ;DEASSIGN ALL DRIVES ?
026370  001403                          BEQ     1$              ;BR IF YES
026372  111504                          MOVB    (R5),R4         ;GET DRIVE NUMBER
026374  012703  000001                  MOV     #1,R3           ;SET R3 FOR ONE DRIVE
026400  136437  040300  001542  1$:     BITB    ATABIT(R4),ASNLST       ;DRIVE ASSIGNED ?
026406  001417                          BEQ     3$              ;BR IF NOT
026410  146437  040300  001542          BICB    ATABIT(R4),ASNLST       ;DELETE THE DRIVE FROM THE ASSIGNED LIST
026416  146437  040300  032012          BICB    ATABIT(R4),AUTLST       ;DELETE DRIVE FROM AUTO ASSIGN LIST
026424  006304                          ASL     R4              ;MAKE ADDR INTO A WORD INDEX
026426  016464  002056  001544          MOV     BLKADR(R4),DDRVS(R4)    ;PUT ADDRESS IN DEASSIGN LIST
026434  006204                          ASR     R4
026436  005303                  2$:     DEC     R3              ;ANY MORE DRIVES ?
026440  001410                          BEQ     4$              ;BR IF NOT
026442  005204                          INC     R4
026444  000755                          BR      1$
026446  012737  075333  031200  3$:     MOV     #UNTNOT,ASNMSG  ;ADDR OF 'NOT ASSIGNED' MESSAGE
026454  004737  031154                  JSR     PC,ASNERR       ;REPORT IT
026460  000766                          BR      2$
026462  000207                  4$:     RTS     PC

                                        ;'S' COMMAND (ROUTINE TO TYPE DRIVE PERFORMANCE SUMMARY)

026464                          SCMND:
026464  013746  001542                  MOV     ASNLST,-(SP)    ;;PUSH ASNLST ON STACK
026470  122715  000101                  CMPB    #'A,(R5)        ;ALL STATISTICS ?
026474  001416                          BEQ     2$              ;BR IF YES
026476  111504                          MOVB    (R5),R4         ;GET DRIVE NUMBER
026500  136416  040300                  BITB    ATABIT(R4),(SP) ;IS THIS DRIVE ASSIGNED ?
026504  001404                          BEQ     1$              ;BR IF NO
026506  116437  040300  001542          MOVB    ATABIT(R4),ASNLST       ;GET DRIVE ASSIGN BIT
026514  000411                          BR      3$

026516  012737  075333  031200  1$:     MOV     #UNTNOT,ASNMSG  ;ADDR OF 'NOT ASSIGNED' MSG
026524  004737  031154                  JSR     PC,ASNERR       ;TYPE ERROR MESSAGE
026530  000407                          BR      4$              ;EXIT
026532  105737  001542          2$:     TSTB    ASNLST          ;ANY DRIVE ASSIGNED ?
026536  001404                          BEQ     4$              ;BR IF NO
026540  004737  023626          3$:     JSR     PC,STATPR       ;TYPE ALL STATISTICS
026544  104401  075617                  TYPE    ,STAR5          ;TYPE '****...ETC'
026550                          4$:
026550  012637  001542                  MOV     (SP)+,ASNLST    ;;POP STACK INTO ASNLST
026554  000207                          RTS     PC

                                        ;'T' COMMAND (ROUTINE TO TEST A DRIVE)

026556  005037  001320          NEWASN: CLR     PACK            ;SET 'T' COMMAND INDICATOR
026562  000137  025666                  JMP     ASSIGN          ;GO TO THE ASSIGN ROUTINE

                                        ;'R' COMMAND (ROUTINE TO READ A DATA PACK)

026566  012737  000001  001320  REDAPK: MOV     #1,PACK         ;SET THE 'READ' INDICATOR
026574  000137  025666                  JMP     ASSIGN          ;ASSIGN THE REQUESTED DRIVE
```

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

026600  012737  177777  001320  DATAPK: MOV     #-1,PACK        ;SET THE 'W' COMMAND INDICATOR
026606  000137  025666                  JMP     ASSIGN          ;ASSIGN REQUESTED DRIVE

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

026612  116515  000001          WATPAK: MOVB    1(R5),(R5)      ;ADJUST DRIVE NUMBER ADDRESS
026616  012737  177776  001320          MOV     #-2,PACK        ;PACK WRITE COMMAND
026624  000137  025666                  JMP     ASSIGN          ;JUMP TO ASSIGN ROUTINE
```

GENERAL SUPPORT SUBROUTINES

```
                                        ;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

026630                                  CLRDPB:
026630  010146                                 MOV     R1,-(SP)        ;;PUSH R1 ON STACK
026632  010346                                 MOV     R3,-(SP)        ;;PUSH R3 ON STACK
026634  010446                                 MOV     R4,-(SP)        ;;PUSH R4 ON STACK
026636  010546                                 MOV     R5,-(SP)        ;;PUSH R5 ON STACK
026640  005737  037716                        TST     PWRFLG          ;RETURNING FROM POWER FAIL ?
026644  001076                                 BNE     4$              ;BRANCH IF YES
026646  010004                                 MOV     R0,R4           ;GET THE DPB ADDRESS
026650  062704  000002                        ADD     #2,R4           ;ADDRESS OF FIRST LOCN TO BE CLEARED
026654  012703  000012                        MOV     #<$CYL-$COMND>+2,R3     ;NUMBER OF LOCNS TO BE CLEARED
026660  005024                          1$:   CLR     (R4)+           ;CLEAR THE LOCATION
026662  162703  000002                        SUB     #2,R3           ;DONE CLEARING YET ?
026666  001374                                 BNE     1$              ;BR IF NO
026670  062704  000002                        ADD     #2,R4           ;SKIP OVER THE '$REG' LOCATION
026674  012703  000106                        MOV     #<$NEXT-$TATUS>+2,R3    ;NUMBER OF LOCNS TO BE CLEARED
026700  005024                          2$:   CLR     (R4)+           ;CLEAR THE LOCATION
026702  162703  000002                        SUB     #2,R3           ;DONE CLEARING YET ?
026706  001374                                 BNE     2$              ;BR IF NO
026710  062704  002004                        ADD     #<$DRVSN-$FIRST>,R4     ;SKIP OVER FIRST FLAG, MIN/MAX ADRS
                                                                        ;LIMITS AND BAD SECTOR TABLE
026714  012703  000064                        MOV     #<$RMCS3-$DRVSN>+2,R3   ;NUMBER OF LOCNS TO BE CLEARED
026720  012764  177777  177776                MOV     #-1,-2(R4)      ;INITIALIZE TERMINATOR FOR BAD SECTOR TABLE
026726  005024                          3$:   CLR     (R4)+           ;CLEAR A LOCATION
026730  162703  000002                        SUB     #2,R3           ;DONE CLEARING YET ?
026734  001374                                 BNE     3$              ;BR IF NO
026736  113760  001514  000024                MOVB    BEGCOD,$CODE(R0)        ;INITIAL COMMAND CODE
026744  013701  001514                        MOV     BEGCOD,R1       ;GET THE ACTUAL OP CODE
026750  116160  002076  000002                MOVB    COMTBL(R1),$COMND(R0)   ;OPERATION CODE
026756  113760  001512  000030                MOVB    BEGPAT,$PATTC(R0)       ;PATTERN CODE
026764  106360  000030                        ASLB    $PATTC(R0)      ;CONVERT CODE TO A TABLE INDEX
026770  013760  001516  000020                MOV     BEGWC,$WRDL(R0) ;BEGINNING WORD COUNT
026776  013760  001516  000004                MOV     BEGWC,$WCNT(R0) ;VALUE FOR DATA TRANSFER
027004  005460  000004                        NEG     $WCNT(R0)       ;MAKE IT INTO 2'S COMPLEMENT
027010  012760  000400  000022                MOV     #256.,$SSEC(R0) ;INITIAL VALUE OF SECTOR SIZE
027016  012760  000001  000104                MOV     #1,$PASSC(R0)   ;PRESET PASS COUNT TO 1
027024  132760  000001  000024                BITB    #1,$CODE(R0)    ;HEADER COMMAND ?
027032  001403                                 BEQ     4$              ;BR IF NOT
027034  062760  000002  000022                ADD     #2,$SSEC(R0)    ;ADD HEADER SIZE TO SECTOR SIZE
027042                                  4$:
027042  012605                                 MOV     (SP)+,R5        ;;POP STACK INTO R5
027044  012604                                 MOV     (SP)+,R4        ;;POP STACK INTO R4
027046  012603                                 MOV     (SP)+,R3        ;;POP STACK INTO R3
027050  012601                                 MOV     (SP)+,R1        ;;POP STACK INTO R1
027052  000207                                 RTS     PC              ;RETURN

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

```
                                        ;       JSR     PC,DRVPRM       ;CALL ROUTINE
                                        ;
                                        ;RO = DPB ADDRESS BEFORE CALLING THE ROUTINE
                                        DRVPRM: MOV     R3,-(SP)        ;SAVE R3
                                                MOV     R4,-(SP)        ;SAVE R4
                                                TSTB    $AUTOB          ;RUNNING IN AUTO MODE ?
                                                BNE     1$              ;BR IF YES
                                                TST     CHGADR          ;PROGRAM STARTED AT 200 ?
                                                BGT     1$              ;BR IF YES
                                                TST     DRVPAR          ;CHANGE DRIVE PARAMETERS ?
                                                BNE     1$              ;BR IF NO
                                                TYPE    ,ENTLMT         ;'ENTER ADDRESS LIMITS'

                                        1$:     JSR     PC,GETLMT       ;GET ADDRESS LIMITS
                                                ADD     #-1,$FIRST(R0)  ;SEE IF FIRST TIME STARTED
                                                BCS     4$              ;BR IF NOT
                                                MOV     CYLIMT,MAXCYL(R0)       ;LOAD MAXIMUM CYLINDER
                                                MOV     TRKLMT,MAXTRK(R0)       ;LOAD MAXIMUM TRACK
                                                MOV     SECLMT,MAXSEC(R0)       ;LOAD MAXIMUM SECTOR
                                                TST     FEFLAG          ;USING FE CYLINDERS ONLY ?
                                                BNE     2$              ;BR IF NO
                                                MOV     FE1,MINCYL(R0)  ;RESET MINIMUM CYLINDER ADDRESS
                                                BR      3$
                                        2$:     CLR     MINCYL(R0)      ;CLEAR MINIMUM CYLINDER
                                        3$:     CLR     MINTRK(R0)      ;CLEAR MINIMUM TRACK
                                                CLR     MINSEC(R0)      ;CLEAR MINIMUM SECTOR

                                        4$:     TSTB    $AUTOB          ;RUNNING IN AUTO MODE ?
                                                BNE     9$              ;BR IF YES
                                                TST     CHGADR          ;PROGRAM STARTED AT 200 ?
                                                BGT     9$              ;BR IF YES
                                                TST     DRVPAR          ;CHANGE DRIVE PARAMETERS ?
                                                BNE     9$              ;BR IF NO
                                                MOV     TABLE(R4),R3    ;PARAMETER TABLE ADDRESS
                                                MOV     CYLIMT,2(R3)    ;LOAD CYLINDER LIMIT FOR MINCYL
                                                MOV     CYLIMT,10(R3)   ;LOAD CYLINDER LIMIT FOR MAXCYL
                                                MOV     TRKLMT,16(R3)   ;LOAD TRACK LIMIT FOR MINTRK
                                                MOV     TRKLMT,24(R3)   ;LOAD TRACK LIMIT FOR MAXTRK
                                                MOV     SECLMT,32(R3)   ;LOAD SECTOR LIMIT FOR MINSEC
                                                MOV     SECLMT,40(R3)   ;LOAD SECTOR LIMIT FOR MAXSEC
                                                JSR     PC,PARENT       ;GET THE DRIVE'S PARAMETERS

                                                MOV     MINCYL(R0),R3   ;STORE MINCYL VALUE
                                                MOV     MAXCYL(R0),R4   ;STORE MAXCYL VALUE
                                                CMP     R3,R4           ;IS MIN. LESS THAN OR EQUAL TO MAX. ?
                                                BLE     5$              ;BR IF YES
                                                MOV     R3,MAXCYL(R0)   ;SWAP MIN. TO MAX.
                                                MOV     R4,MINCYL(R0)   ;SWAP MAX. TO MIN.
                                        5$:     MOV     MINTRK(R0),R3   ;STORE MINTRK VALUE
                                                MOV     MAXTRK(R0),R4   ;STORE MAXTRK VALUE
                                                CMP     R3,R4           ;IS MIN. LESS THAN OR EQUAL TO MAX. ?
                                                BLE     6$              ;BR IF YES
                                                MOV     R3,MAXTRK(R0)   ;SWAP MIN. TO MAX.
                                                MOV     R4,MINTRK(R0)   ;SWAP MAX. TO MIN.
                                        6$:     MOV     MINSEC(R0),R3   ;STORE MINSEC VALUE
                                                MOV     MAXSEC(R0),R4   ;STORE MAXSEC VALUE
```

```
                                        CMP     R3,R4           ;IS MIN. LESS THAN OR EQUAL TO MAX. ?
                                        BLE     7$              ;BR IF YES
                                        MOV     R3,MAXSEC(R0)   ;SWAP MIN. TO MAX.
                                        MOV     R4,MINSEC(R0)   ;SWAP MAX. TO MIN.

                                7$:     TST     FEFLAG          ;USING FE CYLINDERS ONLY ?
                                        BNE     9$              ;BR IF NO
                                        CMP     MINCYL(R0),FE1  ;IS MIN. CYLINDER < 1ST FE CYLINDER ?
                                        BHI     8$              ;BR IF NO
                                        MOV     FE1,MINCYL(R0)  ;YES, RESET MIN. CYLINDER
                                8$:     CMP     MAXCYL(R0),FE1  ;IS MAX. CYLINDER < 1ST FE CYLINDER ?
                                        BHIS    9$              ;BR IF NO
                                        MOV     FE2,MAXCYL(R0)  ;YES, RESET MAX. CYLINDER

                                9$:     MOV     MINCYL(R0),$CYL(R0)     ;INITIAL CYLINDER VALUE
                                        MOVB    MINTRK(R0),$TRK(R0)     ;INITIAL TRACK VALUE
                                        MOVB    MINSEC(R0),$SEC(R0)     ;INITIAL SECTOR VALUE
                                        MOV     (SP)+,R4        ;;POP STACK INTO R4
                                        MOV     (SP)+,R3        ;;POP STACK INTO R3
                                        RTS     PC              ;RETURN

                                ;ROUTINE TO GET THE ADDRESS LIMITS FOR THE CURRENT DRIVE TYPE
                                ;CALL:
                                ;       JSR     PC,GETLMT       ;CALL ROUTINE
                                ;
                                ;R0 = DPB ADDRESS BEFORE CALLING THE ROUTINE

                                GETLMT: TST     FEFLAG          ;USING FE CLYINDERS ONLY ?
                                        BNE     1$              ;BR IF NO
                                        MOV     FE2,CYLIMT      ;GET 2ND FE CYLINDER
                                        BR      2$

                                1$:     MOV     #558.,CYLIMT    ;GET LAST CYLINDER
                                2$:     MOV     #13.,TRKLMT     ;GET LAST TRACK FOR AN RM80
                                        MOV     #30.,SECLMT     ;GET LAST SECTOR
                                        BIT     #SSEI,$RMOF(R0) ;IS SKIP SECTOR INHIBIT SET ?
                                        BNE     3$              ;BR IF YES
                                        TST     $SSENB(R0)      ;WAS SKIP SECTORING ENABLED DURING XFER ?
                                        BEQ     4$              ;BR IF NO
                                3$:     MOV     #31.,SECLMT     ;GET LAST SECTOR
                                4$:     RTS     PC              ;RETURN
```

GENERAL SUPPORT SUBROUTINES

```
                                        ;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         ;DP3 ADDRESS
                                        ;       JSR     PC,GETID        ;CALL ROUTINE
                                        ;
                                        ;R0 = DPB ADDRESS BEFORE CALLING THE ROUTINE

027550                                  GETID:
027550  010046                                 MOV     R0,-(SP)        ;;PUSH R0 ON STACK
027552  010146                                 MOV     R1,-(SP)        ;;PUSH R1 ON STACK
027554  010246                                 MOV     R2,-(SP)        ;;PUSH R2 ON STACK
027556  010546                                 MOV     R5,-(SP)        ;;PUSH R5 ON STACK
027560  010002                                 MOV     R0,R2           ;GET INDEX TO DPB
027562  004737  045062                        JSR     PC,SVRH70       ;SAVE ALL REGISTERS
027566  012702  000004                        MOV     #4,R2           ;FOUR DIGITS TO STORE
027572  016001  002170                        MOV     $RMSN(R0),R1    ;SERIAL NUMBER
027576  005005                          1$:   CLR     R5              ;ZERO
027600  006101                                 ROL     R1              ;PUT THE NEXT DIGIT
027602  006105                                 ROL     R5              ;INTO R5
027604  006101                                 ROL     R1
027606  006105                                 ROL     R5
027610  006101                                 ROL     R1
027612  006105                                 ROL     R5
027614  006101                                 ROL     R1
027616  006105                                 ROL     R5
027620  062705  000060                        ADD     #'0,R5          ;MAKE IT ASCII
027624  110560  002130                        MOVB    R5,$DRVSN(R0)   ;SAVE DRIVE (MBA) SERIAL NUMBER DIGIT
027630  005200                                 INC     R0              ;GET NEXT INDEX FOR DRIVE (MBA) SERIAL NUMBER
027632  005302                                 DEC     R2              ;ALL DIGITS TYPED?
027634  003360                                 BGT     1$              ;NO -- BRANCH
027636  012605                                 MOV     (SP)+,R5        ;;POP STACK INTO R5
027640  012602                                 MOV     (SP)+,R2        ;;POP STACK INTO R2
027642  012601                                 MOV     (SP)+,R1        ;;POP STACK INTO R1
027644  012600                                 MOV     (SP)+,R0        ;;POP STACK INTO R0
027646  000207                                 RTS     PC              ;RETURN
```

READ DEC144 FILE

```
                                        .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 HDA 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 HDA
                                        ;       SERIAL NUMBER CHANGED SINCE THE LAST TIME IT WAS
                                        ;       READ. (DEFAULT)
                                        ;
                                        ;NOTE: IF THE SERIAL NUMBER HAS CHANGED, THIS MOST LIKELY MEANS THAT THE
                                        ;      HDA OR DRIVE HAD BEEN REPLACED WHILE THE DRIVE WAS DEASSIGNED.
                                        ;
                                        ;THIS ROUTINE CHECKS THAT THE TWO SERIAL NUMBER WORDS ARE NOT ZERO
                                        ;AND THE ENTIRE SERIAL NUMBER IS POSITIVE. 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

027650                                  GETADR:
027650  010146                                 MOV     R1,-(SP)        ;;PUSH R1 ON STACK
027652  010246                                 MOV     R2,-(SP)        ;;PUSH R2 ON STACK
027654  010346                                 MOV     R3,-(SP)        ;;PUSH R3 ON STACK
027656  004737  027462                        JSR     PC,GETLMT       ;GET ADDRESS LIMITS
027662  010001                                 MOV     R0,R1           ;DPB ADDRESS
027664  062701  000146                        ADD     #$BDSEC,R1      ;ADDRESS OF BAD SECTOR TABLE
027670  010146                                 MOV     R1,-(SP)        ;;PUSH R1 ON STACK
027672  111037  067556                        MOVB    (R0),GENDPB     ;DRIVE NUMBER
027676  012737  001056  067570                MOV     #558.,GENDPB+$CYL       ;LAST CYLINDER
027704  113737  001426  067567                MOVB    TRKLMT,GENDPB+$TRK      ;GET LAST TRACK
027712  112737  000000  067566                MOVB    #0,GENDPB+$SEC  ;GET STARTING SECTOR OF 16 BIT MFG FILE
027720  012737  177400  067562                MOV     #-256.,GENDPB+$WCNT     ;ONE SECTOR WORD COUNT
027726  112737  000171  067560                MOVB    #RDDAT,GENDPB+$COMND    ;READ DATA COMMAND
027734  012737  000010  001270                MOV     #8.,$CDW2       ;GET LAST SECTOR OF 16 BIT MFG FILE
027742  012703  101174                1$:     MOV     #CYLNDR,R3      ;GET READ BUFFER ADDRESS
027746  004037  041000                        JSR     R0,RM80         ;READ CURRENT SECTOR
027752  067556                               GENDPB
027754  000772                               BR      1$              ;WAIT FOR QUE
027756  005737  067574                2$:     TST     GENDPB+$STATUS  ;READ DONE YET ?
027762  001775                               BEQ     2$              ;BR IF NO
027764  100010                               BPL     3$              ;BR IF NO ERROR, ELSE
027766  062737  000002  067566                ADD     #2,GENDPB+$SEC  ;INCREMENT NEXT SECTOR TO READ
027774  123737  001270  067566                CMPB    $CDW2,GENDPB+$SEC       ;WERE ALL SECTORS TRIED ?
030002  103357                               BHIS    1$              ;BR IF NO
```

```
 55 030004  000470                          BR      10$             ;BR IF UNSUCCESSFUL ON RETRIES
 56 030006  005723                  3$:     TST     (R3)+           ;DON'T CHECK LSB'S OF SERIAL NUMBER
 57 030010  005723                          TST     (R3)+           ;ARE MSB'S OF SERIAL NUMBER VALID ?
 58 030012  100462                          BMI     9$              ;BR IF MINUS (CORRUPT)
 59 030014  001003                          BNE     4$              ;BR IF NOT ZERO (PLUS)
 60 030016  005763  177774                  TST     -4(R3)          ;ARE SERIAL NUMBERS ZERO ?
 61 030022  001456                          BEQ     9$              ;BR IF YES (CORRUPT)
 62 030024  005723                  4$:     TST     (R3)+           ;IS 3RD WORD ALL 0'S ?
 63 030026  001054                          BNE     9$              ;BR IF NO (CORRUPT)
 64 030030  005723                          TST     (R3)+           ;IS 4TH WORD ALL 0'S ?
 65 030032  001052                          BNE     9$              ;BR IF NO (CORRUPT)
 66 030034  123727  067566  000012          CMPB    GENDPB+$SEC,#10.        ;READING USR BAD FILE ?
 67 030042  103021                          BHIS    6$              ;BR IF YES
 68 030044  005737  001510                  TST     BADBLK          ;INIT. BAD SECTOR TABLE ENTRIES ?
 69 030050  001010                          BNE     5$              ;BR IF YES
 70 030052  026360  177770  000142          CMP     -10(R3),$HSNL(R0)       ;ARE LSB'S OF S/N SAME AS BEFORE ?
 71 030060  001004                          BNE     5$              ;BR IF NO
 72 030062  026360  177772  000144          CMP     -6(R3),$HSNM(R0)        ;ARE MSB'S OF S/N SAME AS BEFORE ?
 73 030070  001464                          BEQ     13$             ;BR IF YES
 74 030072  016360  177770  000142  5$:     MOV     -10(R3),$HSNL(R0)       ;STORE HDA SERIAL NUMBER
 75 030100  016360  177772  000144          MOV     -6(R3),$HSNM(R0)
 76
 77 030106  012702  000176          6$:     MOV     #126.,R2        ;NUMBER OF ENTRIES PER FILE (MFG/USR)
 78 030112  012321                  7$:     MOV     (R3)+,(R1)+     ;STORE BAD CYLINDER ADDRESS
 79 030114  012321                          MOV     (R3)+,(R1)+     ;STORE BAD TRK/SEC ADDRESS
 80 030116  005302                          DEC     R2              ;DONE WITH ENTRIES ?
 81 030120  001374                          BNE     7$              ;BR IF NO
 82 030122  123727  067566  000012          CMPB    GENDPB+$SEC,#10.        ;USR BAD FILE DONE YET ?
 83 030130  103044                          BHIS    13$             ;BR IF YES
 84 030132  112737  000012  067566          MOVB    #10.,GENDPB+$SEC        ;GET STARTING SECTOR OF USR FILE
 85 030140  012737  000036  001270          MOV     #30.,$CDW2      ;GET LAST SECTOR OF USR FILE
 86 030146  011601                          MOV     (SP),R1         ;GET BEGINNING OF $BDSEC TABLE
 87 030150  005721                  8$:     TST     (R1)+           ;IS THIS TERMINATOR ?
 88 030152  100376                          BPL     8$              ;BR IF NO
 89 030154  005741                          TST     -(R1)           ;FOUND TERMINATOR, BACKUP 1 WORD
 90 030156  000671                          BR      1$
 91
 92 030160  104401  076442          9$:     TYPE    ,MERR2          ;REPORT, INVALID DEC144 FILE STRUCTURE
 93 030164  000402                          BR      11$
 94 030166  104401  076362          10$:    TYPE    ,MERR1          ;REPORT, FAILED TO RETRIEVE DEC144 FILES
 95 030172  104401  075304          11$:    TYPE    ,UNTMSG         ;TYPE 'ON DRIVE'
 96 030176  011046                          MOV     (R0),-(SP)      ;;SAVE (R0) FOR TYPEOUT
    030200  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
    030202     002                          .BYTE   2               ;;TYPE 2 DIGIT(S)
    030203     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
 97 030204  104401  001203                  TYPE    ,$CRLF          ;CR-LF
 98 030210  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
 99 030212  012702  000374                  MOV     #252.,R2        ;TOTAL NUMBER OF ENTRIES ALLOWED
100 030216  012721  177777          12$:    MOV     #-1,(R1)+       ;INITIALIZE CYLINDER LOCATIONS TO -1
101 030222  012721  177777                  MOV     #-1,(R1)+       ;INITIALIZE TRK/SEC LOCATIONS TO -1
102 030226  005302                          DEC     R2              ;DONE YET ?
103 030230  001372                          BNE     12$             ;BR IF NO
104 030232  012760  177777  000144          MOV     #-1,$HSNM(R0)   ;INDICATE SERIAL NUMBER IS UNKNOWN
105 030240  000401                          BR      14$
106 030242  005726                  13$:    TST     (SP)+           ;RESTORE STACK
107 030244                          14$:
    030244  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
```

READ DEC144 FILE

```
        030246  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
        030250  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
    108 030252  000205                          RTS     R5              ;EXIT
```

ENTER BAD SECTOR ROUTINE

```
                                .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

030254                          MANTER:
030254  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
030256  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
030260  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
030262  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
030264  105737  001150                  TSTB    $AUTOB          ;RUNNING IN AUTO MODE ?
030270  001162                          BNE     19$             ;BRANCH IF SO
030272  005737  001442                  TST     DRVPAR          ;CHANGE DRIVE PARAMETERS ?
030276  001157                          BNE     19$             ;BR IF NO
030300  005037  001340          1$:     CLR     CFLAG           ;CLEAR THE CONTROL-C FLAG
030304  104401  076302                  TYPE    ,ENTADR         ;MESSAGE TO ENTER...

030310  012704  000146                  MOV     #$BDSEC,R4      ;INDEX VALUE OF TABLE ADDRESS
030314  060004                          ADD     R0,R4           ;TABLE STARTING ADDRESS
030316  012701  000374                  MOV     #<126.*2>,R1    ;256. TOTAL BAD SECTORS ALLOWED
030322  022714  177777          2$:     CMP     #-1,(R4)        ;ENTRY IN THE TABLE ?
030326  001407                          BEQ     3$              ;BRANCH IF SO
030330  062704  000004                  ADD     #4,R4           ;ADJUST THE TABLE ENTRY POINTER
030334  005301                          DEC     R1              ;DECREMENT THE BAD SECTOR COUNT
030336  001371                          BNE     2$              ;BR IF TO NEXT ENTRIES POSITION
030340  104401  076506                  TYPE    ,MSFULL         ;TYPE 'BAD SECTOR TABLE IS FULL'
030344  000534                          BR      19$             ;EXIT..

030346  010146                  3$:     MOV     R1,-(SP)        ;SAVE THE COUNTER AND FIRST
030350  010446                          MOV     R4,-(SP)        ;ENTRY POINTER PAIR
030352  012714  177777          4$:     MOV     #-1,(R4)        ;RESET CYLINDER TO -1
030356  012764  177777  000002          MOV     #-1,2(R4)       ;RESET TRACK/SECTOR FIELD TO -1
030364  104401  076544                  TYPE    ,MSGCTS         ;TYPE 'CYL,TRK,SEC = '
030370  104411                          RDLIN                   ;READ THE ADDRESS
030372  012601                          MOV     (SP)+,R1        ;READ IN TEXT ADDRESS
030374  005737  001340                  TST     CFLAG           ;CONTROL-C ENTERED ?
030400  001011                          BNE     5$              ;BRANCH IF YES
030402  105761  000001                  TSTB    1(R1)           ;WAS IT TERMINATED WITH CARRIAGE RETURN ?
030406  001021                          BNE     7$              ;BR IF NO
030410  122711  000114                  CMPB    #'L,(R1)        ;WAS CHARACTER AN 'L' ?
030414  001016                          BNE     7$              ;BR IF NO
030416  004737  030650                  JSR     PC,TYLIST       ;TYPE BAD SECTOR LIST FOR USER
030422  000753                          BR      4$
030424  012604                  5$:     MOV     (SP)+,R4        ;RETRIEVE THE ENTRY POINTER
030426  012601                          MOV     (SP)+,R1        ;RETRIEVE THE COUNT
030430  012724  177777          6$:     MOV     #-1,(R4)+       ;RESET THE TABLE
030434  012724  177777                  MOV     #-1,(R4)+       ;TO -1
030440  005301                          DEC     R1              ;ALL DONE ?
030442  001372                          BNE     6$              ;BRANCH IF NOT
030444  104401  077262                  TYPE    ,ALOST          ;TYPE ' * ALL CURRENT ENTRIES LOST *'
030450  000713                          BR      1$              ;ENTER AGAIN FROM THE FIRST POINTER

030452                          7$:
```

```
ENTER BAD SECTOR ROUTINE

        030452  013702  001422                  MOV     CYLIMT,R2       ;UPPER LIMIT OF INPUT
        030456  004537  033072                  JSR     R5,CK.DIG       ;CHECK THE DIGIT(S)
        030462  030632                          18$                     ;CARRIAGE RETURN ONLY ENTERED
        030464  030632                          18$                     ;PERIOD ONLY ENTERED
        030466  030624                          17$                     ;ILLEGAL INPUT
        030470  030476                          8$                      ;TERMINATED WITH A CARRIAGE RETURN
        030472  030506                          10$                     ;TERMINATED WITH A ','
        030474  030502                          9$                      ;TERMINATED WITH A '.'
     54 030476  010214                  8$:     MOV     R2,(R4)         ;CYLINDER ADDRESS
     55 030500  000444                          BR      16$             ;FINISH WITH THE CURRENT ADDRESS
     56 030502  010214                  9$:     MOV     R2,(R4)         ;CYLINDER ADDRESS
     57 030504  000452                          BR      18$             ;EXIT,PERIOD ENTERED
     58 030506  010214                  10$:    MOV     R2,(R4)         ;CYLINDER ADDRESS FOLLOWED BY ','
     59
     60 030510  013702  001426                  MOV     TRKLMT,R2       ;UPPER LIMIT OF INPUT
        030514  004537  033072                  JSR     R5,CK.DIG       ;CHECK THE DIGIT(S)
        030520  030632                          18$                     ;CARRIAGE RETURN ONLY ENTERED
        030522  030632                          18$                     ;PERIOD ONLY ENTERED
        030524  030624                          17$                     ;ILLEGAL INPUT
        030526  030534                          11$                     ;TERMINATED WITH A CARRIAGE RETURN
        030530  030550                          13$                     ;TERMINATED WITH A ','
        030532  030542                          12$                     ;TERMINATED WITH A '.'
     61 030534  110264  000003          11$:    MOVB    R2,3(R4)        ;TRACK ADDRESS
     62 030540  000424                          BR      16$             ;TRACK NUMBER FOLLOWED BY CR
     63 030542  110264  000003          12$:    MOVB    R2,3(R4)        ;TRACK ADDRESS
     64 030546  000431                          BR      18$             ;EXIT, TRACK NUMBER FOLLOWED BY '.'
     65 030550  110264  000003          13$:    MOVB    R2,3(R4)        ;TRACK ADDRESS FOLLOWED BY ','
     66
     67 030554  013702  001424                  MOV     SECLMT,R2       ;UPPER LIMIT OF INPUT
        030560  004537  033072                  JSR     R5,CK.DIG       ;CHECK THE DIGIT(S)
        030564  030632                          18$                     ;CARRIAGE RETURN ONLY ENTERED
        030566  030632                          18$                     ;PERIOD ONLY ENTERED
        030570  030624                          17$                     ;ILLEGAL INPUT
        030572  030606                          15$                     ;TERMINATED WITH A CARRIAGE RETURN
        030574  030624                          17$                     ;TERMINATED WITH A ','
        030576  030600                          14$                     ;TERMINATED WITH A '.'
     68 030600  110264  000002          14$:    MOVB    R2,2(R4)        ;SECTOR ADDRESS
     69 030604  000412                          BR      18$             ;EXIT,SECTOR ADDRESS FOLLOWED BY '.'
     70 030606  110264  000002          15$:    MOVB    R2,2(R4)        ;SECTOR ADDRESS
     71
     72 030612  005303                  16$:    DEC     R3              ;MORE ENTRYS ?
     73 030614  001406                          BEQ     18$             ;BRANCH IF EXHAUSTED
     74 030616  062704  000004                  ADD     #4,R4           ;ADJUST FOR THE NEXT TABLE ENTRY
     75 030622  000653                          BR      4$              ;ENTER NEXT SECTOR ADDRESS
     76
     77 030624  104401  076341          17$:    TYPE    ,BADENT         ;MESSAGE BAD ENTRY
     78 030630  000650                          BR      4$              ;ENTER SECTOR ADDRESS AGAIN
     79 030632  062706  000004          18$:    ADD     #4,SP           ;CLEAR OFF THE STACK POINT
     80 030636                          19$:
        030636  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
        030640  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
        030642  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
        030644  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
     81 030646  000207                          RTS     PC              ;EXIT
     82
     83                                 .SBTTL  TYPE BAD SECTOR LIST
     84
```

TYPE BAD SECTOR LIST

```
                                        ;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

030650                                  TYLIST:
030650  010146                                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
030652  104401  077216                          TYPE    ,LSTHDR         ;TYPE 'DEC144 AND MANUAL BAD SECTOR LIST'
030656  012701  000146                          MOV     #$BDSEC,R1      ;INDEX VALUE OF TABLE ADDRESS
030662  060001                                  ADD     R0,R1           ;TABLE STARTING ADDRESS
030664  010146                                  MOV     R1,-(SP)        ;SAVE ADDRESS FOR LATER
030666  022711  177777                1$:       CMP     #-1,(R1)        ;TERMINATOR OR NO ENTRY IN THE TABLE ?
030672  001444                                  BEQ     6$              ;BRANCH IF YES
030674  011146                                  MOV     (R1),-(SP)      ;GET CYLINDER NUMBER
030676  004737  033230                          JSR     PC,$SB2D        ;CONVERT NUMBER
030702  004737  032264                          JSR     PC,SUPRSL       ;LEFT JUSTIFY AND TYPE
030706  005046                                  CLR     -(SP)           ;CLEAR HI BYTE AND PUSH STACK
030710  116116  000003                          MOVB    3(R1),(SP)      ;GET TRACK NUMBER
030714  100407                                  BMI     2$              ;BR IF ALL BAD
030716  104401  075276                          TYPE    ,COMMA          ;TYPE ','
030722  004737  033230                          JSR     PC,$SB2D        ;CONVEPT NUMBER
030726  004737  032264                          JSR     PC,SUPRSL       ;LEFT JUSTIFY AND TYPE
030732  000401                                  BR      3$
030734  005726                        2$:       TST     (SP)+           ;RESTORE STACK
030736  005046                        3$:       CLR     -(SP)           ;CLEAR HI BYTE AND PUSH STACK
030740  116116  000002                          MOVB    2(R1),(SP)      ;GET SECTOR NUMBER
030744  100407                                  BMI     4$              ;BR IF ALL BAD
030746  104401  075276                          TYPE    ,COMMA          ;TYPE ','
030752  004737  033230                          JSR     PC,$SB2D        ;CONVERT NUMBER
030756  004737  032264                          JSR     PC,SUPRSL       ;LEFT JUSTIFY AND TYPE
030762  000401                                  BR      5$
030764  005726                        4$:       TST     (SP)+           ;RESTORE STACK
030766  104401  001203                5$:       TYPE    ,$CRLF          ;CR-LF
030772  062701  000004                          ADD     #4,R1           ;INCREMENT POINTER
030776  005737  001340                          TST     CFLAG           ;CONTROL-C ENTERED ?
031002  001731                                  BEQ     1$              ;BRANCH IF NO
031004  022601                        6$:       CMP     (SP)+,R1        ;ANY ENTRIES ?
031006  001002                                  BNE     7$              ;BR IF YES
031010  104401  077176                          TYPE    ,NOENTY         ;TYPE 'NO ENTRIES'
031014  104401  001203                7$:       TYPE    ,$CRLF          ;CR-LF
031020  005037  001340                          CLR     CFLAG           ;CLEAR CONTROL FLAG
031024  012601                                  MOV     (SP)+,R1        ;;POP STACK INTO R1
031026  000207                                  RTS     PC              ;RETURN
```

```
                                .SBTTL  PARAMETER ENTRY ROUTINE

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

 031030  010346                 PARENT: MOV     R3,-(SP)        ;SAVE THE PARAMETER TABLE ADDRESS
 031032  005037  001340                 CLR     CFLAG           ;CLEAR THE 'CONTROL C' FLAG
 031036  012337  031046         1$:     MOV     (R3)+,3$        ;ADDRESS OF PARAMETER NAME
 031042  001442                         BEQ     9$              ;BR IF AT END OF TABLE
 031044  104401                         TYPE                    ;TYPE THE PARAMETER NAME
 031046  000000                 3$:     .WORD   0               ;ADDRESS OF PARAMETER NAME TEXT
 031050  012302                         MOV     (R3)+,R2        ;MAXIMUM PARAMETER VALUE
 031052  012305                         MOV     (R3)+,R5        ;ADDRESS OF PARAMETER
 031054  011546                         MOV     (R5),-(SP)      ;CURRENT VALUE OF PARAMETER
 031056  104405                         TYPDS                   ;TYPE THE CURRENT VALUE OF THE PARAMETER
 031060  104401  077572                 TYPE    ,SLASH          ;' / '
 031064  104411                         RDLIN                   ;READ THE KEYBOARD
 031066  012601                         MOV     (SP)+,R1        ;INPUT ASCII STRING ADDRESS
 031070  005737  001340                 TST     CFLAG           ;'CONTROL C' ENTERED ?
 031074  001021                         BNE     8$              ;BR IF IT WAS
 031076  004537  033072                 JSR     R5,CK.DIG       ;CHECK THE DIGIT(S)
 031102  031036                         1$                      ;CARRIAGE RETURN ONLY ENTERED
 031104  031150                         9$                      ;PERIOD ONLY ENTERED
 031106  031122                         6$                      ;ILLEGAL INPUT
 031110  031116                         5$                      ;TERMINATED WITH A CARRIAGE RETURN
 031112  031122                         6$                      ;TERMINATED WITH A ','
 031114  031134                         7$                      ;TERMINATED WITH A '.'
 031116  010215                 5$:     MOV     R2,(R5)         ;MOVE NEW VALUE TO PARAMETER LOCATION
 031120  000746                         BR      1$              ;GET MORE PARAMETERS
 031122  104401  076341         6$:     TYPE    ,BADENT         ;'BAD ENTRY'
 031126  162703  000006                 SUB     #6,R3           ;DECREMENT THE TABLE POINTER
 031132  000741                         BR      1$              ;TRY AGAIN
 031134  010215                 7$:     MOV     R2,(R5)         ;NEW VALUE
 031136  000404                         BR      9$              ;EXIT
 031140  005037  001340         8$:     CLR     CFLAG           ;CLEAR THE 'CONTROL C' FLAG
 031144  011603                         MOV     (SP),R3         ;RELOAD THE PARAMETER TABLE ADDRESS
 031146  000733                         BR      1$              ;TRY AGAIN
 031150  005726                 9$:     TST     (SP)+           ;CORRECT THE STACK POINTER
 031152  000207                         RTS     PC              ;RETURN
```

```
PARAMETER ENTRY ROUTINE

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

031154  104401  001203          ASNERR: TYPE    ,$CRLF          ;CR-LF
031160  104401  075272                  TYPE    ,QUES           ;'?'
031164  104401  075304                  TYPE    ,UNTMSG         ;TYPE 'DRIVE'
031170  010446                          MOV     R4,-(SP)        ;;SAVE R4 FOR TYPEOUT
                                                                ;;TYPE DRIVE NUMBER
031172  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
031174     002                          .BYTE   2               ;;TYPE 2 DIGIT(S)
031175     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
031176  104401                          TYPE                    ;TYPE SPECIFIC MESSAGE
031200  000000                  ASNMSG: .WORD   0               ;MESSAGE ADDRESS
031202  000207                          RTS     PC

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

031204  005004                  DROP:   CLR     R4              ;CLEAR R4 FOR DRIVE NUMBER
031206  111004                          MOVB    (R0),R4         ;MOVE DRIVE NUMBER TO R4
031210  146437  040300  001542          BICB    ATABIT(R4),ASNLST       ;REMOVE DRIVE FROM ASSIGNED LIST
031216  146437  040300  032012          BICB    ATABIT(R4),AUTLST       ;DELETE DRIVE FROM AUTO ASSIGN LIST
031224  006304                          ASL     R4              ;MAKE DRIVE NUMBER INTO A TABLE INDEX
031226  010064  001544                  MOV     R0,DDRVS(R4)    ;PUT DRIVE IN DROP LIST
031232  104401  001203                  TYPE    ,$CRLF
031236  104401  075700                  TYPE    ,DROPNG         ;TYPE 'FATAL OR EXCESSIVE ERRORS'
031242  104401  075753                  TYPE    ,MSGON          ;TYPE 'ON'
031246  104401  075304                  TYPE    ,UNTMSG         ;TYPE 'DRIVE'
031252  006204                          ASR     R4              ;DRIVE NUMBER
031254  010446                          MOV     R4,-(SP)        ;;SAVE R4 FOR TYPEOUT
                                                                ;;TYPE DRIVE NUMBER
031256  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
031260     002                          .BYTE   2               ;;TYPE 2 DIGIT(S)
031261     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
031262  000207                  1$:     RTS     PC

                                        ;ROUTINE TO DEASSIGN DRIVE IF ERRORS BECOMES EXCESSIVE

031264  032777  000020  147662  ABNRML: BIT     #SW04,@SWR      ;SEE IF SWITCH 4 SET
031272  001006                          BNE     1$              ;BR IF IT'S SET
031274  023760  001456  000072          CMP     MAXER,$TOTAL(R0)        ;CHECK TOTAL ERROR VALUE
031302  101002                          BHI     1$              ;BR IF ERRORS DO NOT EXCEED MAX
031304  000137  031204                  JMP     DROP            ;DEASSING THE DRIVE
031310  000207                  1$:     RTS     PC              ;RETURN

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

```
     1                                  .SBTTL  END OF PASS ROUTINE

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

031312                                  $EOP:
031312  005737  001500                          TST     ENDING          ;END OF PASS DETERMINED BY SEEKS OR DATA WORDS ?
031316  001412                                BEQ     EOP1            ;BR IF SEEKS
031320  026037  000040  001450                CMP     $ENDAT+2(R0),ENDCON+2   ;CHECK MSW OF WORDS DATA COUNT
031326  101020                                BHI     EOP2            ;BR IF MSW GREATER THAN LIMIT
031330  103404                                BLO     1$              ;BR IF MSW LESS THAN LIMIT
031332  026037  000036  001446                CMP     $ENDAT(R0),ENDCON       ;CHECK LSW AGAINST LIMIT
031340  103013                                BHIS    EOP2            ;BR IF EQUAL OR GREATER
031342  000207                          1$:   RTS     PC

031344  026037  000044  001454        EOP1:   CMP     $ENDSK+2(R0),ENDSEK+2   ;CHECK MSW OF SEEK COUNT
031352  101006                                BHI     EOP2            ;BR IF MSW GREATER THAN LIMIT
031354  103404                                BLO     1$              ;EXIT IF MSW LESS THAN LIMIT
031356  026037  000042  001452                CMP     $ENDSK(R0),ENDSEK       ;CHECK LSW OF SEEK COUNT
031364  103001                                BHIS    EOP2            ;BR IF EQUAL OR GREATER
031366  000207                          1$:   RTS     PC

031370  010446                          EOP2:   MOV     R4,-(SP)        ;SAVE R4
031372  032777  000400  147554                BIT     #SW08,@SWR      ;INHIBIT END OF PASS TYPEOUT ?
031400  001023                                BNE     1$              ;BR IF YES
031402  104401  001203                        TYPE    ,$CRLF          ;CR-LF
031406  104401  075734                        TYPE    ,ENDPAS         ;END OF PASS FOR THE DRIVE
031412  016046  000104                        MOV     $PASSC(R0),-(SP)        ;;SAVE $PASSC(R0) FOR TYPEOUT
031416  104405                                TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
031420  111037  001324                        MOVB    (R0),DRVNO      ;STORE THE DRIVE NUMBER
031424  104401  075753                        TYPE    ,MSGON          ;TYPE 'ON'
031430  104401  075304                        TYPE    ,UNTMSG         ;'DRIVE '
031434  013746  001324                        MOV     DRVNO,-(SP)     ;;SAVE DRVNO FOR TYPEOUT
031440  104403                                TYPOS                   ;;GO TYPE--OCTAL ASCII
031442     002                                .BYTE   2               ;;TYPE 2 DIGIT(S)
031443     000                                .BYTE   0               ;;SUPPRESS LEADING ZEROS
031444  104401  001203                        TYPE    ,$CRLF          ;CR-LF
031450  111004                          1$:   MOVB    (R0),R4         ;MOVE DRIVE NUMBER

031452  105737  001150                        TSTB    $AUTOB          ;RUNNING IN AUTO MODE ?
031456  001410                                BEQ     2$              ;BR IF NO
031460  136437  040300  032012                BITB    ATABIT(R4),AUTLST       ;IS DRIVE ALREADY ASSIGNED TO AUTO LIST ?
031466  001071                                BNE     6$              ;BR IF YES
031470  156437  040300  032012                BISB    ATABIT(R4),AUTLST       ;ADD DRIVE TO AUTO ASSIGN LIST
031476  000443                                BR      3$

031500  026037  000104  001470  2$:   CMP     $PASSC(R0),PASSES       ;SEE IF AT END OF TEST
031506  103437                                BLO     3$              ;BR IF NOT
031510  032777  000020  147436                BIT     #SW04,@SWR      ;TYPE END OF TEST MESSAGE ?
031516  001033                                BNE     3$              ;BR IF NO
031520  104401  075760                        TYPE    ,ENDTST         ;TYPE 'END OF TEST'
031524  104401  075776                        TYPE    ,MSGFOR         ;TYPE 'FOR'
031530  104401  075304                        TYPE    ,UNTMSG         ;'DRIVE '
031534  013746  001324                        MOV     DRVNO,-(SP)     ;;SAVE DRVNO FOR TYPEOUT
031540  104403                                TYPOS                   ;;GO TYPE--OCTAL ASCII
```

```
031542  002                                     .BYTE   2               ;;TYPE 2 DIGIT(S)
031543  000                                     .BYTE   0               ;;SUPPRESS LEADING ZEROS
031544  146437  040300  001542                  BICB    ATABIT(R4),ASNLST       ;DELETE DRIVE FROM ASSIGNED LIST
031552  006304                                  ASL     R4              ;MAKE DRIVE NUMBER INTO TABLE INDEX
031554  010064  001544                          MOV     R0,DDRVS(R4)    ;PUT BLOCK ADDRESS INTO DROP LIST
031560  105737  001542                          TSTB    ASNLST          ;ALL DRIVES ARE DEASSIGNED ?
031564  001041                                  BNE     7$              ;BR IF NO
031566  005237  001216                          INC     $DEVCT          ;INCREMENT DEVICE COUNT
031572  005237  001214                          INC     $PASS           ;INCREMENT THE PASS COUNT
031576  042737  100000  001214                  BIC     #100000,$PASS   ;AVOID NEGATIVE NUMBER
031604  000431                                  BR      7$

031606  032777  000400  147340  3$:     BIT     #SW08,@SWR      ;INHIBIT END OF PASS TYPEOUT ?
031614  001002                          BNE     4$              ;BR IF YES
031616  004737  023714                  JSR     PC,SUMARY       ;TYPE THE DRIVE'S STATISTICS SUMMARY
031622  010346                  4$:     MOV     R3,-(SP)        ;SAVE R3
031624  010004                          MOV     R0,R4           ;DRIVE'S BLOCK ADDRESS
031626  062704  000036                  ADD     #$ENDAT,R4      ;ADD THE STARTING ADDR OF SECTIONS TO CLEAR
031632  012703  000006                  MOV     #6,R3           ;NUMBER OF LOCNS TO BE CLEARED
                                                                ;(CLEAR $ENDAT, $ENDSK AND $OPERC COUNTERS)
031636  005024                  5$:     CLR     (R4)+           ;CLEAR THE LOCN
031640  005303                          DEC     R3              ;DECREMENT THE LOCATION COUNTER
031642  001375                          BNE     5$              ;BR IF MORE TO GO
031644  012603                          MOV     (SP)+,R3        ;RESTORE R3
031646  005260  000104                  INC     $PASSC(R0)      ;INCREMENT THE PASS COUNT

031652  105737  001150          6$:     TSTB    $AUTOB          ;RUNNING IN AUTO MODE ?
031656  001404                          BEQ     7$              ;BR IF NO
031660  023737  0015'2  032012          CMP     ASNLST,AUTLST   ;HAVE ALL DRIVES COMPLETED PASS IN AUTO MODE ?
031666  001402                          BEQ     8$              ;BR IF YES
031670  012604                  7$:     MOV     (SP)+,R4        ;RESTORE R4
031672  000207                          RTS     PC              ;RETURN

031674  005237  032012          8$:     INC     AUTLST          ;CLEAR AUTO ASSIGN LIST FOR NEXT PASS AND
031700  001375                          BNE     8$              ;WAIT FOR TTY
031702  012737  000000  177776          MOV     #PR0,PS         ;ALLOW INTERRUPTS
031710  005237  001216                  INC     $DEVCT          ;INCREMENT DEVICE COUNT
031714  005237  001214                  INC     $PASS           ;;INCREMENT THE PASS NUMBER
031720  042737  100000  001214          BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
031726  005327                          DEC     (PC)+           ;;LOOP?
031730  000001          $EOPCT: .WORD   1
031732  003013                  BGT     $DOAGN          ;;YES
031734  012737                  MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
031736  000001          $ENDCT: .WORD   1
031740  031730                  $EOPCT
031742  013700  000042  $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
031746  001405                  BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
031750  000005                  RESET                   ;;CLEAR THE WORLD
031752  004710          $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
031754  000240                  NOP                     ;;SAVE ROOM
031756  000240                  NOP                     ;;FOR
031760  000240                  NOP                     ;;ACT11
031762                  $DOAGN:
031762  000137                  JMP     @(PC)+          ;;RETURN
031764  031766          $RTNAD: .WORD   RTURN

2
3 031766  012706  001100  RTURN:  MOV     #STACK,SP       ;RESTORE STACK
```

END OF PASS ROUTINE

```
      4 031772  005237  001212                    INC     $TESTN          ;INCREMENT THE TEST NUMBER IN THE MAIL BOX
      5 031776  004737  033326                    JSR     PC,$TKINT       ;MAKE SURE KEYBOARD INTERRUPT AND
      6 032002  004737  023422                    JSR     PC,CKCLK        ;SYSTEM CLOCK ARE STILL ON.
      7 032006  000137  006240                    JMP     MAIN            ;RETURN TO LOOP
      8
      9 032012  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
                                        ;
 032014  013746  037124         GETREM: MOV     $LONUM,-(SP)    ;STORE RANDOM NUMBER ON THE STACK FOR DIVIDE
 032020  013746  037122                 MOV     $HINUM,-(SP)    ;UPPER PART
 032024  010546                         MOV     R5,-(SP)        ;PUT THE DIVISOR ONTO THE STACK
 032026  004737  032040                 JSR     PC,$DIV         ;DIVIDE THE RANDOM NUMBERS
 032032  012605                         MOV     (SP)+,R5        ;PUT THE REMAINDER INTO R5
 032034  005726                         TST     (SP)+           ;ADJUST THE STACK POINTER
 032036  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').

 032040  104412                 $DIV:   SAVREG                  ;STORE R0 - R5
 032042  016605  000026                 MOV     26(SP),R5       ;DIVISOR
 032046  005004                         CLR     R4              ;OTHER DIVISOR WORD
 032050  016602  000030                 MOV     30(SP),R2       ;UPPER DIVIDEND WORD
 032054  016603  000032                 MOV     32(SP),R3       ;LOWER DIVIDEND WORD
 032060  005000                         CLR     R0              ;CLEAR OTHER DIVIDEND REGISTERS
 032062  005001                         CLR     R1
 032064  004737  032106                 JSR     PC,M.DPID       ;GO TO THE DIVIDE ROUTINE
 032070  010166  000030                 MOV     R1,30(SP)       ;REMAINDER ON THE STACK
 032074  010366  000032                 MOV     R3,32(SP)       ;QUOTIENT ON THE STACK
 032100  104413                         RESREG                  ;RESTORE R0 - R5
 032102  012616                         MOV     (SP)+,(SP)      ;MOVE RETURN UP THE STACK
 032104  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)
                                
032106  012746  000040        M.DPID: MOV     #40,-(SP)       ;COUNTER FOR DIVISION CYCLES
032112  010446                        MOV     R4,-(SP)        ;HIGH ORDER
032114  010546                        MOV     R5,-(SP)        ;LOW ORDER DIVISOR TO THE STACK
032116  005466  000002                NEG     2(SP)           ;FORM NEGATIVE
032122  005416                        NEG     @SP             ;VERSION OF THE DIVISOR
032124  005666  000002                SBC     2(SP)
032130  061601                        ADD     @SP,R1
032132  005500                        ADC     R0              ;PERFORM THE INITIAL SUBTRACTION
032134  066600  000002                ADD     2(SP),R0
032140  103445                        BCS     M.DP50          ;IF CARRY THEN OVERFLOW HAS OCCURRED
032142  005046                        CLR     -(SP)           ;THIS IS A LONGER LASTING CARRY BIT
032144  006103                M.DP40: ROL     R3
032146  006102                        ROL     R2
032150  006101                        ROL     R1
032152  006100                        ROL     R0
032154  005716                        TST     @SP             ;TEST "CARRY" INDICATOR
032156  001410                        BEQ     M.DP41          ;IF NO "CARRY" THEN ADD ELSE SUBTRACT
032160  005016                        CLR     @SP             ;CLEAR UP FOR NEXT TIME
032162  066601  000002                ADD     2(SP),R1
032166  005500                        ADC     R0              ;ADD -(DIVISOR)
032170  005516                        ADC     @SP     ;  I    ;SET "CARRY"
032172  066600  000004                ADD     4(SP),R0;<-
032176  000404                        BR      M.DP42
032200  060501                M.DP41: ADD     R5,R1
032202  005500                        ADC     R0              ;ADD +(DIVISOR)
032204  005516                        ADC     @SP     ;  I    ;SET "CARRY"
032206  060400                        ADD     R4,R0   ;<-
032210  005516                M.DP42: ADC     @SP             ;SET "CARRY"
032212  005716                        TST     @SP             ;TEST THE UPDATE INDICATOR
032214  001401                        BEQ     .+4     ;->     ;IF ZERO FORGET IT
032216  005203                        INC     R3      ;  I    ;NO CARRY POSSIBLE HERE
032220  005366  000006                DEC     6(SP)   ;<-     ;DECREMENT COUNTER
032224  003347                        BGT     M.DP40          ;BRANCH IF MORE TO DO
032226  006003                        ROR     R3
032230  103404                        BCS     M.DP44
032232  060501                        ADD     R5,R1
032234  005500                        ADC     R0
032236  060400                        ADD     R4,R0
032240  000241                        CLC
032242  006103                M.DP44: ROL     R3
032244  062706  000010                ADD     #10,SP          ;ADJUST STACK BY 4 WORDS
032250  000242                        CLV
032252  000207                        RTS     PC
032254  062706  000006        M.DP50: ADD     #6,SP
032260  000262                        SEV
032262  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
                                
032264  010046                  SUPRSL: MOV     R0,-(SP)        ;SAVE R0
032266  016600  000004                  MOV     4(SP),R0        ;GET POINTER TO MESSAGE
032272  005037  032354                  CLR     SUPR2
032276  000405                          BR      SUPR1
                                
032300  010046                  SUPRS:  MOV     R0,-(SP)        ;SAVE R0
032302  016600  000004                  MOV     4(SP),R0        ;GET POINTER TO MESSAGE
032306  010037  032354                  MOV     R0,SUPR2        ;GET POINTER FOR TYPING
032312                          SUPR1:
032312  105710                  1$:     TSTB    (R0)            ;TEST FOR TERMINATOR
032314  001406                          BEQ     2$              ;YES
032316  122710  000060                  CMPB    #'0,(R0)        ;IS THIS A '0' ?
032322  001006                          BNE     3$              ;NO
032324  112720  000040                  MOVB    #40,(R0)+       ;REPLACE IT WITH A 'BLANK'
032330  000770                          BR      1$              ;NEXT CHAR.
032332  005300                  2$:     DEC     R0              ;BACKUP 1
032334  112710  000060                  MOVB    #'0,(R0)        ;MAKE IT '0'
032340  005737  032354          3$:     TST     SUPR2           ;LEFT JUSTIFY ?
032344  001002                          BNE     4$              ;NO
032346  010037  032354                  MOV     R0,SUPR2        ;YES
032352  104401                  4$:     TYPE
032354  000000                  SUPR2:  .WORD   0
032356  012600                          MOV     (SP)+,R0        ;RESTORE R0
032360  012616                          MOV     (SP)+,(SP)      ;RESTORE STACK
032362  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
                                        ;
032364  010046                  $SUPRL: MOV     R0,-(SP)        ;SAVE R0
032366  016600  000004                  MOV     4(SP),R0        ;GET POINTER TO MESSAGE
032372  005037  032454                  CLR     $SUPR2
032376  000405                          BR      $SUPR1
                                        ;
032400  010046                  $SUPRS: MOV     R0,-(SP)        ;SAVE R0
032402  016600  000004                  MOV     4(SP),R0        ;GET POINTER TO MESSAGE
032406  010037  032454                  MOV     R0,$SUPR2       ;GET POINTER FOR TYPING
032412                          $SUPR1:
032412  105710                  1$:     TSTB    (R0)            ;TEST FOR TERMINATOR
032414  001406                          BEQ     2$              ;YES
032416  122710  000060                  CMPB    #'0,(R0)        ;IS THIS A '0' ?
032422  001006                          BNE     3$              ;NO
032424  112720  000040                  MOVB    #40,(R0)+       ;REPLACE IT WITH A 'BLANK'
032430  000770                          BR      1$              ;NEXT CHAR.
032432  005300                  2$:     DEC     R0              ;BACKUP 1
032434  112710  000060                  MOVB    #'0,(R0)        ;MAKE IT '0'
032440  005737  032454          3$:     TST     $SUPR2          ;LEFT JUSTIFY ?
032444  001002                          BNE     4$              ;NO
032446  010037  032454                  MOV     R0,$SUPR2       ;YES
032452  104414                  4$:     DISPLY                  ;TYPE, UNLESS SW13=1
032454  000000                  $SUPR2: .WORD   0
032456  012600                          MOV     (SP)+,R0        ;RESTORE R0
032460  012616                          MOV     (SP)+,(SP)      ;RESTORE STACK
032462  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

032464  005237  032572          FILLZ:  INC     FILLO           ;LEAVE ZERO'S

032470  010046                  REPLZ:  MOV     R0,-(SP)        ;SAVE R0
032472  016600  000004                  MOV     4(SP),R0        ;ADDRESS OF NUMBER TO R0
032476  005737  032572                  TST     FILLO           ;LEAVE PRECEDING ZEROS ?
032502  001014                          BNE     3$              ;BR IF YES
032504  122710  000060          1$:     CMPB    #'0,(R0)        ;BYTE EQUAL TO ASCII '0' ?
032510  001004                          BNE     2$              ;BR IF NOT
032512  112710  000040                  MOVB    #40,(R0)        ;REPLACE THE ZERO WITH A SPACE
032516  005200                          INC     R0              ;INCREMENT THE BYTE ADDRESS
032520  000771                          BR      1$              ;GO BACK AND LOOK FOR MORE LEADING ZEROS
032522  105710          2$:             TSTB    (R0)            ;SEE IF ZERO BYTE TERMINATOR
032524  001003                          BNE     3$              ;BR IF NOT
032526  005300                          DEC     R0              ;BACKUP STRING POINTER
032530  112710  000060                  MOVB    #'0,(R0)        ;PUT A ZERO BACK IN
032534  016600  000004          3$:     MOV     4(SP),R0        ;PUT ADDRESS OF FIRST CHARACTER ON STACK
032540  105720                  4$:     TSTB    (R0)+           ;SEE IF ZERO BYTE TERMINATOR
032542  001376                          BNE     4$              ;BR IF NOT
032544  005300                          DEC     R0              ;BACKUP STRING POINTER
032546  162500                          SUB     (R5)+,R0        ;ADJUST ADDRESS
032550  010037  032556                  MOV     R0,5$           ;GET ADDRESS FOR TYPEOUT
032554  104401                          TYPE                    ;TYPE THE NUMBER
032556  000000                  5$:     .WORD   0               ;ADDRESS OF NUMBER
032560  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
032562  012616                          MOV     (SP)+,(SP)      ;RESTORE STACK
032564  005037  032572                  CLR     FILLO           ;RESET FILL FLAG
032570  000205                          RTS     R5              ;RETURN

032572  000000          FILLO:  .WORD   0               ;IF SET, LEAVE PRECEDING ZEROS FOR TYPE

                                        ;ROUTINE TO TYPE AT PRIORITY 4

032574  013746  177776          TYPRI4: MOV     @#PS,-(SP)      ;SAVE THE PRESENT STATUS
032600  012737  000200  177776          MOV     #200,@#PS       ;CHANGE THE PRIORITY TO 4
032606  012537  032616                  MOV     (R5)+,1$        ;MESSAGE ADDRESS
032612  004737  035550                  JSR     PC,$TYPE        ;TYPE THE MESSAGE
032616  000000          1$:             .WORD   0               ;MESSAGE ADDRESS GOES HERE
032620  000205                          RTS     R5              ;RETURN

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

```
                                        ;R0 = DPB ADDRESS BEFORE CALLING THE ROUTINE
                                        ;
032622  104401  077332          TYDRV:  TYPE    ,DRVSN          ;TYPE 'DRV S/N:'
032626  010037  032642          TYPDRV: MOV     R0,1$           ;ADDRESS OF DPB
032632  062737  002130  032642        ADD     #$DRVSN,1$      ;INDEX TO DRIVE (MBA) SERIAL NUMBER
032640  104401                        TYPE                    ;TYPE THE DRIVE (MBA) SERIAL NUMBER
032642  000000                1$:     .WORD   0               ;ADDRESS OF DRIVE (MBA) SERIAL NUMBER FIELD
032644  104401  076141                TYPE    ,PERIOD         ;TYPE '.'
032650  000207                        RTS     PC              ;RETURN
                                        ;
                                        ;ROUTINE TO TYPE THE HDA SERIAL NUMBER IN DECIMAL
                                        ;CALL:
                                        ;       MOV     #DPB,R0         ;ADDRESS OF DRIVE PARAMETER BLOCK
                                        ;       JSR     PC,TYHDA        ;CALL ROUTINE
                                        ;   OR
                                        ;       MOV     #DPB,R0         ;ADDRESS OF DRIVE PARAMETER BLOCK
                                        ;       JSR     PC,TYPHDA       ;CALL ROUTINE(WITH NO HEADER MESSAGE)
                                        ;
                                        ;R0 = DPB ADDRESS BEFORE CALLING THE ROUTINE
                                        ;
032652  104401  077320          TYHDA:  TYPE    ,HDASN          ;TYPE 'HDA S/N:'
032656  005760  000144          TYPHDA: TST     $HSNM(R0)       ;IS SERIAL NUMBER VALID ?
032662  100003                        BPL     1$              ;YES TYPE IT, ELSE
032664  104401  076334                TYPE    ,NONE
032670  000207                        RTS     PC              ;RETURN
032672  010046                1$:     MOV     R0,-(SP)        ;DPB ADDRESS
032674  062716  000142                ADD     #$HSNL,(SP)     ;ADDRESS OF LOW NUMBER
032700  004737  037222                JSR     PC,$DB2D        ;CONVERT TO DOUBLE DECIMAL NUMBER
032704  004737  032264                JSR     PC,SUPRSL       ;AND TYPE IT LEFT JUSTIFIED
032710  104401  076141                TYPE    ,PERIOD         ;TYPE '.'
032714  000207                        RTS     PC              ;RETURN
                                        ;
                                        ;ROUTINE TO TYPE ERRORS
                                        ;CALL:
                                        ;       DISPLY                  ;MUST DEFINED IN 'TRAP' TABLE
                                        ;       MESADR                  ;ADDRESS OF MESSAGE
                                        ;       RETURN
                                        ;
032716  032777  020000  146230  $DSPLY: BIT     #BIT13,@SWR     ;INHIBIT ERROR TYPEOUT ?
032724  001004                        BNE     1$              ;BR IF YES
032726  005037  177776                CLR     @#PS            ;SET PRIORITY TO ZERO
032732  000137  035550                JMP     $TYPE           ;TYPE THE MESSAGE
032736  062716  000002          1$:     ADD     #2,(SP)         ;INCREMENT THE RETURN
032742  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
                                        ;
032744  121127  000060          CK.OCT: CMPB    (R1),#'0        ;LESS THAN ZERO?
032750  103407                        BLO     1$              ;YES -- BRANCH
032752  121127  000067                CMPB    (R1),#'7        ;GREATER THAN SEVEN?
```

```
                                        BHI     1$              ;YES -- BRANCH
                                        MOVB    (R1),R2         ;GET THE CHARACTER
                                        BIC     #^C7,R2         ;STRIP AWAY THE ASCII
                                        TST     (R5)+           ;ADJUST FOR RETURN
                                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

                                CK.DEC: CMPB    (R1),#'0        ;LESS THAN ZERO?
                                        BLO     1$              ;YES -- BRANCH
                                        CMPB    (R1),#'9        ;GREATER THAN NINE?
                                        BHI     1$              ;YES -- BRANCH
                                        MOVB    (R1),R2         ;GET THE CHARACTER
                                        BIC     #'0,R2          ;STRIP AWAY THE ASCII
                                        TST     (R5)+           ;ADJUST FOR RETURN
                                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+1
                                ;       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+1

                                CK.CHR: TSTB    (R1)            ;"CARRIAGE RETURN"?
                                        BEQ     3$              ;YES -- BRANCH
                                        CMPB    (R1),#',        ;"COMMA"?
                                        BEQ     2$              ;YES -- BRANCH
                                        CMPB    (R1),#'.        ;"PERIOD"?
                                        BEQ     1$              ;YES -- BRANCH
                                        JSR     R5,CK.DEC       ;"DIGIT"?
                                        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

033072  010446                  CK.DIG: MOV     R4,-(SP)        ;SAVE R4
033074  010346                          MOV     R3,-(SP)        ;SAVE R3
033076  010246                          MOV     R2,-(SP)        ;SAVE THE MAX. SIZE ON THE STACK
033100  005002                          CLR     R2              ;START WITH 0
033102  005003                          CLR     R3
033104  005004                          CLR     R4
033106  004537  033020                  JSR     R5,CK.CHR       ;CHECK ONE CHARACTER
033112  033206                          6$                      ;ILLEGAL CHARACTER
033114  033214                          9$                      ;CARRIAGE RETURN
033116  033206                          6$                      ;","
033120  033210                          7$                      ;"."
033122  033126                          1$                      ;DIGIT 0-7
033124  033126                          1$                      ;DIGIT 8-9
033126  062705  000004          1$:     ADD     #4,R5           ;STEP RETURN POINTER PAST "CR" & "PERIOD" RETURNS
033132  006303                  2$:     ASL     R3              ;INPUT NUMBER *2
033134  010346                          MOV     R3,-(SP)        ;SAVE *2
033136  006303                          ASL     R3              ;*4
033140  006303                          ASL     R3              ;*8
033142  062603                          ADD     (SP)+,R3        ;(*2)+(*8) = *10
033144  060203                          ADD     R2,R3           ;UPDATE THE INPUT NUMBER
033146  004537  033020                  JSR     R5,CK.CHR       ;CHECK ONE CHARACTER
033152  033212                          8$                      ;ILLEGAL CHARACTER
033154  033176                          5$                      ;CARRIAGE RETURN
03315   033174                          4$                      ;","
03316   033166                          3$                      ;"."
033162  033132                          2$                      ;DIGIT 0-7
033164  033132                          2$                      ;DIGIT 8-9
033166  105711                  3$:     TSTB    (R1)            ;DOES A "CR" FOLLOW THE "PERIOD"
033170  001010                          BNE     8$              ;BR IF NOT
033172  005724                          TST     (R4)+           ;INCREMENT THE RETURN
033174  005724                  4$:     TST     (R4)+           ;INCREMENT THE RETURN
033176  005724                  5$:     TST     (R4)+           ;INCREMENT THE RETURN
033200  020316                          CMP     R3,(SP)         ;CHECK THE MAGNITUDE OF THE NUMBER
033202  101004                          BHI     9$              ;BR IF ENTERED NUMBER TOO LARGE
033204  000402                          BR      8$              ;BYPASS INCREMENT
033206  005725                  6$:     TST     (R5)+           ;INCREMENT RETURN PAST INVALID RETURN
033210  005725                  7$:     TST     (R5)+           ;INCREMENT RETURN
033212  060405                  8$:     ADD     R4,R5           ;SETUP RETURN POINTER
033214  010302                  9$:     MOV     R3,R2           ;ENTERED VALUE
033216  005726                          TST     (SP)+           ;CLEAN MAX. SIZE OFF OF STACK
033220  012603                          MOV     (SP)+,R3        ;RESTORE R3
033222  012604                          MOV     (SP)+,R4        ;RESTORE R4
033224  011505                          MOV     (R5),R5         ;GET RETURN ADDRESS
033226  000205                          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

033230  016637  000002  033254  $SB2D:  MOV     2(SP),1$        ;SAVE THE BINARY NUMBER
033236  012746  033254          MOV     #1$,-(SP)       ;SET THE POINTER
033242  004737  037222          JSR     PC,$DB2D        ;CALL THE DOUBLE LENGTH CONVERT
033246  012666  000002          MOV     (SP)+,2(SP)     ;PICKUP THE POINTER
033252  000207                  RTS     PC              ;RETURN
033254  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

033260  016637  000002  033304  $SB2O:  MOV     2(SP),1$        ;SAVE THE BINARY NUMBER
033266  012746  033304          MOV     #1$,-(SP)       ;SET THE POINTER
033272  004737  037416          JSR     PC,$DB2O        ;CALL THE DOUBLE LENGTH CONVERT
033276  012666  000002          MOV     (SP)+,2(SP)     ;PICKUP THE POINTER
033302  000207                  RTS     PC              ;RETURN
033304  000000  000000  1$:     .WORD   0,0
```

```
     1                                  .SBTTL   TY INPUT ROUTINE

                                        ;;*****************************************************************
                                        .ENABL  LSB
033310  000000                          $TKCNT: .WORD   0               ;;NUMBER OF ITEMS IN QUEUE
033312  000000                          $TKQIN: .WORD   0               ;;INPUT POINTER
033314  000000                          $TKQOUT: .WORD  0               ;;OUTPUT POINTER
033316                                  $TKQSRT: .BLKB  7               ;;TTY KEYBOARD QUEUE
        033325                          $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
                                        ;
033326  005037  033310                  $TKINT: CLR     $TKCNT          ;;CLEAR COUNT OF ITEMS IN QUEUE
033332  012737  033316  033312                  MOV     #$TKQSRT,$TKQIN ;;MOVE THE STARTING ADDRESS OF THE
033340  013737  033312  033314                  MOV     $TKQIN,$TKQOUT  ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
033346  012737  033376  000060                  MOV     #$TKSRV,@#TKVEC ;;INITIALIZE THE KEYBOARD VECTOR
033354  012737  000200  000062                  MOV     #200,@#TKVEC+2  ;;'BR' LEVEL 4
033362  005777  145574                          TST     @$TKB           ;;CLEAR DONE FLAG
033366  012777  000100  145564                  MOV     #100,@$TKS      ;;ENABLE TTY KEYBOARD INTERRUPT
033374  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)
                                        ;
033376  117746  145560                  $TKSRV: MOVB    @$TKB,-(SP)     ;;PICKUP THE CHARACTER
033402  042716  177600                          BIC     #^C177,(SP)     ;;STRIP THE JUNK
033406  021627  000021                          CMP     (SP),#$XON      ;;IS IT A RANDOM XON?
033412  001002                                  BNE     30$             ;;BRANCH IF NO
033414  005726                                  TST     (SP)+           ;;CLEAN RANDOM XON OFF STACK
033416  000002                                  RTI                     ;;RETURN
033420                                  30$:
033420  021627  000003                          CMP     (SP),#3         ;;IS IT A CONTROL C?
033424  001007                                  BNE     1$              ;;BRANCH IF NO
033426  104401  034533                          TYPE    ,$CNTLC         ;;TYPE A CONTROL-C (^C)
033432  004737  033326                          JSR     PC,$TKINT       ;;INIT THE KEYBOARD
033436  005726                                  TST     (SP)+           ;;CLEAN UP STACK
033440  000137  034574                          JMP     CTRAP           ;;CONTROL C RESTART
033444  021627  000007                  1$:     CMP     (SP),#7         ;;IS IT A CONTROL G?
033450  001004                                  BNE     2$              ;;BRANCH IF NO
033452  022737  000176  001154                  CMP     #SWREG,SWR      ;;IS SOFT-SWR SELECTED?
033460  001500                                  BEQ     6$              ;;GO TO SWR CHANGE

033462                                  2$:
033462  022737  000007  033310                  CMP     #7,$TKCNT       ;;IS THE QUEUE FULL?
033470  001004                                  BNE     3$              ;;BRANCH IF NO
033472  104401  001176                          TYPE    ,$BELL          ;;RING THE TTY BELL
```

```
033476  005726                          TST     (SP)+           ;;CLEAN CHARACTER OFF OF STACK
033500  000451                          BR      5$              ;;EXIT
033502  021627  000023          3$:     CMP     (SP),#23        ;;IS IT A CONTROL-S?
033506  001021                          BNE     32$             ;;BRANCH IF NO
033510  005077  145444                  CLR     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
033514  005726                          TST     (SP)+           ;;CLEAN CHAR OFF STACK
033516  105777  145436          31$:    TSTB    @$TKS           ;;WAIT FOR A CHAR
033522  100375                          BPL     31$             ;;LOOP UNTIL ITS THERE
033524  117746  145432                  MOVB    @$TKB,-(SP)     ;;GET THE CHARACTER
033530  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
033534  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
033540  001366                          BNE     31$             ;;BRANCH IF NO
033542  012777  000100  145410          MOV     #100,@$TKS      ;;REENABLE TTY KEYBOARD INTERRUPTS
033550  000002                          RTI                     ;;RETURN
033552  005237  033310          32$:    INC     $TKCNT          ;;COUNT THIS CHARACTER
033556  021627  000140                  CMP     (SP),#140       ;;IS IT UPPER CASE?
033562  002405                          BLT     4$              ;;BRANCH IF YES
033564  021627  000175                  CMP     (SP),#175       ;;IS IT A SPECIAL CHAR?
033570  003002                          BGT     4$              ;;BRANCH IF YES
033572  042716  000040                  BIC     #40,(SP)        ;;MAKE IT UPPER CASE
033576  112677  177510          4$:     MOVB    (SP)+,@$TKQIN   ;;AND PUT IT IN QUEUE
033602  005237  033312                  INC     $TKQIN          ;;UPDATE THE POINTER
033606  023727  033312  033325          CMP     $TKQIN,#$TKQEND ;;GO OFF THE END?
033614  001003                          BNE     5$              ;;BRANCH IF NO
033616  012737  033316  033312          MOV     #$TKQSRT,$TKQIN ;;RESET THE POINTER
033624  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.
033626  022737  000176  001154  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED
033634  001124                          BNE     15$             ;;EXIT IF NOT
033636  105777  145316                  TSTB    @$TKS           ;;IS A CHAR WAITING?
033642  100121                          BPL     15$             ;;IF NOT, EXIT
033644  117746  145312                  MOVB    @$TKB,-(SP)     ;;YES
033650  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT  7-BIT ASCII
033654  021627  000007                  CMP     (SP),#7         ;;IS IT A CONTROL-G?
033660  001300                          BNE     2$              ;;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.
033662  123727  001150  000001  6$:     CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
033670  001674                          BEQ     2$              ;;BRANCH IF YES
033672  005726                          TST     (SP)+           ;;CLEAR CONTROL-G OFF STACK
033674  004737  033326                  JSR     PC,$TKINT       ;;FLUSH THE TTY INPUT QUEUE
033700  005077  145254                  CLR     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
033704  112737  000001  001151          MOVB    #1,$INTAG       ;;SET INTERRUPT MODE INDICATOR

033712  104401  034545                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
033716  104401  034552  $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
033722  013746  000176                  MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
033726  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
```

```
033730  104401  034563          TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
033734  005046          19$:    CLR     -(SP)           ;;CLEAR COUNTER
033736  005046                  CLR     -(SP)           ;;THE NEW SWR
033740  105777  145214    7$:   TSTB    @$TKS           ;;CHAR THERE?
033744  100375                  BPL     7$              ;;IF NOT TRY AGAIN

033746  117746  145210          MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
033752  042716  177600          BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII

033756  021627  000003          CMP     (SP),#3         ;;IS IT A CONTROL-C?
033762  001015                  BNE     9$              ;;BRANCH IF NOT
033764  104401  034533          TYPE    ,$CNTLC         ;;YES, ECHO CONTROL-C (^C)
033770  062706  000006          ADD     #6,SP           ;;CLEAN UP STACK
033774  123727  001151  C00001  CMPB    $INTAG,#1       ;;REENABLE TTY KEYBCARD INTERRUPTS?
034002  001003                  BNE     8$              ;;BRANCH IF NO
034004  012777  000100  145146  MOV     #100,@$TKS      ;;ALLOW TTY KEYBOARD INTERRUPTS
034012  000137  034574    8$:   JMP     CTRAP           ;;CONTROL-C RESTART


034016  021627  000025    9$:   CMP     (SP),#25        ;;IS IT A CONTROL-U?
034022  001005                  BNE     10$             ;;BRANCH IF NOT
034024  104401  034540          TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
034030  062706  000006    20$:  ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
034034  000737                  BR      19$             ;;LET'S TRY IT AGAIN


034036  021627  000015    10$:  CMP     (SP),#15        ;;IS IT A <CR>?
034042  001022                  BNE     16$             ;;BRANCH IF NO
034044  005766  000004          TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
034050  001403                  BEQ     11$             ;;BRANCH IF YES
034052  016677  000002  145074  MOV     2(SP),@SWR      ;;SAVE NEW SWR
034060  062706  000006    11$:  ADD     #6,SP           ;;CLEAR UP STACK
034064  104401  001203    14$:  TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
034070  123727  001151  000001  CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
034076  001003                  BNE     15$             ;;BRANCH IF NOT
034100  012777  000100  145052  MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
034106  000002          15$:    RTI                     ;;RETURN
034110  004737  035762    16$:  JSR     PC,$TYPEC       ;;ECHO CHAR
034114  021627  000060          CMP     (SP),#60        ;;CHAR < 0?
034120  002420                  BLT     18$             ;;BRANCH IF YES
034122  021627  000067          CMP     (SP),#67        ;;CHAR > 7?
034126  003015                  BGT     18$             ;;BRANCH IF YES
034130  042726  000060          BIC     #60,(SP)+       ;;STRIP-OFF ASCII
034134  005766  000002          TST     2(SP)           ;;IS THIS THE FIRST CHAR
034140  001403                  BEQ     17$             ;;BRANCH IF YES
034142  006316                  ASL     (SP)            ;;NO, SHIFT PRESENT
034144  006316                  ASL     (SP)            ;;   CHAR OVER TO MAKE
034146  006316                  ASL     (SP)            ;;   ROOM FOR NEW ONE.
034150  005266  000002    17$:  INC     2(SP)           ;;KEEP COUNT OF CHAR
034154  056616  177776          BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
034160  000667                  BR      7$              ;;GET THE NEXT ONE
034162  104401  001202    18$:  TYPE    ,$QUES          ;;TYPE ?<CR><LF>
034166  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
                                ;

034170  011646                  $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC AND
034172  016666  000004  000002          MOV     4(SP),2(SP)     ;;THE PS
034200  005066  000004                  CLR     4(SP)           ;;GET READY FOR A CHARACTER
034204  005046                          CLR     -(SP)           ;;PUT NEW PS ON STACK
034206  012746  034214                  MOV     #64$,-(SP)      ;;PUT NEW PC ON  STACK
034212  000002                          RTI                     ;;POP NEW PC AND PS
034214                          64$:
034214  005737  033310          1$:     TST     $TKCNT          ;;WAIT ON A CHARACTER
034220  001775                          BEQ     1$
034222  005337  033310                  DEC     $TKCNT          ;;DECREMENT THE COUNTER
034226  117766  177062  000004          MOVB    @$TKQOUT,4(SP)  ;;GET ONE CHARACTER
034234  005237  033314                  INC     $TKQOUT         ;;UPDATE THE POINTER
034240  023727  033314  033325          CMP     $TKQOUT,#$TKQEND ;;DID IT GO OFF OF THE END?
034246  001003                          BNE     2$              ;;BRANCH IF NO
034250  012737  033316  033314          MOV     #$TKQSRT,$TKQOUT ;;RESET THE POINTER
034256  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

034260  010346                  $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
034262  005046                          CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
034264  012703  034514          1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
034270  022703  034533          2$:     CMP     #$TTYIN+15.,R3  ;;BUFFER FULL?
034274  [illegible]                     BLOS    4$              ;;BR IF YES
[illegible]                             RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
034300  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
034302  122713  000177          10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
034306  001022                          BNE     5$              ;;BR IF NO
034310  005716                          TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
034312  001007                          BNE     6$              ;;BR IF NO
034314  112737  000134  034512          MOVB    #'\,9$          ;;TYPE A BACK SLASH
034322  104401  034512                  TYPE    ,9$
034326  012716  177777                  MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
034332  005303                  6$:     DEC     R3              ;;BACKUP BY ONE
034334  020327  034514                  CMP     R3,#$TTYIN      ;;STACK EMPTY?
034340  103434                          BLO     4$              ;;BR IF YES
034342  111337  034512                  MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR.
034346  104401  034512                  TYPE    ,9$             ;;GO TYPE
034352  000746                          BR      2$              ;;GO READ ANOTHER CHAR.
034354  005716                  5$:     TST     (SP)            ;;RUBOUT KEY SET?
034356  001406                          BEQ     7$              ;;BR IF NO
034360  112737  000134  034512          MOVB    #'\,9$          ;;TYPE A BACK SLASH
034366  104401  034512                  TYPE    ,9$
034372  005016                          CLR     (SP)            ;;CLEAR THE RUBOUT KEY
034374  122713  000025          7$:     CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
034400  001003                          BNE     8$              ;;BR IF NO
```

TTY INPUT ROUTINE

```
034402  104401  034540                  TYPE    ,$CNTLU         ;;TYPE A CONTROL 'U'
034406  000726                          BR      1$              ;;GO START OVER
034410  122713  000022          8$:     CMPB    #22,(R3)        ;;IS CHARACTER A '^R'?
034414  001011                          BNE     3$              ;;BRANCH IF NO
034416  105013                          CLRB    (R3)            ;;CLEAR THE CHARACTER
034420  104401  001203                  TYPE    ,$CRLF          ;;TYPE A 'CR' & 'LF'
034424  104401  034514                  TYPE    ,$TTYIN         ;;TYPE THE INPUT STRING
034430  000717                          BR      2$              ;;GO PICKUP ANOTHER CHACTER
034432  104401  001202          4$:     TYPE    ,$QUES          ;;TYPE A '?'
034436  000712                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
034440  111337  034512          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
034444  104401  034512                  TYPE    ,9$
034450  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
034454  001305                          BNE     2$              ;;LOOP IF NOT RETURN
034456  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
034462  104401  001204                  TYPE    ,$LF            ;;TYPE A LINE FEED
034466  005726                          TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
034470  012603                          MOV     (SP)+,R3        ;;RESTORE R3
034472  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
034474  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
034502  012766  034514  000004          MOV     #$TTYIN,4(SP)
034510  000002                          RTI                     ;;RETURN
034512     000          9$:     .BYTE   0                       ;;STORAGE FOR ASCII CHAR. TO TYPE
034513     000                  .BYTE   0                       ;;TERMINATOR
034514                  $TTYIN: .BLKB   15.                     ;;RESERVE 15. BYTES FOR TTY INPUT
034533     136     103     015  $CNTLC: .ASCIZ  /^C/<15><12>    ;;CONTROL 'C'
034540     136     125     015  $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL 'U'
034545     136     107     015  $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL 'G'
034552     015     012     123  $MSWR:  .ASCIZ  <15><12>/SWR = /
034563     040     040     116  $MNEW:  .ASCIZ  /  NEW = /

                                ;THIS ROUTINE WILL PROCESS THE (^C) CHARACTER

034574  012737  000001  001340  CTRAP:  MOV     #1,CFLAG        ;SET THE 'CONTROL C' FLAG
034602  005237  033310                  INC     $TKCNT          ;COUNT THIS CHARACTER
034606  112777  000015  176476          MOVB    #15,@$TKQIN     ;PUT 'RETURN' CHARACTER IN QUEUE
034614  005237  033312                  INC     $TKQIN          ;UPDATE THE POINTER
034620  023727  033312  033325          CMP     $TKQIN,#$TKQEND ;GO OFF THE END ?
034626  001003                          BNE     1$              ;BR IF YES
034630  012737  033316  033312          MOV     #$TKQSRT,$TKQIN ;RESET THE POINTER
034636  000002                  1$:     RTI                     ;RETURN
```

```
     1                                  .SBTTL  ERROR HANDLER ROUTINE

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

034640  105037  035226          $ERROR: CLRB    IBSAVE          ;;CLEAR THE ITEM BYTE SAVE LOCATION
034644  104407                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
034646  010337  001322                  MOV     R3,ATTN         ;SAVE THE ATTENTION REGISTER CONTENTS
034652  010137  001220                  MOV     R1,DRIVE        ;DRIVE NUMBER
034656  032777  020000  144270          BIT     #SW13,@SWR      ;INHIBIT PRINTOUTS ?
034664  001004                          BNE     .+12            ;BR IF YES
034666  104401  001203                  TYPE    ,$CRLF          ;CR-LF
034672  104401  001203                  TYPE    ,$CRLF          ;CR-LF
034676  004737  024764                  JSR     PC,$TIME        ;TYPE THE TIME
034702  105237  001117          7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
034706  001775                          BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
034710  013777  001116  144240          MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
034716  032777  002000  144230          BIT     #BIT10,@SWR     ;;BELL ON ERROR?
034724  001402                          BEQ     1$              ;;NO - SKIP
034726  104401  001176                  TYPE    ,$BELL          ;;RING BELL
034732  005237  001126          1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
034736  011637  001132                  MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
034742  162737  000002  001132          SUB     #2,$ERRPC
034750  117737  144156  001130          MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
034756  032777  001000  144170          BIT     #BIT09,@SWR     ;;SEE IF LOOP ON ERROR IS SET
034764  001060                          BNE     1004$           ;;BRANCH AROUND ROUTINE IF SO
034766  122737  000177  001130          CMPB    #177,$ITEMB     ;;SEE IF THIS IS THE POWER FAIL CALL
034774  001454                          BEQ     1004$           ;;BRANCH AROUND ROUTINE IF IT IS
034776  105737  035226                  TSTB    IBSAVE          ;;SEE IF THIS IS THE 2ND ERROR CALL IN THIS ROUTINE
035002  001047                          BNE     1003$           ;;BRANCH IF SO
035004  022737  177777  035224          CMP     #-1,CPSAVE      ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
035012  001445                          BEQ     1004$           ;;BRANCH IF SO
035014  013746  000004                  MOV     ERRVEC,-(SP)    ;;SAVE CONTENTS OF ERROR VECTOR
035020  012737  035036  000004          MOV     #1000$,ERRVEC   ;;SETUP 'TRAP' RETURN ADDRESS
035026  013737  177766  035224          MOV     177766,CPSAVE   ;;MOVE CPU ERROR REGISTER TO CPSAVE FOR TEST
035034  000406                          BR      1001$
035036  012737  177777  035224  1000$:  MOV     #-1,CPSAVE      ;;SET CPU ERROR REGISTER TIMEOUT INDICATOR
035044  012716  035052                  MOV     #1001$,(SP)     ;;SETUP RETURN ADDRESS
035050  000002                          RTI
035052  012637  000004          1001$:  MOV     (SP)+,ERRVEC    ;;RESTORE CONTENTS OF ERROR VECTOR

035056  022737  177777  035224  1003$:  CMP     #-1,CPSAVE      ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
035064  001420                          BEQ     1004$           ;;BRANCH IF SO
035066  032737  000001  035224          BIT     #BIT00,CPSAVE   ;;SEE IF POWER MONITOR BIT IS SET IN CPU ERR REG
035074  001414                          BEQ     1004$           ;;BRANCH IF OK
035076  042737  000001  177766          BIC     #BIT00,177766   ;;CLEAR THE BIT FOUND SET
035104  113737  001130  035226          MOVB    $ITEMB,IBSAVE   ;;MAKE IBSAVE NON-ZERO FOR DUAL ERROR CALL
035112  112737  000177  001130          MOVB    #177,$ITEMB     ;;SET $ITEMB TO SPECIAL POWER FAIL POINTER
035120  000402                          BR      1004$           ;;BRANCH OVER IBSAVE CLEARING
```

ERROR HANDLER ROUTINE

```
035122  105037  035226          1003$:  CLRB    IBSAVE          ;;CLEAR IBSAVE SO 2ND TIME THROUGH EXITS
035126                          1004$:
035126  032777  020000  144020          BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
035134  001004                          BNE     20$             ;;SKIP TYPEOUTS
035136  004737  035230                  JSR     PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
035142  104401  001203                  TYPE    ,$CRLF
035146                          20$:
035146  122737  000001  001226          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
035154  001007                          BNE     2$              ;;NO,SKIP APT ERROR REPORT
035156  113737  001130  035170          MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
035164  004737  036574                  JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
035170     000                  21$:    .BYTE   0
035171     000                          .BYTE   0
035172  000777                  22$:    BR      22$             ;;APT ERROR LOOP
035174  105737  035226          2$:     TSTB    IBSAVE          ;;SEE IF IBSAVE IS LOADED
035200  001005                          BNE     3$              ;;BRANCH IF NOT - NO HALT ON PWR MON BIT ERROR
035202  005777  143746                  TST     @SWR            ;;HALT ON ERROR
035206  100002                          BPL     3$              ;;SKIP IF CONTINUE
035210  000000                          HALT                    ;;HALT ON ERROR!
035212  104407                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
035214                          3$:
035214  105737  035226                  TSTB    IBSAVE          ;;SEE IF ITEM BYTE SAVE LOCATION HAS AN ERROR CALL
035220  001230                          BNE     7$              ;;BRANCH BACK TO CALL ORIGINAL ERROR
035222  000002                          RTI                     ;;RETURN
035224  000000          CPSAVE: .WORD   0               ;;LOCATION TO SAVE CPU ERROR REG CONTENTS
035226  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.

035230                          $ERRTYP:
035230  104401  001203                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
035234  010046                          MOV     R0,-(SP)        ;;SAVE R0
035236  005000                          CLR     R0              ;;PICKUP THE ITEM INDEX
035240  153700  001130                  BISB    @#$ITEMB,R0
035244  001004                          BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
                                                                ;;TYPE THE PC OF THE ERROR
035246  013746  001132                  MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
                                                                ;;ERROR ADDRESS
035252  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
035254  000437                          BR      6$              ;;GET OUT
035256  122700  000177          1$:     CMPB    #177,R0         ;;SEE IF THIS ERROR CALL IS SPECIAL POWER FAIL CALL
035262  001006                          BNE     1000$           ;;BRANCH IF NOT
035264  013737  001212  035546          MOV     $TESTN,PFTSTN   ;;GET TEST NUMBER
035272  012700  035406                  MOV     #PFECH,R0       ;;MOVE POWER FAIL ERROR CALL TABLE TO R0
035276  000406                          BR      1001$           ;;BRANCH TO CALL ERROR
035300  005300                  1000$:  DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
035302  006300                          ASL     R0              ;;      WORK FOR THE ERROR TABLE
035304  006300                          ASL     R0
035306  006300                          ASL     R0
035310  062700  003370                  ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
035314  012037  035324          1001$:  MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
035320  001404                          BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
035322  104401                          TYPE                    ;;TYPE THE "ERROR MESSAGE"
035324  000000                  2$:     .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
035326  104401  001203                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
035332  012037  035342          3$:     MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
035336  001404                          BEQ     5$              ;;SKIP TYPEOUT IF 0
035340  104401                          TYPE                    ;;TYPE THE "DATA HEADER"
035342  000000                  4$:     .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
035344  104401  001203                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
035350  011000                  5$:     MOV     (R0),R0         ;;PICKUP "DATA TABLE" POINTER
035352  001004                          BNE     7$              ;;GO TYPE THE DATA
035354  012600                  6$:     MOV     (SP)+,R0        ;;RESTORE R0
035356  104401  001203                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
035362  000207                          RTS     PC              ;;RETURN
035364                          7$:
035364  013046                          MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
035366  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
035370  005710                          TST     (R0)            ;;IS THERE ANOTHER NUMBER?
035372  001770                          BEQ     6$              ;;BR IF NO
035374  104401  035402                  TYPE    ,8$             ;;TYPE TWO(2) SPACES
035400  000771                          BR      7$              ;;LOOP
035402     040     040     000  8$:     .ASCIZ  /  /            ;;TWO(2) SPACES
                                        .EVEN
035406  035416  035500  035532  PFECH:  PFECH1,PFECH2,PFECH3,PFECH4 ;;WORDS DEFINING TABLES BELOW
035416     120     117     127  PFECH1: .ASCIZ  ?POWER MONITOR BIT IN CPU ERROR REGISTER FOUND SET?
035500     124     105     123  PFECH2: .ASCIZ  ?TESTNO  ERR PC  CPUERREG?
                                        .EVEN
035532  035546  001132  035224  PFECH3: .WORD   PFTSTN,$ERRPC,CPSAVE,0
```

ERROR MESSAGE TYPEOUT ROUTINE

```
035542     000     000     000  PFECH4: .BYTE   0,0,0,0
035546  000000                  PFTSTN: .WORD   0          ;;CONTAINS TEST NUMBER FOR PF BIT ERROR
```

TYPE ROUTINE

```
                                .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
                                ;*

035550  105737  001173          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
035554  100002                          BPL     1$              ;;BR IF YES
035556  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
035560  000430                          BR      3$              ;;LEAVE
035562  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
035564  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
035570  122737  000001  001226          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
035576  001011                          BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
035600  132737  000100  001227          BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
035606  001405                          BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
035610  010037  035620                  MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
035614  004737  036564                  JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
035620  000000                  61$:    .WORD   0               ;;MESSAGE ADDRESS
035622  132737  000040  001227  62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
035630  001003                          BNE     60$             ;;YES,SKIP TYPE OUT
035632  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
035634  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
035636  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
035640  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
035642  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
035646  000002                          RTI                     ;;RETURN
035650  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
035654  001430                          BEQ     8$
035656  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
035662  001006                          BNE     5$
035664  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
035666  104401                          TYPE                    ;;TYPE A CR AND LF
035670  001203                          $CRLF
035672  105037  036100                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
035676  000755                          BR      2$              ;;GET NEXT CHARACTER
035700  004737  035762          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
035704  123726  001172          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
035710  001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
035712  013746  001170                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                ;;AND THE NULL CHAR.
035716  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
035722  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
035724  004737  035762                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
035730  105337  036100                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
035734  000770                          BR      7$              ;;LOOP
```

```
                                        ;HORIZONTAL TAB PROCESSOR

035736  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
035742  004737  035762          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
035746  132737  000007  036100          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
035754  001372                          BNE     9$              ;;TAB STOP
035756  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
035760  000724                          BR      2$              ;;GET NEXT CHARACTER
035762                          $TYPEC:
035762  105777  143172                  TSTB    @$TKS           ;;CHAR IN KYBD BUFFER?
035766  100022                          BPL     10$             ;;BR IF NOT
035770  017746  143166                  MOV     @$TKB,-(SP)     ;;GET CHAR
035774  042716  177600                  BIC     #177600,(SP)    ;;STRIP EXTRANEOUS BITS
036000  122716  000023                  CMPB    #$XOFF,(SP)     ;;WAS CHAR XOFF
036004  001012                          BNE     102$            ;;BR IF NOT
036006                          101$:
036006  105777  143146                  TSTB    @$TKS           ;;WAIT FOR CHAR
036012  100375                          BPL     101$
036014  117716  143142                  MOVB    @$TKB,(SP)      ;;GET CHAR
036020  042716  177600                  BIC     #177600,(SP)    ;;STRIP IT
036024  122716  000021                  CMPB    #$XON,(SP)      ;;WAS IT XON?
036030  001366                          BNE     101$            ;;BR IF NOT
036032                          102$:
036032  005726                          TST     (SP)+           ;;FIX STACK
036034                          10$:
036034  105777  143124                  TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
036040  100375                          BPL     10$
036042  116677  000002  143116          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
036050  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
036056  001003                          BNE     1$              ;;BRANCH IF NO
036060  105037  036100                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
036064  000406                          BR      $TYPEX          ;;EXIT
036066  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
036074  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
036076  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
036100  000000          $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
036102  000207          $TYPEX: RTS     PC
```

```
                                .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE

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

036104  017646  000000         $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
036110  116637  000001 036327          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
036116  112637  036331                 MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
036122  062716  000002                 ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
036126  000406                         BR      $TYPON
036130  112737  000001 036327 $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
036136  112737  000006 036331         MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
036144  112737  000005 036326 $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
036152  010346                       MOV     R3,-(SP)        ;;SAVE R3
036154  010446                       MOV     R4,-(SP)        ;;SAVE R4
036156  010546                       MOV     R5,-(SP)        ;;SAVE R5
036160  113704  036331               MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
036164  005404                       NEG     R4
036166  062704  000006               ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
036172  110437  036330               MOVB    R4,$OMODE       ;;SAVE IT FOR USE
036176  113704  036327               MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
036202  016605  000012               MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
036206  005003                       CLR     R3              ;;CLEAR THE OUTPUT WORD
036210  006105                1$:    ROL     R5              ;;ROTATE MSB INTO "C"
036212  000404                       BR      3$              ;;GO DO MSB
036214  006105                2$:    ROL     R5              ;;FORM THIS DIGIT
036216  006105                       ROL     R5
036220  006105                       ROL     R5
036222  010503                       MOV     R5,R3
036224  006103                3$:    ROL     R3              ;;GET LSB OF THIS DIGIT
036226  105337  036330               DECB    $OMODE          ;;TYPE THIS DIGIT?
036232  100016                       BPL     7$              ;;BR IF NO
036234  042703  177770               BIC     #177770,R3      ;;GET RID OF JUNK
036240  001002                       BNE     4$              ;;TEST FOR 0
036242  005704                       TST     R4              ;;SUPPRESS THIS 0?
036244  001403                       BEQ     5$              ;;BR IF YES
036246  005204                4$:    INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
```

BINARY TO OCTAL (ASCII) AND TYPE

```
036250  052703  000060            BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
036254  052703  000040      5$:   BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
036260  110337  036324            MOVB    R3,8$           ;;SAVE FOR TYPING
036264  104401  036324            TYPE    ,8$             ;;GO TYPE THIS DIGIT
036270  105337  036326      7$:   DECB    $OCNT           ;;COUNT BY 1
036274  003347                    BGT     2$              ;;BR IF MORE TO DO
036276  002402                    BLT     6$              ;;BR IF DONE
036300  005204                    INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
036302  000744                    BR      2$              ;;GO DO THE LAST DIGIT
036304  012605      6$:           MOV     (SP)+,R5        ;;RESTORE R5
036306  012604                    MOV     (SP)+,R4        ;;RESTORE R4
036310  012603                    MOV     (SP)+,R3        ;;RESTORE R3
036312  016666  000002  000004    MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
036320  012616                    MOV     (SP)+,(SP)
036322  000002                    RTI                     ;;RETURN
036324     000      8$:           .BYTE   0               ;;STORAGE FOR ASCII DIGIT
036325     000                    .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
036326     000      $OCNT:        .BYTE   0               ;;OCTAL DIGIT COUNTER
036327     000      $OFILL:       .BYTE   0               ;;ZERO FILL SWITCH
036330  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

036332                          $TYPDS:
036332  010046                         MOV     R0,-(SP)        ;;PUSH R0 ON STACK
036334  010146                         MOV     R1,-(SP)        ;;PUSH R1 ON STACK
036336  010246                         MOV     R2,-(SP)        ;;PUSH R2 ON STACK
036340  010346                         MOV     R3,-(SP)        ;;PUSH R3 ON STACK
036342  010546                         MOV     R5,-(SP)        ;;PUSH R5 ON STACK
036344  012746  020200                 MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
036350  016605  000020                 MOV     20(SP),R5       ;;GET THE INPUT NUMBER
036354  ,00004                        BPL     1$              ;;BR IF INPUT IS POS.
036356  005405                         NEG     R5              ;;MAKE THE BINARY NUMBER POS.
036360  112766  000055  000001         MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
036366  005000                1$:      CLR     R0              ;;ZERO THE CONSTANTS INDEX
036370  012703  036546                 MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
036374  112723  000040                 MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
036400  005002                2$:      CLR     R2              ;;CLEAR THE BCD NUMBER
036402  016001  036536                 MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
036406  160105                3$:      SUB     R1,R5           ;;FORM THIS BCD DIGIT
036410  002402                         BLT     4$              ;;BR IF DONE
036412  005202                         INC     R2              ;;INCREASE THE BCD DIGIT BY 1
036414  000774                         BR      3$
036416  060105                4$:      ADD     R1,R5           ;;ADD BACK THE CONSTANT
036420  0C5702                         TST     R2              ;;CHECK IF BCD DIGIT=0
036422  001002                         BNE     5$              ;;FALL THROUGH IF 0
036424  105716                         TSTB    (SP)            ;;STILL DOING LEADING 0'S?
036426  100407                         BMI     7$              ;;BR IF YES
036430  106316                5$:      ASLB    (SP)            ;;MSD?
036432  103003                         BCC     6$              ;;BR IF NO
036434  116663  000001  177777         MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
036442  052702  000060        6$:      BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
036446  052702  000040        7$:      BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
036452  110223                         MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
036454  005720                         TST     (R0)+           ;;JUST INCREMENTING
036456  020027  000010                 CMP     R0,#10          ;;CHECK THE TABLE INDEX
036462  002746                         BLT     2$              ;;GO DO THE NEXT DIGIT
036464  003002                         BGT     8$              ;;GO TO EXIT
036466  010502                         MOV     R5,R2           ;;GET THE LSD
036470  000764                         BR      6$              ;;GO CHANGE TO ASCII
036472  105726                8$:      TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
036474  100003                         BPL     9$              ;;BR IF NO
036476  116663  177777  177776         MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
036504  105013                9$:      CLRB    (R3)            ;;SET THE TERMINATOR
036506  012605                         MOV     (SP)+,R5        ;;POP STACK INTO R5
036510  012603                         MOV     (SP)+,R3        ;;POP STACK INTO R3
036512  012602                         MOV     (SP)+,R2        ;;POP STACK INTO R2
036514  012601                         MOV     (SP)+,R1        ;;POP STACK INTO R1
```

```
036516  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
036520  104401  036546                  TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
036524  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
036532  012616                          MOV     (SP)+,(SP)
036534  000002                          RTI                     ;;RETURN TO USER
036536  023420                  $DTBL:  10000.
036540  001750                          1000.
036542  000144                          100.
036544  000012                          10.
036546                          $DBLK:  .BLKW   4
```

```
                                .SBTTL  APT COMMUNICATIONS ROUTINE

                                ;;**********************************************************************
036556  112737  000001  037022  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
036564  112737  000001  037020  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
036572  000403                  BR      $ATYC
036574  112737  000001  037022  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
036602                          $ATYC:
036602  010046                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
036604  010146                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
036606  105737  037020          TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
036612  001450                  BEQ     5$              ;;IF NOT:  BR
036614  122737  000001  001226  CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
036622  001031                  BNE     3$              ;;IF NOT:  BR
036624  132737  000100  001227  BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
036632  001425                  BEQ     3$              ;;IF NOT:  BR
036634  017600  000004          MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
036640  062766  000002  000004  ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
036646  005737  001206  1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
036652  001375                  BNE     1$              ;;IF NOT:  WAIT
036654  010037  001222          MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
036660  105720          2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
036662  001376                  BNE     2$
036664  163700  001222          SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
036670  006200                  ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
036672  010037  001224          MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
036676  012737  000004  001206  MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
036704  000413                  BR      5$
036706  017637  000004  036732  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
036714  062766  000002  000004  ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
036722  013746  177776          MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
036726  004737  035550          JSR     PC,$TYPE        ;;CALL TYPE MACRO
036732  000000          4$:     .WORD   0
036734                  5$:
036734  105737  037022  10$:     TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
036740  001416                  BEQ     12$             ;;IF NOT:  BR
036742  005737  001226          TST     $ENV            ;;RUNNING UNDER APT?
036746  001413                  BEQ     12$             ;;IF NOT:  BR
036750  005737  001206  11$:     TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
036754  001375                  BNE     11$             ;;IF NOT:  WAIT
036756  017637  000004  001210  MOV     @4(SP),$FATAL   ;;GET ERROR #
036764  062766  000002  000004  ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
036772  005237  001206          INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
036776  105037  037022  12$:     CLRB    $FFLG           ;;CLEAR FATAL FLAG
037002  105037  037021          CLRB    $LFLG           ;;CLEAR LOG FLAG
037006  105037  037020          CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
037012  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
037014  012600                  MOV     (SP)+,R0        ;;POP STACK INTO R0
037016  000207                  RTS     PC              ;;RETURN
037020     000  $MFLG:  .BYTE   0       ;;MESSG. FLAG
037021     000  $LFLG:  .BYTE   0       ;;LOG FLAG
037022     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

 037024                         $RAND:
 037024  010046                         MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 037026  010146                         MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 037030  010246                         MOV     R2,-(SP)        ;;PUSH R2 ON STACK
 037032  013700  037124                 MOV     $LONUM,R0       ;;SET R0 WITH LOW
 037036  013701  037122                 MOV     $HINUM,R1       ;;SET R1 WITH HIGH
 037042  012702  177771                 MOV     #-7,R2          ;;SET SHIFT COUNT
 037046  006300                 1$:     ASL     R0              ;;SHIFT R0 LEFT AND
 037050  006101                         ROL     R1              ;;ROTATE CARRY INTO R1 AND
 037052  005202                         INC     R2              ;;CHECK FOR DONE
 037054  001374                         BNE     1$              ;;CONTINUE SHIFT LOOP
 037056  063700  037124                 ADD     $LONUM,R0       ;;ADD NUMBER TO MAKE X 129
 037062  005501                         ADC     R1              ;;PROPOGATE CARRY
 037064  063701  037122                 ADD     $HINUM,R1       ;;ADD NUMBER TO MAKE X 129
 037070  062700  001057                 ADD     #1057,R0        ;;ADD LOW CONSTANT
 037074  005501                         ADC     R1              ;;PROPOGATE CARRY
 037076  062701  047401                 ADD     #47401,R1       ;;ADD HIGH CONSTANT
 037102  010037  037124                 MOV     R0,$LONUM       ;;SAVE R0
 037106  010137  037122                 MOV     R1,$HINUM       ;;SAVE R1
 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  000207                         RTS     PC              ;;RETURN
 037122  176543         $HINUM: .WORD   176543
 037124  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

037126                          $SAVREG:
037126  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
037130  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
037132  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
037134  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
037136  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
037140  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
037142  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
037146  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
037152  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
037156  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
037162  000002                          RTI

                                ;*RESTORE R0-R5
                                ;*CALL:
                                ;*      RESREG
037164                          $RESREG:
037164  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
037170  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
037174  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
037200  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
037204  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
037206  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
037210  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
037212  0126C2                          MOV     (SP)+,R2        ;;POP STACK INTO R2
037214  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
037216  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
037220  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


037222  104412                  $DB2D:  SAVREG                  ;;SAVE REGISTERS
037224  016602  000002                  MOV     2(SP),R2        ;;PICKUP THE DATA POINTER
037230  012700  037402                  MOV     #$DECVL,R0      ;;GET ADDRESS OF "$DECVL" STRING
037234  010066  000002                  MOV     R0,2(SP)        ;;PUT ADDRESS OF ASCIZ STRING ON STACK
037240  012201                          MOV     (R2)+,R1        ;;PICKUP THE BINARY NUMBER
037242  012202                          MOV     (R2)+,R2
037244  012737  000012  037320          MOV     #10.,4$         ;;SET UP TO DO 10 CONVERSIONS
037252  012704  037332                  MOV     #$TNPWR,R4      ;;ADDRESS OF TEN POWER
037256  012705  037334                  MOV     #$TNPWR+2,R5
037262  005003                  1$:     CLR     R3              ;;CLEAR PARTIAL
037264  161401                  2$:     SUB     (R4),R1         ;;SUBTRACT TEN POWER
037266  005602                          SBC     R2
037270  161502                          SUB     (R5),R2
037272  002402                          BLT     3$              ;;BR IF TEN POWER TO LARGE
037274  005203                          INC     R3              ;;ADD 1 TO PARTIAL
037276  000772                          BR      2$              ;;LOOP
037300  062401                  3$:     ADD     (R4)+,R1        ;;RESTORE SUBTRACTED VALUE
037302  005502                          ADC     R2
037304  062402                          ADD     (R4)+,R2
037306  022525                          CMP     (R5)+,(R5)+     ;;MOVE TO NEXT TEN POWER
037310  052703  000060                  BIS     #'0,R3          ;;CHANGE PARTIAL TO ASCII
037314  110320                          MOVB    R3,(R0)+        ;;SAVE IT
037316  005327                          DEC     (PC)+           ;;DONE?
037320  000000                  4$:     .WORD   0
037322  001357                          BNE     1$              ;;BR IF NO
037324  105020                          CLRB    (R0)+           ;;TERMINATOR
037326  104413                          RESREG                  ;;RESTORE REGISTERS
037330  000207                          RTS     PC              ;;RETURN
037332  145000                  $TNPWR: 145000                  ;;1.0E09
037334  035632                          35632
037336  160400                          160400                  ;;1.0E08
037340  002765                          2765
037342  113200                          113200                  ;;1.0E07
037344  000230                          230
037346  041100                          041100                  ;;1.0E06
037350  000017                          17
037352  103240                          103240                  ;;1.0E05
037354  000001                          1
037356  023420                          23420                   ;;1.0E04
037360  000000                          0
037362  001750                          1750                    ;;1.0E03
037364  000000                          0
037366  000144                          144                     ;;1.0E02
037370  000000                          0
```

```
037372  000012                          12                      ;;1.0E01
037374  000000                          0
037376  000001                          1                       ;;1.0E00
037400  000000                          0
037402                          $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


037416  104412          $DB20:  SAVREG                  ;;SAVE ALL REGISTERS
037420  016601  000002          MOV     2(SP),R1        ;;PICKUP THE POINTER TO LOW WORD
037424  012705  037535          MOV     #$OCTVL+13.,R5  ;;POINTER TO DATA TABLE
037430  012704  000014          MOV     #12.,R4         ;;DO ELEVEN CHARACTERS
037434  012703  177770          MOV     #^C7,R3         ;;MASK
037440  012100                  MOV     (R1)+,R0        ;;LOWER WORD
037442  012101                  MOV     (R1)+,R1        ;;HIGH WORD
037444  005002                  CLR     R2              ;;TERMINATOR
037446  110245          1$:     MOVB    R2,-(R5)        ;;PUT CHARACTER IN DATA TABLE
037450  010002                  MOV     R0,R2           ;;GET THIS DIGIT
037452  005304                  DEC     R4              ;;COUNT THIS CHARACTER
037454  003007                  BGT     3$              ;;BR IF NOT THE LAST DIGIT
037456  001405                  BEQ     2$              ;;BR IF IT IS THE LAST DIGIT
037460  005205                  INC     R5              ;;ALL DIGITS DONE-ADJUST POINTER FOR FIRST
037462  010566  000002          MOV     R5,2(SP)        ;;ASCIZ CHAR. & PUT IT ON THE STACK
037466  104413                  RESREG                  ;;RESTORE ALL REGISTERS
037470  000207                  RTS     PC              ;;RETURN TO USER
037472  006203          2$:     ASR     R3              ;;POSITION THE MASK FOR THE LAST DIGIT
037474  006001          3$:     ROR     R1              ;;POSITION THE BINARY NUMBER FOR
037476  006000                  ROR     R0              ;;      THE NEXT OCTAL DIGIT
037500  006001                  ROR     R1
037502  006000                  ROR     R0
037504  006001                  ROR     R1
037506  006000                  ROR     R0
037510  040302                  BIC     R3,R2           ;;MASK OUT ALL JUNK
037512  062702  000060          ADD     #'0,R2          ;;MAKE THIS CHAR. ASCII
037516  000753                  BR      1$              ;;GO PUT IT IN THE DATA TABLE
037520                  $OCTVL: .BLKB   14.             ;;RESERVE DATA TABLE
```

```
                                .SBTTL  POWER DOWN AND UP ROUTINES

                                ;;*****************************************************************
                                ;POWER DOWN ROUTINE
037536  012737  037710  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
037544  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
037552  010046                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
037554  010146                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
037556  010246                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
037560  010346                  MOV     R3,-(SP)        ;;PUSH R3 ON STACK
037562  010446                  MOV     R4,-(SP)        ;;PUSH R4 ON STACK
037564  010546                  MOV     R5,-(SP)        ;;PUSH R5 ON STACK
037566  017746  141362          MOV     @$WR,-(SP)      ;;PUSH @$WR ON STACK
037572  010637  037714          MOV     SP,$SAVR6       ;;SAVE SP
037576  012737  037610  000024  MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
037604  000000                  HALT
037606  000776                  BR      .-2             ;;HANG UP

                                ;;*****************************************************************
                                ;POWER UP ROUTINE
037610  012737  037710  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
037616  013706  037714          MOV     $SAVR6,SP       ;;GET SP
037622  005037  037714          CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
037626  005237  037714  1$:     INC     $SAVR6          ;;WAIT FOR THE INC
037632  001375                  BNE     1$              ;;OF  WORD
037634  005337  037716  2$:     DEC     PWRFLG          ;WAIT AND SET POWER FAIL FLAG
037640  003775                  BLE     2$              ;WAIT FOR FLAG= 1
037642  012677  141306          MOV     (SP)+,@$WR      ;;POP STACK INTO @$WR
037646  012605                  MOV     (SP)+,R5        ;;POP STACK INTO R5
037650  012604                  MOV     (SP)+,R4        ;;POP STACK INTO R4
037652  012603                  MOV     (SP)+,R3        ;;POP STACK INTO R3
037654  012602                  MOV     (SP)+,R2        ;;POP STACK INTO R2
037656  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
037660  012600                  MOV     (SP)+,R0        ;;POP STACK INTO R0
037662  012737  037536  000024  MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
037670  012737  000340  000026  MOV     #340,@#PWRVEC+2 ;;PRIO:7
037676  104401                  TYPE                    ;;REPORT THE POWER FAILURE
037700  037720          $PWRMG: .WORD   $POWER          ;;POWER FAIL MESSAGE POINTER
037702  012716                  MOV     (PC)+,(SP)      ;;RESTART AT SATPOW
037704  037736          $PWRAD: .WORD   SATPOW          ;;RESTART ADDRESS
037706  000002                  RTI
037710  000000          $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
037712  000776                  BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
037714  000000          $SAVR6: 0                       ;;PUT THE SP HERE
037716  000000          PWRFLG: .WORD   0               ;INDICATES POWER FAIL OCCURRED WHEN SET
037720      200     042     120 $POWER: .ASCIZ  <CRLF>/'POWER UP'/<CRLF>
                                .EVEN


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

037736  005227  000000  SATPOW: INC     #0              ;TTY LOOP, WAIT FOR INCREMENT
037742  001375                  BNE     .-4             ;OF WORD
037744  000005                  RESET                   ;CLEAR THE WORLD
037746  005037  001350          CLR     SECOND          ;RESET SECOND COUNT
037752  005037  001466          CLR     INTRVL+2        ;RESET THE INTERVAL COUNT
037756  005037  177776          CLR     PS              ;CLEAR PSW
```

POWER DOWN AND UP ROUTINES

```
 15 037762  012706  001100                MOV     #STACK,SP       ;SETUP STACK POINTER
 16 037766  004737  033326                JSR     PC,$TKINT       ;MAKE SURE KEYBOARD INTERRUPT AND
 17 037772  004737  023422                JSR     PC,CKCLK        ;SYSTEM CLOCK ARE STILL ON.
 18 037776  005037  001340                CLR     CFLAG           ;CLEAR THE 'CONTROL C' FLAG
 19 040002  105737  001542                TSTB    ASNLST          ;ANY DRIVES ASSIGNED ?
 20 040006  001414                        BEQ     2$              ;BR IF NO
 21 040010  104401  077406                TYPE    ,MSWAIT         ;TYPE 'WAITING 2 MINUTES...TO START'
 22 040014  005737  001340          1$:   TST     CFLAG           ;CONTROL C INTERRUPT ?
 23 040020  001007                        BNE     2$              ;BR IF YES
 24 040022  023727  001466  000002        CMP     INTRVL+2,#2     ;TWO MINUTES YET ?
 25 040030  002771                        BLT     1$              ;WAIT IF NOT
 26 040032  012737  000400  001336        MOV     #400,CHGADR     ;FUDGE 200 START
 27 040040  012705  001520          2$:   MOV     #ORDERQ,R5      ;CLEAR UP THE QUE AND BUFFER
 28 040044  005025                  3$:   CLR     (R5)+
 29 040046  022705  002056                CMP     #BLKADR,R5      ;ALL DONE ?
 30 040052  001374                        BNE     3$              ;BRANCH IF NOT
 31 040054  000137  005024                JMP     SIZMEM          ;LOOP BACK
```

```
      1                                 .SBTTL  TRAP DECODER

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

 040060  010046                         $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
 040062  016600  000002                         MOV     2(SP),R0        ;;GET TRAP ADDRESS
 040066  005740                         .       TST     -(R0)           ;;BACKUP BY 2
 040070  111000                         .       MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
 040072  006300                         .       ASL     R0              ;;POSITION FOR INDEXING
 040074  016000  040114                 .       MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
 040100  000200                         .       RTS     R0              ;;GO TO ROUTINE


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

 040102  011646                         $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
 040104  016666  000004  000002         .       MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
 040112  000002                         .       RTI                     ;;RESTORE THE PSW

                                        .SBTTL  TRAP TABLE

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

                                        ;       ROUTINE
                                        ;       -------
 040114  040102                         $TRPAD: .WORD   $TRAP2
 040116  035550                         .       $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
 040120  036130                         .       $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
 040122  036104                         .       $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
 040124  036144                         .       $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
 040126  036332                         .       $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)

 040130  033716                         .       $GTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING

 040132  033626                         .       $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
 040134  034170                         .       $RDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
 040136  034260                         .       $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
 040140  037126                         .       $SAVREG ;;CALL=SAVREG   TRAP+12(104412) SAVE R0-R5 ROUTINE
 040142  037164                         .       $RESREG ;;CALL=RESREG   TRAP+13(104413) RESTORE R0-R5 ROUTINE
 040144  032716                       2 .       $DSPLY  ;;CALL=DISPLY   TRAP+14(104414) ROUTINE TO TYPE ERROR MESSAGES
         000032                       3 $TERM=.-$TRPAD
```

TRAP TABLE

```
 7
17
27
28                                      .SBTTL  SINGLE/DUAL PORT RH70/RM80 DRIVER (REV 0.17)
29
30                                      ;COPYRIGHT (C) 1979,1981
31                                      ;DIGITAL EQUIPMENT CORP.
32                                      ;MAYNARD, MA 01754
33                                      ;AUTHOR(S): CHUCK HESS
34                                      ;REVISED BY: MIKE LEAVITT
35
36                                      ;;*****************************************************************
37
38                                      ;STORAGE FOR RMDS, RMER1 AND RMER2 ON AN ERROR '2'
39                                              ;RMERRS   = RMDS
40                                              ;RMERRS+2 = RMER1
41                                              ;RMERRS+4 = RMER2
42
43 040146  000000  000000  000000   RMERRS: .WORD   0,0,0
44
45                                      ;TABLE OF DRIVE ACTIVE INDICATORS (DRVACT=8 BYTES)
46                                              ;DRVACT=0 IF DRIVE IS IDLE
47                                              ;DRVACT>0 IF DRIVE IS ACTIVE WITH A COMMAND
48                                              ;DRVACT<0 IF DRIVE IS ACTIVE WITH AN ERROR RECOVERY OPERATION
49
50 040154      000              DRVACT: .BYTE   0               ;DRIVE 0
53 040155      000                      .BYTE   0               ;DRIVE 1
   040156      000                      .BYTE   0               ;DRIVE 2
   040157      000                      .BYTE   0               ;DRIVE 3
   040160      000                      .BYTE   0               ;DRIVE 4
   040161      000                      .BYTE   0               ;DRIVE 5
   040162      000                      .BYTE   0               ;DRIVE 6
   040163      000                      .BYTE   0               ;DRIVE 7
54
55                                      ;TABLE OF DRIVE STATUS INDICATORS (DRVSTA=8 BYTES)
56                                              ;DRVSTA=0 IF DRIVE IS OFFLINE OR NONEXSITENT
57                                              ;DRVSTA>0 IF DRIVE IS ONLINE
58                                              ;DRVSTA<0 IF DRIVE IS UNSAFE
59
60 040164      000              DRVSTA: .BYTE   0               ;DRIVE 0
63 040165      000                      .BYTE   0               ;DRIVE 1
   040166      000                      .BYTE   0               ;DRIVE 2
   040167      000                      .BYTE   0               ;DRIVE 3
   040170      000                      .BYTE   0               ;DRIVE 4
   040171      000                      .BYTE   0               ;DRIVE 5
   040172      000                      .BYTE   0               ;DRIVE 6
   040173      000                      .BYTE   0               ;DRIVE 7
64
65                                      ;TABLE OF DRIVE TYPES (DRVTYP=8 BYTES)
66                                              ;DRVTYP=0 IF DRIVE IS NONEXISTENT (DRVSTA=0, ALSO)
67                                              ;DRVTYP=1 IF DRIVE IS RM80
68                                              ;DRVTYP=-1 IF NOT RM80
69
70 040174      000              DRVTYP: .BYTE   0               ;DRIVE 0
73 040175      000                      .BYTE   0               ;DRIVE 1
   040176      000                      .BYTE   0               ;DRIVE 2
   040177      000                      .BYTE   0               ;DRIVE 3
```

```
        040200      000                         .BYTE   0               ;DRIVE 4
        040201      000                         .BYTE   0               ;DRIVE 5
        040202      000                         .BYTE   0               ;DRIVE 6
        040203      000                         .BYTE   0               ;DRIVE 7
     74
     75                                 ;TABLE OF DUAL PORT INITIALIZATION INDICATORS
     76                                         ;DPINT=0 IF INITIALIZATION IS NOT ACTIVE ON THE DRIVE
     77                                         ;DPINT<0 IF INITIALIZATION IS IN PROGRESS
     78
     79 040204      000                 DPINT:  .BYTE   0               ;DRIVE 0
     82 040205      000                         .BYTE   0               ;DRIVE 1
        040206      000                         .BYTE   0               ;DRIVE 2
        040207      000                         .BYTE   0               ;DRIVE 3
        040210      000                         .BYTE   0               ;DRIVE 4
        040211      000                         .BYTE   0               ;DRIVE 5
        040212      000                         .BYTE   0               ;DRIVE 6
        040213      000                         .BYTE   0               ;DRIVE 7
     83
     84                                 ;TABLE OF PENDING DUAL PORT REQUESTS
     85                                         ;DPRQS=0 IF THAT A DUAL PORT REQUEST IS NOT PENDING FOR THAT DRIVE
     86                                         ;DPRQS<0 IF THAT A DUAL PORT REQUEST IS PENDING FOR THAT DRIVE
     87
     88 040214      000                 DPRQS:  .BYTE   0               ;DRIVE 0
     91 040215      000                         .BYTE   0               ;DRIVE 1
        040216      000                         .BYTE   0               ;DRIVE 2
        040217      000                         .BYTE   0               ;DRIVE 3
        040220      000                         .BYTE   0               ;DRIVE 4
        040221      000                         .BYTE   0               ;DRIVE 5
        040222      000                         .BYTE   0               ;DRIVE 6
        040223      000                         .BYTE   0               ;DRIVE 7
     92
     93                                 ;DRIVE REQUEST QUE WORDS
     94
     95 040224   000000                 QDRV:   .WORD   0               ;DRIVE 0
     98 040226   000000                         .WORD   0               ;DRIVE 1
        040230   000000                         .WORD   0               ;DRIVE 2
        040232   000000                         .WORD   0               ;DRIVE 3
        040234   000000                         .WORD   0               ;DRIVE 4
        040236   000000                         .WORD   0               ;DRIVE 5
        040240   000000                         .WORD   0               ;DRIVE 6
        040242   000000                         .WORD   0               ;DRIVE 7
     99
    100                                 ;TRANSFER WAIT FLAG (TRNSWT=1 WORD)
    101                                         ;THIS IS A ONE WORD QUEUE. IT WILL CONTAIN THE ADDRESS OF
    102                                         ;"DPB" OF THE I/O OPERATION.
    103
    104 040244   000000                 TRNSWT: .WORD   0
    105
    106                                 ;SEARCH WAIT KEYS (SRCHWT=1 WORD)
    107                                         ;THIS IS A ONE WORD QUEUE THAT WILL CONTAIN A KEY FOR EACH OF
    108                                         ;THE DRIVES THAT ARE PERFORMING A SEARCH COMMAND FOR THE I/O
    109                                         ;REQUEST THAT IS AT THE TOP OF THEIR REQUEST QUEUE.
    110                                         ;EACH DRIVE IS ASSIGNED ONE BIT, STARTING AT BIT00 FOR DRIVE 0.
    111
    112 040246   000000                 SRCHWT: .WORD   0
    113
    114                                 ;RM DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)
```

```
                                        ;ACTDRV=0 IF DRIVER IS INACTIVE
                                        ;ACTDRV>0 IF DRIVER IS ACTIVE

040250      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

040251      000                 ACTSTR: .BYTE   0

                                ;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.

040252  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

040254  177777                  SEEKFG: .WORD   -1

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

040256  177777                  TIMER:  .WORD   -1               ;DRIVE 0
040260  177777                          .WORD   -1               ;DRIVE 1
040262  177777                          .WORD   -1               ;DRIVE 2
040264  177777                          .WORD   -1               ;DRIVE 3
040266  177777                          .WORD   -1               ;DRIVE 4
040270  177777                          .WORD   -1               ;DRIVE 5
040272  177777                          .WORD   -1               ;DRIVE 6
040274  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

040276  177777                  DTUW:   .WORD   -1

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

040300      001     002     004 ATABIT: .BYTE   1,2,4,10,20,40,100,200

                                ;STORAGE FOR RMADR (THE FIRST ADDRESS (776700) OF THE RH70/RM80),
                                ;RMVEC (THE VECTOR ADDRESS (254)), AND RMVEC+2 (THE BR LEVEL (5)).

040310  176700                  RMADR:  .WORD   176700
040312  000254  000240          RMVEC:  .WORD   254,5*32.
040316  000050                  RHEXT:  .WORD   50               ;OFFSET TO RMBAE
```

```
                                ;SEARCH DIFFERENCE IS 5 FIVE SECTORS

040320  000005                  MXWNDW: .WORD   5

                                ;DEFINITIONS OF THE RH70/RM80 ADDRESS INDEXES

        000000                  RMCS1   = 0             ;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 0)
        000002                  RMWC    = 2             ;WORD COUNT REGISTER (NOT A DRIVE REG)
        000004                  RMBA    = 4             ;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)
        000006                  RMDA    = 6             ;DESIRED TRK/SEC ADDRESS REGISTER (DRIVE REG. 5)
        000010                  RMCS2   = 10            ;CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
        000012                  RMDS    = 12            ;DRIVE STATUS REGISTER (DRIVE REG 1)
        000014                  RMER1   = 14            ;ERROR REGISTER #1 (DRIVE REG. 2)
        000016                  RMAS    = 16            ;ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 4)
        000020                  RMLA    = 20            ;LOOK AHEAD REGISTER (DRIVE REG. 7)
        000022                  RMDB    = 22            ;DATA BUFFER REGISTER (NOT A DRIVE REG.)
        000024                  RMMR1   = 24            ;MAINTAINABILITY REGISTER #1  (DRIVE REG. 3)
        000026                  RMDT    = 26            ;DRIVE TYPE REGISTER (DRIVE REG. 6)
        000030                  RMSN    = 30            ;SERIAL NUMBER REGISTER (DRIVE REG. 10)
        000032                  RMOF    = 32            ;OFFSET REGISTER (DRIVE REG. 11)
        000034                  RMDC    = 34            ;DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
        000036                  RMHR    = 36            ;"HOLDING REGISTER" (DRIVE REG. 13)
        000040                  RMMR2   = 40            ;MAINTENANCE REGISTER #2 (DRIVE REG. 14)
        000042                  RMER2   = 42            ;ERROR REGISTER #2 (DRIVE REG. 15)
        000044                  RMEC1   = 44            ;ECC POSITION REGISTER (DRIVE REG. 16)
        000046                  RMEC2   = 46            ;ECC PATTERN REGISTER (DRIVE REG.  17)
        000050                  RMBAE   = 50            ;BUS ADDRESS EXTENTION REGISTER
        000052                  RMCS3   = 52            ;CONTROL AND STATUS REGISIER #3

```

```
                                        ;RH70/RM80 DRIVER INITIALIZATION CODE
                                        ;THIS ROUTINE WILL DETERMINE WHICH RM80 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
                                        ;
040322  104412                  RMINIT: SAVREG                  ;SAVE R0 - R5
040324  013746  177776                  MOV     @#PS,-(SP)      ;SAVE THE PRESENT PROCESSOR STATUS
040330  012737  000240  177776          MOV     #<5*32.>,@#PS   ;CHANGE THE PRIORITY TO 5
040336  004737  045300                  JSR     PC,CLRQUE       ;CLEAR ALL REQUEST QUEUES
040342  012701  040146                  MOV     #RMERRS,R1      ;FIRST ADDRESS TO BE CLEARED
040346  012702  040254                  MOV     #SEEKFG,R2      ;LAST ADDRESS TO BE CLEARED
040352  005021                  1$:     CLR     (R1)+           ;CLEAR
040354  020102                          CMP     R1,R2           ;ARE WE DONE?
040356  101775                          BLOS    1$              ;BRANCH IF NO
040360  012702  040276                  MOV     #DTUW,R2        ;LAST ADDRESS
040364  012721  177777          2$:     MOV     #-1,(R1)+       ;INITIALIZE
040370  020102                          CMP     R1,R2           ;DONE?
040372  101774                          BLOS    2$              ;LOOP IF NO
040374  005037  040164                  CLR     DRVSTA          ;SET ALL DRIVES TO OFFLINE
040400  005037  040166                  CLR     DRVSTA+2
040404  005037  040170                  CLR     DRVSTA+4
040410  005037  040172                  CLR     DRVSTA+6
040414  013703  040312                  MOV     RMVEC,R3        ;SETUP THE RH70/RM80 VECTOR
040420  012723  042452                  MOV     #ISR,(R3)+
040424  013713  040314                  MOV     RMVEC+2,(R3)
040430  013704  040310                  MOV     RMADR,R4        ;FIRST ADDRESS OF RH70/RM80
040434  012764  000040  000010          MOV     #40,RMCS2(R4)   ;MASSBUS INIT
040442  005001                          CLR     R1              ;START WITH DRIVE 0
040444  004037  040534          3$:     JSR     R0,DRVINT       ;INIT THE DRIVE
040450  000401                          BR      4$              ;'DVA' NOT SET
040452  000402                          BR      5$              ;NORMAL RETURN
040454  105061  040164          4$:     CLRB    DRVSTA(R1)      ;SET DRIVE STATUS TO OFFLINE
040460  005201                  5$:     INC     R1              ;GO TO NEXT DRIVE
040462  042701  177770                  BIC     #^C7,R1         ;MASK OUT UNUSED BITS
040466  001366                          BNE     3$              ;BR IF MORE DRIVES TO GO
040470  012701  000007                  MOV     #7,R1           ;START WITH DRIVE 7
040474  005037  177776                  CLR     @#PS            ;CLEAR THE PROCESSOR STATUS
040500  105761  040204          6$:     TSTB    DPINT(R1)       ;WAITING FOR DRIVE TO SWITCH PORTS ?
040504  001405                          BEQ     8$              ;BR NOT WAITING
040506  004737  045226                  JSR     PC,SET.IE       ;SET INTERRUPT
040512  105761  040204          7$:     TSTB    DPINT(R1)       ;DRIVE SWITCHED PORTS ?
040516  001375                          BNE     7$              ;BR IF NOT
040520  005301                  8$:     DEC     R1              ;GO TO THE NEXT DRIVE
040522  100366                          BPL     6$              ;CHECK NEXT DRIVE
040524  012637  177776                  MOV     (SP)+,@#PS      ;RESTORE THE PROCESSOR STATUS
040530  104413                          RESREG                  ;RESTORE R0 - R5
040532  000207                          RTS     PC              ;BYE-BYE

                                        ;DRIVE INITILIZATION ROUTINE
```

```
                                                ;THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
                                                ;AN RM80. 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 RH70/RM80 (RMCS1)
                                        ;       JSR     R0,DRVINT       ;CALLED BY A JSR
                                        ;       RETURN1                 ;ERROR OCCURRED ('NED')
                                        ;       RETURN2                 ;NORMAL RETURN
                                        ;
040534  010546                  DRVINT: MOV     R5,-(SP)        ;SAVE R5
040536  105061  040164                  CLRB    DRVSTA(R1)      ;START DRIVE STATUS AS OFFLINE
040542  105061  040174                  CLRB    DRVTYP(R1)      ;CLEAR THE DRIVE TYPE INDICATOR
040546  010164  000010                  MOV     R1,RMCS2(R4)    ;SELECT A DRIVE
040552  112764  000111  000000          MOVB    #111,RMCS1(R4)  ;DO A DRIVE CLEAR COMMAND (& SEIZE DRIVE)
040560  032764  010000  000010          BIT     #BIT12,RMCS2(R4) ;NONEXISTENT DRIVE?
040566  001403                          BEQ     1$              ;NO---BRANCH
040570  004737  045226                  JSR     PC,SET.IE       ;GO SET "IE" WITHOUT A "TRE"
040574  000476                          BR      4$              ;LEAVE THIS ROUTINE
040576  105061  040164          1$:     CLRB    DRVSTA(R1)      ;SET DRIVE STATUS TO OFFLINE
040602  032764  004000  000000          BIT     #BIT11,RMCS1(R4)  ;SEE IF DRIVE AVAILABLE
040610  001470                          BEQ     4$              ;BR IF DRIVE NOT AVAILABLE
040612  004037  044670                  JSR     R0,RD.RM        ;CALL THE READ ROUTINE
040616  000026                          RMDT                    ;REGISTER OFFSET
040620  040774                          5$                      ;'NED' RETURN
040622  012605                          MOV     (SP)+,R5        ;PUT DRIVE TYPE IN R5
040624  112761  000001  040174          MOVB    #1,DRVTYP(R1)   ;SET RM80 INDICATOR
040632  022705  020026                  CMP     #20026,R5       ;IS IT A SINGLE PORT RM80?
040636  001407                          BEQ     2$              ;BRANCH IF YES
040640  022705  024026                  CMP     #24026,R5       ;IS IT A DUAL PORT RM80?
040644  001404                          BEQ     2$              ;BR IF YES
040646  112761  177777  040174          MOVB    #-1,DRVTYP(R1)  ;SET INDICATOR TO 'OTHER'
040654  000446                          BR      4$              ;EXIT
040656  012746  000121          2$:     MOV     #121,-(SP)      ;DO A "READ-IN PRESET"
040662  004037  044750                  JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
040666  000000                          RMCS1                   ;REGISTER OFFSET
040670  040774                          5$                      ;'NED' RETURN
040672  012746  010000                  MOV     #BIT12,-(SP)    ;SET FMT16=1
040676  004037  044750                  JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
040702  000032                          RMOF                    ;REGISTER OFFSET
040704  040774                          5$                      ;'NED' RETURN
040706  004037  044670                  JSR     R0,RD.RM        ;CALL THE READ ROUTINE
040712  000012                          RMDS                    ;REGISTER OFFSET
040714  040774                          5$                      ;'NED' RETURN
040716  012605                          MOV     (SP)+,R5        ;AND SAVE IT IN R5
040720  100015                          BPL     3$              ;BRANCH IF ATA=0
040722  116164  040300  000016          MOVB    ATABIT(R1),RMAS(R4) ;CLEAR ATTENTION BIT
040730  004037  044670                  JSR     R0,RD.RM        ;CALL THE READ ROUTINE
040734  000014                          RMER1                   ;REGISTER OFFSET
040736  040774                          5$                      ;'NED' RETURN
040740  006126                          ROL     (SP)+           ;IS IT UNSAFE?
040742  100004                          BPL     3$              ;BR IF NOT
040744  112761  177777  040164          MOVB    #-1,DRVSTA(R1)  ;SET UNSAFE INDICATOR
040752  000407                          BR      4$              ;EXIT
```

```
    105 040754  005105                    3$:     COM     R5                      ;CHECK MOL, DPR, DRY, AND VV
    106 040756  042705  167077                    BIC     #^C<BIT12!BIT08!BIT07!BIT06>,R5
    107 040762  001003                            BNE     4$                      ;BRANCH IF MOL, DPR, DRY, OR VV IS CLEAR
    108 040764  112761  000001  040164            MOVB    #1,DRVSTA(R1)           ;SET DRIVE STATUS TO ONLINE
    109 040772  005720                    4$:     TST     (R0)+                   ;STEP OVER THE ERROR RETURN
    110 040774  012605                    5$:     MOV     (SP)+,R5                ;RESTORE R5
    111 040776  000200                            RTS     R0                      ;EXIT
```

```
                                        ;REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
                                        ;
                                        ;CALL:
                                        ;
                                        ;       JSR     R0,RM80         ;CALL THE RM80 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

041000  013746  177776          RM80:   MOV     @#PS,-(SP)      ;SAVE THE CALLING STATUS
041004  013737  040314  177776          MOV     RMVEC+2,@#PS    ;DON'T ALLOW ANY RM80 INTERRUPTS
041012  112737  000001  040250          MOVB    #1,ACTDRV       ;SET "ACTIVE DRIVER" FLAG
041020  104412                          SAVREG                  ;SAVE R0 - R5
041022  011002                          MOV     (R0),R2         ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
041024  005062  000016                  CLR     16(R2)          ;CLEAR THE STATUS/ERROR INDICATOR
041030  111201                          MOVB    (R2),R1         ;PICKUP THE DRIVE NUMBER
041032  013704  040310                  MOV     RMADR,R4        ;UNIBUS ADDRESS OF RMCS1
041036  105761  040164                  TSTB    DRVSTA(R1)      ;CHECK DRIVES STATUS
041042  003011                          BGT     1$              ;BRANCH IF ONLINE
041044  105761  040204                  TSTB    DPINT(R1)       ;TRYING TO INIT THE DRIVE
041050  001027                          BNE     4$              ;BR IF YES
041052  004037  040534                  JSR     R0,DRVINT       ;GO INIT. THE DRIVE
041056  000421                          BR      3$              ;ERROR RETURN
041060  105761  040164                  TSTB    DRVSTA(R1)      ;IS DRIVE STATUS ONLINE?
041064  003432                          BLE     5$              ;BR IF NOT
041066  105761  040214          1$:     TSTB    DPRQS(R1)       ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
041072  001016                          BNE     4$              ;BR IF YES
041074  010164  000010                  MOV     R1,RMCS2(R4)    ;SELECT THE DRIVE
041100  004037  045344                  JSR     R0,DRVQUE       ;PUT THIS REQUEST IN QUEUE
041104  000445                          BR      8$              ;QUEUE IS FULL
041106  105761  040154          2$:     TSTB    DRVACT(R1)      ;IS THIS DRIVE ACTIVE?
041112  001037                          BNE     7$              ;BR IF YES
041114  004737  041236                  JSR     PC,OPT          ;CALL THE OPTIMIZER
041120  000434                          BR      7$
041122  004737  042334          3$:     JSR     PC,CI8          ;GO HANDLE THE 'NED'
041126  000431                          BR      7$
041130  004037  045344          4$:     JSR     R0,DRVQUE       ;PUT REQUEST IN QUEUE
041134  000431                          BR      8$              ;QUEUE IS FULL
041136  032714  000100                  BIT     #BIT06,(R4)     ;IS 'IE' SET ALREADY ?
041142  001023                          BNE     7$              ;BR IF IT IS
041144  004737  045226                  JSR     PC,SET.IE       ;SET INTERRUPT
041150  000420                          BR      7$              ,RETURN, REQUEST IN QUEUE
041152  105761  040164          5$:     TSTB    DRVSTA(R1)      ;SEE IF DRIVE OFFLINE OR UNSAFE
041156  002412                          BLT     6$              ;BR IF UNSAFE
041160  012762  140000  000016          MOV     #BIT15!BIT14,16(R2)     ;SET OFFLINE ERROR INDICATOR
041166  105761  040174                  TSTB    DRVTYP(R1)      ;SEE IF OFFLINE OR NONEXISTENT
041172  001007                          BNE     7$              ;BR IF OFFLINE
041174  012762  100002  000016          MOV     #BIT15!BIT01,16(R2)     ;REPORT DRIVE NONEXISTENT
041202  000403                          BR      7$              ;GO TO EXIT
041204  012762  110000  000016  6$:     MOV     #BIT15!BIT12,16(R2)     ;DRIVE IS UNSAFE
041212  104413                  7$:     RESREG                  ;RESTORE R0 - R5
041214  005720                          TST     (R0)+           ;SETUP FOR NORMAL RETURN
041216  000401                          BR      9$              ;FINISH UP, THEN
041220  104413                  8$:     RESREG                  ;RESTORE R0 - R5
041222  005720                  9$:     TST     (R0)+           ;CORRECT THE RETURN ADDRESS
041224  105037  040250                  CLRB    ACTDRV          ;CLEAR "ACTIVE DRIVER" FLAG
```

```
     041230  012637  177776                  MOV     (SP)+,@#PS      ;RETURN 'PS" TO USER LEVEL
     041234  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
                                     ;
     041236  104412                  OPT:    SAVREG                  ;SAVE R0 - R5
     041240  013746  177776                  MOV     @#PS,-(SP)      ;SAVE PROC. STATUS
     041244  004737  045366                  JSR     PC,GETREQ       ;GET "DPB" POINTER OF REQUEST
     041250  005702                          TST     R2              ;IS THERE A REQUEST IN QUEUE?
     041252  001510                          BEQ     9$              ;NO--BRANCH TO EXIT
     041254  032764  004000  000000          BIT     #BIT11,RMCS1(R4)        ;IS DVA SET?
     041262  001407                          BEQ     2$              ;BRANCH IF NOT
     041264  032764  000100  000012          BIT     #BIT6,RMDS(R4)  ;IS VV SET ?
     041272  001003                          BNE     2$              ;BR IF IT IS
     041274  004037  040534          1$:     JSR     R0,DRVINT       ;SEE IF DRIVE STILL ONLINE ?
     041300  000473                          BR      8$              ;'DVA' NOT SET
     041302  105761  040164          2$:     TSTB    DRVSTA(R1)      ;IS DRIVE ONLINE?
     041306  003014                          BGT     3$              ;YES--BRANCH
     041310  004737  045400                  JSR     PC,POPQUE       ;NO--REMOVE REQUEST FROM QUEUE
     041314  012762  140000  000016          MOV     #BIT15!BIT14,16(R2)     ;SET OFFLINE STATUS/ERROR INDICATOR
     041322  105761  040164                  TSTB    DRVSTA(R1)      ;IS DRIVE UNSAFE ?
     041326  100067                          BPL     10$             ;BR TO EXIT IF NOT
     041330  012762  110000  000016          MOV     #BIT15!BIT12,16(R2)     ;SET UNSAFE STATUS/ERROR INDICATOR
     041336  000463                          BR      10$             ;BRANCH TO EXIT
     041340  012746  000111          3$:     MOV     #111,-(SP)      ;LOAD COMMAND ONTO THE STACK
     041344  004037  044750                  JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
     041350  000000                          RMCS1                   ;REGISTER OFFSET
     041352  041470                          8$                      ;'NED' RETURN
     041354  032714  004000                  BIT     #BIT11,(R4)     ;DRIVE AVAILABLE ?
     041360  001432                          BEQ     7$              ;BR IF NOT
     041362  122762  000150  000002          CMPB    #150,2(R2)      ;IS THE REQUEST FOR I/O?
     041370  002403                          BLT     4$              ;YES--BRANCH
     041372  004737  041756                  JSR     PC,CI4          ;CALL THE COMMAND INITIATOR
     041376  000443                          BR      10$             ;BRANCH TO EXIT
     041400  005737  040276          4$:     TST     DTUW            ;DATA TRANSFER UNDERWAY?
     041404  002015                          BGE     6$              ;YES--GO START A SEARCH
     041406  136137  040300  040246          BITB    ATABIT(R1),SRCHWT       ;FINISHED A SEARCH ?
     041414  001003                          BNE     5$              ;IF NE, YES
     041416  005737  040254                  TST     SEEKFG          ;DO IMPLIED SEEKS?
     041422  100006                          BPL     6$              ;IF PL DO A SEARCH
     041424  146137  040300  040246  5$:     BICB    ATABIT(R1),SRCHWT       ;CLEAR "SEARCH WAIT" KEY
     041432  004737  041516                  JSR     PC,CI1          ;START A DATA TRANSFER
     041436  000423                          BR      10$
     041440  004737  041650          6$:     JSR     PC,CI3          ;START A SEARCH
     041444  000420                          BR      10$             ;GO TO THE EXIT
     041446  112761  177777  040214  7$:     MOVB    #-1,DPRQS(R1)   ;SET PORT REQUEST INDICATOR
     041454  010103                          MOV     R1,R3           ;SET UP TO ADDRESS WORDS
     041456  006303                          ASL     R3              ;CONVERT TO WORD INDEX
     041460  012763  035230  040256          MOV     #15000.,TIMER(R3)       ;START 15 SECOND TIMER
     041466  000402                          BR      9$              ;EXIT
     041470  004737  042334          8$:     JSR     PC,CI8          ;PROCESS THE 'NED'
     041474  032714  000100          9$:     BIT     #BIT06,(R4)     ;SEE IF 'IE' ALREADY SET
     041500  001002                          BNE     10$             ;BR IF SET
```

```
                                        JSR     PC,SET.IE       ;SET "IE" WITHOUT A "TRE"
                                10$:    MOV     (SP)+,@#PS      ;RESTORE PROC. STATUS
                                        RESREG                  ;RESTORE R0 - R5
                                        RTS     PC

                                ;COMMAND INITIATOR
                                ;
                                ;CALL:
                                ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER
                                ;       MOV     #DPB,R2         ;ADDRESS OF DPB
                                ;       JSR     PC,CI?          ;CI?= CI1,CI3, OR CI4
                                ;                               ;WHERE:
                                ;                               ;CI1=DATA TRANSFER
                                ;                               ;CI2=SEARCH REQUESTED BY DATA XFER
                                ;                               ;CI4=NOT DATA TRANSFER

                                ;START A DATA TRANSFER

                                CI1:    JSR     PC,POPQUE       ;REMOVE REQUEST FROM "DRIVES WAIT" QUEUE
                                        MOV     R2,TRNSWT       ;PUT REQ. IN TRANSFER WAIT QUEUE
                                        MOV     R2,R3           ;DPB ADDRESS TO R3
                                        MOV     RMADR,R4        ;RMCS1 ADDRESS
                                        MOV     R1,RMCS2(R4)    ;SELECT DRIVE
                                        ADD     #4,R3           ;DESIRED WORD COUNT
                                        ADD     #2,R4           ;RMWC ADDRESS
                                        MOV     (R3)+,(R4)+     ;LOAD WORD COUNT
                                        MOV     (R3)+,(R4)+     ;LOAD BUFFER ADDRESS
                                        MOV     (R3)+,-(SP)     ;LOAD SECTOR AND TRACK
                                        JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
                                        RMDA                    ;REGISTER OFFSET
                                        CI8                     ;'NED' RETURN
                                        MOV     (R3)+,-(SP)     ;LOAD CYLINDER ADDRESS
                                        JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
                                        RMDC                    ;REGISTER OFFSET
                                        CI8                     ;'NED' RETURN
                                        JSR     R0,RD.RM        ;CALL THE READ ROUTINE
                                        RMOF                    ;REGISTER OFFSET
                                        CI8                     ;'NED' RETURN
                                        BIC     #BIT09,(SP)     ;CLEAR 'SKIP SECTOR ERROR INHIBIT'
                                        JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
                                        RMOF                    ;REGISTER OFFSET
                                        CI8                     ;'NED' RETURN
                                        MOV     2(R2),-(SP)     ;LOAD "COMMAND+GO", "A17&A16", AND "PSEL"
                                        JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
                                        RMCS1                   ;REGISTER OFFSET
                                        CI8                     ;'NED' RETURN
                                        MOV     R1,DTUW         ;SET "DATA TRANSFER UNDERWAY"
                                        JMP     CI5

                                ;START A SEARCH

                                CI3:    MOV     RMADR,R4        ;RMCS1 ADDRESS
                                        MOV     R1,RMCS2(R4)    ;SELECT DRIVE
                                        MOV     12(R2),-(SP)    ;DESIRED CYLINDER ADDRESS
                                        JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
                                        RMDC                    ;REGISTER OFFSET
                                        CI8                     ;'NED' RETURN
```

```
                                                MOVB    10(R2),R3       ;PICKUP SECTOR ADDRESS
                                                SUB     MXWNDW,R3       ;BACKUP BY MAX. SEARCH FOR I/O WINDOW
                                                BGE     1$
                                                ADD     #31.,R3
                                        1$:     MOV     R3,-(SP)        ;COMBINE THE ADJUSTED SECTOR WITH
                                                MOVB    11(R2),1(SP)    ;THE DESIRED TRACK
                                                JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
                                                RMDA                    ;REGISTER OFFSET
                                                CI8                     ;'NED' RETURN
                                                MOV     #131,-(SP)      ;START A SEARCH
                                                JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
                                                RMCS1                   ;REGISTER OFFSET
                                                CI8                     ;'NED' RETURN
                                                BISB    ATABIT(R1),SRCHWT       ;SET ''SEARCH WAIT'' KEY
                                                BR      CI5

                                        ;INITIATE A NON-I/O OPERATION

                                        CI4:    MOV     RMADR,R4        ;RMCS1 ADDRESS
                                                MOV     R1,RMCS2(R4)    ;SELECT DRIVE
                                                MOVB    2(R2),R3        ;PICKUP THE REQUESTED COMMAND
                                                CMPB    #131,R3         ;IS IT A SEARCH COMMAND?
                                                BNE     1$              ;BRANCH IF NO
                                                MOV     10(R2),-(SP)    ;LOAD DESIRED TRACK & SECTOR
                                                JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
                                                RMDA                    ;REGISTER OFFSET
                                                CI8                     ;'NED' RETURN
                                                BR      2$              ;GO LOAD CYLINDER
                                        1$:     CMPB    #105,R3         ;IS IT A SEEK COMMAND
                                                BNE     3$              ;BRANCH IF NO
                                        2$:     MOV     12(R2),-(SP)    ;LOAD DESIRED CYLINDER
                                                JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
                                                RMDC                    ;REGISTER OFFSET
                                                CI8                     ;'NED' RETURN
                                                BR      14$
                                        3$:     CMPB    #115,R3         ;IS IT AN ''OFFSET'' COMMAND?
                                                BNE     4$              ;BR IF NO
                                                JSR     R0,RD.RM        ;CALL THE READ ROUTINE
                                                RMOF                    ;REGISTER OFFSET
                                                CI8                     ;'NED' RETURN
                                                MOVB    1(R2),(SP)      ;BYTE WHEN LOADING THE REGISTER
                                                JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
                                                RMOF                    ;REGISTER OFFSET
                                                CI8                     ;'NED' RETURN
                                                BR      14$
                                        4$:     CMPB    #107,R3         ;IS IT A ''RECALIBRATE'' COMMAND?
                                                BNE     5$              ;IF NE, NO
                                                CLR     -(SP)           ;CYLINDER ZERO
                                                JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
                                                RMDC                    ;REGISTER OFFSET
                                                CI8                     ;'NED' RETURN
                                                BR      14$             ;CONTINUE
                                        5$:     CMPB    #143,R3         ;IS IT A ''SET FORMAT'' COMMAND?
                                                BNE     6$              ;BRANCH IF NO
                                                JSR     R0,RD.RM        ;CALL THE READ ROUTINE
                                                RMOF                    ;REGISTER OFFSET
                                                CI8                     ;'NED' RETURN
```

```
042136  116266  000001  000001          MOVB    1(R2),1(SP)     ;COMBINE 'FMT16','ECI','HCI', & 'SSEI'
042144  004037  044750                  JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
042150  000032                          RMOF                    ;REGISTER OFFSET
042152  042334                          CI8                     ;'NED' RETURN
042154  000436                          BR      12$
042156  122703  000141          6$:     CMPB    #141,R3         ;IS IT A "GET REGISTER" COMMAND?
042162  001023                          BNE     10$             ;BRANCH IF NO
042164  016203  000006          7$:     MOV     6(R2),R3        ;POINTS TO 1ST ADDRESS OF WHERE
                                                                ;TO PUT THE REGISTER(S)
042170  116237  000010  042206          MOVB    10(R2),9$       ;INIT. THE INDEX FOR THE FIRST REG.
042176  116205  000011                  MOVB    11(R2),R5       ;INDEX OF LAST REG. TO MOVE
042202  004037  044670          8$:     JSR     R0,RD.RM        ;READ RM80 REGISTER
042206  000000          9$:     RMCS1                           ;INDEX OF REG. TO READ
042210  042334                          CI8
042212  012623                          MOV     (SP)+,(R3)+     ;GET THE CONTENTS OF RH70/RM80 REG.
042214  023705  042206                  CMP     9$,R5           ;LAST REG. BEEN READ?
042220  001414                          BEQ     12$             ;GET OUT IF YES
042222  062737  000002  042206          ADD     #2,9$           ;INCREASE THE INDEX BY 2
042230  000764                          BR      8$              ;LOOP--MORE TO READ
042232  122703  000145          10$:    CMPB    #145,R3         ;IS IT A "SELECT DRIVE" COMMAND?
042236  001405                          BEQ     12$             ;BRANCH IF YES
042240  010346          11$:    MOV     R3,-(SP)        ;LOAD THE COMMAND
042242  004037  044750                  JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
042246  000000                          RMCS1                   ;REGISTER OFFSET
042250  042334                          CI8                     ;'NED' RETURN
042252  004737  045400          12$:    JSR     PC,POPQUE       ;REMOVE REQ. FROM QUEUE
042256  052762  000200  000016          BIS     #BIT07,16(R2)   ;SET THE "DONE" BIT
042264  005737  040252                  TST     SAVEFG          ;SAVE THE RH70/RM80 REGISTERS?
042270  100002                          BPL     13$             ;BRANCH IF NO
042272  004737  045062                  JSR     PC,SVRH70       ;YES--GO SAVE THE REGISTERS
042276  000207          13$:    RTS     PC              ;RETURN TO USER

                        ;START A NON-DATA TRANSFER OPERATION

042300  010346          14$:    MOV     R3,-(SP)        ;LOAD THE COMMAND
042302  004037  044750                  JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
042306  000000                          RMCS1                   ;REGISTER OFFSET
042310  042334                          CI8                     ;'NED' RETURN

                        ;START THE COMMAND TIMER

042312  006301  CI5:     ASL     R1
042314  012761  023420  040256          MOV     #10000.,TIMER(R1)       ;SET A 10 SECOND TIMER
042322  006201                          ASR     R1
042324  112761  000001  040154          MOVB    #1,DRVACT(R1)   ;SET THE DRIVE ACTIVE
042332  000207                          RTS     PC              ;RETURN TO THE USER

                        ;PROCESS A NON-EXISTENT DRIVE

042334  104412  CI8:     SAVREG                          ;SAVE R0 - R5
042336  105761  040154                  TSTB    DRVACT(R1)      ;DRIVE ACTIVE?
042342  001431                          BEQ     3$              ;BRANCH IF NO
042344  013702  040244                  MOV     TRNSWT,R2       ;GET THE "TRANSFER WAIT" QUEUE
042350  020137  040276                  CMP     R1,DTUW         ;DID THIS DRIVE HAVE AN I/O IN PROGRESS?
042354  001402                          BEQ     1$              ;BRANCH IF YES
042356  004737  045366                  JSR     PC,GETREQ       ;GET THE DPB POINTER
042362  005702          1$:     TST     R2              ;QUEUE ENTRY FOR DRIVE ?
```

```
250 042364  001403                          BEQ     2$                  ;BR IF NOT
251 042366  012762  100002  000016          MOV     #BIT15!BIT01,16(R2)     ;SET 'DRIVE NON-EXISTENT' INDICATOR
252 042374  012763  177777  040256  2$:     MOV     #-1,TIMER(R3)       ;STOP THE TIMER
253 042402  105061  040154                  CLRB    DRVACT(R1)          ;SET 'DRIVE ACTIVE'' TO IDLE
254 042406  020137  040276                  CMP     R1,DTUW             ;IS THIS DRIVE SETUP FOR A TRANSFER
255 042412  001005                          BNE     3$                  ;BR IF NOT
256 042414  012737  177777  040276          MOV     #-1,DTUW            ;RESET THE INDICATOR
257 042422  005037  040244                  CLR     TRNSWT              ;CLEAR THE TRANSFER QUEUE
258 042426  004737  045332          3$:     JSR     PC,EMPTYQ           ;CLEAR THE DRIVE'S QUEUE
259 042432  105061  040164                  CLRB    DRVSTA(R1)          ;SET DRIVE TO OFFLINE
260 042436  105061  040174                  CLRB    DRVTYP(R1)          ;CLEAR THE DRIVE TYPE INDICATOR
261 042442  004737  045226          4$:     JSR     PC,SET.IE           ;SET ''IE'' WITHOUT ''TRE''
262 042446  104413                          RESREG                      ;RESTORE R0 - R5
263 042450  000207                          RTS     PC                  ;RETURN
```

```
                                        ;INTERRUPT SERVICE ROUTINE
                                        
042452  112737  000001  040250  ISR:    MOVB    #1,ACTDRV       ;SET "ACTIVE DRIVER" FLAG
042460  104412                          SAVREG                  ;SAVE R0 - R5
042462  013704  040310                  MOV     RMADR,R4        ;ADDRESS OF RMCS1
042466  013701  040276                  MOV     DTUW,R1         ;GET "DATA TRANSFER UNDERWAY" INDICATOR
042472  002402                          BLT     1$              ;BRANCH IF NO DATA TRANSFER UNDERWAY
042474  004737  042514                  JSR     PC,TD           ;CALL TRANSFER DONE
042500  004737  043334          1$:     JSR     PC,SC           ;CALL SPECIAL CONDITIONS
042504  104413                          RESREG                  ;RESTORE R0 - R5
042506  105037  040250                  CLRB    ACTDRV          ;CLEAR "ACTIVE DRIVER" FLAG
042512  000002                          RTI                     ;RETURN
                                        
                                        ;TRANSFER DONE ROUTINE
                                        
042514  105061  040154          TD:     CLRB    DRVACT(R1)      ;SET DRIVE ACTIVE INDICATOR TO IDLE
042520  012737  177777  040276          MOV     #-1,DTUW        ;NO DATA TRANSFERS UNDERWAY
042526  006301                          ASL     R1
042530  012761  177777  040256          MOV     #-1,TIMER(R1)   ;CANCEL TIMEOUT
042536  006201                          ASR     R1
042540  013702  040244                  MOV     TRNSWT,R2       ;GET "DPB" ADDRESS FROM THE
042544  005037  040244                  CLR     TRNSWT          ;TRANSFER WAIT QUEUE--CLEAR QUEUE
042550  052762  000200  000016          BIS     #BIT07,16(R2)   ;SET DONE
042556  010164  000010                  MOV     R1,RMCS2(R4)    ;SELECT THE DRIVE
042562  004037  044670                  JSR     R0,RD.RM        ;CALL THE READ ROUTINE
042566  000000                          RMCS1                   ;REGISTER OFFSET
042570  042334                          CI8                     ;'NED' RETURN
042572  006126                          ROL     (SP)+           ;IS TRE=1 ?
042574  100421                          BMI     3$              ;BR IF YES
042576  005737  040252                  TST     SAVEFG          ;SAVE THE RH70/RM80 REGISTERS?
042602  100002                          BPL     1$              ;BRANCH IF NO
042604  004737  045062                  JSR     PC,SVRH70       ;YES--SAVE THE REGISTERS
042610  004737  043246          1$:     JSR     PC,WC.HK        ;SEE IF WRITE CHECK TO BE PUT IN QUEUE
042614  004737  04[366                  JSR     PC,GETREQ       ;GET DPB POINTER
042620  005702                          TST     R2              ;ENTRY FOR DRIVE ?
042622  001403                          BEQ     2$              ;BR IF NOT
042624  004737  041236                  JSR     PC,OPT          ;CALL OPTIMIZER
042630  000207                          RTS     PC              ;RETURN
042632  012714  000113          2$:     MOV     #113,(R4)       ;RELEASE THE DRIVE
042636  000207                          RTS     PC              ;RETURN
                                        
042640                          3$:
042640  004037  044670                  JSR     R0,RD.RM        ;CALL THE READ ROUTINE
042644  000014                          RMER1                   ;REGISTER OFFSET
042646  042334                          CI8                     ;'NED' RETURN
042650  032726  000600                  BIT     #BIT8!BIT7,(SP)+        ;SEE IF HCRC OR HCE ERRORS
042654  001016                          BNE     4$              ;IF NE, YES
042656  004037  044670                  JSR     R0,RD.RM        ;CALL THE READ ROUTINE
042662  000042                          RMER2                   ;REGISTER OFFSET
042664  042334                          CI8                     ;'NED' RETURN
042666  032726  000040                  BIT     #SSE,(SP)+      ;SEE IF SKIP SECTOR ERROR
042672  001407                          BEQ     4$              ;IF EQ, NO
042674  004037  044670                  JSR     R0,RD.RM        ;CALL THE READ ROUTINE
042700  000032                          RMOF                    ;REGISTER OFFSET
042702  042334                          CI8                     ;'NED' RETURN
042704  032726  001000                  BIT     #SSEI,(SP)+     ;IS THE INHIBIT BIT ALREADY SET ?
042710  001416                          BEQ     SKIP            ;IF EQ, NO
```

```
                                                4$:     BIS     #BIT15!BIT06,16(R2)     ;SET DATA ERROR FLAG
                                                        JSR     PC,EMPTYQ               ;EMPTY THE 'DRIVE'S WAIT'' QUEUE
                                                        JSR     PC,SVRH70               ;SAVE THE RH70/RM80 REGISTERS
                                                        MOV     #40111,(R4)             ;ISSUE A 'DRIVE CLEAR''
                                                        MOV     #113,(R4)               ;ISSUE A RELEASE TO THE DRIVE
                                                        RTS     PC                      ;RETURN

                                                ;SKIP SECTOR HANDLING ROUTINE

                                                SKIP:   MOV     R1,DTUW                 ;LOAD ACTIVE DRIVE NUMBER
                                                        MOV     R2,TRNSWT               ;RESTORE TRANSFER FLAG
                                                        CLR     $TATUS(R2)              ;CLEAR THE DRIVER STATUS
                                                        BITB    #2,$COMND(R2)           ;SEE IF HEADER ORDER
                                                        BEQ     4$                      ;IF EQ, NO
                                                        JSR     R0,RD.RM                ;CALL THE READ ROUTINE
                                                        RMDC                            ;REGISTER OFFSET
                                                        CI8                             ;'NED' RETURN
                                                        MOV     (SP)+,$CYL(R2)          ;SAVE NEW STARTING CYLINDER ADDRESS
                                                        JSR     R0,RD.RM                ;CALL THE READ ROUTINE
                                                        RMDA                            ;REGISTER OFFSET
                                                        CI8                             ;'NED' RETURN
                                                        MOV     (SP)+,$SEC(R2)          ;SAVE NEW STARTING TRK/SEC ADDRESS
                                                        JSR     R0,RD.RM                ;CALL THE READ ROUTINE
                                                        RMWC                            ;REGISTER OFFSET
                                                        CI8                             ;'NED' RETURN
                                                        MOV     (SP)+,R0                ;GET REMAINING WORD COUNT
                                                        BNE     2$                      ;BR IF PARTIAL SECTOR LEFT
                                                        MOV     $WCNT(R2),R0            ;STARTING WORD COUNT
                                                                                        ;IF RMWC WAS AT ZERO AFTER TRANSFER,
                                                1$:     SUB     #-258.,R0               ;FIND THE # NUMBER OF WORDS FOR LAST SECTOR
                                                        BLT     1$                      ;BR IF NOT DONE YET
                                                2$:     ADD     #-258.,R0               ;ADD 1 SECTOR BACK
                                                        CMP     R0,#-6                  ;IS WORD COUNT AT LEAST 6 ?
                                                        BLE     3$                      ;BR IF YES
                                                        MOV     #-6,R0                  ;SET WORD COUNT TO 6
                                                3$:     MOV     R0,$WCNT(R2)            ;STORE NEW WORD COUNT IN DPB
                                                        MOV     R0,$WRDL(R2)            ;STORE NEW WORD COUNT LENGTH IN DPB AND
                                                        NEG     $WRDL(R2)               ;MAKE IT POSITIVE
                                                        MOV     $BUF(R2),R3             ;STARTING BUFFER ADDRESS
                                                        BR      5$

                                                4$:
                                                        JSR     R0,RD.RM                ;CALL THE READ ROUTINE
                                                        RMWC                            ;REGISTER OFFSET
                                                        CI8                             ;'NED' RETURN
                                                        SUB     $WCNT(R2),(SP)          ;CALCULATE THE NUMBER OF WORD TRANSFERED
                                                        BIC     #377,(SP)               ;LEAVE ONLY SECTOR MULTIPLES
                                                        MOV     (SP),R3                 ;COPY THE DIFFERENCE
                                                        ADD     $WCNT(R2),(SP)          ;NEW WORD COUNT
                                                        MOV     (SP)+,R0                ;COPY THE WORD COUNT
                                                        ASL     R3                      ;CONVERT WORD DIFFERENCE TO A BYTE DIFFERENCE
                                                        ADD     $BUF(R2),R3             ;NEW BUFFER ADDRESS
                                                5$:     MOV     R3,-(SP)                ;BUFFER ADDRESS
                                                        JSR     R0,WRT.RM               ;CALL THE WRITE ROUTINE
                                                        RMBA                            ;REGISTER OFFSET
                                                        CI8                             ;'NED' RETURN
                                                        MOV     R0,-(SP)                ;WORD COUNT
```

```
                                        JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
                                        RMWC                    ;REGISTER OFFSET
                                        CI8                     ;'NED' RETURN
                                        MOV     #40111,(R4)     ;CLEAR THE DRIVE
                                        JSR     R0,RD.RM        ;CALL THE READ ROUTINE
                                        RMOF                    ;REGISTER OFFSET
                                        CI8                     ;'NED' RETURN
                                        BIS     #SSEI,(SP)      ;SET THE INHIBIT BIT
                                        JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
                                        RMOF                    ;REGISTER OFFSET
                                        CI8                     ;'NED' RETURN
                                        MOV     #1,$SSENB(R2)   ;INDICATE THAT SKIP SECTORING WAS ENABLED
                                        MOV     $COMND(R2),-(SP)        ;GET THE ORIGINAL COMMAND
                                        JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
                                        RMCS1                   ;REGISTER OFFSET
                                        CI8                     ;'NED' RETURN
                                        RTS     PC              ;RETURN

                                ;FORCED WRITE CHECK ROUTINE

                        WC.HK:  TST     WRTCHK          ;DO WRITE CHECK ?
                                BEQ     2$              ;BR IF NOT
                                CMPB    #WRTDAT,$COMND(R2)      ;LAST OPERATION A WRITE COMMAND ?
                                BNE     2$              ;BR IF NOT
                        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$              ;BRANCH IF ANY 'ATA' BITS SET
                                JSR     R0,RD.RM        ;CALL THE READ ROUTINE
                                RMCS1                   ;REGISTER OFFSET
                                CI8                     ;'NED' RETURN
                                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

                                ;PROCESS ALL DRIVES WITH "ATA" SET

                        SC3:    BIT     R3,(SP)         ;ATA=1?
                                BNE     SC5             ;YES--BRANCH
                        SC4:    INC     R1              ;MOVE TO THE NEXT DRIVE
                                ASLB    R3
```

```
                                        BNE     SC3             ;BRANCH IF MORE TO CHECK?
                                        TST     (SP)+           ;CLEAN OFF THE STACK
                                        RTS     PC              ;RETURN TO USER

                                ;DETERMINE IF THE DRIVE WITH "ATA" SET IS ACTIVE WITH A COMMAND

                                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    DRVACT(R1)      ;DRIVE ACTIVE ?
                                        BNE     SC6             ;BR IF ACTIVE

                                ;THE DRIVE WAS NOT ACTIVE, FIND OUT WHY IT INTERRUPTED

                                        JSR     PC,SC12         ;SAVE RMDS, RMER1 AND RMER2
                                                                ;ALSO DO A DRIVE INIT (DRVINT)
                                        TSTB    DPINT(R1)       ;TRYING TO INIT THE DRIVE ?
                                        BNE     SC4             ;BR IF YES, CHECK ON MORE DRIVES
                                        BIT     #BIT13,RMERRS+6 ;ADDRESS PLUG CHANGED ?
                                        BNE     4$              ;BR IF YES
                                3$:     MOV     (SP),R5         ;PICKUP (RMAS) BEFORE THE ERROR CALL:
                                        EMT     2               ;REPORT THE UNEXPECTED ATTENTION
                                        BR      SC4             ;GO CHECK FOR MORE ATA'S
                                4$:
                                        EMT     5               ;REPORT THE ADDRESS PLUG CHANGE
                                        BR      SC4             ;CHECK FOR MORE DRIVES

                                ;THE DRIVE COMPLETED A NON-I/O COMMAND

                                SC6:    ASL     R1              ;SETUP TO ADDRESS WORDS
                                        MOV     #-1,TIMER(R1)   ;STOP THE TIMER
                                        ASR     R1              ;RESTORE THE DRIVE ADDRESS
                                        JSR     PC,GETREQ       ;GET THE DPB POINTER FROM THE QUEUE
                                        MOV     R1,RMCS2(R4)    ;SELECT DRIVE
                                        JSR     R0,RD.RM        ;CALL THE READ ROUTINE
                                        RMDS                    ;REGISTER OFFSET
                                        SC8                     ;'NED' RETURN
                                        MOV     (SP),R5         ;AND PUT IT IN R5
                                        ROL     (SP)+           ;WAS THERE AN ERROR?
                                        BPL     SC11            ;BR IF NO ERROR
                                        JSR     R0,RD.RM        ;CALL THE READ ROUTINE
                                        RMER1                   ;REGISTER OFFSET
                                        SC8                     ;'NED' RETURN
                                        MOV     (SP)+,R5        ;AND SAVE IT IN R5
                                        JSR     PC,SVRH70       ;SAVE RH70/RM80 REGISTERS
                                        MOV     #111,-(SP)      ;ISSUE A DRIVE CLEAR
                                        JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
                                        RMCS1                   ;REGISTER OFFSET
                                        SC8                     ;'NED' RETURN
                                        ROL     R5              ;WAS 'UNSAFE" CONDITION =1?
                                        BMI     1$              ;BRANCH IF YES
                                        BIS     #BIT15!BIT07!BIT05,16(R2)       ;INFORM USER OF ERROR
                                        BR      2$
                                1$:     BIS     #BIT15!BIT07!BIT04,16(R2)       ;INFORM USER OF UNSAFE ERROR
```

```
                                2$:     CLRB    DRVACT(R1)              ;SET DRIVE TO IDLE
                                        JSR     PC,EMPTYQ               ;DUMP THE QUEUE
                                        BICB    ATABIT(R1),SRCHWT       ;CLEAR THE SEARCH WAIT FLAG
                                        MOV     #113,-(SP)              ;RELEASE COMMAND
                                        JSR     R0,WRT.RM               ;CALL THE WRITE ROUTINE
                                        RMCS1                           ;REGISTER OFFSET
                                        SC8                             ;'NED' RETURN
                                        JMP     SC4                     ;CHECK FOR MORE DRIVES

                                ;ISR 'NED'' PROCESSOR

                                SC8:    JSR     PC,EMPTYQ               ;CLEAR THE DRIVE'S QUEUE
                                        JSR     PC,CI8                  ;PROCESS THE 'NED'
                                        JMP     SC4                     ;CHECK MORE DRIVES

                                ;NON-I/O COMMAND TERMINATION ROUTINE

                                SC11:   CLRB    DRVACT(R1)              ;SET DRIVE IDLE
                                        BITB    ATABIT(R1),SRCHWT               ;DOING A SEARCH OPERATION FOR
                                                                                ;AN I/O COMMAND?
                                        BNE     1$                      ;BRANCH IF YES
                                        JSR     PC,POPQUE               ;REMOVE REQUEST FROM QUEUE
                                        BIS     #BIT07,16(R2)           ;SET 'DONE'' BIT
                                        TST     SAVEFG                  ;SAVE THE REGISTERS?
                                        BPL     1$                      ;BRANCH IF NO
                                        JSR     PC,SVRH70               ;YES--SAVE ALL OF THE RH70/RM80 REG'S
                                1$:     MOVB    ATABIT(R1),RMAS(R4)     ;CLEAR ATTENTION BIT
                                        JSR     PC,OPT                  ;START A REQUEST
                                        JMP     SC4                     ;CHECK FOR MORE DRIVES

                                ;ERROR PROCESSOR

                                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
                                        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

                                ;DUAL PORT REQUEST PROCESSOR

                                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
                                        BIT     #BIT11,(R4)             ;DRIVE AVAILABLE ?
                                        BNE     1$                      ;BR IF AVAILABLE
                                        ASL     R1
                                        MOV     #15000.,TIMER(R1)       ;START 15 SECOND TIMER AGAIN
                                        ASR     R1
                                        BR      3$                      ;EXIT
                                1$:     TSTB    DPINT(R1)               ;INITIALIZING THE DRIVE ?
                                        BEQ     2$                      ;BR IF NOT
```

```
277 044066  105061  040204                    CLRB    DPINT(R1)          ;CLEAR THE INIT INDICATOR
278 044072  004037  040534                    JSR     R0,DRVINT          ;GO INIT THE DRIVE
279 044076  000240                            NOP                        ;DUMMY PARITY ERROR RETURN
280 044100  105761  040164                    TSTB    DRVSTA(R1)         ;DRIVE ONLINE ?
281 044104  003014                            BGT     2$                 ;BR IF YES -- START COMMAND
282 044106  005702                            TST     R2                 ;QUEUE ENTRY FOR THE DRIVE
283 044110  001416                            BEQ     3$                 ;BR IF NOT
284 044112  004737  045366                    JSR     PC,GETREQ          ;GET DPB ADDRESS
285 044116  052762  140000  000016            BIS     #BIT15!BIT14,16(R2)     ;INFORM USER THAT DRIVE OFFLINE
286 044124  004737  045062                    JSR     PC,SVRH70          ;SAVE THE REGISTERS
287 044130  004737  045332                    JSR     PC,EMPTYQ          ;EMPTY THE REQUEST QUEUE
288 044134  000404                            BR      3$
289 044136  105061  040214          2$:       CLRB    DPRQS(R1)          ;CLEAR THE PORT REQUEST INDICATOR
290 044142  004737  041236                    JSR     PC,OPT             ;START THE PENDING REQUEST
291 044146  000137  043404          3$:       JMP     SC4                ;PROCESS OTHER DRIVES
```

```
                                        ;RM80 TIMER ROUTINE
                                        ;CALL:
                                        ;       MOV     #TIME,-(SP)     ;ELASPED TIME IN MILLISECONDS ON THE STACK
                                        ;       JSR     PC,RMTMR        ;CALL RM80 TIME ROUTINE
                                        ;
 044152  005737  040250         RMTMR: TST     ACTDRV          ;CHECK "ACTDRV & ACTSTR"
 044156  001030                        BNE     4$              ;IF NON ZERO EXIT
 044160  112737  000001  040251        MOVB    #1,ACTSTR       ;SET "ACTSTR"
 044166  104412                        SAVREG                  ;SAVE R0 - R5
 044170  005001                        CLR     R1              ;START WITH DRIVE 0
 044172  005003                        CLR     R3                      )
 044174  005763  040256         1$:    TST     TIMER(R3)       ;IS THE TIMER RUNNING?
 044200  002407                        BLT     2$              ;BRANCH IF NO
 044202  166663  000002  040256        SUB     2(SP),TIMER(R3) ;COUNT THE INTERVAL
 044210  003003                        BGT     2$              ;BR IF NO SOFTWARE TIMEOUT
 044212  004737  044244                JSR     PC,STO          ;CALL SOFTWARE TIMEOUT ROUTINE
 044216  000405                        BR      3$              ;GO TO THE EXIT
 044220  005201                2$:     INC     R1              ;MOVE TO NEXT DRIVE
 044222  005723                        TST     (R3)+
 044224  022701  000010                CMP     #8.,R1          ;OUT OF DRIVES?
 044230  003361                        BGT     1$              ;BRANCH IF NO
 044232  104413                3$:     RESREG                  ;RESTORE R0 - R5
 044234  105037  040251                CLRB    ACTSTR          ;ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
 044240  012616                4$:     MOV     (SP)+,(SP)      ;ADJUST THE STACK
 044242  000207                        RTS     PC              ;RETURN

                                        ;SOFTWARE TIMEOUT ROUTINE
                                        ;
                                        ;NOTE: THIS ROUTINE MUST BE ENTERED AT PRIORITY 6
                                        ;       OR GREATER
                                        ;
                                        ;CALL:  STO
                                        ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER
                                        ;       JSR     PC,STO          ;CALL:
                                        ;       RETURN

 044244  010146                 STO:    MOV     R1,-(SP)        ;SAVE R1
 044246  010346                         MOV     R3,-(SP)        ;SAVE R3
 044250  013704  040310                 MOV     RMADR,R4        ;GET ADDRESS OF "RMCS1"
 044254  010164  000010                 MOV     R1,RMCS2(R4)    ;SELECT THE DRIVE
 044260  004037  044670                 JSR     R0,RD.RM        ;CALL THE READ ROUTINE
 044264  000012                         RMDS                    ;REGISTER OFFSET
 044266  044556                         STO5                    ;'NED' RETURN
 044270  105726                         TSTB    (SP)+           ;IS "DRY"=1?
 044272  100473                         BMI     STO2            ;BR IF YES
 044274  105761  040204         STO1:   TSTB    DPINT(R1)       ;TRYING TO INTIALIZE THE DRIVE ?
 044300  001070                         BNE     STO2            ;BR IF YES
 044302  105761  040214                 TSTB    DPRQS(R1)       ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
 044306  001065                         BNE     STO2            ;BR IF YES
 044310  013702  040244                 MOV     TRNSWT,R2       ;PICKUP TRANSFER WAIT QUEUE
 044314  020137  040276                 CMP     R1,DTUW         ;TRANSFER UNDERWAY ON THIS DRIVE?
 044320  001402                         BEQ     1$              ;BRANCH IF YES
 044322  004737  045306                 JSR     PC,GETREQ       ;GET DPB ADDRESS
 044326  052762  101000  000016  1$:    BIS     #BIT15!BIT09,16(R2)     ;SET THE ERROR FLAGS
 044334  004737  045062                 JSR     PC,SVRH70       ;SAVE RH70/RM80 REGISTERS
 044340  012764  000040  000010         MOV     #BIT05,RMCS2(R4)        ;"INIT" THE MASS BUS
 044346  105061  040154                 CLRB    DRVACT(R1)      ;DRIVE IS IDLE
```

```
                                                CLR     R1              ;START WITH DRIVE 0
                                                CLR     R3
                                        2$:     JSR     R0,DRVINT       ;INIT. THIS DRIVE
                                                BR      ST05            ;PARITY ERROR RETURN
                                                TSTB    DRVACT(R1)      ;DRIVE IDLE BEFORE THE INIT.?
                                                BEQ     4$              ;YES--BRANCH
                                                MOV     TRNSWT,R2       ;GET TRANSFER WAIT QUEUE
                                                CMP     DTUW,R1         ;WAS THERE I/O ON THIS DRIVE?
                                                BEQ     3$              ;YES--BRANCH
                                                JSR     PC,GETREQ       ;GET THE DPB POINTER FROM QUEUE
                                        3$:     BIS     #BIT15!BIT08,16(R2)     ;INFORM USER OF INIT.
                                                CLRB    DRVACT(R1)      ;SET DRIVE ACTIVE TO IDLE
                                        4$:     MOV     #-1,TIMER(R3)   ;STOP THE TIMER
                                                TST     (R3)+           ;UPDATE THE INDEX
                                                INC     R1              ;INCREMENT THE DRIVE NUMBER
                                                CMP     #8.,R1          ;LAST DRIVE BEEN CHECKED?
                                                BGT     2$              ;NO--LOOP
                                                MOV     #-1,DTUW        ;NO DATA TRANSFERS UNDERWAY
                                                CLR     TRNSWT          ;CLEAR TRANSFER WAIT QUEUE
                                                JSR     PC,CLRQUE       ;CLEAR ALL REQUEST QUEUES
                                                BR      ST09            ;EXIT
                                        ST02:   MOVB    RMAS(R4),R5     ;READ ATTENTION REG
                                                BITB    ATABIT(R1),R5   ;IS ATTENTION FOR THIS DRIVE UP ?
                                                BNE     ST03            ;YES--BRANCH
                                                TSTB    DPINT(R1)       ;TRYING TO INTIALIZE THE DRIVE ?
                                                BNE     ST06            ;BR IF YES - DRIVE NOT ONLINE
                                                TSTB    DPRQS(R1)       ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
                                                BNE     ST07            ;BR IF YES - NO RESPONSE TO REQUEST
                                                CMP     R1,DTUW         ;DATA TRANSFER UNDERWAY FOR THIS DRIVE
                                                BNE     ST01            ;BR IF NO
                                                JSR     R0,RD.RM        ;CALL THE READ ROUTINE
                                                'CS1                    ;REGISTER OFFSET
                                                S05                     ;'NED' RETURN
                                                TSTB    (SP)+
                                                BPL     ST01            ;BR IF 'RDY'=0
                                        ST03:   TSTB    DPINT(R1)       ;INITIALIZING THE DRIVE ?
                                                BNE     1$              ;BR IF INIT PENDING
                                                TSTB    DPRQS(.1)       ;PORT REQUEST PENDING ?
                                                BEQ     ST09            ;BR IF NOT
                                        1$:     MOV     #-1,TIMER(R3)   ;STOP THE TIMER
                                                BR      ST09            ;EXIT
                                        ST05:   JSR     PC,C18          ;GO HANDLE THE 'NED'
                                                BR      ST09
                                        ST06:   CLRB    DPINT(R1)       ;CLEAR THE INITIALIZE INDICATOR
                                                CLRB    DRVSTA(R1)      ;SET DRIVE OFFLINE
                                                MOV     #-1,TIMER(R3)   ;STOP THE TIMER
                                                JSR     PC,GETREQ       ;GET THE DPB ADDRESS
                                                TST     R2              ;REQUEST IN QUEUE ?
                                                BEQ     ST09            ;BR IF NOT
                                                BIS     #BIT15!BIT14,16(R2)     ;INFORM THE USER DRIVE NOT AVAILABLE
                                                BR      ST08            ;FINISH
                                        ST07:   MOV     #-1,TIMER(R3)   ;STOP THE TIMER
                                                CLRB    DPRQS(R1)       ;CLEAR PORT REQUEST INDICATOR
                                                JSR     PC,GETREQ       ;GET DPB ADDRESS
                                                TST     R2              ;QUEUE ENTRY FOR DRIVE ?
                                                BEQ     ST09            ;BR IF NONE
                                                MOV     #BIT15!BIT2,16(R2)      ;INFORM USER OF PORT REQUEST ERROR
```

```
    111 044652  004737  045332          ST08:   JSR     PC,EMPTYQ       ;CLEAR THE QUEUE FOR THE DRIVE
    112 044656  004737  045062                  JSR     PC,SVRH70       ;SAVE THE REGISTERS
    113 044662  012603                  ST09:   MOV     (SP)+,R3        ;RESTORE R3
    114 044664  012601                          MOV     (SP)+,R1        ;RESTORE R1
    115 044666  000207                          RTS     PC              ;RETURN
```

```
                                        ;ROUTINE TO READ A RH70/RM80 REGISTER
                                        ;
                                        ;CALL:
                                        ;       JSR     R0,RD.RM        ;GO READ A REGISTER
                                        ;       INDEX                   ;REG. INDEX FROM BASE
                                        ;       ERRADR                  ;ERROR ADDRESS--PROCESS ERROR STARTING
                                        ;                               ;AT THIS ADDRESS
                                        ;       RETURN                  ;CONTENTS OF REG. IS ON THE STACK
                                        ;
044670  011646                  RD.RM:  MOV     (SP),-(SP)      ;SAVE R0 FOR RETURN
044672  013737  040310  044706          MOV     RMADR,RD.ADR    ;FORM THE DESIRED ADDRESS
044700  062037  044706                  ADD     (R0)+,RD.ADR    ;USING THE BASE AND THE INDEX
044704  013727                  RD.RM1: MOV     @(PC)+,(PC)+    ;READ THE DESIRED REGISTER OF THE RM80
044706  000000                  RD.ADR: .WORD   0               ;ADDRESS IS FORMED HERE
044710  000000                  RD.WRD: .WORD   0               ;REG. CONTENTS PUT HERE
044712  013766  044710  000002          MOV     RD.WRD,2(SP)    ;RETURN IT TO THE USER
044720  013746  040310                  MOV     RMADR,-(SP)     ;PUT THE ADDRESS ON THE STACK
044724  062716  000010                  ADD     #RMCS2,(SP)     ;FORM THE ADDRESS OF RMCS2
044730  032736  010000                  BIT     #BIT12,@(SP)+   ;CHECK THE 'NED' BIT
044734  001002                          BNE     RD.RM3          ;IF NE, DRIVE NOT PRESENT
044736  005720                          TST     (R0)+           ;ERROR FREE RETURN
044740  000402                          BR      RD.RM4          ;EXIT
044742  011000                  RD.RM3: MOV     (R0),R0         ;ERROR EXIT
044744  012616                          MOV     (SP)+,(SP)
044746  000200                  RD.RM4: RTS     R0

                                        ;ROUTINE TO WRITE A REGISTER
                                        ;
                                        ;CALL:
                                        ;       MOV     DATA,-(SP)      ;DATA TO BE LOADED ON THE STACK
                                        ;       JSR     R0,WRT.RM       ;CALL THE ROUTINE TO LOAD (WRITE) THE REG.
                                        ;       INDEX                   ;INDEX OF THE REGISTER TO BE LOADED
                                        ;       'NED' RETURN            ;ADDRESS TO RETURN TO IF 'NED' ERROR
                                        ;       RETURN                  ;ERROR FREE RETURN

044750  016637  000002  045030  WRT.RM: MOV     2(SP),WRT.WD    ;SAVE THE WORD TO WRITE
044756  012616                          MOV     (SP)+,(SP)      ;ADJUST THE STACK
044760  012037  045032                  MOV     (R0)+,WRT.AD    ;GET INDEX OF REGISTER TO BE WRITTEN
044764  001015                          BNE     1$              ;BRANCH IF NOT RMCS1
044766  122737  000150  045030          CMPB    #150,WRT.WD     ;IS THE COMMAND FOR DATA TRANSFERS?
044774  002411                          BLT     1$              ;YES--DON'T GET THE OLD A16 & A17, & PSEL
044776  004037  044670                  JSR     R0,RD.RM        ;CALL THE READ ROUTINE
045002  000000                          RMCS1                   ;REGISTER OFFSET
045004  045052                          WRT.R3                  ;'NED' RETURN
                                                                ;THE COMMAND BEFORE SENDING TO THE
                                                                ;THE RH70/RM80

045006  000316                          SWAB    (SP)
045010  042716  177770                  BIC     #^C7,(SP)
045014  112637  045031                  MOVB    (SP)+,WRT.WD+1
045020  063737  040310  045032  1$:     ADD     RMADR,WRT.AD    ;FORM THE ADDRESS OF THE DISK REG.
045026  012737                  WRT.R1: MOV     (PC)+,@(PC)+    ;LOAD THE DESIRED REG.
045030  000000                  WRT.WD: .WORD   0               ;WORD TO WRITE GOES HERE
045032  000000                  WRT.AD: .WORD   0               ;ADDRESS IS FORMED HERE
045034  013746  040310                  MOV     RMADR,-(SP)     ;PUT THE ADDRESS ON THE STACK
045040  062716  000010                  ADD     #RMCS2,(SP)     ;FORM THE ADDRESS OF RMCS2
045044  032736  010000                  BIT     #BIT12,@(SP)+   ;CHECK THE 'NED' BIT
045050  001402                          BEQ     WRT.R4          ;IF EQ, DRIVE IS PRESENT
```

```
045052  011000                  WRT.R3: MOV     (R0),R0         ;TAKE THE 'NED' EXIT
045054  000401                          BR      WRT.R5          ;EXIT
045056  005720                  WRT.R4: TST     (R0)+           ;ADJUST FOR ERROR FREE EXIT
045060  000200                  WRT.R5: RTS     R0

                                ;ROUTINE TO SAVE THE RH70/RM80 REGISTERS
                                ;
                                ;CALL:
                                ;       MOV     #DPBADR,R2      ;DPB POINTER TO R2
                                ;       JSR     PC,SVRH70       ;SAVE THE DRIVES REG'S

045062  104412                  SVRH70: SAVREG                  ;SAVE R0 - R5
045064  005702                          TST     R2              ;QUEUE ENTRY FOR THE DRIVE ?
045066  001442                          BEQ     6$              ;BR IF NONE
045070  013704  040310                  MOV     RMADR,R4
045074  111264  000010                  MOVB    (R2),RMCS2(R4)  ;SELECT DRIVE
045100  016203  000014                  MOV     14(R2),R3       ;GET THE ERROR TABLE POINTER
045104  001433                          BEQ     6$              ;EXIT IF NO ADDRESS
045106  005037  045142                  CLR     3$              ;COUNTER & POINTER
045112  023727  045142  000022  1$:     CMP     3$,#RMDB        ;REACHED THE BUFFER REGISTER ?
045120  001006                          BNE     2$              ;BR IF NOT
045122  032764  000200  000010          BIT     #BIT07,RMCS2(R4)        ;'OR' SET ?
045130  001002                          BNE     2$              ;BR IF SET
045132  005023                          CLR     (R3)+           ;STORE RMDB AS ZEROES
045134  000405                          BR      4$              ;CONTINUE
045136  004037  044670          2$:     JSR     R0,RD.RM        ;READ THE SELECTED REGISTER
045142  000000                  3$:     .WORD   0               ;REGISTER INDEX
045144  045170                          5$                      ;ERROR RETURN ADDRESS
045146  012623                          MOV     (SP)+,(R3)+     ;STORE THE REGISTER CONTENTS
045150  023727  045142  000046  4$:     CMP     3$,#RMEC2       ;REACHED THE END ?
045156  001406                          BEQ     6$              ;BR IF YES
045160  062737  000002  045142          ADD     #2,3$           ;INCREMENT THE REGISTER INDEX
045166  000751                          BR      1$              ;CONTINUE READING THE REGISTERS
045170  004737  042334          5$:     JSR     PC,CI8          ;PROCESS THE 'NED'
045174  013746  001234          6$:     MOV     $CPUOP,-(SP)    ;CHECK THE CPU (RH) TYPE
045200  042716  003777                  BIC     #^C174000,(SP)  ;LEAVE THE CPU TYPE BITS
045204  022726  030000                  CMP     #30000,(SP)+    ;SEE IF RH70
045210  001004                          BNE     7$              ;IF NE, NO
045212  063704  040316                  ADD     RHEXT,R4        ;POINT TO RMBAE
045216  012423                          MOV     (R4)+,(R3)+     ;STORE THE CONTENTS
045220  011413                          MOV     (R4),(R3)       ;GET RMCS3
045222  104413                  7$:     RESREG                  ;RESTORE R0 - R5
045224  000207                          RTS     PC              ;RETURN

                                ;ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A "TRE"
                                ;CALL:
                                ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
                                ;       JSR     PC,SET.IE       ;SET "IE"
                                ;       RETURN

045226  010446                  SET.IE: MOV     R4,-(SP)        ;SAVE R4
045230  013704  040310                  MOV     RMADR,R4        ;PICKUP ADDRESS OF RMCS1
045234  010164  000010                  MOV     R1,RMCS2(R4)    ;SELECT DRIVE
045240  011446                          MOV     (R4),-(SP)      ;READ RMCS1
045242  052716  040000                  BIS     #BIT14,(SP)     ;SET THE "TRE" BIT OF THE WORD READ
045246  000316                          SWAB    (SP)            ;ADJUST FOR DATO
045250  112714  000100                  MOVB    #BIT06,(R4)     ;SET "IE"
```

```
                                        BIT     #BIT12,RMCS2(R4)        ;IS 'NED'=1?
                                        BNE     1$              ;YES--CLEAR 'TRE'
                                        TST     (SP)+           ;CLEAN OFF THE STACK
                                        BR      2$
                                1$:     MOVB    (SP)+,1(R4)     ;CLEAR 'TRE'
                                2$:     MOV     (SP)+,R4        ;RESTORE R4
                                        RTS     PC              ;RETURN TO CALLER

                                ;ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
                                ;
                                ;CALL:
                                ;       JSR     PC,CLRQUE

                                CLRQUE: SAVREG                  ;SAVE R0 - R5
                                        MOV     #QDRV,R2        ;QUEUE BASE ADDRESS
                                        CLR     (R2)+           ;CLEAR ENTRY
                                        CLR     (R2)+           ;CLEAR ENTRY
                                        CLR     (R2)+           ;CLEAR ENTRY
                                        CLR     (R2)+           ;CLEAR ENTRY
                                        CLR     (R2)+           ;CLEAR ENTRY
                                        CLR     (R2)+           ;CLEAR ENTRY
                                        CLR     (R2)+           ;CLEAR ENTRY
                                        CLR     (R2)+           ;CLEAR ENTRY
                                        CLR     (R2)+           ;CLEAR ENTRY
                                        RESREG                  ;RESTORE R0 - R5
                                        RTS     PC

                                ;EMPTY THE QUEUE SPECIFIED BY R1
                                ;
                                ;CALL:
                                ;       MOV     DRVNUM,R1       ;DRIVE NUMBER TO R1
                                ;       JSR     PC,EMPTYQ

                                EMPTYQ: ASL     R1
                                        CLR     QDRV(R1)        ;CLEAR DRIVE QUEUE
                                        ASR     R1              ;RESTORE R1
                                        RTS     PC

                                ;ROUTINE TO PUT A REQUEST IN QUEUE
                                ;
                                ;CALL:
                                ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER
                                ;       MOV     #DPB,R2         ;ADDRESS OF PARAMETER BLOCK
                                ;       JSR     R0,DRVQUE       ;GO PUT REQUEST IN QUEUE
                                ;       RETURN1                 ;RETURN HERE IF QUEUE IS FULL
                                ;       RETURN2                 ;RETURN HERE IF REQUEST IS IN QUEUE

                                DRVQUE: ASL     R1
                                        TST     QDRV(R1)        ;TEST THE QUEUE ENTRY
                                        BNE     1$              ;IF NE, QUEUE ENTRY ALREADY THERE
                                        MOV     R2,QDRV(R1)     ;ADD THE QUEUE ENTRY
                                        TST     (R0)+           ;TAKE RETURN 2
                                1$:     ASR     R1
                                2$:     RTS     R0              ;RETURN TO USER

                                ;ROUTINE TO GET THE 'DPB' ADDRESS OF NEXT REQUEST IN QUEUE
                                ;
                                ;CALL:
```

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

045366  006301                  GETREQ: ASL     R1                      ;
045370  016102  040224                  MOV     QDRV(R1),R2             ;GET THE REQUEST
045374  006201                          ASR     R1                      ;
045376  000207                          RTS     PC                      ;RETURN

                                ;ROUTINE TO "POP" THE REQUEST FROM QUEUE

                                ;CALL:
                                ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
                                ;       JSR     PC,POPQUE       ;CALL TO REMOVE REQUEST
                                ;       RETURN                  ;R2=ADDRESS OF DPB REMOVED

045400  006301                  POPQUE: ASL     R1
045402  016102  040224                  MOV     QDRV(R1),R2     ;GET THE QUEUE ENTRY
045406  005061  040224                  CLR     QDRV(R1)        ;CLEAR THE QUEUE
045412  006201                          ASR     R1
045414  000207                          RTS     PC              ;RETURN TO USER
```

DATA, CONTROL, & STATUS BLOCKS

```
                                .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 COUNT FOR 'RELBUF' ROUTINE
000112                          $SSENB  = $WRDL+72              ;SKIP SECTOR INDICATOR

                                ;INDEX EQUATES TO THE NEXT OPERATION PARAMETERS

000114                          $NCODE  = $WRDL+74              ;NEXT OPERATION CODE
000115                          $NPATC  = $NCODE+1              ;NEXT PATTERN
000116                          $NSEC   = $NCODE+2              ;NEXT SECTOR
000117                          $NTRK   = $NCODE+3              ;NEXT TRACK
000120                          $NCYL   = $NCODE+4              ;NEXT CYLINDER
000122                          $NEXT   = $NCODE+6              ;PARAMETER SELECTION INDICATOR
000124                          $FIRST  = $NCODE+10             ;FIRST OPERATION INDICATOR

                                ;INDEX EQUATES FOR MAXIMUM/MINIMUM ADDRESSES

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

DATA, CONTROL, & STATUS BLOCKS

```
 58          000134                  MINTRK  = MAXCYL+6              ;MINIMUM TRACK ADDRESS
 59          000136                  MAXSEC  = MAXCYL+10             ;MAXIMUM SECTOR ADDRESS
 60          000140                  MINSEC  = MAXCYL+12             ;MINIMUM SECTOR ADDRESS
 61
 62                                  ;HDA SERIAL NUMBER CONTAINED IN DEC144 FILE
 63
 64          000142                  $HSNL   = MAXCYL+14             ;LSB'S OF SERIAL NUMBER (DECIMAL)
 65          000144                  $HSNM   = $HSNL+2               ;MSB'S OF SERIAL NUMBER (DECIMAL)
 66
 67                                  ;DEC144 BAD SECTOR ADDRESS STORAGE AREA INDEX EQUATE
 68
 69          000146                  $BDSEC  = $HSNL+4               ;BAD SECTOR STORAGE TABLE PLUS TERMINATOR
 70
 71                                  ;DRIVE (MBA) SERIAL NUMBER AREA INDEX EQUATE
 72
 73          002130                  $DRVSN  = $BDSEC+<126.*8.>+2    ;DRIVE (MBA) SERIAL NUMBER
 74
 75                                  ;RH/RM REGISTER EQUATES
 76
 77          002140                  $RMCS1  = $DRVSN+10             ;RM REGISTER STORAGE
 78          002142                  $RMWC   = $RMCS1+2
 79          002144                  $RMBA   = $RMCS1+4
 80          002146                  $RMDA   = $RMCS1+6
 81          002150                  $RMCS2  = $RMCS1+10
 82          002152                  $RMDS   = $RMCS1+12
 83          002154                  $RMER1  = $RMCS1+14
 84          002156                  $RMAS   = $RMCS1+16
 85          002160                  $RMLA   = $RMCS1+20
 86          002162                  $RMDB   = $RMCS1+22
 87          002164                  $RMMR1  = $RMCS1+24
 88          002166                  $RMDT   = $RMCS1+26
 89          002170                  $RMSN   = $RMCS1+30
 90          002172                  $RMOF   = $RMCS1+32
 91          002174                  $RMDC   = $RMCS1+34
 92          002176                  $RMHR   = $RMCS1+36
 93          002200                  $RMMR2  = $RMCS1+40
 94          002202                  $RMER2  = $RMCS1+42
 95          002204                  $RMEC1  = $RMCS1+44
 96          002206                  $RMEC2  = $RMCS1+46
 98          002210                  $RMBAE  = $RMCS1+50
 99          002212                  $RMCS3  = $RMCS1+52
101
113                                  ;BLOCK FOR DRIVE 0
     045416     000     000          DRIVE0: .BYTE   0,0             ;DRIVE NUMBER 0
     045420                                  .BLKW   5
     045432  047556                         .WORD   .+$RMCS1-$REG
     045434                                  .BLKB   $RMCS3-$REG

                                     ;BLOCK FOR DRIVE 1
     047632     001     000          DRIVE1: .BYTE   1,0             ;DRIVE NUMBER 1
     047634                                  .BLKW   5
     047646  051772                         .WORD   .+$RMCS1-$REG
     047650                                  .BLKB   $RMCS3-$REG

                                     ;BLOCK FOR DRIVE 2
     052046     002     000          DRIVE2: .BYTE   2,0             ;DRIVE NUMBER 2
     052050                                  .BLKW   5
```

```
052062  054206                          .WORD   .+$RMCS1-$REG
052064                                  .BLKB   $RMCS3-$REG

                                ;BLOCK FOR DRIVE 3
054262      003     000         DRIVE3: .BYTE   3,0             ;DRIVE NUMBER 3
054264                                  .BLKW   5
054276  056422                          .WORD   .+$RMCS1-$REG
054300                                  .BLKB   $RMCS3-$REG

                                ;BLOCK FOR DRIVE 4
056476      004     000         DRIVE4: .BYTE   4,0             ;DRIVE NUMBER 4
056500                                  .BLKW   5
056512  060636                          .WORD   .+$RMCS1-$REG
056514                                  .BLKB   $RMCS3-$REG

                                ;BLOCK FOR DRIVE 5
060712      005     000         DRIVE5: .BYTE   5,0             ;DRIVE NUMBER 5
060714                                  .BLKW   5
060726  063052                          .WORD   .+$RMCS1-$REG
060730                                  .BLKB   $RMCS3-$REG

                                ;BLOCK FOR DRIVE 6
063126      006     000         DRIVE6: .BYTE   6,0             ;DRIVE NUMBER 6
063130                                  .BLKW   5
063142  065266                          .WORD   .+$RMCS1-$REG
063144                                  .BLKB   $RMCS3-$REG

                                ;BLOCK FOR DRIVE 7
065342      007     000         DRIVE7: .BYTE   7,0             ;DRIVE NUMBER 7
065344                                  .BLKW   5
065356  067502                          .WORD   .+$RMCS1-$REG
065360                                  .BLKB   $RMCS3-$REG

114
115                             ;GENERAL PURPOSE PARAMETER BLOCK
116
117 067556      000             GENDPB: .BYTE   0               ;DRIVER PARAMETER BLOCK, DRIVE #
118 067557      000                     .BYTE   0               ;OFFSET VALUE OR FMT16, HCI OR ECI
119 067560      000                     .BYTE   0               ;COMMAND CODE
120 067561      000                     .BYTE   0               ;PSEL, A16 AND A17
121 067562  177776                      .WORD   -2              ;WORD COUNT (NEG)
122 067564  101174                      .WORD   CYLNDR          ;BUFFER ADDRESS
123 067566      000                     .BYTE   0               ;SECTOR ADDRESS
124 067567      000                     .BYTE   0               ;TRACK ADDRESS
125 067570  000000                      .WORD   0               ;CYLINDER ADDRESS
126 067572  067576                      .WORD   GENREG          ;ADDRESS TO SAVE ALL RH/RM REG'S
127 067574  000000                      .WORD   0               ;STATUS WORD
128
129 067576                      GENREG: .BLKW   24              ;REGISTER STORAGE
```

```
                                        .SBTTL  ERROR MESSAGES

067646  122  110  040  EM1:    .ASCIZ  /RH CONTROLLER INTERRUPT OCCURRED (RMAS=0)/
067720  125  116  105  EM2:    .ASCIZ  /UNEXPECTED ATTENTION OCCURRED/
067756  115  101  123  EM3:    .ASCIZ  /MASSBUS PARITY ERROR (MCPE=1)/
070014  115  101  123  EM4:    .ASCIZ  /MASSBUS PARITY ERROR (PAR=1)/
070051  101  104  104  EM5:    .ASCIZ  /ADDRESS PLUG CHANGE BIT SET/
070105  122  110  040  EM6:    .ASCIZ  /RH CONTROLLER DIDN'T RESPOND TO ADDRESSING/
070160  125  116  103  EM10:   .ASCIZ  /UNCORRECTABLE MASSBUS PARITY ERROR/
070223  106  101  124  EM11:   .ASCIZ  /FATAL MASSBUS PARITY ERROR/
070256  120  105  122  EM12:   .ASCIZ  /PERSISTENT DEVICE UNSAFE/
070307  117  120  105  EM13:   .ASCIZ  /OPERATION NOT COMPLETED WITHIN TIME LIMIT/
070361  104  122  111  EM14:   .ASCIZ  /DRIVE WENT OFFLINE/
070404  116  117  040  EM15:   .ASCIZ  /NO RESPONSE TO PORT REQUEST/
070440  110  105  101  EM20:   .ASCIZ  /HEADER CRC ERROR/
070461  104  101  124  EM21:   .ASCIZ  /DATA CHECK ('DCK') ERROR/
070512  127  122  111  EM22:   .ASCIZ  /WRITE CHECK ERROR - DATA CHECK ('DCK') SET/
070565  127  122  111  EM23:   .ASCIZ  /WRITE CHECK ERROR - DATA CHECK ('DCK') NOT SET/
070644  110  105  101  EM24:   .ASCIZ  /HEADER READ ERROR - 'FMT' BIT DROPPED/
070712  110  105  101  EM25:   .ASCIZ  /HEADER READ ERROR - HEADER COMPARE ('HCE') ERROR/
070773  106  117  122  EM26:   .ASCIZ  /FORMAT ERROR ('FER')/
071020  110  105  101  EM27:   .ASCIZ  /HEADER COMPARE ('HCE') ERROR/
071055  115  111  123  EM30:   .ASCIZ  /MISCELLANEOUS DRIVE ERROR/
071107  117  120  105  EM31:   .ASCIZ  /OPERATION INCOMPLETE ('OPI') ERROR/
071152  104  122  111  EM32:   .ASCIZ  /DRIVE TIMING ('DTE') ERROR/
071205  120  101  122  EM33:   .ASCIZ  /PARITY ('PAR') ERROR AFTER OPERATION STARTED/
071262  127  122  111  EM34:   .ASCIZ  /WRITE CLOCK FAILURE ('WCF') ERROR/
071324  111  116  126  EM35:   .ASCIZ  /INVALID ADDRESS ('IAE') ERROR/
071362  127  122  111  EM36:   .ASCIZ  /WRITE LOCK ('WLE') ERROR/
071413  104  101  124  EM37:   .ASCIZ  /DATA CHECK ('DCK') SET DURING WRITE CHECK COMMAND/
071475  122  110  040  EM40:   .ASCIZ  /RH CONTROLLER OR UNIBUS TRANSFER ERROR/
071544  102  125  123  EM41:   .ASCIZ  /BUS ADDRESS OR WORD COUNT INCORRECT/
071610  104  101  124  EM42:   .ASCIZ  /DATA COMPARE ERRORS - NO OTHER ERROR(S) DETECTED/
071671  103  101  116  EM43:   .ASCIZ  /CAN'T MATCH DATA READ WITH A PATTERN - UNKNOWN DATA PATTERN/
071765  105  122  122  EM44:   .ASCIZ  /ERROR BIT(S) SET, BUT NO ERROR SIGNALED BY THE RH CONTROLLER/
072062  105  103  103  EM45:   .ASCIZ  /ECC LOGIC FAILURE - POSITION REGISTER VALUE NOT VALID/
072150  102  125  123  EM46:   .ASCIZ  /BUS ADDRESS AND WORD COUNT NOT CONSISTENT/
072222  105  103  103  EM47:   .ASCIZ  /ECC LOGIC FAILURE - PATTERN REGISTER IS ZERO/
072277  123  105  105  EM50:   .ASCIZ  /SEEK INCOMPLETE ('SKI') ERROR/
072335  120  122  117  EM51:   .ASCIZ  /PROGRAM DETECTED POSITIONING ERROR/
072400  105  103  110  EM52:   .ASCIZ  /ECH ERROR - UNCORRECTABLE ECC ERROR/
072444  104  122  111  EM60:   .ASCIZ  /DRIVE UNSAFE ERROR/
```

ERROR MESSAGFS

```
      1 072467      122      115      101  DH1:     .ASCIZ  /RMAS/
      2 072474      104      122      111  DH2:     .ASCIZ  /DRIVE   RMDS    RMER1   RMER2  RMMR2   RMAS/
      3 072550      104      122      111  DH3:     .ASCIZ  /DRIVE   REG ADR  DATA/
      4 072576      104      122      111  DH4:     .ASCIZ  /DRIVE   REG ADR    GOOD    BAD/
      5 072635      044      122      115  DH6:     .ASCIZ  /$RMADR/
      6 072644      104      122      126  DH14:    .ASCII  /DRV RMCS1   RMCS2   RMDS    RMER1   /
      7 072710      122      115      115           .ASCIZ  /RMMR2  RMER2   RMEC1   RMEC2/<CRLF>
      8 072746      122      115      127  DH15:    .ASCII  /RMWC    RMBA    RMDA    RMAS    RMLA    /
      9 073016      122      115      104           .ASCIZ  /RMDB    RMMR1   RMDT/<CRLF>
     10 073044      122      115      123  DH16:    .ASCIZ  /RMSN    RMOF    RMDC    RMHR    STATUS/<CRLF>
     12 073114      122      115      102  DH17:    .ASCIZ  /RMBAE   RMCS3/<CRLF>
     14                                    .EVEN
```

ERROR MESSAGES

```
      1 073134  001322  000000          DT1:    .WORD   ATTN,0
      2 073140  001220  040146  040150  DT2:    .WORD   DRIVE,RMERRS,RMERRS+2,RMERRS+4,RMERRS+6,ATTN,0
      3 073156  001220  044706  044710  DT3:    .WORD   DRIVE,RD.ADR,RD.WRD,0
      4 073166  001220  045032  045030  DT4:    .WORD   DRIVE,WRT.AD,WRT.WD,RD.WRD,0
      5 073200  001272  000000          DT6:    .WORD   $RMADR,0
      6 073204  002140  002150  002152  DT14:   .WORD   $RMCS1,$RMCS2,$RMDS,$RMER1,$RMMR2,$RMER2,$RMEC1,$RMEC2,0
      7 073226  002142  002144  002146  DT15:   .WORD   $RMWC,$RMBA,$RMDA,$RMAS,$RMLA,$RMDB,$RMMR1,$RMDT,0
      8 073250  002170  002172  002174  DT16:   .WORD   $RMSN,$RMOF,$RMDC,$RMHR,$STATUS,0
     10 073264  002210  002212  000000  DT17:   .WORD   $RMBAE,$RMCS3,0
```

ERROR MESSAGES

```
      1 073272      120      122      123  LIN2C:   .ASCIZ  /PRSNT COMMAND= /
      2 073312      040      040      120  LIN2P:   .ASCIZ  /  PREV COMMAND= /
      3 073333      052      040      105  LIN2S:   .ASCIZ  @* ERROR AT BAD TRACK/SECTOR@
      4 073367      105      122      122  LINM3:   .ASCIZ  /ERROR AT C/
      5 073402      040      124      000  T:       .ASCIZ  / T/
      6 073405      120      122      123  LINN3:   .ASCIZ  /PRSNT ADDR= C/
      7 073423      040      123      000  S:       .ASCIZ  / S/
      8 073426      040      040      120  LINP3:   .ASCIZ  /  PREV ADDR= C/
      9 073445      123      124      101  LINS3:   .ASCIZ  /START CYL= /
     10 073461      040      040      105  LINEN3:  .ASCIZ  /  END CYL= /
     11 073475      040      040      101  LINA3:   .ASCIZ  /  ACTUAL CYL= /
     12 073514      040      040      124  LINT3:   .ASCIZ  /  TRK= /
     13 073524      040      040      122  LINCA3:  .ASCIZ  /  RMDC= /
     14 073535      122      115      104  LINDA3:  .ASCIZ  /RMDA= /
     15 073544      122      115      102  LINB3:   .ASCIZ  /RMBA= /
     16 073553      040      040      122  LINW3:   .ASCIZ  /  RMWC= /
     17 073564      123      124      101  LINST3:  .ASCIZ  /START TRK= /
     18 073600      123      124      101  LINSS3:  .ASCIZ  /START SEC= /
     19 073614      102      125      106  LINM4:   .ASCIZ  /BUFFER ADDR= /
     20 073632      040      040      127  LINS4:   .ASCIZ  /  WRD CNT= /
     21 073646      040      040      101  LINX4:   .ASCIZ  /  ACTUAL NMBR WRDS XFRD= /
     22 073700      105      130      120  LIND5:   .ASCIZ  /EXPCTD DATA= /
     23 073716      040      040      122  LINB5:   .ASCIZ  /  RECEVD DATA= /
     24 073736      040      040      127  LINP5:   .ASCIZ  /  WORD POS= /
     25 073753      110      105      101  LINS5:   .ASCIZ  /HEADER FROM ERROR SECTOR= /
     26 074006      122      115      105  LINEP5:  .ASCIZ  /RMEC1= /
     27 074016      040      040      122  LINEO5:  .ASCIZ  /  RMEC2= /
     28 074030      123      105      103  LINB6:   .ASCIZ  /SECTOR IS ECC CORRECTABLE /
     29 074063      123      105      103  LINC6:   .ASCIZ  /SECTOR READ CORRECTLY /
     30 074112      103      117      122  LING6:   .ASCIZ  /CORRECTED ON /
     31 074130      040      040      122  LINR6:   .ASCIZ  /  RETRIES/
     32 074142      125      116      103  LINU06:  .ASCIZ  /UNCORRECTABLE AFTER /
     33 074167      040      040      124  LIN7M:   .ASCIZ  /  TOTAL MISPOS ERR= /
     37 074214      124      117      124  LIN7P:   .ASCIZ  /TOTAL SEEKS= /
     38 074232      040      040      124  LIN7S:   .ASCIZ  /  TOTAL SKI ERR= /
     39 074254      124      117      124  LIN7T:   .ASCIZ  /TOTAL ERRORS:/
     40 074272      040      040      127  LIN70X:  .ASCIZ  /  WOFL:/
     41 074302      040      127      122  LIN7X:   .ASCIZ  / WRDS WRITN:/
     42 074317      040      040      122  LIN70R:  .ASCIZ  /  ROFL:/
     43 074327      040      127      122  LIN7R:   .ASCIZ  / WRDS READ:/
     44 074343      105      122      122  LIN8M:   .ASCIZ  /ERROR DURING RETRY/
     45 074366      104      101      124  LIN9B:   .ASCIZ  /DATA COMPARISON ERRORS/
     46 074415      040      040      040  LIN9H:   .ASCII  /          EXPCTD  RECEVD/<CRLF>
     47 074444      114      117      103           .ASCIZ  /LOC       DATA    DATA/<CRLF>
     48 074473      040      040      040  LIN9I:   .ASCII  /          RECEVD/<CRLF>
     49 074512      114      117      103           .ASCIZ  /LOC       DATA/<CRLF>
     50 074531      124      117      124  LIN9E:   .ASCIZ  /TOTAL COMPARE ERRORS= /
     51 074560      124      110      105  LIN9G:   .ASCIZ  /THE DATA COMPARED OK/<CRLF>
     52 074606      105      122      122  LIN10A:  .ASCIZ  /ERROR BURST BEGINS AT WORD /
     53 074642      040      111      116  LIN10B:  .ASCIZ  / IN DATA FIELD OF ERROR SECTOR/<CRLF>
     54 074702      105      122      122  LIN10C:  .ASCII  /ERROR WAS NOT IN THE DATA READ - /<CRLF>
     55 074744      105      103      103           .ASCIZ  /ECC CORRECTION CAN'T BE PERFORMED/
     56 075006      105      103      103  LIN10H:  .ASCII  /ECC CORRECTION RESULTS/<CRLF>
     57 075035      040      040      040           .ASCII  /          BAD     CORRECTED /<CRLF>
     58 075070      101      104      104           .ASCIZ  /ADDR      DATA    DATA/<CRLF>
     59 075116      103      117      116  LIN11H:  .ASCIZ  /CONTENTS OF ERROR SECTOR (REPORTED ABOVE)/<CRLF>
     60 075171      101      104      104  LIN11:   .ASCIZ  /ADDR      HEADER/<CRLF>
```

ERROR MESSAGES

```
     61 075212      101      104      104  LIN11A: .ASCIZ  /ADDR     DATA/<CRLF>
     62 075231      040                    BLNKS4: .ASCII  / /
     63 075232      040                    BLNKS3: .ASCII  / /
     64 075233      040                    BLNKS2: .ASCII  / /
     65 075234      040      000           BLNKS1: .ASCIZ  / /
     66 075236      122      105      124  LINX5: .ASCIZ   /RETRIEVED BY A RDHD COMMAND/
```

TELETYPE MESSAGES

```
                                        .SBTTL  TELETYPE MESSAGES

075272  077     000             QUES:   .ASCIZ  /?/
075274  075     000             EQUAL:  .ASCIZ  /=/
075276  054     000             COMMA:  .ASCIZ  /,/
075300  055     000             DASH:   .ASCIZ  /-/
075302  011     000             TAB:    .ASCIZ  <11>
075304  104     122     111     UNTMSG: .ASCIZ  /DRIVE/
075312  040     117     106     UNTOFF: .ASCIZ  / OFFLINE/
075323  040     117     116     UNTON:  .ASCIZ  / ONLINE/
075333  040     116     117     UNTNOT: .ASCIZ  / NOT BEING TESTED/
075355  040     101     114     UNTASN: .ASCIZ  / ALREADY BEING TESTED/
075403  040     116     117     NOTRM:  .ASCIZ  @ NOT AN RM80@
075420  040     116     117     NOTPRS: .ASCIZ  / NOT PRESENT/
075435  040     116     117     NOTAVL: .ASCIZ  / NOT AVAILABLE/
075454  040     125     116     NOTSAF: .ASCIZ  / UNSAFE/
075464  040     114     117     LODEV:  .ASCIZ  / LOAD DEVICE/
075501  200     104     122     SYSTAT: .ASCIZ  <CRLF>/DRIVE STATUS:/
075520  122     115     070     $RM80:  .ASCIZ  /RM80/
075525  200     012     052     REPHD:  .ASCIZ  <CRLF><LF>/***** PERFORMANCE REPORT *****/
075566  052     052     052     STAR30: .ASCII  /**************************/
075617  052     052     052     STAR5:  .ASCIZ  /*****/<CRLF>
075626  200     104     122     SUMHD:  .ASCIZ  <CRLF>/DRIVE SUMMARY,  (OFLOW= 2,147,483,647.)/<CRLF>
075700  007     077     106     DROPNG: .ASCIZ  <07>/?FATAL OR EXCESSIVE ERRORS/
075734  200     105     116     ENDPAS: .ASCIZ  <CRLF>/END OF PASS #/
075753  040     117     116     MSGON:  .ASCIZ  / ON /
075760  200     105     116     ENDTST: .ASCIZ  <CRLF>/END OF TEST /
075776  106     117     122     MSGFOR: .ASCIZ  /FOR /
076003  200     052     052     DEASSG: .ASCIZ  <CRLF>/***** DRIVE DEASSIGNED/<CRLF>
076034  200     052     052     DRNUM:  .ASCIZ  <CRLF>/***** DRIVE #/
076053  040     123     124     ASGND:  .ASCIZ  / STARTED/
076064  200     007     077     NEDCLK: .ASCIZ  <CRLF><07>/? 'L' OR 'P' CLOCK REQUIRED ON SYSTEM/<CRLF>
076135  116     000             N:      .ASCIZ  /N/
076137  131     000             Y:      .ASCIZ  /Y/
076141  056     000             PERIOD: .ASCIZ  /./
076143  040     077     103     MSWRO:  .ASCIZ  / ?CAN'T WRITE IN 'READ ONLY' MODE/<CRLF>
076206  040     077     111     INVLD:  .ASCIZ  / ?INVALID COMMAND/<CRLF>
076231  200     105     116     ENTCOM: .ASCIZ  <CRLF>/ENTER COMMAND: /
076252  200     105     116     ENTLMT: .ASCIZ  <CRLF>/ENTER ADDRESS LIMITS:/<CRLF>
076302  200     105     116     ENTADR: .ASCIZ  <CRLF>/ENTER BAD SECTR ADRS:/<CRLF>
076332  072     000             COLON:  .ASCIZ  /:/
076334  116     117     116     NONE:   .ASCIZ  /NONE/
076341  040     077     111     BADENT: .ASCIZ  / ?INVALID ENTRY/<CRLF>
076362  200     106     101     MERR1:  .ASCIZ  <CRLF>/FAILED TO RETRIEVE BAD SECTOR FILE(DEC144) ON /
076442  200     111     116     MERR2:  .ASCIZ  <CRLF>/INVALID FILE(DEC144) STRUCTURE ON /
076506  052     040     102     MSFULL: .ASCIZ  /* BAD SECTOR TABLE IS FULL */<CRLF>
076544  103     131     114     MSGCTS: .ASCIZ  /CYL,TRK,SEC= /
076562  200     104     117     MESFE:  .ASCIZ  <CRLF>/DO YOU WISH TO EXERCISE ONLY FE CYLINDERS (L) Y ? /
076646  101     122     105     SURE:   .ASCIZ  /ARE YOU SURE (L) N ? /
076674  200     052     040     FEONLY: .ASCIZ  <CRLF>/* EXERCISER WILL OPERATE ON FE CYLINDERS ONLY */<CRLF>
076756  007     007             OVRWRT: .ASCII  <07><07>
076760  200     011     041             .ASCII  <CRLF>/ ! CUSTOMER DATA WILL BE OVERWRITTEN !/
077027  200     011     055             .ASCII  <CRLF>/ ----------------------------------/
077076  007     007     200             .ASCIZ  <07><07><CRLF><LF>
077103  200     012     116     NODRVS: .ASCIZ  <CRLF><LF>/NO DRIVES ASSIGNED/<CRLF>
077131  200     120     122     MREAD:  .ASCIZ  <CRLF>/PROGRAM LOCKED IN "READ ONLY" MODE/<CRLF>
077176  052     040     116     NOENTY: .ASCIZ  /* NO ENTRIES */<CRLF>
```

TELETYPE MESSAGES

```
     58 077216      200      104      105  LSTHDR: .ASCIZ  <CRLF>/DEC144 AND MANUAL BAD SECTOR LIST/<CRLF>
     59 077262      052      040      101  ALOST:  .ASCIZ  /* ALL CURRENT ENTRIES LOST */<CRLF>
     60 077320      110      104      101  HDASN:  .ASCIZ  @HDA S/N: @
     61 077332      104      122      126  DRVSN:  .ASCIZ  @DRV S/N: @
     62 077344      200      103      110  MSPRM:  .ASCIZ  <CRLF>/CHANGE DRIVE PARAMETERS (L) N ? /
     63 077406      200      124      131  MSWAIT: .ASCII  <CRLF>/TYPE ^C TO ABORT/
     64 077427      200      127      101          .ASCIZ  <CRLF>/WAITING 2 MINUTES...TO START/<CRLF>
     65                                    .EVEN
```

```
                                        ;PARAMETER ENTRY TABLE

 077466  077632  017400  001462  PARLST: .WORD   PAR1,7936.,WRDCNT
 077474  077642  077777  001464          .WORD   PAR2,32767.,INTRVL
 077502  100012  077777  001470          .WORD   PAR19,32767.,PASSES
 077510  077652  000017  001472          .WORD   PAR3,15.,PATTERN
 077516  077752  000001  001474          .WORD   PAR11,1,RANDWC
 077524  077762  000007  001476          .WORD   PAR14,7,RATIO
 077532  100002  000001  001500          .WORD   PAR16,1,ENDING
 077540  077772  000001  001502          .WORD   PAR15,1,WRTCHK
 077546  100022  000001  001504          .WORD   PAR20,1,MESSAGE
 077554  100032  000001  001506          .WORD   PAR21,1,RANDOM
 077562  077742  000001  001510          .WORD   PAR10,1,BADBLK
 077570  000000                          .WORD   0                       ;TABLE TERMINATOR

     077572      040     057     040  SLASH:  .ASCIZ  @ / @
     077576      200     103     110  ASKPAR: .ASCIZ  <CRLF>/CHANGE PARAMETERS (L) N ? /
     077632      127     122     104  PAR1:   .ASCIZ  /WRDCNT /
     077642      111     116     124  PAR2:   .ASCIZ  /INTRVL /
     077652      120     101     124  PAR3:   .ASCIZ  /PATTERN/
     077662      115     101     130  PAR4:   .ASCIZ  /MAXCYL /
     077672      115     111     116  PAR5:   .ASCIZ  /MINCYL /
     077702      115     101     130  PAR6:   .ASCIZ  /MAXTRK /
     077712      115     111     116  PAR7:   .ASCIZ  /MINTRK /
     077722      115     101     130  PAR8:   .ASCIZ  /MAXSEC /
     077732      115     111     116  PAR9:   .ASCIZ  /MINSEC /
     077742      102     101     104  PAR10:  .ASCIZ  /BADBLK /
     077752      122     101     116  PAR11:  .ASCIZ  /RANDWC /
     077762      122     101     124  PAR14:  .ASCIZ  /RATIO  /
     077772      127     122     124  PAR15:  .ASCIZ  /WRTCHK /
     100002      105     116     104  PAR16:  .ASCIZ  /ENDING /
     100012      120     101     123  PAR19:  .ASCIZ  /PASSES /
     100022      115     105     123  PAR20:  .ASCIZ  /MESSAGE/
     100032      122     101     116  PAR21:  .ASCIZ  /RANDOM /

                                        .EVEN

                                        ;PARAMETER TABLE POINTERS FOR ADDRESS LIMITS
 100042  100062                         TABLE:  .WORD   TABLE0           ;PARAMETER TABLE FOR DRIVE 0
 100044  100130                                 .WORD   TABLE1           ;PARAMETER TABLE FOR DRIVE 1
 100046  100176                                 .WORD   TABLE2           ;PARAMETER TABLE FOR DRIVE 2
 100050  100244                                 .WORD   TABLE3           ;PARAMETER TABLE FOR DRIVE 3
 100052  100312                                 .WORD   TABLE4           ;PARAMETER TABLE FOR DRIVE 4
 100054  100360                                 .WORD   TABLE5           ;PARAMETER TABLE FOR DRIVE 5
 100056  100426                                 .WORD   TABLE6           ;PARAMETER TABLE FOR DRIVE 6
 100060  100474                                 .WORD   TABLE7           ;PARAMETER TABLE FOR DRIVE 7

                                        ;PARAMETER TABLE FOR ADDRESS LIMITS
 100062  077672  000000  045546  TABLE0: .WORD   PAR5,0,MINCYL+DRIVE0
 100070  077662  000000  045544          .WORD   PAR4,0,MAXCYL+DRIVE0
 100076  077712  000000  045552          .WORD   PAR7,0,MINTRK+DRIVE0
 100104  077702  000000  045550          .WORD   PAR6,0,MAXTRK+DRIVE0
 100112  077732  000036  045556          .WORD   PAR9,30.,MINSEC+DRIVE0
 100120  077722  000036  045554          .WORD   PAR8,30.,MAXSEC+DRIVE0
 100126  000000                          .WORD   0                       ;TERMINATOR

 100130  077672  000000  047762  TABLE1: .WORD   PAR5,0,MINCYL+DRIVE1
```

TELETYPE MESSAGES

```
100136  077662  000000  047760          .WORD   PAR4,0,MAXCYL+DRIVE1
100144  077712  000000  047766          .WORD   PAR7,0,MINTRK+DRIVE1
100152  077702  000000  047764          .WORD   PAR6,0,MAXTRK+DRIVE1
100160  077732  000036  047772          .WORD   PAR9,30.,MINSEC+DRIVE1
100166  077722  000036  047770          .WORD   PAR8,30.,MAXSEC+DRIVE1
100174  000000                          .WORD   0                       ;TERMINATOR

100176  077672  000000  052176  TABLE2: .WORD   PAR5,0,MINCYL+DRIVE2
100204  077662  000000  052174          .WORD   PAR4,0,MAXCYL+DRIVE2
100212  077712  000000  052202          .WORD   PAR7,0,MINTRK+DRIVE2
100220  077702  000000  052200          .WORD   PAR6,0,MAXTRK+DRIVE2
100226  077732  000036  052206          .WORD   PAR9,30.,MINSEC+DRIVE2
100234  077722  000036  052204          .WORD   PAR8,30.,MAXSEC+DRIVE2
100242  000000                          .WORD   0                       ;TERMINATOR

100244  077672  000000  054412  TABLE3: .WORD   PAR5,0,MINCYL+DRIVE3
100252  077662  000000  054410          .WORD   PAR4,0,MAXCYL+DRIVE3
100260  077712  000000  054416          .WORD   PAR7,0,MINTRK+DRIVE3
100266  077702  000000  054414          .WORD   PAR6,0,MAXTRK+DRIVE3
100274  077732  000036  054422          .WORD   PAR9,30.,MINSEC+DRIVE3
100302  077722  000036  054420          .WORD   PAR8,30.,MAXSEC+DRIVE3
100310  000000                          .WORD   0                       ;TERMINATOR

100312  077672  000000  056626  TABLE4: .WORD   PAR5,0,MINCYL+DRIVE4
100320  077662  000000  056624          .WORD   PAR4,0,MAXCYL+DRIVE4
100326  077712  000000  056632          .WORD   PAR7,0,MINTRK+DRIVE4
100334  077702  000000  056630          .WORD   PAR6,0,MAXTRK+DRIVE4
100342  077732  000036  056636          .WORD   PAR9,30.,MINSEC+DRIVE4
100350  077722  000036  056634          .WORD   PAR8,30.,MAXSEC+DRIVE4
100356  000000                          .WORD   0                       ;TERMINATOR

100360  077672  000000  061042  TABLE5: .WORD   PAR5,0,MINCYL+DRIVE5
100366  077662  000000  061040          .WORD   PAR4,0,MAXCYL+DRIVE5
100374  077712  000000  061046          .WORD   PAR7,0,MINTRK+DRIVE5
100402  077702  000000  061044          .WORD   PAR6,0,MAXTRK+DRIVE5
100410  077732  000036  061052          .WORD   PAR9,30.,MINSEC+DRIVE5
100416  077722  000036  061050          .WORD   PAR8,30.,MAXSEC+DRIVE5
100424  000000                          .WORD   0                       ;TERMINATOR

100426  077672  000000  063256  TABLE6: .WORD   PAR5,0,MINCYL+DRIVE6
100434  077662  000000  063254          .WORD   PAR4,0,MAXCYL+DRIVE6
100442  077712  000000  063262          .WORD   PAR7,0,MINTRK+DRIVE6
100450  077702  000000  063260          .WORD   PAR6,0,MAXTRK+DRIVE6
100456  077732  000036  063266          .WORD   PAR9,30.,MINSEC+DRIVE6
100464  077722  000036  063264          .WORD   PAR8,30.,MAXSEC+DRIVE6
100472  000000                          .WORD   0                       ;TERMINATOR

100474  077672  000000  065472  TABLE7: .WORD   PAR5,0,MINCYL+DRIVE7
100502  077662  000000  065470          .WORD   PAR4,0,MAXCYL+DRIVE7
100510  077712  000000  065476          .WORD   PAR7,0,MINTRK+DRIVE7
100516  077702  000000  065474          .WORD   PAR6,0,MAXTRK+DRIVE7
100524  077732  000036  065502          .WORD   PAR9,30.,MINSEC+DRIVE7
100532  077722  000036  065500          .WORD   PAR8,30.,MAXSEC+DRIVE7
100540  000000                          .WORD   0                       ;TERMINATOR
```

```
     1                                  .SBTTL  ROUTINE TO SIZE MEMORY

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

 100542  010046                         $SIZE:  MOV     R0,-(SP)        ;;SAVE R0 ON THE STACK
 100544  010146                                 MOV     R1,-(SP)        ;;SAVE R1 ON THE STACK
 100546  013746  000114                         MOV     @#114,-(SP)     ;;SAVE MEMORY ERROR VECTOR PS & PC
 100552  013746  000116                         MOV     @#116,-(SP)
 100556  012737  000116  000114                 MOV     #116,@#114      ;;IGNORE PARITY ERRORS WHILE SIZING
 100564  012737  000002  000116                 MOV     #RTI,@#116
 100570  013746  000004                         MOV     @#ERRVEC,-(SP)  ;;SAVE PRESENT ERROR VECTOR PS & PC
 100576  013746  000006                         MOV     @#ERRVEC+2,-(SP)
 100602  010600                                 MOV     SP,R0           ;;SAVE THE STACK POINTER
                                        ;;SET THE ERRVEC PS TO THE PRESENT PS
 100604  104400                                 TRAP                    ;;PUSH OLD PSW AND PC ON STACK
 100606  012637  000006                         MOV     (SP)+,@#ERRVEC+2        ;;SAVE THE PSW IN @#ERRVEC+2
 100612  012737  100632  000004                 MOV     #2$,@#ERRVEC    ;;SET FOR TIMEOUT
 100620  012701  020000                         MOV     #20000,R1       ;;FIRST ADDRESS
 100624  005711                         1$:     TST     (R1)            ;;TEST THIS ADDRESS
 100626  005721                                 TST     (R1)+           ;;STEP TO NEXT ADDRESS
 100630  000775                                 BR      1$              ;;TRY ANOTHER
 100632  162701  000002                 2$:     SUB     #2,R1           ;;DROP BACK
 100636  010006                                 MOV     R0,SP           ;;RESTORE THE STACK
 100640  012637  000006                         MOV     (SP)+,@#ERRVEC+2 ;;RESTORE ERROR VECTOR
 100644  012637  000004                         MOV     (SP)+,@#ERRVEC
 100650  012637  000116                         MOV     (SP)+,@#116     ;;RESTORE MEMORY ERROR VECTOR
 100654  012637  000114                         MOV     (SP)+,@#114
 100660  010137  100672                         MOV     R1,$LSTAD       ;;LAST ADDRESS
 100664  012601                                 MOV     (SP)+,R1        ;;RESTORE R1
 100666  012600                                 MOV     (SP)+,R0        ;;RESTORE R0
 100670  000207                                 RTS     PC
 100672  000000                 $LSTAD: .WORD   0               ;;CONTAINS THE LAST ADDRESS
```

```
                                .SBTTL  BUSADR - GET BUS ADDRESS AND VECTOR ADDRESS
 2
 3                              ;THIS ROUTINE IS USED TO INSURE THE BUS ADDRESS OF THE RH/RM
 4                              ;IS SETUP FOR THE PROPER ADDRESS. IT WILL ALSO READ THE ADDRESS
 5                              ;FROM THE TTY IF REQUIRED.
 6                              ;
 7                              ;NOTE: THIS ROUTINE DESTROYS R0-R4
 8                              ;CALL:
 9                              ;
10                              ;       JSR     PC,BUSADR
11                              ;       RETURN
12
13 100674  005737  001336       BUSADR: TST     CHGADR          ;INPUT FROM TTY REQUESTED?
14 100700  002053                       BGE     3$              ;NO--BRANCH
15 100702  005037  001336               CLR     CHGADR          ;YES--CLEAR THE REQUEST FLAG
16 100706  104401  001203               TYPE    ,$CRLF          ;TYPE A CR-LF
17 100712  005227  177777               INC     #-1             ;FIRST TIME THRU ?
18 100716  001402                       BEQ     1$              ;BR IF YES
19 100720  104401  001203               TYPE    ,$CRLF          ;CR-LF
20 100724  005037  001340       1$:     CLR     CFLAG           ;CLEAR CONTROL C FLAG
21 100730  012700  001272               MOV     #$RMADR,R0      ;FIRST ADDRESS
22 100734  104401  101046               TYPE    ,MRMCS1         ;'RMCS1='
23 100740  011046                       MOV     (R0),-(SP)      ;PRESENT RMCS1 ADDRESS
24 100742  104402                       TYPOC                   ;TYPE IT
25 100744  104401  075233               TYPE    ,BLNKS2         ;TYPE 2 BLANKS
26 100750  104411                       RDLIN                   ;GET THE ENTRY
27 100752  012601                       MOV     (SP)+,R1        ;ADDRESS OF ASCII TEXT
28 100754  005737  001340               TST     CFLAG           ;WAS IT CONTROL C ?
29 100760  001361                       BNE     1$              ;BR IF YES
30 100762  004537  101064               JSR     R5,CK.NUM       ;ENTER AND STORE THE NEW ADDRESS
31 100766  000756                       BR      1$              ;ERROR EXIT
32 100770  012700  001274       2$:     MOV     #$RMVEC,R0      ;VECTOR ADDRESS
33 100774  104401  101055               TYPE    ,MRMVEC         ;'RMVEC='
34 101000  011046                       MOV     (R0),-(SP)      ;PRESENT RH/RM VECTOR ADDRESS ON THE STACK
35 101002  104402                       TYPOC                   ;TYPE IT
36 101004  104401  075233               TYPE    ,BLNKS2         ;TYPE 2 BLANKS
37 101010  104411                       RDLIN                   ;READ THE ENTRY
38 101012  012601                       MOV     (SP)+,R1        ;ASCII TEXT ADDRESS
39 101014  005737  001340               TST     CFLAG           ;WAS IT CONTROL C ?
40 101020  001341                       BNE     1$              ;BR IF YES
41 101022  004537  101064               JSR     R5,CK.NUM       ;ENTER AND STORE NEW ADDRESS
42 101026  000760                       BR      2$              ;ERROR EXIT
43 101030  012700  001272       3$:     MOV     #$RMADR,R0      ;FIRST ADDRESS OF NEW PARAMETERS
44 101034  012701  040310               MOV     #RMADR,R1       ;FIRST ADDRESS OF WHERE TO PUT THEM
45 101040  012021                       MOV     (R0)+,(R1)+     ;BUS ADDRESS
46 101042  012021                       MOV     (R0)+,(R1)+     ;VECTOR ADDRESS
47 101044  000207                       RTS     PC              ;RETURN
48
49 101046     122     115     103  MRMCS1: .ASCIZ  @RMCS1=@
50 101055     122     115     126  MRMVEC: .ASCIZ  @RMVEC=@
```

CK.NUM - CHECK NUMBER (OCTAL)

```
                                        .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

101064  010246                 CK.NUM: MOV     R2,-(SP)        ;SAVE R2
101066  010346                         MOV     R3,-(SP)        ;SAVE R3
101070  010446                         MOV     R4,-(SP)        ;SAVE R4
101072  012703  000006                 MOV     #6,R3           ;MAX OCTAL DIGITS IN THE NUMBER
101076  005002                         CLR     R2              ;FINAL OCTAL VALUE
101100  112104                 1$:     MOVB    (R1)+,R4        ;GET CURRENT POINTED BYTE
101102  001424                         BEQ     3$              ;BRANCH,IF TERMINATOR DETECTED
101104  120427  000060                 CMPB    R4,#'0          ;SMALLER THAN ASCII-0 ?
101110  103425                         BLO     5$              ;YES,ERROR EXIT
101112  120427  000067                 CMPB    R4,#'7          ;LARGER THAN ASCII-7 ?
101116  101022                         BHI     5$              ;YES,ERROR EXIT
101120  006302                         ASL     R2              ;SHIFT LEFT
101122  103420                         BCS     5$              ;
101124  006302                         ASL     R2              ;ONE
101126  103416                         BCS     5$              ;
101130  006302                         ASL     R2              ;OCTAL DIGIT
101132  103414                         BCS     5$              ;ERROR IF CARRY BIT SET
101134  042704  177770                 BIC     #177770,R4      ;CHOP OFF HIGHER BITS
101140  060402                         ADD     R4,R2           ;APPENDING CURRENT DIGIT TO NUMBER
101142  005303                         DEC     R3              ;DECREMENT BYTE COUNT
101144  001401                         BEQ     2$              ;BRANCH,IF LAST BYTE
101146  000754                         BR      1$              ;LOOPING BACK
101150  112104                 2$:     MOVB    (R1)+,R4        ;CHECK TERMINATOR
101152  001004                         BNE     5$              ;ERROR EXIT
101154  005702                 3$:     TST     R2              ;FINAL VALUE= 0
101156  001401                         BEQ     4$              ;YES,THEN NOT REPLACE THE ORIGINAL VALUE
101160  010210                         MOV     R2,(R0)         ;REPLACE THE ORIGINAL VALUE
101162  005725                 4$:     TST     (R5)+           ;ADJUST FOR NORMAL RETURN
101164  012604                 5$:     MOV     (SP)+,R4        ;RESTORE R4
101166  012603                         MOV     (SP)+,R3        ;RESTORE R3
101170  012602                         MOV     (SP)+,R2        ;RESTORE R2
101172  000205                         RTS     R5              ;EXIT

101174                          CYLNDR: .BLKW   258.            ;ONE SECTOR WORD CTR MAX SIZE

        102200                  ENDPGM=.

        000200                  .END    200
```

SYMBOL TABLE

```
ABASE = 176700        ASGN6   026302        BIT5  = 000040        COMMA   075276        DRIVE4  056476
ABNRML  031264        ASGN7   026314        BIT6  = 000100        COMTBL  002076        DRIVE5  060712
ACDW1 = 000000        ASKPAR  077576        BIT7  = 000200        CPSAVE  035224        DRIVE6  063126
ACDW2 = 000000        ASNERR  031154        BIT8  = 000400        CR    = 000015        DRIVE7  065342
ACK   = 000123        ASNLST  001542        BIT9  = 001000        CRLF  = 000200        DRNUM   076034
ACPUOP= 000000        ASNMSG  031200        BLKADR  002056        CTRAP   034574        DROP    031204
ACTDRV  040250        ASSIGN  025666        BLNKS1  075234        CYLIMT  001422        DROPNG  075700
ACTSTR  040251        ASWREG= 000000        BLNKS2  075233        CYLNDR  101174        DRQ   = 004000
ADDW0 = 000000        ATA   = 100000        BLNKS3  075232        DASH    075300        DRVACT  040154
ADDW1 = 000000        ATABIT  040300        BLNKS4  075231        DATAPK  026600        DRVCLR= 000111
ADDW10= 000000        ATESTN= 000000        BPTVEC= 000014        DATA0   002370        DRVER   011200
ADDW11= 000000        ATTN    001322        BSE   = 100000        DATA1   002430        DRVINT  040534
ADDW12= 000000        AT0   = 000001        BUFTBL  001654        DATA10  003070        DRVNO   001324
ADDW13= 000000        AT1   = 000002        BUSADR  100674        DATA11  003130        DRVPAR  001442
ADDW14= 000000        AT2   = 000004        CFLAG   001340        DATA12  003170        DRVPRM  027054
ADDW15= 000000        AT3   = 000010        CHGADR  001336        DATA13  003230        DRVQUE  045344
ADDW2 = 000000        AT4   = 000020        CHKWC   020330        DATA14  003270        DRVSN   077332
ADDW3 = 000000        AT5   = 000040        CI1     041516        DATA15  003330        DRVSTA  040164
ADDW4 = 000000        AT6   = 000100        CI3     041650        DATA2   002470        DRVTYP  040174
ADDW5 = 000000        AT7   = 000200        CI4     041756        DATA3   002530        DRY   = 000200
ADDW6 = 000000        AUNIT = 000000        CI5     042312        DATA4   002570        DSWR  = 177570
ADDW7 = 000000        AUSWR = 000000        CI8     042334        DATA5   002630        DTE   = 010000
ADDW8 = 000000        AUTLST  032012        CKBUS   013154        DATA6   002670        DTEER   011772
ADDW9 = 000000        AVAIL   001610        CKCLK   023422        DATA7   002730        DTUW    040276
ADEVCT= 000000        AVECT1= 000254        CKCLK1  023516        DATA8   002770        DT00  = 000001
ADEVM = 000000        AVECT2= 000000        CKCLK2  023566        DATA9   003030        DT01  = 000002
AENV  = 000000        A16   = 000400        CKCLK3  023620        DCK   = 100000        DT02  = 000004
AENVM = 000000        A17   = 001000        CKERR   013062        DCKER   010072        DT03  = 000010
AFATAL= 000000        BADBLK  001510        CKFMT   011232        DCKER1  010254        DT04  = 000020
ALOST   077262        BADENT  076341        CKHCE   011406        DDISP = 177570        DT05  = 000040
AMADR1= 000000        BADSEC  001342        CKLMTS  020130        DDRVS   001544        DT06  = 000100
AMADR2= 000000        BADTMO  003450        CKSWR = 104407        DEASGN  026356        DT07  = 000200
AMADR3= 000000        BAI   = 000010        CK.CHR  033020        DEASSG  076003        DT08  = 000400
AMADR4= 000000        BEGCOD  001514        CK.DEC  032772        DEC2    001436        DT1     073134
AMAMS1= 000000        BEGPAT  001512        CK.DIG  033072        DH1     072467        DT14    073204
AMAMS2= 000000        BEGWC   001516        CK.NUM  101064        DH14    072644        DT15    073226
AMAMS3= 000000        BIT0  = 000001        CK.OCT  032744        DH15    072746        DT16    073250
AMAMS4= 000000        BIT00 = 000001        CLKFLG  001312        DH16    073044        DT17    073264
AMSGAD= 000000        BIT01 = 000002        CLOCK   025110        DH17    073114        DT2     073140
AMSGLG= 000000        BIT02 = 000004        CLR   = 000040        DH2     072474        DT3     073156
AMSGTY= 000000        BIT03 = 000010        CLRDPB  026630        DH3     072550        DT4     073166
AMTYP1= 000000        BIT04 = 000020        CLRQUE  045300        DH4     072576        DT6     073200
AMTYP2= 000000        BIT05 = 000040        CMCNT   001370        DH6     072635        DVA   = 004000
AMTYP3= 000000        BIT06 = 000100        CMCYL   001372        DISPLA  001156        DVC   = 000200
AMTYP4= 000000        BIT07 = 000200        CMDAT   013500        DISPLY= 104414        ECBAD0  001412
AOE   = 001000        BIT08 = 000400        CMHED   013424        DISPRE  000174        ECBAD1  001420
APASS = 000000        BIT09 = 001000        CMPAR   013240        DLT   = 100000        ECBIT   001376
APRIOR= 000000        BIT1  = 000002        CMPARD  013244        DONE    007550        ECC     014472
APTCSU= 000040        BIT10 = 002000        CMPLMT  001460        DPE   = 000010        ECCX    015230
APTENV= 000001        BIT11 = 004000        CMPRES  020750        DPINT   040204        ECC1    015074
APTSIZ= 000200        BIT12 = 010000        CMPRT   014060        DPR   = 000400        ECC2    015224
APTSPO= 000100        BIT13 = 020000        CMPRX   014052        DPRQS   040214        ECGD    001410
ASGND   076053        BIT14 = 040000        CMSEC   001374        DRIVE = 001220        ECGD1   001416
ASGN1   025764        BIT15 = 100000        CMSTR   013342        DRIVE0  045416        ECH   = 000100
ASGN2   026040        BIT2  = 000004        CMSTR2  013466        DRIVE1  047632        ECI   = 004000
ASGN3   026130        BIT3  = 000010        CMTRK   001375        DRIVE2  052046        ECMSK0  001402
ASGN4   026300        BIT4  = 000020        COLON   076332        DRIVE3  054262        ECMSK1  001404
```

SYMBOL TABLE

```
ECSEC   001400        EOP2    031370        INCMIS  024714        LINR6   074130        MDPE   = 000400
ECWRD   001406        EQUAL   075274        INCSKI  024670        LINSS3  073600        MERR1    076362
ECWRD1  001414        ERCTR   001364        INCSOF  024620        LINST3  073564        MERR2    076442
EMPTYQ  045332        ERPRC1  007356        INCTOT  024740        LINS3   073445        MESFE    076562
EMTVEC= 000030        ERPROC  007342        INTRVL  001464        LINS4   073632        MESSAG   001504
EM1     067646        ERR    = 040000        INVLD   076206        LINS5   073753        MINCYL=  000130
EM10    070160        ERROR  = 104000        IOTVEC= 000020        LINT3   073514        MINSEC=  000140
EM11    070223        ERRVEC= 000004        IP     = 000100        LINU06  074142        MINTRK=  000134
EM12    070256        FACTOR  016240        ISR     042452        LINW3   073553        MINUTE   001346
EM13    070307        FAIRNS  001332        IVC    = 010000        LINX4   073646        MNTBL    002124
EM14    070361        FEFLAG  001434        KSR     025254        LINX5   075236        MOH    = 020000
EM15    070404        FEONLY  076674        KSR1    025262        LIN10A  074606        MOL    = 010000
EM2     067720        FER    = 000020        LBC    = 002000        LIN10B  074642        MREAD    077131
EM20    070440        FE1     001430        LBT    = 002000        LIN10C  074702        MRMCS1   101046
EM21    070461        FE2     001432        LF     = 000012        LIN10H  075006        MRMVEC   101055
EM22    070512        FILBUF  016632        LIMIT   001366        LIN11   075171        MSFULL   076506
EM23    070565        FILLZ   032464        LINA3   073475        LIN11A  075212        MSGCTS   076544
EM24    070644        FILLO   032572        LINB3   073544        LIN11H  075116        MSGFOR   075776
EM25    070712        FMTER   012166        LINB5   073716        LIN2C   073272        MSGON    075753
EM26    070773        FMT16  = 010000        LINB6   074030        LIN2P   073312        MSPRM    077344
EM27    071020        FRSTER  001356        LINCA3  073524        LIN2S   073333        MSWAIT   077406
EM3     067756        F0     = 000002        LINC6   074063        LIN3.1  022120        MSWRO    076143
EM30    071055        F1     = 000004        LINDA3  073535        LIN3.3  022226        MXF    = 001000
EM31    071107        F2     = 000010        LINDEC  023234        LIN3.4  022260        MXWNDW   040320
EM32    071152        F3     = 000020        LIND5   073700        LIN6.2  022722        M.DPID   032106
EM33    071205        F4     = 000040        LINEN3  073461        LIN7M   074167        M.DP40   032144
EM34    071262        GENDPB  067556        LINEO5  074016        LIN7OR  074317        M.DP41   032200
EM35    071324        GENPAR  017370        LINEP5  074006        LIN7OX  074272        M.DP42   032210
EM36    071362        GENRCG  067576        LINE1   021134        LIN7P   074214        M.DP44   032242
EM37    071413        GETADR  027650        LINE2   021214        LIN7R   074327        M.DP50   032254
EM4     070014        GETBUF  016242        LINE2A  021364        LIN7S   074232        N        076135
EM40    071475        GETID   027550        LINE2B  021402        LIN7T   074254        NED    = 010000
EM41    071544        GETLMT  027462        LINE3   021654        LIN7X   074302        NEDCLK   076064
EM42    071610        GETPAT  020070        LINE3A  021662        LIN8M   074343        NEM    = 004000
EM43    071671        GETREG= 000141        LINE3B  021670        LIN9B   074366        NEWASN   026556
EM44    071765        GETREM  032014        LINE3C  021702        LIN9E   074531        NEWUNT   001566
EM45    072062        GETREQ  045366        LINE3D  021712        LIN9G   074560        NODRVS   077103
EM46    072150        GO     = 000001        LINE3E  021760        LIN9H   074415        NOENTY   077176
EM47    072222        GODRIV  016710        LINE3F  022046        LIN9I   074473        NOMTCH   012646
EM5     070051        GTSWR  = 104406        LINE4   022324        LKPAR   005232        NONE     076334
EM50    072277        HCE    = 000200        LINE5   022414        LODEV   075464        NOTAVL   075435
EM51    072335        HCEER   012230        LINE5A  022556        LODPAR  020500        NOTPRS   075420
EM52    072400        HCI    = 002000        LINE5B  022624        LSC    = 004000        NOTRM    075403
EM6     070105        HCRC   = 000400        LINE6   022666        LSTAD   001334        NOTSAF   075454
EM60    072444        HCRCER  011054        LINE6A  022700        LSTHDR  077216        NSA    = 100000
ENDCMP  014334        HDASN   077320        LINE6C  022706        MAIN    006240        OFFDIR=  000001
ENDCON  001446        HOUR    001344        LINE6D  022714        MAINLA  006266        OFFON  = 000001
ENDING  001500        HT     = 000011        LINE7   022746        MAIN1   006404        OFLIN    007446
ENDPAS  075734        HZ      001314        LINE7A  023076        MAIN2   006522        ONES     003272
ENDPGM= 102200        IAE    = 002000        LINE8   023170        MANTER  030254        ONESEC   001352
ENDSEK  001452        IAEER   012106        LING6   074112        MASK    001326        OPE    = 020000
ENDTST  075760        IBSAVE  035226        LINM3   073367        MATCH   014402        OPI    = 020000
ENTADR  076302        IDLE    007104        LINM4   073614        MAXCYL=  000126        OPIER    011666
ENTCOM  076231        IE     = 000100        LINN3   073405        MAXER   001456        OPIER1   011732
ENTLMT  076252        ILF    = 000001        LINOCT  023202        MAXSEC=  000136        OPT      041236
ENTPR   005542        ILR    = 000002        LINP3   073426        MAXTRK=  000132        OPTBL    002104
EOP1    031344        INCHRD  024644        LINP5   073736        MCPE   = 020000        OR     = 000200
```

SYMBOL TABLE

```
ORDERQ  001520        PWRFLG  037716        RMHR   = 000036        SPOTCK  020764        SW5    = 000040
OVRWRT  076756        PWRVEC= 000024        RMINIT  040322        SRCHWT  040246        SW6    = 000100
PACK    001320        QDRV    040224        RMLA   = 000020        SSE    = 000040        SW7    = 000200
PAR    = 000010       QUES    075272        RMMR1  = 000024        SSEI   = 001000        SW8    = 000400
PARENT  031030        RANCYL  017600        RMMR2  = 000040        STA      006006        SW9    = 001000
PARER   012014        RANDOM  001506        RMOF   = 000032        STAC.  = 001100        SYSTAT  075501
PARLST  077466        RANDWC  001474        RMR    = 000004        STAPT    003532        T       073402
PAR1    077632        RANPAT  020040        RMSN   = 000030        STAPT1   003542        TAB     075302
PAR10   077742        RANSEC  017670        RMTMR    044152        START2   003560        TABLE   100042
PAR11   077752        RANSIZ  017746        RMVEC    040312        STAR30   075566        TABLE0  100062
PAR14   077762        RANTRK  017634        RMWC   = 000002        STAR5    075617        TABLE1  100130
PAR15   077772        RANXIT  020060        RM80     041000        STATIN   001316        TABLE2  100176
PAR16   100002        RATIO   001476        RNOP   = 000101        STATIS   016072        TABLE3  100244
PAR19   100012        RDCHR = 104410        RTC    = 000117        STATPR   023626        TABLE4  100312
PAR2    077642        RDDAT = 000171        RTNCTR   015510        STKLMT= 177774        TABLE5  100360
PAR20   100022        RDHD   = 000173       RTURN    031766        STNDAT   002324        TABLE6  100426
PAR21   100032        RDLIN = 104411        R6     =%000006        STO      044244        TABLE7  100474
PAR3    077652        RDONLY  001440        R7     =%000007        STO1     044274        TAB.XY= 001114
PAR4    077662        RDY    = 000200       S        073423        STO2     044462        TAP    = 040000
PAR5    077672        RD.ADR  044706        SATPOW   037736        STO3     044532        TBITVE= 000014
PAR6    077702        RD.RM   044670        SAVEFG   040252        STO5     044556        TD      042514
PAR7    077712        RD.RM1  044704        SAVER1   001360        STO6     044564        THEAD   017514
PAR8    077722        RD.RM3  044742        SAVER5   001362        STO7     044622        TIMER   040256
PAR9    077732        RD.RM4  044746        SAVREG= 104412        STO8     044652        TKVEC  = 000060
PASSES  001470        RD.WRD  044710        SC       043334        STO9     044662        TPVEC  = 000064
PAT    = 000020       READDR  023254        SCMND    026464        SUMARY   023714        TRAPVE= 000034
PATTER  001472        READHD  01565?        SCOPE  = 000004        SUMHD    075626        TRE    = 040000
PCLOCK  001310        READIN= 00012?        SC04   = 000400        SUPRS    032300        TRFER   012272
PERFEX= 000001        RECAL  = 000107       SC1    = 000100        SUPRSL   032264        TRKLMT  001?66
PERIOD  076141        RECALT  015534        SC10   = 001000        SUPR1    032312        TRNSWT  040244
PFECH   035406        RECALO  015624        SC11     043672        SUPR2    032354        TRTVF?= 000014
PFECH1  035416        REDAPK  026566        SC12     043750        SURE     076646        TST     003550
PFECH2  035500        RELBUF  016376        SC13     044012        SVRH70   045062        TY?RV   032622
PFECH3  035532        RELSE  = 000113       SC2    = 000200        SWR      001154        TY?DA   032652
PFECH4  035542        REPHD   075525        SC20   = 002000        SWREG    000176        T?LIST  030650
PFTSTN  035546        REPLZ   032470        SC3      043400        SWTIM    007410        TYPDRV  032626
PGE    = 002000       RESREG= 104413        SC4      043404        SW0    = 000001        TYPDS  = 104405
PGM    = 001000       RESVEC= 000010        SC5      043416        SW00   = 000001        TYPE   = 104401
PIP    = 020000       RETRY   001330        SC6      043504        SW01   = 000002        TYPHDA  032656
PIRQ   = 177772       RHEXT   040316        SC8      043656        SW02   = 000004        TYPOC  = 104402
PIRQVE= 000240        RMADR   040310        SDETAL   023752        SW03   = 000010        TYPON  = 104404
POPQUE  045400        RMAS   = 000016       SEARCH= 000131        SW04   = 000020        TYPOS  = 104403
POSER   011614        RMBA   = 000004       SECLMT   001424        SW05   = 000040        TYPRI4  032574
PROCES  007252        RMBAE  = 000050       SECOND   001350        SW06   = 000100        UNS    = 040000
PRTBAD  015236        RMCS1  = 000000       SEEK   = 000105        SW07   = 000200        UNSAF   012532
PRTIM   007512        RMCS2  = 000010       SEEKFG   040254        SW08   = 000400        UNTASN  075355
PR0    = 000000       RMCS3  = 000052       SELDRV= 000145        SW09   = 001000        UNTMSG  075304
PR1    = 000040       RMDA   = 000006       SETFMT= 000143        SW1    = 000002        UNTNOT  075333
PR2    = 000100       RMDB   = 000022       SETVEC   005570        SW10   = 002000        UNTOFF  075312
PR3    = 000140       RMDC   = 000034       SET.IE   045226        SW11   = 004000        UNTON   075323
PR4    = 000200       RMDS   = 000012       SHDTYP   023?42        SW12   = 010000        UPE    = 020000
PR5    = 000240       RMDT   = 000026       SIZE70   004?56        SW13   = 020000        US1    = 000001
PR6    = 000300       RMEC1  = 000044       SIZMEM   005024        SW14   = 040000        US2    = 000002
PR7    = 000340       RMEC2  = 000046       SKI    = 040000        SW15   = 100000        US4    = 000004
PS     = 177776       RMEPRS  040146        SKIER    012372        SW2    = 000004        VV     = 000100
PSEL   = 002000       RMER1  = 000014       SKIP     042746        SW3    = 000010        WAIT    001632
PSW    = 177776       RMER2  = 000042       SLASH    077572        SW4    = 000020        WATPAK  026612
```

SYMBOL TABLE

```
WCE    = 040000        $DBLK   036546        $LKS    001304        $PWRUP  037610        $SUPR2  032454
WCF    = 000020        $DB2D   037222        $LLVEC  001306        $QUES   001202        $SVPC = 000210
WCFER    012434        $DB2O   037416        $LONUM  037124        $RAND   037024        $SWR  = 122000
WCHKX = 000000        $DECVL  037402        $LPADR  001122        $RDCHR  034170        $SWREG  001230
WCKD   = 000151        $DEVCT  001216        $LPERR  001124        $RDLIN  034260        $TATUS= 000016
WCKER    010546        $DEVM   001264        $LPVEC  001302        $RDOFL= 000064        $TERM = 000032
WCKHD = 000153        $DIV    032040        $LSTAD  100672        $RDSZ = 000017        $TESTN  001212
WC.HK    043246        $DOAGN  031762        $MADR1  001240        $READ = 000066        $TIME   024764
WLE    = 004000        $DRVSN= 002130        $MADR2  001244        $REG  = 000014        $TKB    001162
WLEER    012140        $DSPLY  032716        $MADR3  001250        $RESRE  037164        $TKCNT  033310
WRDCNT   001462        $DTBL   036536        $MADR4  001254        $RETRY  015744        $TKINT  033326
WRL    = 004000        $ENDAD  031752        $MAIL   001206        $RHEXT= 000001        $TKQEN= 033325
WRTCHK   001502        $ENDAT= 000036        $MAMS1  001236        $RMADR  001272        $TKQIN  033312
WRTDAT= 000161        $ENDCT  031736        $MAMS2  001242        $RMAS = 002156        $TKQOU  033314
WRTHD  = 000163        $ENDSK= 000042        $MAMS3  001246        $RMBA = 002144        $TKQSR  033316
WRTPK    017000        $ENV    001226        $MAMS4  001252        $RMBAE= 002210        $TKS    001160
WRT.AD   045032        $ENVM   001227        $MBADR  001102        $RMCS1= 002140        $TKSRV  033376
WRT.RM   044750        $EOP    031312        $MFLG   037020        $RMCS2= 002150        $TMPO   001174
WRT.R1   045026        $EOPCT  031730        $MISPO= 000102        $RMCS3= 002212        $TN   = 000002
WRT.R3   045052        $ERFLG  001117        $MNEW   034563        $RMDA = 002146        $TNPWR  037332
WRT.R4   045056        $ERMAX  001131        $MSGAD  001222        $RMDB = 002162        $TOTAL= 000072
WRT.R5   045060        $ERROR  034640        $MSGLG  001224        $RMDC = 002174        $TPB    001166
WRT.WD   045030        $ERRPC  001132        $MSGTY  001206        $RMDS = 002152        $TPFLG  001173
XXDP     001444        $ERRTB  003370        $MSWR   034552        $RMDT = 002166        $TPS    001164
Y        076137        $ERRTY  035230        $MTYP1  001237        $RMEC1= 002204        $TRAP   040060
ZEROS    002370        $ERTTL  001126        $MTYP2  001243        $RMEC2= 002206        $TRAP2  040102
ZROIND   001354        $ETABL  001226        $MTYP3  001247        $RMER1= 002154        $TRK  = 000011
$APTHD   001100        $ETEND  001272        $MTYP4  001253        $RMER2= 002202        $TRP  = 000015
$ATYC    036602        $FAIR = 000106        $NCODE= 000114        $RMHR = 002176        $TRPAD  040114
$ATY1    036556        $FATAL  001210        $NCYL = 000120        $RMLA = 002160        $TSTM   001104
$ATY3    036564        $FFLG   037022        $NEXT = 000122        $RMMR1= 002164        $TSTNM  001116
$ATY4    036574        $FILLC  001172        $NPATC= 000115        $RMMR2= 002200        $TTYIN  034514
$AUTOB   001150        $FILLS  001171        $NSEC = 000116        $RMOF = 002172        $TYPDS  036332
$BASE    001262        $FIRST= 000124        $NTRK = 000117        $RMSN = 002170        $TYPE   035550
$BDADR   001136        $FMT  = 000001        $NULL   001170        $RMVEC  001274        $TYPEC  035762
$BDDAT   001142        $GDADR  001134        $NWTST= 000000        $RMWC = 002142        $TYPEX  036102
$BDSEC= 000146        $GDDAT  001140        $OCNT   036326        $RM80   075520        $TYPOC  036130
$BELL    001176        $GET42  031742        $OCTVL  037520        $RTNAD  031764        $TYPON  036144
$BUF   = 000006        $GTSWR  033716        $OMODE  036330        $SAVRE  037126        $TYPOS  036104
$CDW1    001266        $HARD = 000076        $OPERC= 000046        $SAVR6  037714        $UNIT   001220
$CDW2    001270        $HD   = 000000        $PACK = 000026        $SB2D   033230        $UNITM  001110
$CHARC   036100        $HIBTS  001100        $PASS   001214        $SB2O   033260        $USWR   001232
$CKSWR   033626        $HINUM  037122        $PASSC= 000104        $SEC  = 000010        $VECT1  001256
$CMTAG   001114        $HLDWC= 000110        $PASTM  001106        $SETUP= 000156        $VECT2  001260
$CM3   = 000000        $HSNL = 000142        $PATTC= 000030        $SIZE   100542        $WCNT = 000004
$CM4   = 000001        $HSNM = 000144        $POSIT= 000052        $SKI  = 000100        $WRDL = 000020
$CNTLC   034533        $ICNT   001120        $POWER  037720        $SOFT = 000074        $WRITN= 000060
$CNTLG   034545        $ILLUP  037710        $PREVA= 000032        $SSEC = 000022        $WTOFL= 000056
$CNTLU   034540        $INTAG  001151        $PREVO= 000027        $SSENB= 000112        $XOFF = 000023
$CODE  = 000024        $ITEMB  001130        $PSEL = 000003        $STUP = 177777        $XON  = 000021
$COMND= 000002        $LF     001204        $PWRAD  037704        $SUPRL  032364        $$GET4= 000000
$CPUOP   001234        $LFLG   037021        $PWRDN  037536        $SUPRS  032400        $OFILL  036327
$CRLF    001203        $LKCSB  001300        $PWRMG  037700        $SUPR1  032412        .$X   = 001100
$CYL   = 000012        $LKCSR  001276

. ABS.   102200      000
         000000      001
```

CZRNAA0 RM80 PERF EXER MACRO V04.00 14-JAN-82 15:16:58 PAGE 68-5 SEQ 0221
SYMBOL TABLE

ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 62720 WORDS (245 PAGES)
DYNAMIC MEMORY AVAILABLE FOR 71 PAGES
,A:CZRNAA/C=A:CZRNAA.DOC,CZRNAA,SYSMAC/M

CROSS REFERENCE TABLE (CREF V04.00)

```
$$GET4  35-1      35-1#
$OFILL  44-1      44-1#     44-1*     44-1*
$40CAT  41-1
$APTHD   5-8       5-8#
$ASTAT  46-1      46-1
$ATY1   46-1#
$ATY3   43-1      46-1#
$ATY4   41-1      46-1#
$ATYC   46-1      46-1#
$AUTOB   6-0#      9-30*     9-76      9-168     9-184     9-235     10-12     11-82     23-71     29-60     29-82     32-11     35-1      35-1
        40-1      40-1      40-1
$BASE    6-0#      9-71      9-73
$BDADR   6-0#
$BDDAT   6-0#
$BDSEC  21-22     31-36     32-18     32-95     59-69#    59-73
$BELL    6-0#     22-10     40-1      40-1      40-1      41-1      41-1      41-1
$BUF    11-98*    11-134*   12-738    13-10     14-10     14-96     14-102    14-107    14-111    14-153    22-102    22-252    22-259    53-206
        56-96     56-111    59-9#
$CDW1    6-0#
$CDW2    6-0#     31-44*    31-53     31-85*
$CHARC  43-1      43-1#     43-1*     43-1*     43-1*
$CKSWR  40-1#     52-1      52-1
$CM3     6-0       6-0#
$CM4     6-0       6-0       6-0#      6-0#
$CMTAG   6-0#      9-25      9-25      9-25
$CNTLC  40-1      40-1      40-1      40-1      40-1#
$CNTLG  40-1      40-1#
$CNTLU  40-1      40-1      40-1#
$CODE   13-5      13-31     13-92     13-260    13-356    13-362    14-14     14-16     14-24     14-151    15-42*    15-64*    15-69*    20-25*
        20-33*    20-39     29-35*    29-45     56-146*   59-20#
$COMND  13-400*   13-414*   13-434*   13-471*   15-65*    15-70*    20-27*    20-28     20-32*    29-15     29-37*    31-43*    53-203    56-68
        56-125    56-135    56-147*   59-6#
$CPUOP   6-0#      9-115*    9-142*    22-89     22-271    58-91
$CRLF    6-0#      9-50      9-273     9-319     11-39     11-61     12-96     12-697    12-706    12-706    12-706    12-706    13-164    13-187
        13-199    13-325    13-334    13-360    13-373    13-387    13-389    13-390    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
        22-383    22-415    22-442    24-21     24-72     24-150    27-17     27-161    27-165    31-97     32-119    32-126    34-7      34-26
        35-1      35-1      40-1      40-1      40-1      40-1      41-1      41-1      41-1      41-1      41-1      42-1      42-1      42-1
        42-1      43-1      43-1      43-1      67-16     67-19
$CYL    13-16     13-424*   13-469*   14-181    15-26     15-28*    15-41*    20-15     20-37*    22-165    29-15     29-125*   31-39*    53-209
        56-75*    56-144    59-12#
$DB2D   22-405    22-413    22-434    24-98     24-109    24-117    39-85     39-228    49-1#
$DB20   39-245    50-1#
$DBLK   45-1      45-1      45-1#
$DECVL  49-1      49-1#
$DEVCT   6-0#     35-1*     35-1*
$DEVM    6-0#
$DIV    13-255    13-349    22-291    36-11     36-39#
$DOAGN  35-1      35-1      35-1#
$DRVSN  29-24     29-27     30-25*    39-62     59-73#    59-77
$DSPLY  39-96#    52-2
$DTBL   45-1      45-1#
$ENDAD   5-5      35-1#
$ENDAT  14-26*    14-27*    35-1      35-1      35-1      59-25#
$ENDCT  35-1#
$ENDSK  14-183*   14-184*   35-1      35-1      59-26#
```

CROSS REFERENCE TABLE (CREF V04.00)

```
$ENV     6-0#     9-30     9-66     10-6     41-1     43-1     46-1     46-1
$ENVM    6-0#     9-25     43-1     43-1     46-1
$EOP    11-176    35-1#
$EOPCT   35-1     35-1#
$ERFLG   6-0#     41-1     41-1     41-1*
$ERMAX   6-0#
$ERROR   9-25     41-1#
$ERRPC   6-0#     41-1     41-1     41-1     41-1*    41-1*    42-1     42-1
$ERRTB   8-0#     42-1
$ERRTY   41-1     42-1#
$ERTTL   6-0#     41-1     41-1     41-1*
$ESCAP  56-158    56-196   56-198   58-196
$ETABL   6-0#
$ETEND   5-8      6-0#
$FAIR   11-101*   11-138*  11-146*  11-147   59-39#
$FATAL   6-0#     46-1*
$FFLG    46-1     46-1#    46-1*    46-1*    46-1*
$FILLC   6-0#     43-1     43-1     43-1
$FILLS   6-0#     43-1     43-1
$FIRST   9-103*   9-103*   9-103*   9-103*   9-103*   9-103*   9-103*   9-103*   10-34*   29-24    29-69*   59-51#
$FMT    53-202    59-5#    59-6     59-7     59-8     59-9     59-10    59-11    59-12    59-13    59-14    59-18
$GDADR   6-0#
$GDDAT   6-0#
$GET42   11-91    23-73    35-1#
$GTSWR   40-1#    52-1     52-1
$HARD   24-135    24-147   25-17    25-17*   59-35#
$HD      4-57     4-57     4-57
$HIBTS   5-8#
$HINUM   9-25*    36-9     47-1     47-1     47-1#    47-1*
$HLDWC   14-94    14-103   14-108   14-112   15-58*   20-43*   59-40#
$HSNL    31-70    31-74*   39-84    59-64#   59-65    59-69
$HSNM    9-155*   9-155*   9-155*   9-155*   9-155*   9-155*   9-155*   9-155*   31-72    31-75*   31-104*  39-79    59-65#
$ICNT    6-0#
$ILLUP   51-1     51-1     51-1#
$INTAG   6-0#     40-1     40-1     40-1     40-1     40-1*
$ITEMB   6-0#     41-1     41-1     41-1     41-1     41-1     41-1*    41-1*    42-1
$LF      6-0#     40-1     40-1     40-1     41-1     41-1     43-1     43-1
$LFLG    46-1#    46-1*
$LKCSB   7-0#     23-53*
$LKCSR   7-0#     23-47    23-54*
$LKS     7-0#     23-60    23-65*
$LLVEC   7-0#     23-62
$LONUM   9-25*    16-20    36-8     47-1     47-1     47-1#    47-1*
$LPADR   6-0#
$LPERR   6-0#
$LPVEC   7-0#     23-50
$LSTAD   9-157    66-1#    66-1*
$MADR1   6-0#
$MADR2   6-0#
$MADR3   6-0#
$MADR4   6-0#
$MAIL    5-8      5-8      6-0#     9-19     9-25     9-30     41-1     43-1
$MAMS1   6-0#
$MAMS2   6-0#
$MAMS3   6-0#
$MAMS4   6-0#
```

CROSS REFERENCE TABLE (CREF V04.00)

```
$MBADR    5-8#
$MFLG     46-1      46-1#     46-1*     46-1*
$MISPO    22-437    24-141    24-147    25-19     25-19*    59-37#
$MNEW     40-1      40-1#
$MSGAD    6-0#      46-1      46-1*
$MSGLG    6-0#      46-1*
$MSGTY    6-0#      46-1      46-1      46-1*     46-1*
$MSWR     40-1      40-1#
$MTYP1    6-0#
$MTYP2    6-0#
$MTYP3    6-0#
$MTYP4    6-0#
$NCODE    16-24*    17-59     17-68     20-13     20-25     20-26     59-45#    59-46     59-47     59-48     59-49     5  0      59-51     59-55
$NCYL     16-26*    16-47*    17-10*    20-37     59-49#
$NEXT     11-100*   11-123    11-137*   15-73*    15-78*    17-72*    29-20     59-50#
$NPATC    17-71*    20-35     59-46#
$NSEC     16-25*    16-42     16-45*    17-32*    17-36     17-55     20-36     59-47#
$NTRK     16-46*    17-21*    59-48#
$NULL     6-0#      43-1      43-1      43-1
$NWTST    9-19#
$OCNT     44-1#     44-1*     44-1*
$OCTVL    50-1      50-1#
$OMODE    44-1      44-1#     44-1*     44-1*     44-1*     44-1*
$OPERC    14-179*   14-180*   15-15     15-17     16-37     16-39     59-27#
$PACK     10-37*    11-123    15-43     15-62     15-77*    20-9      27-170*   59-21#
$PASS     6-0#      9-25*     35-1      35-1      35-1*     35-1*     35-1*     35-1*
$PASSC    24-81     29-44*    35-1      35-1      35-1*     59-38#
$PASTM    5-8#
$PATTC    14-155    15-72*    20-35*    29-38*    29-39*    59-23#
$POSIT    14-185*   14-186*   22-432    24-116    59-28#
$POWER    51-1      51-3#
$PREVA    13-421*   13-422*   13-423*   14-181    15-25*    15-26*    20-15*    20-16*    20-19*    20-20*    20-21*    22-210    22-214    22-218
          22-226    56-144*   56-145*   59-24#
$PREVO    13-413*   15-24*    20-11*    20-12*    20-30     22-46     56-143*   59-22#
$PSEL     53-204    59-7#
$PWRAD    51-1#
$PWRDN    9-25      51-1      51-1#
$PWRMG    51-1#
$PWRUP    51-1      51-1#
$QUES     6-0#      40-1      40-1      40-1      40-1      41-1      41-1      43-1      43-1
$R2A      52-1
$RAND     16-8      17-39     17-52     17-82     47-1#
$RDCHR    40-1#     52-1      52-1
$RDDEC    52-1
$RDLIN    40-1#     52-1      52-1
$RDOCT    52-1
$RDOFL    14-32*    22-408    24-104    59-31#
$RDSZ     40-1      40-1#
$READ     14-28*    14-29*    14-31*    22-411    24-108    59-32#
$REG      53-210    59-13#    59-113    59-113    59-113    59-113    59-113    59-113    59-113    59-113    59-113    59-113    59-113    59-113
          59-113    59-113    59-113    59-113
$RESRE    48-1#     52-1
$RETRY    12-222    12-321    12-388    12-437    12-481    12-517    12-545    12-621    12-653    12-673    13-491#   13-509
$RHEXT    4-9#      9-110     22-28     22-88     22-270    29-26     53-167    53-197    58-90     59-97     59-113    59-113    59-113    59-113
          59-113    59-113    59-113    59-113    61-11     62-9
$RM80     9-307     24-78     64-19#
```

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

```
$TERM   52-3#
$TESTN   6-0#    9-19*   35-4*   42-1    42-1
$TIME   22-18    24-69   26-4#   27-18   41-1
$TKB     6-0#    27-84   40-1    40-1    40-1    40-1    40-1    40-1    40-1    43-1    43-1
$TKCNT  40-1     40-1    40-1#   40-1*   40-1*   40-1*   40-6*
$TKINT   9-65    10-3    27-19   35-5    40-1    40-1    40-1#   51-16
$TKQEN  40-1     40-1    40-1#   40-9
$TKQIN  40-1     40-1    40-1#   40-1*   40-1*   40-1*   40-1*   40-7*   40-8*   40-9    40-11*
$TKQOU  40-1     40-1    40-1#   40-1*   40-1*   40-1*
$TKQSR  40-1     40-1    40-1    40-1#   40-11
$TKS     6-0#    27-85*  40-1    40-1    40-1    40-1    40-1    40-1*   40-1*   40-1*   40-1*   40-1*   40-1*   43-1
        43-1
$TKSRV  40-1     40-1#
$TMPO    6-0#
$TN      4-48#    4-57     9-19    9-19    9-19    9-19#
$TNPWR  49-1     49-1    49-1#
$TOTAL  22-397   24-144  25-20   25-20*  34-38   59-33#
$TPB     6-0#    43-1    43-1    43-1*
$TPFLG   6-0#    43-1    43-1    43-1
$TPS     6-0#    43-1    43-1    43-1
$TRAP    9-25    52-1#
$TRAP2  52-1     52-1#
$TRK    13-467*  15-40*  22-170  29-126*  31-40*  53-208  59-11#
$TRP    52-1     52-1    52-1    52-1     52-1    52-1    52-1    52-1    52-1    52-1    52-1    52-1    52-1    52-1
        52-1     52-1    52-1    52-1     52-1    52-1    52-1    52-1    52-1    52-1    52-1    52-1    52-1    52-1
        52-1     52-1    52-1    52-1     52-1    52-1    52-1    52-1    52-1    52-1    52-1    52-1    52-1    52-1
        52-1     52-1    52-1#   52-1#    52-1#   52-1#   52-1#   52-1#   52-1#   52-1#   52-1#   52-1#   52-1#   52-1#
        52-2     52-2    52-2    52-2     52-2#
$TRPAD  52-1     52-1#   52-3
$TSTM    5-8#
$TSTNM   6-0#    41-1    41-1    41-1
$TTYIN  40-1     40-1    40-1    40-1     40-1    40-1#
$TYPBN  52-1
$TYPDS  45-1#    52-1    52-1
$TYPE   39-46    39-99   43-1#   46-1     52-1    52-1
$TYPEC  40-1     43-1    43-1    43-1     43-1#
$TYPEX  43-1     43-1    43-1#
$TYPOC  44-1#    52-1    52-1
$TYPON  44-1     44-1#   52-1
$TYPOS  44-1#    52-1
$UNIT    6-0#     7-0
$UNITM   5-8#
$USWR    6-0#
$VECT1   6-0#    9-68    9-70
$VECT2   6-0#
$WCNT   13-472*  15-57*  15-59*  20-42*  20-44*  29-41*  29-42*  31-42*  53-205  56-82   56-93*  56-105  56-108  59-8#
$WRDL   12-736   13-11   13-249  13-342  14-51   14-58   14-60*  14-61   14-62*  14-154  15-53*  15-54*  15-56*  15-57
        15-58    17-64*  20-42   20-43   22-99   22-255  22-288  29-40*  56-94*  56-95*  59-18#  59-19   59-20   59-21
        59-22    59-23   59-24   59-25   59-26   59-27   59-28   59-29   59-30   59-31   59-32   59-33   59-34   59-35
        59-36    59-37   59-38   59-39   59-40   59-41   59-45
$WRITN  14-18*   14-19*  14-21*  22-403  24-97   59-30#
$WTOFL  14-22*   22-400  24-93   59-29#
$XOFF   43-1     43-1
$XON    40-1     43-1    43-1
.$ASTA  46-1     46-1
.$X      5-8      5-8#
```

CROSS REFERENCE TABLE (CREF V04.00)

```
A16       4-84#    9-136    9-140
A17       4-85#    9-136    9-140
ABASE     4-264#   6-0      6-0
ABNRML   11-175   34-36#
ACDW1     6-0      6-0
ACDW2     6-0      6-0
ACK       4-252#
ACPUOP    6-0      6-0
ACTDRV   53-118#  55-13*   55-57*   56-3*    56-11*   57-6
ACTSTR   53-124#  57-8*    57-23*
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    64-59#
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       4-156#
APASS     6-0      6-0
APRIOR    6-0
APTCSU   43-1     46-1#
APTENV   41-1     43-1     46-1     46-1#
APTSIZ    9-25    46-1#
APTSPO   43-1     46-1     46-1#
ASGN1    27-113#
ASGN2    27-111   27-126#
```

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

```
BIT13    4-76#    9-142    12-152   12-515   12-617   39-96    41-1     56-193
BIT14    4-76#    12-8     12-10    12-41    12-128   12-132   12-136   12-186   12-332   12-669   12-671   12-680   55-46    55-81
         56-285   57-103   58-114
BIT15    4-76#    12-6     12-188   12-241   12-311   12-355   12-617   55-46    55-49    55-51    55-81    55-84    55-251   56-53
         56-219   56-221   56-285   57-52    57-66    57-103   57-110
BIT2     4-76#    12-43    57-110
BIT3     4-76#    12-156
BIT4     4-76#    12-435
BIT5     4-76#    12-160
BIT6     4-76#    12-263   22-97    55-74
BIT7     4-76#    12-255   12-479   13-498   56-45
BIT8     4-76#    12-386   12-410   12-416   12-449   12-454   12-617   56-45
BIT9     4-76#    12-172   12-617
BLKADR   7-0#     9-90     10-27    24-25    27-148   27-169   28-15    51-29
BLNKS1   13-370   13-382   22-19    22-74    24-75    24-77    24-86    24-88    63-65#
BLNKS2   9-302    12-706   12-706   12-706   12-706   13-181   13-184   13-321   13-321   13-321   13-330   13-330   13-330   13-379
         22-77    22-116   22-167   22-172   22-188   22-307   22-307   22-318   63-64#   67-25    67-36
BLNKS3   13-367   63-63#
BLNKS4   9-275    63-62#
BPTVEC   4-76#
BSE      4-232#
BUFTBL   7-0#     9-158*   9-159*   9-160*   9-163*   9-164*   9-166*   9-167*   9-170*   9-171    9-173    14-48    14-50    14-64*
         14-91    14-92    14-104*  14-109*  14-114   14-115   14-128*
BUSADR   9-78     67-13#
CFLAG    7-0#     9-100*   9-191*   9-195    9-214    9-234*   9-240    10-47*   11-3     11-84*   27-16*   27-24    27-92*   27-97
         32-15*   32-36    32-121   32-127*  33-9*    33-21    33-31*   40-5*    51-18*   51-22    67-20*   67-28    67-39
CHGADR   7-0#     9-15*    9-18*    10-14    10-44*   29-62    29-84    51-26*   67-13    67-15*
CHKWC    15-52    17-56    19-16#
CI1      55-101   55-131#
CI3      55-103   55-154#
CI4      55-92    55-172#
CI5      55-150   55-168   55-234#
CI8      55-36    55-110   55-141   55-143   55-144   55-146   55-148   55-157   55-164   55-166   55-178   55-183   55-187   55-189
         55-194   55-198   55-200   55-210   55-219   55-230   55-242#  56-25    56-44    56-47    56-50    56-74    56-76    56-78
         56-99    56-117   56-119   56-121   56-123   56-126   56-155   56-232   57-95    58-89
CK.CHR   39-152#  39-189   39-197
CK.DEC   39-130#  39-158
CK.DIG   32-53    32-60    32-67    33-23    39-183#
CK.NUM   67-30    67-41    68-12#
CK.OCT   39-112#  39-160
CKBUS    12-16    12-734#
CKCLK    9-262    23-43#   35-6     51-17
CKCLK1   23-46    23-57#
CKCLK2   23-59    23-68#
CKCLK3   23-55    23-66    23-76#
CKERR    12-15    12-715#
CKFMT    12-146   12-410#
CKHCE    12-150   12-449#
CKLMTS   15-20    15-37    16-43    17-37    18-15#
CKSWR    41-1     41-1     52-1#
CLKFLG   7-0#     23-43*   23-48*   23-61*   26-4
CLOCK    23-51    23-63    26-31#
CLR      4-104#   27-15
CLRDPB   10-29    27-154   29-10#
CLRQUE   54-17    57-75    58-130#
CMCNT    7-0#     13-11*   13-12*   13-24    13-26    13-27*   13-30*   13-94    13-97*   13-102   13-127*
```

CROSS REFERENCE TABLE (CREF V04.00)

```
CMCYL    7-0#    13-16*  13-17*  13-36   13-112*  13-113  13-119
CMDAT   13-32    13-45   13-54#  13-98   13-100
CMHED   13-36#
CMPAR   12-19    13-3#
CMPARD  12-301   13-5#
CMPLMT   7-0#    12-710  13-19
CMPRES  11-44    11-110  11-159  11-182  21-7#    21-10
CMPRT   13-49    13-68   13-78   13-89   13-136#
CMPRX   13-46    13-96   13-103  13-117  13-128   13-131#
CMSEC    7-0#    13-18*  13-104* 13-105  13-107*
CMSTR   13-21#
CMSTR2  13-40    13-43   13-48#  13-123  13-126
CMTRK    7-0#    13-108* 13-109  13-111*
COLON   26-17    26-22   64-41#
COMMA   24-79    32-106  32-114  64-5#
COMTBL   7-0#    20-27   29-37
CPSAVE  41-1     41-1    41-1    41-1    41-1     41-1#   41-1*   41-1*   42-1
CR       4-76#   43-1    43-1
CRLF     4-76#    9-6     9-30    9-30    9-44     9-55    9-61   24-68   24-92  24-103  24-114  24-130  24-131  43-1
        43-1     51-3    51-3    61-7    61-9    61-10   61-12   63-46   63-47  63-48   63-49   63-51   63-53   63-54
        63-56    63-57   63-58   63-59   63-60    63-61   64-18   64-20   64-22  64-23   64-23   64-25   64-27   64-29
        64-29    64-30   64-32   64-32   64-36    64-37   64-38   64-39   64-39  64-40   64-40   64-43   64-44   64-45
        64-46    64-48   64-50   64-50   64-52    64-53   64-54   64-55   64-55  64-56   64-56   64-57   64-58   64-58
        64-59    64-62   64-63   64-64   64-64    65-17
CTRAP   40-1     40-1    40-5#
CYLIMT   7-0#    13-115  29-71   29-89   29-90    29-139* 29-142* 32-53
CYLNDR  12-456   12-457* 12-458  12-461* 12-497*  22-236  22-307  22-307  31-45  59-122  68-45#
DASH    24-76    64-6#
DATA0    7-0      7-0#
DATA1    7-0      7-0#
DATA10   7-0      7-0#
DATA11   7-0      7-0#
DATA12   7-0      7-0#
DATA13   7-0      7-0#
DATA14   7-0      7-0#
DATA15   7-0      7-0#
DATA2    7-0      7-0#
DATA3    7-0      7-0#
DATA4    7-0      7-0#
DATA5    7-0      7-0#
DATA6    7-0      7-0#
DATA7    7-0      7-0#
DATA8    7-0      7-0#
DATA9    7-0      7-0#
DATAPK  27-58    28-59#
DCK      4-162#
DCKER   12-184   12-197#
DCKER1  12-237#  12-533
DDISP    4-76#    6-0     9-25
DDRVS    7-0#    11-14   28-15*  34-25*  35-1*
DEASGN  27-46    28-4#
DEASSG  11-40    64-29#
DEC2     7-0#    21-24*  21-27*  23-36*  23-38*
DH1      8-5     61-1#
DH14    22-73    61-6#
DH15    22-82    61-8#
```

CROSS REFERENCE TABLE (CREF V04.00)

```
DH16    22-85   61-10#
DH17    22-93   61-12#
DH2      8-12    8-33   61-2#
DH3      8-19   61-3#
DH4      8-26   61-4#
DH6      8-40   61-5#
DISPLA   6-0#    9-25*   9-25*  41-1*
DISPLY  12-86   12-98   12-109  12-205  12-215  12-218  12 234  12-258  12-305  12-314  12-336  12-374  12-401  12-423
        12-467  12-494  12-510  12-532  12-538  12-557  12-567  12-576  12-594  12-612  12-633  12-645  12-665  12-691
        12-696  12-697  12-698  12-706  12-706  12-706  12-706  12-706  12-706  12-706  12-706  12-724  12-742  13-158
        13-163  13-164  13-165  13-181  13-184  13-187  13-196  13-199  13-274  13-279  13-316  13-321  13-321  13-321
        13-325  13-330  13-330  13-330  13-333  13-334  13-360  13-361  13-364  13-367  13-370  13-373  13-375  13-379
        13-382  13-387  13-389  13-390  13-515  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  22-343  22-357  22-363  22-382  22-383  22-396  22-399  22-402  22-407  22-410  22-415  22-431
        22-436  22-439  22-442  22-451  22-465  38-36   52-2#
DISPRE   5-1#    9-25
DLT      4-114#
DONE    12-26   12-124#
DPE      4-224#
DPINT   53-79#  54-46   54-49   55-21   56-178  56-191  56-275  56-277* 57-44   57-80   57-89   57-97*
DPR      4-136#
DPRQS   53-88#  55-27   55-105* 56-181  56-289* 57-46   57-82   57-91   57-106*
DRIVE    7-0#   41-1*   62-2    62-3    62-4
DRIVE0   7-0     9-103*  9-155* 59-113# 65-54  65-54  65-54  65-54  65-54  65-54
DRIVE1   7-0     9-103*  9-155* 59-113# 65-54  65-54  65-54  65-54  65-54  65-54
DRIVE2   7-0     9-103*  9-155* 59-113# 65-54  65-54  65-54  65-54  65-54  65-54
DRIVE3   7-0     9-103*  9-155* 59-113# 65-54  65-54  65-54  65-54  65-54  65-54
DRIVE4   7-0     9-103*  9-155* 59-113# 65-54  65-54  65-54  65-54  65-54  65-54
DRIVE5   7-0     9-103*  9-155* 59-113# 65-54  65-54  65-54  65-54  65-54  65-54
DRIVE6   7-0     9-103*  9-155* 59-113# 65-54  65-54  65-54  65-54  65-54  65-54
DRIVE7   7-0     9-103*  9-155* 59-113# 65-54  65-54  65-54  65-54  65-54  65-54
DRNUM   27-159  64-30#
DROP    12-104  34-20#   34-40
DROPNG  34-27   64-24#
DRQ      4-192#
DRVACT  53-50#  55-32   55-237* 55-243  55-253* 56-16*  56-184  56-222* 56-237* 57-55*  57-60   57-67*
DRVCLR   4-245#
DRVER   12-190  12-400#
DRVINT  54-37   54-71#  55-23   55-76   56-256  56-278  57-58
DRVNO    7-0#   12-3*   22-75   35-1    35-1    35-1*
DRVPAR   7-0#   27-93*  27-109* 27-157  29-64   29-86   32-13
DRVPRM  10-33   27-167  29-58#
DRVQUE  55-30   55-38   56-140  58-162#
DRVSN   39-60   64-61#
DRVSTA   9-276  10-24   27-151  53-60#  54-27*  54-28*  54-29*  54-30*  54-40*  54-72*  54-80*  54-103* 54-108* 55-19
        55-25   55-44   55-78   55-82   55-259* 56-280  57-98*
DRVTYP   9-280  27-177  53-70#  54-73*  54-85*  54-90*  55-47   55-260*
DRY      4-135#
DSWR     4-76#   6-0     9-25
DT00     4-183#
DT01     4-184#
DT02     4-185#
```

CROSS REFERENCE TABLE (CREF V04.00)

```
DT03      4-186#
DT04      4-187#
DT05      4-188#
DT06      4-189#
DT07      4-190#
DT08      4-191#
DT1       8-6       62-1#
DT14     22-78      62-6#
DT15     22-83      62-7#
DT16     22-86      62-8#
DT17     22-94      62-10#
DT2       8-13       8-34     62-2#
DT3       8-20      62-3#
DT4       8-27      62-4#
DT6       8-41      62-5#
DTE       4-159#
DTEER    12-180     12-529#
DTUW     27-10      53-154#  54-23    55-94    55-149*  55-246   55-254   55-256*  56-6     56-17*   56-65*   57-49    57-63    57-73*
         57-84
DVA       4-129#
DVC       4-226#
ECBAD0    7-0#      13-293*  13-321
ECBAD1    7-0#      13-310*  13-330
ECBIT     7-0#      13-266*  13-267*  13-268   13-269*  13-287*
ECC      12-292     13-247#
ECC1     13-308     13-316#
ECC2     13-283     13-306   13-333#
ECCX     13-324     13-331   13-334#
ECGD      7-0#
ECGD1     7-0#
ECH       4-153#    12-200
ECI       4-213#
ECMSK0    7-0#      13-285*  13-289*  13-294   13-296   13-305
ECMSK1    7-0#      13-286*  13-290*  13-299   13-311   13-313
ECSEC     7-0#      13-247*  13-259*  13-262*  13-264*  13-280
ECWRD     7-0#      13-268*  13-270*  13-271*  13-272*  13-273*  13-275   13-280*  13-281   13-293   13-295   13-296*  13-297*  13-301
         13-321     13-321
EMWRD1    7-0#      13-301*  13-302*  13-303   13-307*  13-310   13-312   13-313*  13-314*  13-323   13-330   13-330
EM1       8-4       60-3#
EM10     60-9#
EM11     60-10#
EM12     60-11#
EM13     12-86      60-12#
EM14     12-98      60-13#
EM15     12-109     60-14#
EM2       8-11      60-4#
EM20     12-374     60-15#
EM21     12-234     60-16#
EM22     12-331     60-17#
EM23     12-314     60-18#
EM24     12-423     60-19#
EM25     12-467     60-20#
EM26     12-576     60-21#
EM27     12-594     60-22#
EM3       8-18      60-5#
EM30     12-401     60-23#
```

CROSS REFERENCE TABLE (CREF V04.00)

```
EM31     12-510    60-24#
EM32     12-532    60-25#
EM33     12-538    60-26#
EM34     12-645    60-27#
EM35     12-557    60-28#
EM36     12-567    60-29#
EM37     12-334    60-30#
EM4       8-25     60-6#
EM40     12-612    60-31#
EM41     12-742    60-32#
EM42     13-158    60-33#
EM43     12-691    12-696   60-34#
EM44     12-724    60-35#
EM45     12-218    60-36#
EM46     22-105    60-37#
EM47     12-215    60-38#
EM5       8-32     60-7#
EM50     12-633    60-39#
EM51     12-494    60-40#
EM52     12-205    60-41#
EM6       8-39     60-8#
EM60     12-665    60-42#
EMPTYQ   55-258    56-54    56-223   56-231   56-287   57-111   58-148#
EMTVEC    4-76#     9-25*    9-25*
ENDCMP   13-131    13-192#
ENDCON    7-0#     10-4*    10-5*    10-9*    10-10*   35-1     35-1
ENDING    7-0#     35-1     65-9
ENDPAS   35-1      64-25#
ENDPGM    9-159     9-164   68-47#
ENDSEK    7-0#     35-1     35-1
ENDTST   35-1      64-27#
ENTADR   32-16     64-40#
ENTCOM   27-20     64-38#
ENTLMT   29-66     64-39#
ENTPR     9-247     9-253#
EOP1     35-1      35-1#
EOP2     15-45     35-1     35-1     35-1     35-1     35-1#
EQUAL    64-4#
ERCTR     7-0#     13-9*    13-48*   13-64    13-77*   13-194   13-197
ERPRC1   12-25     12-30#   12-259   13-516
ERPROC   12-5      12-9     12-11    12-24#
ERR       4-142#
ERROR     4-76#
ERRVEC    4-76#     9-25     9-25*    9-25*    9-27*    9-28*    9-116    9-117*   9-143*   23-45    23-46*   23-59*   23-76*   41-1
         41-1*     41-1*    66-1     66-1     66-1*    66-1*    66-1*    66-1*
F0        4-124#
F1        4-125#
F2        4-126#
F3        4-127#
F4        4-128#
FACTOR   14-9*     14-10*   14-12*   14-18    14-26    14-28    14-35#
FAIRNS    7-0#
FE1       7-0#     18-42    18-49    19-46    29-76    29-118   29-120   29-121
FE2       7-0#     29-123   29-139
FEFLAG    7-0#      9-182*   9-226*  29-74    29-116   29-137
FEONLY    9-188     9-228   64-50#
```

CROSS REFERENCE TABLE (CREF V04.00)

```
FER       4-151#
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#
FMT16      4-214#
FMTER     12-418    12-575#
FRSTER     7-0#     12-300*   12-302    12-688    12-709*   13-3*     13-81     13-138    13-154    13-167*   13-192
GENDPB    10-26*    13-399*   13-400*   13-402    13-434*   13-436    13-438    13-467*   13-468*   13-469*   13-470*   13-471*   13-472*   13-474
          13-476    27-146*   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*    59-117#
GENPAR    11-130    16-7#
GENREG    12-416    12-454    59-126    59-129#
GETADR    10-31     27-156    31-33#
GETBUF    11-97     11-133    14-45#
GETID     10-30     27-155    30-10#
GETLMT    13-54     15-14     16-7      23-14     29-68     29-137#   31-34
GETPAT    15-71     17-70     17-77#    17-85
GETREG     4-254#
GETREM    16-14     17-8      17-19     17-30     17-49     17-83     36-8#
GETREQ    55-69     55-248    56-33     56-206    56-284    57-51     57-65     57-100    57-107    58-178#
GNS        5-1       5-1       9-6       9-30      9-44      9-55      9-61      24-68     24-80     24-92     24-95     24-103    24-106    24-114
          24-130    24-131    24-134    24-137    24-140    24-143    52-1      52-1      52-1      52-1      52-1      52-1      52-1      52-1
          52-1      52-1      52-1      52-1      52-1      52-1      52-1      52-1      52-1      52-1      52-1      52-1      52-1      52-1
          52-2      52-2
GO         4-123#
GODRIV    11-103    11-139    12-251    12-344    13-491    14-173#
GTSWR      9-30     52-1#
HCE        4-154#
HCEER     12-462    12-593#
HCI        4-212#
HCRC       4-155#
HCRCER    12-142    12-369#   12-412    12-451
HDWSN     39-78     64-60#
HOUR       7-0#      9-95*    11-86*    26-7      26-43*
HT         4-76#    43-1      43-1
HZ         7-0#      9-94     11-85     26-33     26-44
IAE        4-157#
IAEER     12-166    12-556#
IBSAVE    41-1      41-1      41-1      41-1      41-1      41-1#     41-1*     41-1*     41-1*
IDLE      11-7      11-116    11-121    11-164#
IE         4-82#
ILF        4-147#
ILR        4-148#
INCHRD    12-279    25-17#
INCMIS    12-499    25-19#
INCSKI    12-637    25-18#
INCSOF    12-291    12-299    12-384    12-433    12-477    25-16#
INCTOT    12-90     12-102    12-113    12-220    12-257    12-280    12-304    12-319    12-363    12-385    12-404    12-434    12-478    12-500
          12-514    12-542    12-560    12-569    12-587    12-605    12-616    12-636    12-649    12-668    12-728    12-746    13 200    25-20#
INTRVL     7-0#      9-98*    11-89*    26-38*    26-50     26-52     26-52     26-55*    51-13*    51-24     65-4
INVLD     27-81     64-37#
IOTVEC     4-76#
IR         4-105#
ISR       54-32     56-3#
IVC        4-229#
KIPARO    66-1
```

CROSS REFERENCE TABLE (CREF V04.00)

```
KSR      11-8     27-10#   27-11
KSR1     27-12#
LBC       4-227#
LBT       4-138#
LF        4-76#    24-92    43-1     43-1     64-20    64-54    64-55
LIMIT     7-0#     12-710*  13-19*   13-20*   13-170   13-172*
LIN10A   13-274   63-52#
LIN10B   13-279   63-53#
LIN10C   13-333   63-54#
LIN10H   13-316   63-56#
LIN11    13-364   63-60#
LIN11A   13-375   63-61#
LIN11H   13-361   63-59#
LIN2C    22-39    63-1#
LIN2P    22-45    63-2#
LIN2S    22-71    63-3#
LIN3.1   22-126   22-132   22-200#
LIN3.3   22-137   22-144   22-225#
LIN3.4   22-145   22-235#
LIN6.2   22-344   22-358   22-364   22-379#
LIN7M    22-436   63-33#
LIN7OR   22-407   63-42#
LIN7OX   22-399   63-40#
LIN7P    22-431   63-37#
LIN7R    22-410   63-43#
LIN7S    22-439   63-38#
LIN7T    22-396   63-39#
LIN7X    22-402   63-41#
LIN8M    12-258   13-515   22-451   63-44#
LIN9B    13-163   63-45#
LIN9E    13-196   63-50#
LIN9G    12-305   63-51#
LIN9H    13-165   63-46#
LIN9I    12-698   63-48#
LINA3    22-235   63-11#
LINB3    22-152   63-15#
LINB5    22-284   63-23#
LINB6    22-328   63-28#
LINC6    22-343   63-29#
LINCA3   22-189   63-13#
LIND5    22-268   63-22#
LINDA3   22-185   63-14#
LINDEC   13-198   22-76    22-166   22-171   22-176   22-201   22-206   22-208   22-211   22-215   22-219   22-227   22-230   22-238
         22-242   22-256   22-261   22-294   22-381   22-398   22-401   22-409   22-438   22-441   22-477#
LINE1    12-85    12-97    12-108   12-204   12-214   12-217   12-233   12-313   12-330   12-373   12-400   12-422   12-466   12-493
         12-509   12-531   12-537   12-556   12-566   12-575   12-593   12-611   12-632   12-644   12-664   12-690   12-723   12-741
         13-157   22-8#
LINE2    12-87    12-99    12-112   12-219   12-237   12-318   12-340   12-375   12-402   12-424   12-468   12-495   12-511   12-539
         12-558   12-568   12-577   12-595   12-613   12-634   12-646   12-666   12-692   12-725   12-743   13-159   22-35#   22-452
LINE2A   22-44    22-49    22-68#
LINE2B   22-69    22-72#
LINE3    12-88    12-100   12-112   12-238   12-318   12-340   12-376   12-403   12-425   12-469   12-512   12-578   12-596   12-614
         12-667   12-726   22-125#
LINE3A   12-647   12-693   13-160   22-131#
LINE3B   12-635   22-137#
LINE3C   12-496   22-144#
```

CROSS REFERENCE TABLE (CREF V04.00)

```
LINE3D  12-744  22-107  22-150#
LINE3E  12-540  22-164#
LINE3F  12-559  22-183#
LINE4   12-89   12-101  12-112  12-239  12-318  12-340  12-377  12-426  12-470  12-513  12-541  12-579  12-597  12-615
        12-648  12-694  12-727  12-745  13-161  22-108  22-249#
LINE5   12-318  12-340  22-268#
LINE5A  12-378  12-432  12-476  12-498  12-586  12-604  22-301#
LINE5B  12-290  22-315#
LINE6   12-244  12-288  22-328#
LINE6A  12-298  22-343#
LINE6C  12-224  12-323  12-349  12-359  12-390  12-439  12-483  12-519  12-547  12-623  12-655  12-675  22-357#
LINE6D  12-227  12-295  12-326  12-394  12-443  12-487  12-523  12-551  12-626  12-659  12-678  22-363#
LINE7   12-91   12-103  12-114  12-228  12-281  12-306  12-364  12-391  12-395  12-405  12-440  12-444  12-484  12-488
        12-520  12-524  12-548  12-561  12-570  12-588  12-606  12-627  12-656  12-679  12-729  12-747  13-201  13-512
        22-396#
LINE7A  12-501  12-638  22-431#
LINE8   12-278  12-362  13-511  22-451#
LINEN3  22-228  63-10#
LINEO5  22-319  63-27#
LINEP5  22-315  63-26#
LING6   22-357  63-30#
LINM3   22-125  63-4#
LINM4   22-251  63-19#
LINN3   22-131  63-6#
LINOCT  12-706  12-706  12-706  12-706  12-706  12-706  12-706  12-706  13-180  13-183  13-186  13-321  13-321  13-321
        13-330  13-330  13-330  13-366  13-369  13-372  13-378  13-384  22-114  22-154  22-157  22-187  22-191  22-253
        22-283  22-286  22-307  22-307  22-317  22-321  22-461#
LINP3   22-209  63-8#
LINP5   22-293  63-24#
LINR6   22-382  63-31#
LINS3   22-164  22-225  63-9#
LINS4   22-254  63-20#
LINS5   22-302  63-25#
LINSS3  22-173  63-18#
LINST3  22-168  63-17#
LINT3   22-239  63-12#
LINU06  22-363  63-32#
LINW3   22-155  63-16#
LINX4   22-257  63-21#
LINX5   22-308  63-66#
LKPAR    9-179#  9-196   9-206   9-215
LODEV    9-299  27-118  27-132  64-17#
LODPAR  11-131  20-8#
LSC      4-228#
LSTAD    7-0#    9-157*  9-160   9-161
LSTHDR  32-94   64-58#
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#   11-195  15-80   35-7
MAIN1   11-20   11-49#
MAIN2   11-59   11-79#
MAINDA  11-13#
```

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

```
OFLIN    12-42    12-96#
ONES      7-0      7-0#
ONESEC    7-0#     9-94*   11-85*   26-31*   26-33*
OPE       4-230#
OPI       4-160#
OPIER    12-154   12-507#
OPIER1   12-516#
OPT      55-34    55-67#   56-36    56-247   56-290
OPTBL     7-0#    22-51    22-53
OR        4-106#
ORDERQ    7-0#     9-88    11-5     11-104   11-115   11-140   11-164   51-27
OVRWRT    9-209   64-51#
PACK      7-0#     9-99*   15-76*   27-170   28-49*   28-54*   28-59*   28-65*
PAR       4-150#
PAR1     65-3     65-18#
PAR10    65-13    65-27#
PAR11    65-7     65-28#
PAR14    65-8     65-29#
PAR15    65-10    65-30#
PAR16    65-9     65-31#
PAR19    65-5     65-32#
PAR2     65-4     65-19#
PAR20    65-11    65-33#
PAR21    65-12    65-34#
PAR3     65-6     65-20#
PAR4     65-21#   65-54    65-54    65-54    65-54    65-54    65-54    65-54    65-54
PAR5     65-22#   65-54    65-54    65-54    65-54    65-54    65-54    65-54    65-54
PAR6     65-23#   65-54    65-54    65-54    65-54    65-54    65-54    65-54    65-54
PAR7     65-24#   65-54    65-54    65-54    65-54    65-54    65-54    65-54    65-54
PAR8     65-25#   65-54    65-54    65-54    65-54    65-54    65-54    65-54    65-54
PAR9     65-26#   65-54    65-54    65-54    65-54    65-54    65-54    65-54    65-54
PARENT    9-254   29-95    33-8#
PARER    12-158   12-537#
PARLST    9-174*   9-253   65-3#
PASSES    7-0#    35-1     65-5
PAT       4-103#
PATTER    7-0#    13-225   17-78    17-80    65-6
PCLOCK    7-0#    23-44*   23-49*
PERFEX    4-8#    56-69
PERIOD   24-85    24-94    24-100   24-105   24-111   24-119   24-133   24-136   24-139   24-142   24-149   39-65    39-87    64-35#
PFECH     42-1    42-1#
PFECH1    42-1    42-1#
PFECH2    42-1    42-1#
PFECH3    42-1    42-1#
PFECH4    42-1    42-1#
PFTSTN    42-1    42-1#    42-1*
PGE       4-109#
PGM       4-137#
PIP       4-141#
PIRQ      4-76#
PIRQVE    4-76#
POPQUE   55-80    55-131   55-220   56-241   58-190#
POSER    12-460   12-493#
PR0       4-76#   35-1
PR1       4-76#
PR2       4-76#
```

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

```
RHEXT    9-114*  9-123*  9-127*  9-133   53-168#  58-95
RM80    13-401  13-415  13-435  13-473   14-175   31-46   55-11#
RMADR    9-85*  27-14   53-165#  54-34   55-18    55-134  55-154  55-172  56-5    57-39   58-11   58-17   58-48   58-52
        58-70   58-111  67-44
RMAS    53-184#  54-99*  56-153  56-246*  56-268*  57-77
RMBA    53-179#  56-117
RMBAE   53-198#
RMCS1   53-177#  54-75*  54-81   54-93   55-72    55-87   55-148  55-166  55-209  55-219  55-230  56-25   56-126  56-155
        56-216  56-226  57-86   58-42
RMCS2   27-15*  53-181#  54-35*  54-74*  54-76    55-29*  55-135* 55-155* 55-173* 56-24*  56-207* 56-252* 56-267* 57-40*
        57-54*  58-18   58-53   58-71*  58-77    58-112* 58-117
RMCS3   53-199#
RMDA    53-180#  55-141  55-164  55-178  56-76
RMDB    53-186#  58-75
RMDC    53-191#  55-143  55-157  55-183  55-194   56-74
RMDS    53-182#  54-96   55-74   56-208  56-253   57-41
RMDT    53-188#  54-83
RMEC1   53-195#
RMEC2   53-196#  58-85
RMER1   12-416  12-454  53-183#  54-100  56-44    56-212  56-254
RMER2   53-194#  56-47   56-255
RMERRS  53-43#  54-18   56-193  56-253*  56-254*  56-255*  62-2    62-2    62-2    62-2
RMHR    53-192#
RMINIT   9-263  54-14#
RMLA    53-185#
RMMR1   53-187#
RMMR2   53-193#
RMOF    53-190#  54-95   55-144  55-146  55-187   55-189  55-198  55-200  56-50   56-121  56-123
RMR      4-149#
RMSN    53-189#
RMTMR   26-49   57-6#
RMVEC    9-86*  53-166#  54-31   54-33   55-12
RMWC    53-178#  56-78   56-99   56-119
RNOP     4-242#
RTC      4-250#  13-400
RTNCTR  13-399#
RTURN   35-1    35-3#
S       22-207  22-216  63-7#
SATPOW  51-1    51-9#
SAVEFG   9-264*  53-132#  55-222  56-28   56-243
SAVER1   7-0#  13-22*  13-84*  13-136  13-144   13-148*
SAVER5   7-0#  13-23*  13-85*  13-145  13-149*
SAVREG  14-150  27-12   36-39   49-1    50-1     52-1#   54-14   55-14   55-67   55-242  56-4    57-9    58-67   58-130
SC      56-9    56-153#
SC04     4-201#
SC1      4-199#
SC10     4-202#
SC11    56-211  56-237#
SC12    56-189  56-252#
SC13    56-180  56-183  56-264#
SC2      4-200#
SC20     4-203#
SC3     56-168#  56-172
SC4     56-170#  56-192  56-197  56-199  56-227   56-233  56-248  56-291
SC5     56-169  56-178#
SC6     56-185  56-203#
```

CROSS REFERENCE TABLE (CREF V04.00)

```
SC8      56-208  56-212  56-216  56-226  56-231#  56-260
SCMND    27-50   28-28#
SCOPE     4-76#
SDETAL   24-29   24-48   24-68#
SEARCH    4-253#
SECLMT    7-0#   13-105  19-19   23-28   23-33   29-73   29-93   29-94   29-144*  29-149*  32-67
SECOND    7-0#    9-97*  11-88*  26-23   26-34*  26-35   26-37*  51-12*
SEEK      4-243#
SEEKFG   53-140#  54-19   55-98
SELDRV    4-256#
SET.IE   54-48   54-78   55-42   55-113  55-261  56-159  58-110#
SETFMT    4-255#
SETVEC    9-180   9-236   9-243   9-249   9-256   9-262#
SHDTYP   24-22   24-45   24-58#
SIZE70    9-114#
SIZMEM    9-150#  51-31
SKI       4-231#
SKIER    12-192  12-632#
SKIP     56-52   56-65#
SLASH    33-18   65-16#
SPOTCK   12-197  12-328  12-369  12-420  12-464  12-507  12-529  21-21#
SRCHWT   53-112#  55-96   55-100* 55-167* 56-224* 56-238
SSE       4-225#  56-48
SSEI      4-211#  15-30   16-28   29-145  56-51   56-122
STA      10-3#
STACK     4-76#   9-25    9-92    35-3    51-15
STAR30   11-194  64-21#
STAR5    11-42   28-43   64-22#
START     5-3     9-15#  23-75   27-28
START1    5-1     9-11    9-18#
START2    9-16    9-21#
STATIN    7-0#    9-87*  11-187  11-189* 26-54*
STATIS   11-171  14-7#
STATPR   11-193  24-7#   28-42
STKLMT    4-76#
STNDAT    7-0#   13-233  13-239  14-156
STO      57-16   57-37#
STO1     57-44#  57-85   57-88
STO2     57-43   57-45   57-47   57-77#
STO3     57-79   57-89#
STO5     57-41   57-59   57-86   57-95#
STO6     57-81   57-97#
STO7     57-83   57-105#
STO8     57-104  57-111#
STO9     57-76   57-92   57-94   57-96   57-102  57-109  57-113#
SUMARY   11-41   24-43#  35-1
SUMHD    24-59   64-23#
SUPR1    37-15   37-20#
SUPR2    37-14*  37-19*  37-29   37-31*  37-33#
SUPRS    24-99   24-110  24-118  37-17#
SUPRSL   32-102  32-108  32-116  37-12#  39-86
SURE      9-211  64-49#
SVRH70   30-12   55-224  56-30   56-55   56-214  56-245  56-286  57-53   57-112  58-67#
SW0       4-76#   9-105  16-11   27-56   27-68
SW00      4-76    4-76#
SW01      4-76    4-76#  12-17
```

CROSS REFERENCE TABLE (CREF V04.00)

```
SW02      4-76      4-76#     9-186     9-269     11-185
SW03      4-76      4-76#
SW04      4-76      4-76#     15-46     34-36     35-1
SW05      4-76      4-76#     22-80     22-150
SW06      4-76      4-76#     20-23
SW07      4-76      4-76#     13-174
SW08      4-76      4-76#     35-1      35-1
SW09      4-76      4-76#
SW1       4-76#
SW10      4-76#     22-8
SW11      4-76#
SW12      4-76#
SW13      4-76#     22-11     41-1
SW14      4-76#
SW15      4-76#     22-416    22-443
SW2       4-76#
SW3       4-76#     13-66     13-339
SW4       4-76#
SW5       4-76#     22-183
SW6       4-76#
SW7       4-76#
SW8       4-76#
SW9       4-76#
SWR       6-0#      9-25      9-25      9-25*     9-25*     9-25*     9-30      9-105     9-186     9-269     11-185    12-17     13-66     13-174
          13-339    15-46     16-11     20-23     22-8      22-11     22-80     22-150    22-183    22-416    22-443    27-56     27-68     34-36
          35-1      35-1      35-1      39-96     40-1      40-1      40-1*     41-1      41-1      41-1      41-1      41-1      51-1      51-1*
SWREG     5-1#      9-25      9-30      40-1      40-1      40-1
SWTIM     12-40     12-85#
SYSTAT    9-272     64-18#
T         22-202    22-212    63-5#
TAB       27-163    64-7#
TAB.XY    5-9#      6-0
TABLE     29-88     65-39#
TABLE0    65-39     65-54#
TABLE1    65-42     65-54#
TABLE2    65-42     65-54#
TABLE3    65-42     65-54#
TABLE4    65-42     65-54#
TABLE5    65-42     65-54#
TABLE6    65-42     65-54#
TABLE7    65-42     65-54#
TAP       4-194#
TBITVE    4-76#
TD        56-8      56-16#
THEAD     16-35#
TIMER     53-145#   55-108*   55-235*   55-252*   56-19*    56-204*   56-265*   56-272*   57-12     57-14*    57-68*    57-93*    57-99*    57-105*
TKVEC     4-76#     40-1*     40-1*
TPVEC     4-76#
TRAPVE    4-76#     9-25*     9-25*
TRE       4-88#
TRFER     12-138    12-611#
TRKLMT    7-0#      13-109    18-46     19-23     19-50     23-34     29-72     29-91     29-92     29-143*   31-40     32-60
TRNSWT    53-104#   55-132*   55-245    55-257*   56-21     56-22*    56-66*    57-48     57-62     57-74*
TRTVEC    4-76#
TST1      9-19#
TYDRV     24-87     27-162    39-60#
```

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

```
$$CMRE    5-586#
$$CMTM    5-586#   6-0
$$ESCA    4-76#
$$NEWT    4-76#    9-19
$$SET    52-1     52-1     52-1     52-1     52-1     52-1     52-1     52-1     52-1     52-1     52-1     52-1#    52-2
$$SETM    9-25     9-25#
$$SKIP    4-76#
.$ACT1    4-50#    5-5
.$APTB    4-53#    6-0      6-0#
.$APTH    4-53#    5-8
.$APTY    4-53#   46-1
.$CATC    4-51#    5-1
.$CMTA    4-51#    5-586
.$DB2D    4-52#   49-1
.$DB2O    4-52#   50-1
.$EOP     4-53#   35-1
.$ERRO    4-51#   41-1
.$ERRT    4-51#   42-1
.$POWE    4-53#   51-1
.$RAND    4-52#   47-1
.$READ    4-50#   40-1
.$SAVE    4-52#   48-1
.$SIZE    4-52#   66-1
.$TRAP    4-52#   52-1
.$TYPD    4-51#   45-1
.$TYPE    4-50#   43-1
.$TYPO    4-51#   44-1
.EQUAT    4-50#    4-76
.HEADE    4-50#    4-57
.SETUP    4-50#    4-267
.SWRHI    4-50#    4-58
.SWRLO    4-50#    4-58#    4-59     4-60     4-61     4-63     4-65     4-70     4-73     4-74
A        25-2#    25-16    25-17    25-18    25-19    25-20
CKCHR     4-11#   39-189   39-197
CKDIG     4-21#   32-53    32-60    32-67    33-23
CKNUM     4-34#
COMMEN    4-76#
ENDCOM    4-76#
ERENTR   40-13#   41-1
ERRCAL   53-1#    56-158   56-196   56-198
ERROR     4-76#
ESCAPE    4-76#
GETPRI    4-76#   66-1
GETSWR    4-76#    9-30     9-30#
MORETA    5-11#    6-0
MULT      4-76#
NEWTST    4-76#    9-19
POP       4-76#   12-461   12-497   13-150   21-54    22-109   24-33    28-44    29-48    29-128   30-29    31-98    31-107   32-80
         32-128   39-34    45-1     46-1     46-1     47-1     48-1     51-1     51-1
PUSH      4-76#   12-456   13-143   21-21    22-35    22-400   22-408   23-45    24-7     28-28    29-10    30-10    31-33    31-37
         32-10    32-93    45-1     46-1     46-1     46-1     47-1     48-1     51-1     51-1
READ      53-8#   54-83    54-96    54-100   55-144   55-187   55-198   56-25    56-44    56-47    56-50    56-74    56-76    56-78
         56-99    56-121   56-155   56-208   56-212   57-41    57-86    58-42
REPORT    4-76#
SETPRI    4-76#   40-1
SETTRA   52-1     52-1     52-1     52-1     52-1     52-1     52-1     52-1     52-1     52-1     52-1     52-1#    52-2
```

CROSS REFERENCE TABLE (CREF V04.00)