```
.REM @
```

IDENTIFICATION

PRODUCT CODE: AC-T122B-MC

PRODUCT NAME: CZRNJB0 RM80 FORMATTER

PRODUCT DATE: SEPTEMBER 15, 1982

MAINTAINER: CX DIAGNOSTIC GROUP

AUTHOR: C. HESS, M. LEAVITT

CONTENTS

1.0 INTRODUCTION

1.1 ABSTRACT

THE RM80 FIELD FORMAT PROCESS FOR 16 BIT APPLICATIONS IS A RECONSTRUCTION PROCESS THAT RE-ESTABLISHES THE FACTORY FORMAT OF THE HDA. THE FORMATTER IS NOT INTENDED TO BE USED BY FIELD SERVICE AS RM80 TROUBLE SHOOTING TOOL; ALSO, HDAS SHOULD NOT BE FORMATTED UNLESS THE EXISTING FORMAT HAS BEEN DAMAGED BY HARDWARE OR SOFTWARE FAILURES.

BECAUSE OF THE TECHNOLOGY USED IN THE RM80 HDA, IT IS EXPECTED THAT THE NUMBER OF DEFECTS (BAD SECTORS) WILL EXCEED THE NUMBER OF BAD SECTORS THAT THE 16 BIT FILE SYSTEMS CAN ACCEPT. SKIP SECTORING WAS ADDED TO THE RM80 TO REDUCE THE APPARENT NUMBER OF BAD SECTORS: SKIPPED SECTORS ARE VISIBLE ONLY TO THE HOST OPERATING SYSTEM'S RM80 DEVICE DRIVER. (THE FILE SYSTEM VISIBLE DEFECTS ARE LISTED IN THE DEC STD 144 MFG AND USER'S FILES, MDBSF AND UDBSF, RESPECTIVELY.)

THE RM80 HDA MEDIA IS VERIFIED IN MANUFACTURING USING PARAMETERIC VERIFICATION EQUIPMENT. THE DEFECT LIST GENERATED BY THE MANUFACTURING VERIFCATION PROCESS IS THE MASTER LIST OF DEFECTS ON THE HDA. EXPERIENCE WITH PREVIOUS DISK PRODUCTS INDICATES THAT FUNCTIONAL FORMAT/VERIFICATION PROGRAMS CANNOT PERFORM ADEQUATE MEDIA VERIFICATION AND WILL MISS A LARGE NUMBER OF THE DEFECTS WHICH ARE PRESENT. SINCE ONLY A FEW OF THE ACTUAL DEFECTS WHICH ARE PRESENT ARE DETECTED BY A FUNCTIONAL FORMAT/VERIFICATION PROGRAM, MARKING THE ERRORS DISCOVERED BY SUCH A VERIFICATION PROGRAM WILL, IN EFFECT, CREATE AN HDA WHICH HAS A HIGHER APPARENT ERROR RATE THAN IT HAD INITIALLY. CONSEQUENTLY, THE FIELD FORMATTER WILL NOT RELY ON FUNCTIONAL MEDIA VERIFICATION ALONE. THE FIELD FORMAT OPERATION WILL BE BASED ON THE HDA'S 16 BIT MODE BAD SECTOR FILES SUPPORTED BY INCIDENTIAL MEDIA VERIFICATION. FUNCTIONAL MEDIA VERIFICATION, ALTHOUGH LIMITED, WILL BE ATTEMPTED AS A MEANS OF DETECTING DEFECTS WHICH HAVE DEVELOPED SINCE THE HDA LEFT MANUFACTURING.

THE LOCATIONS OF SKIPPED SECTORS ARE LISTED IN A NEW BAD SECTOR SUPPORT FILE CALLED THE SKIPPED SECTOR FILE (SSF). THE LOCATIONS OF SKIPPED SECTORS ARE NECESSARY SO THAT A FIELD FORMAT PROGRAM CAN RECONSTRUCT (SEE SECTION 7) THE FACTORY FORMAT OF AN HDA. THE NON-SKIPPED SECTOR LOCATIONS ARE LISTED IN THE DEC STD 144 FILES. THESE FILES ARE CREATED/INITIALIZED BY THE MANUFACTORING FORMAT PROCESS.

2.0 OPERATING REQUIREMENTS

2.1 HARDWARE REQUIREMENTS

THE FOLLOWING MINIMUM HARDWARE CONFIGURATION, ASSUMED TO BE OPERATIONAL, IS REQUIRED TO LOAD AND EXECUTE THE RM80 SUBSYSTEM FORMATTER:

PDP-11/70 PROCESSOR
28K MEMORY

KW11-L OR KW11-P CLOCK
PROGRAM LOADING DEVICE
TERMINAL
RH70 CONTROLLER
1 TO 8 RM80 DISK DRIVES

2.2 PREREQUISITE DIAGNOSTIC PROGRAMS

RM80 DISKLESS TEST, PART 1 AND 2

3.0 OPERATING PROCEDURE

3.1 LOADING

THE PROGRAM MAY BE LOADED FROM ANY XXDP MEDIA.

3.2 SWITCH OPTIONS

THE PROGRAM DOES NOT SUPPORT ANY HARDWARE OR SOFTWARE SWITCH REGISTER OPTIONS.

3.3 STARTING

THE PROGRAM MUST BE STARTED AT LOCATION 200 . HOWEVER, BECAUSE LOCATION 200 GETS OVER WRITTEN BY THE RSX11 MONITOR, THE PROGRAM CANNOT BE RESTARTED AFTER ITS INITIAL STARTUP.

3.4 HALTING

THE PROGRAM CAN BE HALTED BY PRESSING THE HALT SWITCH ON THE PROCESSOR FRONT PANEL.

NOTE: ONCE THE PROGRAM IS HALTED, IT CAN ONLY BE RESTARTED BY PRESSING CONTINUE ON THE PROCESSOR FRONT PANEL OR RELOADING THE PROGRAM AND STARTING AT ADDRESS 200 .

3.5 RESTARTING

THE PROGRAM CAN ONLY BE RESTARTED BY RELOADING THE PROGRAM AND STARTING AT ADDRESS 200 .

4.0 OPERATOR INTERFACE

4.1 PROGRAM IDENTIFICATION

THE PROGRAM TYPES ITS TITLE THE FIRST TIME IT IS STARTED AFTER BEING LOADED.

4.2 CONSOLE DIALOGUE

THE PROGRAM ENTERS A CONSOLE DIALOGUE SEQUENCE AFTER TYPING THE PROGRAM IDENTIFICATION.

UPON STARTING THE PROGRAM, THE FOLLOWING PROMPT WILL ASK THE USER TO SPECIFY WHICH DRIVE IS TO BE FORMATTED.

```
218
219                         'DRIVE #: '
220
221             ENTER THE DRIVE ADDRESS NUMBER (0-7) FOLLOWED BY  CARRIAGE  RETURN.
222             IF  THE  SELECTED  DRIVE IS NOT AVAILABLE, THE PROGRAM WILL TYPE AN
223             ERROR MESSAGE AND RETURN TO THE THIS PROMPT.
224
225             NOTE: ONLY ONE DRIVE CAN BE FORMATTED AT A TIME.  THIS PROGRAM DOES
226                   NOT ALLOW MULTIPLE DRIVE FORMATTING.
227
228             AFTER  THE USER HAS SELECTED THE DRIVF TO BE FORMATTED, THE PROGRAM
229             WILL ASK FOR AN OPTION TO BE SELECTED WITH THE FOLLOWING PROMPT.
230
231                         'OPTIONS: '
232
233             THE COMMANDS AVAILABLE FOR THE OPTION PROMPT ARE AS FOLLOWS:
234
235                   IN      INITIALIZE  ALL  THE BAD SECTOR FILES (DEC STD 144 AND
236                           SSF).
237
238                           NOTE: THE INITIALIZE COMMAND WILL DESTROY THE CONTENTS
239                                 OF  ALL  THE  BAD  SECTOR FILES (DEC STD 144 AND
240                                 SSF).
241
242                   FO      FORMAT THE DISK.  (ALL CYLINDERS 0 - 560.)
243
244                   <CR>    SAME AS 'FO' COMMAND
245
246                   FO:F    FORMAT THE FE CYLINDERS ONLY.  (CYL 559., TRK 2 -  CYL
247                           560., TRK 13.)
248
249                   LI      LIST  THE  PHYICAL ADDRESSES OF THE DEFECTS IN ALL THE
250                           BAD SECTOR FILES (DEC STD 144 FILES AND SSF).
251
252                   LI:L    LIST THE LOGICAL ADDRESSES OF THE DEFECTS IN  THE  DEC
253                           STD  144  FILES.   (SSF  DOES  NOT CONTAIN ANY LOGICAL
254                           ADDRESSES)
255
256                   VFL=N   VERIFY SECTOR COUNT, WHERE 'N' IS THE NUMBER OF  TIMES
257                           TO  RE-READ  A  TRACK  DURING  VERIFY  OPERATION.  THE
258                           DEFAULT IS 1 (RANGE  1-256.).   AFTER  COMPLETING  THE
259                           FORMAT, 'N' WILL RETURN TO ITS DEFAULT VALUE.
260
261                   ERL=N   ERROR  LIMIT  COUNT, WHERE 'N' IS THE NUMBER OF ERRORS
262                           ALLOWED BEFORE THE FORMAT IS ABORTED.  THE DEFAULT  IS
263                           256  (RANGE  1-256.). AFTER COMPLETING THE FORMAT, 'N'
264                           WILL RETURN TO ITS DEFAULT VALUE.
265
266                   CSR=N   RM BASE ADDRESS, WHERE 'N' IS THE ADDRESS OF RMCS1  IN
267                           OCTAL.    THE  DEFAULT  IS 176700. AFTER COMPLETING THE
268                           FORMAT, 'N' WILL RETAIN ITS CHANGED VALUE.
269
270                   VEC=N   RM  VECTOR  ADDRESS,  WHERE  'N'  IS  THE  ADDRESS  OF
271                           INTERRUPT ADDRESS IN OCTAL.  THE DEFAULT IS 254. AFTER
272                           COMPLETING THE FORMAT, 'N'  WILL  RETAIN  ITS  CHANGED
273                           VALUE.
```

THE USER MAY STRING ANY COMBINATION OF COMMAND OPTIONS TOGETHER BY USING A SLASH (/) TO SEPARATE THE COMMANDS. HOWEVER, IF ONLY A CARRIAGE RETURN IS TYPE FOR THE OPTION PROMPT, THE PROGRAM WILL GO INTO FORMAT (FO) MODE USING THE DEFAULT PARAMETERS.

EXAMPLE #1:

```
OPTION:FO/CSR=176000/VEC=260/VFL=3/ERL=10<CR>
```

THE ABOVE COMMAND INDICATES THAT THE ENTIRE DISK WILL BE FORMATTED, THE RMCS1 ADDRESS WILL BE SET TO 176000, THE VECTOR ADDRESS WILL BE SET TO 260, THE VERIFY COUNT WILL BE SET TO 3 AND ERROR LIMIT WILL BE SET TO 10.

THE FOLLOWING PROMPT WILL OCCUR IF THE OPTION SPECIFIED WAS AN INITIALIZE (IN) OR FORMAT (FO) COMMAND.

```
'CONTINUE (L) N ? '
```

IF THE RESPONSE TO THIS PROMPT IS A 'N' FOLLOWED BY A CARRIAGE RETURN OR JUST A CARRIAGE RETURN, THE PROGRAM WILL ABORT THE CURRENT COMMAND AND RETURN TO THE DRIVE PROMPT. IF THE RESPONSE IS A 'Y' FOLLOWED BY A CARRIAGE RETURN, THE PROGRAM WILL CONTINUE TO THE NEXT STEP IN THE CURRENT COMMAND OPTION.

THE FOLLOWING PROMPT WILL OCCUR IF THE OPTION SPECIFIED WAS A FORMAT (FO) COMMAND.

```
'UPDATE MODE (L) N ? '
```

IF THE RESPONSE TO THIS PROMPT IS A 'N' FOLLOWED BY A CARRIAGE RETURN OR JUST A CARRIAGE RETURN, THE PROGRAM WILL ABORT THE CURRENT COMMAND AND RETURN TO THE DRIVE PROMPT. IF THE RESPONSE IS A 'Y' FOLLOWED BY A CARRIAGE RETURN, THE PROGRAM WILL ENTER THE MANUAL BAD SECTOR UPDATE ROUTINE.

THE FOLLOWING TWO PROMPTS WILL OCCUR IF THE OPTION SPECIFIED WAS AN INITIALIZE (IN) OR FORMAT (FO) COMMAND.

```
'ENTER DATE (DD-MMM-YY): '
```

THE DATE IS ENTERED BY TYPING A DAY IN DECIMAL, THE FIRST 3 LETTERS OF THE MONTH AND THE LAST TWO DECIMAL DIGITS OF THE YEAR.

AFTER ENTERING THE DATE, THE PROGRAM WILL PROMPT THE USER FOR THE HDA SERIAL NUMBER AS FOLLOWS:

```
'HDA SERIAL NUMBER: '
```

THE HDA SERIAL NUMBER CAN BE ANY DECIMAL NUMBER UP TO 10. DIGITS IN LENGTH.

IF THE OPTION WAS A FORMAT (FO) COMMAND AND THE USER HAS ENTERED THE MANUAL UPDATE MODE, THE FOLLOWING PROMPT WILL BE TYPED TO ALLOW THE ENTERING OF SECTORS INTO THE BAD SECTOR FILES.

```
'ENTER BAD SECTR ADRS:
 CYL,TRK,SEC = '
```

THE BAD SECTOR ENTRY IS ENTERED BY TYPING THE CYLINDER ADDRESS, TRACK ADDRESS AND SECTOR ADDRESS IN DECIMAL, FOLLOWED BY A CARRIAGE RETURN. TO TERMNATE THE BAD SECTOR ENTRY LIST, TYPE A CARRIAGE RETURN IN RESPONSE TO THE ENTRY PROMPT.

IF THE BAD SECTOR HAS A HEADER ERROR OR DRIVE TIMING ERROR (EXAMPLE #2), THE LETTER 'H' (HEADER DEFECT) MUST FOLLOW THE DEFECT INDICATED. THIS WILL INDICATE TO THE PROGRAM, WHICH FILE (SSF OR DEC STD 144) THE DEFECT MUST BE STORED IN.

EXAMPLE #1:

```
ENTER BAD SECTR ADRS:
CYL,TRK,SEC = 525,4,25<CR>   ;DEFECT  CYL 524., TRK 4., SEC
                              25.
CYL,TRK,SEC = <CR>           ;TERMINATE INPUT
```

IF THIS DEFECT (EXAMPLE #1) IS NOT ALREADY IN THE SKIP SECTOR FILE (SSF), THE DEFECT WILL BE ENTERED AS A SKIP SECTOR ENTRY. (SEE SECTION 7.0 FOR MORE DETAILS)

EXAMPLE #2:

```
ENTER BAD SECTR ADRS:
CYL,TRK,SEC = 525,4,25H<CR>  ;DEFECT CYL 524., TRK 4.,  SEC
                              25. (HEADER ERROR)
CYL,TRK,SEC = <CR>           ;TERMINATE INPUT
```

THIS SECTOR (EXAMPLE #2) WILL NOT BE ENTERED AS A SKIPPED SECTOR, BECAUSE A HEADER ERROR WAS FLAGGED BY THE USER, BY PLACING THE LETTER 'H' AFTER THE DEFECT. THE DEFECT WILL BE ENTERED IN THE MANUFACTURES SECTION OF THE DEC STD 144 FILE. (SEE SECTION 7.0 FOR MORE DETAILS)

NOTE: THE LETTER LOCATED WITHIN THE BRACKETS () INDICATES THE TYPE OF RESPONSE REQUIRED BY THE USER, D=DECIMAL, O=OCTAL AND L=LETTER.

4.3 WARNING MESSAGES

WARNING MESSAGES WILL BE TYPED AT THE APPROPRIATE TIME DURING THE PROGRAM TO NOTIFY THE USER THAT THE CONTENTS OF THE HDA WILL BE LOST. THE FOLLOWING MESSAGE WILL BE TYPED BEFORE INITIALIZING OR FORMATTING THE HDA.

```
'** WARNING - DATA WILL BE LOST ON DRN: **'
```

TO FURTHER WARN THE USER, THE FOLLOWING MESSAGE WILL BE TYPED JUST BEFORE INITIALIZING THE HDA.

'** WARNING - BAD SECTOR FILES WILL BE OVERWRITTEN ON DRN: **'

FOLLOWING EACH WARNING MESSAGE, THE USER IS ALLOWED THE OPPORTUNITY TO TERMINATE THE PROGRAM WITHOUT DESTROYING THE DATA ON HDA, BY RESPONSING TO A PROGRAM PROMPT MESSAGE. (SEE SECTION 4.2)

4.4 PROGRESS REPORTS

NO PROGRESS REPORTS ARE GIVEN DURING THE EXECUTION OF THE PROGRAM.

4.5 PERFORMANCE REPORT

NO PERFORMANCE REPORTS ARE GIVEN DURING THE EXECUTION OF THE PROGRAM.

4.6 PROGRAM HALTS

THERE ARE NO SCHEDULED HALTS DURING THE EXECUTION OF THE PROGRAM.

4.7 ERROR REPORTS

THE FIRST LINE CONTAINS THE ERROR MESSAGE: ONE LINE OF TEXT WHICH GIVES A BRIEF DESCRIPTION OF THE ERROR. THE ERROR MESSAGE IS FOLLOWED BY ONE LINE CONTAINING DATA HEADERS AND DATA PERTININENT TO THE ERROR. THE LAST LINE CONTAINS THE DECIMAL REQUIVALENT OF THE RMDC AND RMDA REGISTERS, WHICH INDICATE WHERE THE ERROR OCCURED.

THE FOLLOWING PRINTOUT SHOWS A TYPICAL ERROR MESSAGE FOR THIS PROGRAM:

```
HEADER/DATA ERROR DURING VERIFY
RMCS1=144252 RMCS2=040300 RMER1=100000 RMER2=000000 RMDA=001014
CYLINDER=559. TRACK=2. SECTOR=11.
```

4.8 EXECUTION TIME

ONE PASS OF THE PROGRAM TAKES ABOUT 20 MINUTES.

5.0 ENVIRONMENTAL SUPPORT

5.1 PROCESSOR COMPATIBILITY

THE RM80 SUBSYSTEM FORMATTER IS EXECUTABLE ON A PDP-11/70 PROCESSOR, PROVIDING PREVIOUSLY MENTIONED HARDWARE REQUIREMENTS ARE MET.

5.2 DUAL PORT CONFIGURATIONS

THE RM80 SUBSYSTEM FORMATTER DOES NOT SPECIFICALLY TEST DUAL PORT LOGIC IN THE RM80 ADAPTER BUT IS EXECUTABLE ON RM80 SUBSYSTEMS HAVING THE DUAL PORT OPTION PROVIDING THE DUAL PORT SWITCH IS SET TO THE APPROPRIATE PORT (A OR B).

5.3 MEMORY PARITY HARDWARE

MEMORY PARITY HARDWARE IS NOT USED DURING THE EXECUTION OF THE RM80 SUBSYSTEM FORMATTER.

5.4 MEMORY MANAGEMENT HARDWARE

MEMORY MANAGEMENT HARDWARE IS NOT USED DURING THE RM80 SUBSYSTEM FORMATTER.

5.5 ACT11, APT11 COMPATIBILITY

THE RM80 SUBSYSTEM FORMATTER IS COMPATIBLE WITH ACT11 AND APT11 IN DUMP MODE ONLY.

5.6 XXDP COMPATIBILITY

THE RM80 SUSBYSTEM FORMATTER IS COMPATIBLE WITH XXDP IN DUMP MODE ONLY.

5.7 OPERATING SYSTEM COMPATIBILITY

THE PROGRAM IS NOT COMPATIBLE WITH ANY OPERATING SYSTEM WHEN IT IS ASSEMBLED AS AN XXDP STANDALONE PROGRAM.

6.0 FORMAT DESCRIPTION

6.1 FORMAT PROCESS

THE FORMATTER USES THE DEC STD 144 MFG AND USER FILES AND THE SKIPPED SECTOR FILE TO RECONSTRUCT (SEE SECTION 7) AN HDA'S 16 BIT FACTORY FORMAT AND ATTEMPTS TO FIND ADDITIONAL DEFECTS BY PERFORMING A FUNCTIONAL MEDIA VERIFICATION.

BEFORE FORMATTING BEGINS, THE FORMATTER WILL READ AND VERIFY THE DEC STD 144 FILES AND THE SSF FILE. THESE FILES CONTROL THE MARKING OF KNOWN DEFECTS ON THE HDA. THE PROGRAM WILL READ AND VERIFY THE DEFECT FILES USING THE FOLLOWING PROCEDURE:

1. EACH COPY OF THE SKIPPED SECTOR FILE (SSF) IS READ UNTIL A GOOD READ OCCURS AND THE HEADER AND DATA FIELDS ARE VALIDATED USING THE INTERNAL CHECKSUMS.

2. STARTING WITH THE PRIMARY DEC STD 144 TRACK (CYL 558., TRK 13.) AND CONTINUING TO THE SECONDARY DEC STD 144 TRACK (CYL 559., TRK 1.), THE PROGRAM WILL READ EACH COPY OF THE 16 BIT MANUFACTURING AND USER'S DEC STD 144 FILE UNTIL A GOOD READ OCCURS AND THE INTERNAL FORMAT OF EACH FILE IS VALIDATED.

3. VERIFY THAT THE HDA SERIAL NUMBERS IN THE DEC STD 144 (MANUFACTURING AND USER) FILE AGREE WITH THE SERIAL NUMBER IN THE SSF. IF THE SERIAL NUMBER IN THE DEC STD 144 (MANUFACTURING OR USER) FILE DOES NOT COMPARE WITH THE SERIAL NUMBER IN THE SSF, THEN THE NEXT COPY OF THE FILE IN ERROR IS READ AND VALIDATED.

4. VERIFY THAT THE DISK ADDRESSES IN THE DEC STD 144 (MANUFACTURING AND USER) FILE ARE VALID RM80 ADDRESSES. IF

AN ADDRESS ERROR IS FOUND, THE FILE IN ERROR WILL BE CONSIDERED CORRUPTED AND THE NEXT COPY OF THE FILE IN ERROR WILL BE READ AND VALIDATED.

AFTER VERIFYING THE DEFECT FILES, THE PROGRAM WILL THEN FORMAT (WRITE HEADER AND DATA) AND VERIFY (WRITE CHECK HEADER AND DATA) ALL 32 SECTORS OF EACH TRACK IN 16 BIT MODE USING THE FOLLOWING 32 BIT DATA PATTERN TO FILL THE DATA FIELD OF EACH SECTOR.

155555 133333 (OCTAL) OR DB6DB6DB (HEX)

THE FOLLOWING SEQUENCE IS USED FOR FORMATTING:

1. THE SKIPPED SECTOR ERROR INHIBIT BIT (SSEI), BIT 09 IN THE RMOF REGISTER, IS SET BEFORE EACH WRITE OR READ OPERATION. SETTING THIS BIT, IN 16 BIT MODE, ALLOWS ACCESS TO ALL 32 SECTORS ON THE TRACK.

   USING A WRITE HEADER AND DATA COMMAND, EACH TRACK (32 SECTORS) IS FORMATTED IN 16 BIT MODE USING THE ABOVE DATA PATTERN FOR THE DATA PORTION OF EACH SECTOR.

2. USING A WRITE CHECK HEADER AND DATA COMMAND, EACH TRACK IS READ 1 TIME. IF A DATA ERROR OCCURS AT A SECTOR WHICH IS NOT LISTED IN EITHER THE 16 BIT DEC STD 144 FILE OR THE SSF, THE SECTOR IS READ 20 TIMES. IF A DATA ERROR OCCURS DURING THE REREAD SEQUENCE, THE SEQUENCE IS DISCONTINUED, AND THE ADDRESS OF THE FAILING SECTOR IS SAVED IN THE APPROPRIATE DEFECT LIST. THE PROGRAM THEN CONTINUES CHECKING THE REMAINDER OF THE TRACK.

AFTER A TRACK HAS BEEN VERIFIED, THE TRACK IS REFORMATTED WITH THE BAD SECTOR MARKINGS BASED ON THE PREVIOUSLY DETECTED ERRORS AND ANY NEWLY DETECTED ERRORS.

THE PRIMARY AND SECONDARY DEC STD 144 TRACKS (CYL 558, TRK 13 AND CYL 559 TRK 1, RESPECTIVELY), THE SSF TRACK (CYL 559,TRK 0), AND THE FE AREA WILL BE REFORMATTED.

THE SSF REVISION NUMBER AND DATE WILL BE UPDATED ONLY IF THE SSF FILE HAS BEEN UPDATED.

BOTH THE SSF AND THE DEC STD 144 FILES WILL BE SORTED SO THAT THE DEFECT ENTRIES ARE IN ASCENDING ORDER. THE 16 BIT DEC STD 144 FILES AND THE SKIPPED SECTOR FILES WILL BE REWRITTEN.

NOTE: THE 18 BIT MODE DEFECT DEC STD 144 FILE RECORDS AND THE 18 BIT MODE SSF WILL BE REPLACED BY NULL RECORDS DURING A 16 BIT MODE FORMAT.

IF, DURING THE FORMAT, THE NUMBER OF NEWLY DETECTED DEFECTS CAUSES THE TOTAL NUMBER OF DEFECTS IN THE MDBSF AND THE UDBSF TO EXCEED THE MAXIMUM, THE PROGRAM DISCONTINUES THE FORMAT AND DISPLAYS AN ERROR MESSAGE TO THIS EFFECT. THE PROGRAM THEN RESTARTS THE FORMAT PROCESS USING ONLY THE BAD SECTOR INFORMATION ALREADY IN THE MDBSF, UDBSF, AND THE SSF. THE FORMATTER, UNDER THESE CIRCUMSTANCES, WILL

WRITE THE TRACK STRUCTURE AND VERIFY THAT THE WRITE OCCURED CORRECTLY, ONLY - NO MEDIA VERIFICATION WILL BE ATTEMPTED. THE OPERATOR WILL MAKE THE DETERMINATION IF THE HDA IS DEFECTIVE AND IF IT IS TO BE REPLACED.

7.0 RECONSTRUCTION DESCRIPTION

7.1 OLD DEFECTS

THE TRACK IS RECONSTRUCTED ACCORDING TO DEFECTS LISTED IN THE 16 BIT MDBSF, THE 16 BIT UDBSF, AND THE 16 BIT SSF.

1. THE SSF IS CHECKED FOR AN ENTRY ON THE CURRENT TRACK. IF AN ENTRY EXISTS, THE SSE BIT IS SET IN THE HEADER OF THE SECTOR LISTED AND IN ALL FOLLOWING SECTORS, TO THE END OF THE CURRENT TRACK. (SEE SECTION 9.2)

2. THE DEC STD 144 FILE IS CHECKED FOR ANY ENTRIES ON THE CURRENT TRACK.

   FOR DEC STD 144 ENTRIES WHICH OCCUR BEFORE A SKIPPED SECTOR OR ON A TRACK WHICH DOES NOT HAVE A SKIPPED SECTOR, THE APPROPRIATE HEADER BIT IS RESET (MF OR UF, ACCORDING T THE FILE CONTAINING THE DEFECT ADDRESS). ALSO, THE ACTUAL ADDRESS (PHYSICAL ADDRESS) OF THE DEFECT IS LISTED IN THE DEC STD 144 FILE. (SEE SECTION 9.1 AND 9.3)

   FOR DEC STD 144 ENTRIES WHICH OCCUR AFTER A SKIPPED SECTOR, THE APPROPRIATE HEADER BIT IS RESET (MF OR UF, ACCORDING TO THE FILE CONTAINING THE DEFECT ADDRESS). ALSO, THE DEFECT IS LISTED IN THE DEC STD 144 FILE BY LOGICAL ADDRESS. (SEE SECTION 9.4 - 9.11) THE LOGICAL ADDRESS IS:

   LOGICAL ADDRESS = PHYSICAL(CYL,TRK,SEC-1)

7.2 NEW DEFECTS

NEWLY DETECTED ERRORS ON TRACKS WHICH ALREADY HAVE ERRORS, ARE HANDLED AS FOLLOWS:

1. IF THE ERROR OCCURS BEFORE A SKIPPED SECTOR OR IF THERE IS NO SSF ENTRY FOR THE CURRENT TRACK, THE PHYSICAL ADDRESS OF THE SECTOR IS ENTERED IN THE 16 BIT UDBSF AND THE UF BIT IN THE HEADER OF EACH ERRORED SECTOR IS RESET. (SEE SECTION 9.1 AND 9.3)

2. IF THE ERROR OCCURS AFTER A SKIPPED SECTOR, THE LOGICAL ADDRESS OF THE SECTOR IS ENTERED IN THE 16 BIT UDBSF AND THE UF BIT IN THE HEADER OF EACH ERRORED SECTOR IS RESET. (SEE SECTION 9.4 - 9.10) THE LOGICAL ADDRESS IS:

   LOGICAL ADDRESS = PHYSICAL(CYL,TRK,SEC-1)

   ADDITIONALLY, IF THE DEFECT SECTOR HAS A HEADER RELATED ERROR, THE SECTOR'S PHYSICAL ADDRESS IS ALSO ENTERED IN THE 16 BIT UDBSF. (SEE SECTION 9.5) HOWEVER, IF THE HEADER ERROR

IS SECTOR 31, ONLY THE SECTOR'S LOGICAL ADDRESS IS ENTERED IN THE UDBSF. (SEE SECTION 9.10)

FOR THE CASE OF A SECTOR WITH A HEADER FIELD DEFECT WHICH IS IMMEDIATELY FOLLOWED BY A SECTOR WITH A DATA FIELD DEFECT, THE LOGICAL ADDRESSES WILL BE ENTERED IN THE 16 BIT UDBSF FOR EACH SECTOR. (SEE SECTION 9.9)

NEWLY DETECTED ERRORS ON TRACKS WHICH WERE PREVIOUSLY HAD NO ERRORS, ARE HANDLED AS FOLLOWS:

1. SKIPPED SECTORING STARTS WITH THE FIRST (PRIMARY), NON-HEADER DEFECT ON THE TRACK. THE PHYSICAL ADDRESS OF THIS DEFECT IS ENTERED INTO THE SSF. (SEE SECTION 9.2)

   NOTE: SKIPPED SECTORING CANNOT BEGIN WITH A SECTOR WHICH HAS A HEADER FIELD DEFECT.

   THE HEADER OF THE SKIPPED SECTOR AND THE HEADERS OF THE SECTORS ON THE SAME TRACK FOLLOWING THE SKIPPED SECTOR WILL HAVE SSE (BIT 13) SET.

   THE UF BIT WILL BE RESET IN THE HEADER OF ALL NON-SKIPPED SECTORS EVEN THOUGH THE HEADER OF THE ERROR SECTOR MAY NOT BE READABLE.

2. HEADER FIELD AND DATA FIELD DEFECTS PRECEDING A SKIPPED SECTOR WILL BE ENTERED IN THE 16 BIT MDBSF BY PHYSICAL ADDRESS. (SEE SECTION 9.3)

3. IF THE ONLY DEFECT ON A TRACK IS SECTOR 31, THE DEFECT WILL BE MARKED AS A SKIPPED SECTOR, EVEN IF THE DEFECT AFFECTS THE HEADER, AND ITS ADDRESS ENTERED IN THE SSF. (SEE SECTION 9.11)

4. MULTIPLE DEFECTS WITHIN A SECTOR WILL BE CONSIDERED A SINGLE DEFECT. IF BOTH THE HEADER AND THE DATA FIELD ARF AFFECTED BY DEFECTS, THE HEADER DEFECT WILL TAKE REPORTING CEDENCE.

5. NEW DEFECTS DETECTED ON PREVIOUSLY ERROR FREE TRACK. WITHIN THE DEC STD 144 AREA, THE SSF TRACK, OR THE FE TRACKS ARE NOT ENTERED IN THE 16 BIT UDBSF. IF APPROPRIATE, A DEFECT ON ANY OF THESE TRACKS IS SKIPPED AND ITS ADDRESS IS ENTERED IN THE 16 BIT SSF. IF A DEFECT WITHIN THIS ADDRESSING RANGE CANNOT BE SKIPPED, THE DEFECT WILL BE TREATED AS A DEC STD 144 ERROR (I.E., THE UF BIT IS RESET IN THE HEADER) EXCEPT THAT THE ADDRESS OF THE DEFECT IS NOT ENTERED IN THE 16 BIT UDBSF.

8.0 BAD SECTOR FILE LAYOUTS

8.1 SKIPPED SECTOR FILE (SSF)

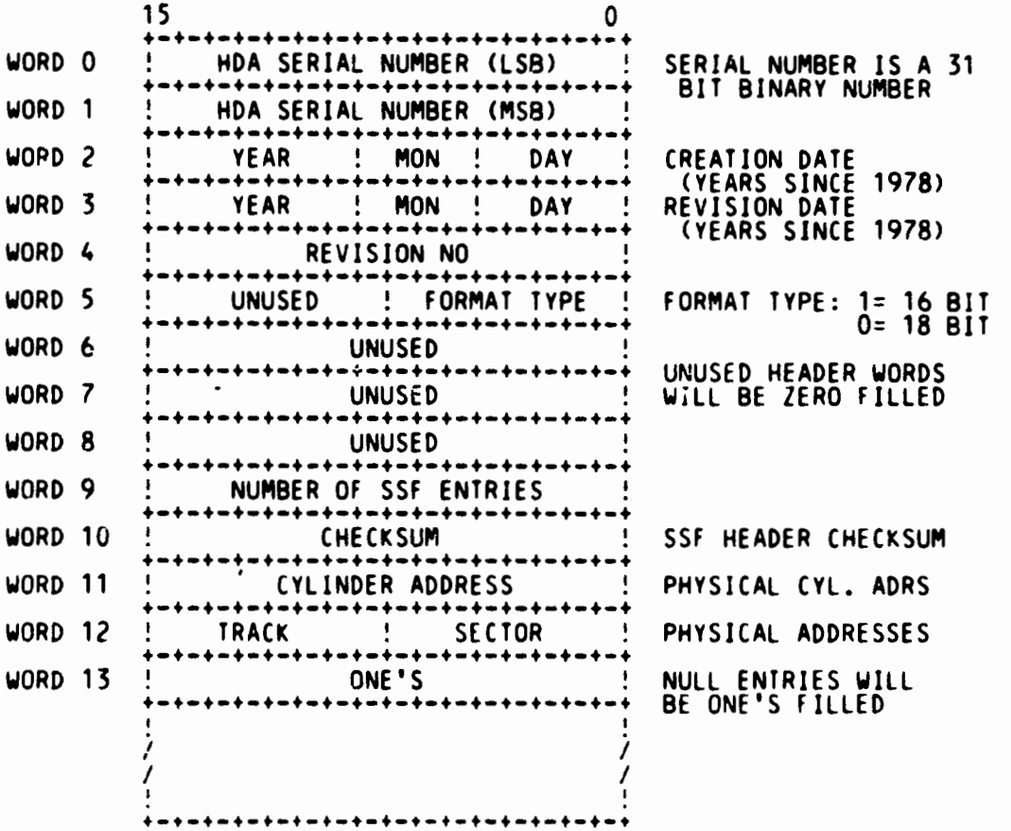
THE SSF HAS STORAGE SPACE FOR THE PHYSICAL ADDRESSES OF 378 SKIPPED SECTORS. DEFECT ADDRESSES ARE STORED IN THE SSF IN ASCENDING SEQUENCE IN THE SAME FORMAT AS IN THE DEC STD 144 FILES, I.E.,

CYLINDER ADDRESS FOLLOWED BY SECTOR AND TRACK ADDRRESSES. THE SSF WILL BE FILLED WITH ONE'S BETWEEN THE LAST DEFECT ADDRESS AND THE CHECKSUM. NOTE THAT BOTH THE HEADER PORTION AND THE DATA PORTION OF THE SSF ARE PROTECTED BY CHECKSUMS. THE 16 BIT SSF IS REPEATED 5 TIMES ON THE FIRST TRACK OF THE FE AREA: CYLINDER 559, TRACK 0. THE SSF TRACK IS WRITTEN IN 16 BIT MODE.

THE SERIAL NUMBER FIELD IN BOTH THE SSF AND THE DEC STD 144 FILE IS IN THE SAME FORMAT.

THE SSF FILE FOR 18 BIT MODE DEFECTS WILL BE REPEATED 5 TIMES FOLLOWING THE 16 BIT SSF FILE. SINCE SKIPPED SECTORING IS A MECHANISM WHICH IS INHIBITED WHEN THE RM80 IS BEING OPERATED IN 18 BIT MODE, THE USE OF THE SSF FOR 18 BIT MODE ADDRESSES IS ONLY TO PASS THE 18 BIT MODE ADDRESSES OF HDA DEFECTS WHICH DO NOT APPEAR IN THE DEC STD 144 FILE. THE 18 BIT SSF COPIES WILL NEITHER BE USED NOR PRESERVED DURING 16 BIT FIELD FORMATTING.

THE SSF FILE LAYOUT FOR BOTH 16 BIT MODE AND 18 BIT MODE FILES ARE AS FOLLOWS:



```
WORD 766 !             ONE'S             !
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
WORD 767 !           CHECKSUM            !  DEFECT ADDRESS FIELD
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   CHECKSUM
```

THE CHECKSUM IS CALCULATED USING THE FOLLOWING ALGORITHM. THE CHECKSUM CALCULATION IS BASED ON 16 BIT QUANTITIES (WORDS) AND IS REPEATED ONCE FOR THE SSF HEADER FIELD AND ONCE FOR THE SSF DATA FIELD.

```
SIZE = FIELD_SIZE_IN_16_BIT_WORDS
CHECKSUM = 1
REPEAT
   CHECKSUM = CHECKSUM + (16_BIT_WORD)
   CHECKSUM = CHECKSUM .ROTATED_LEFT_1_BIT.
   SIZE = SIZE - 1
UNTIL SIZE .EQ. 0
END_REPEAT
```

THE ROTATE REFERED TO ABOVE IS THE FOLLOWING:

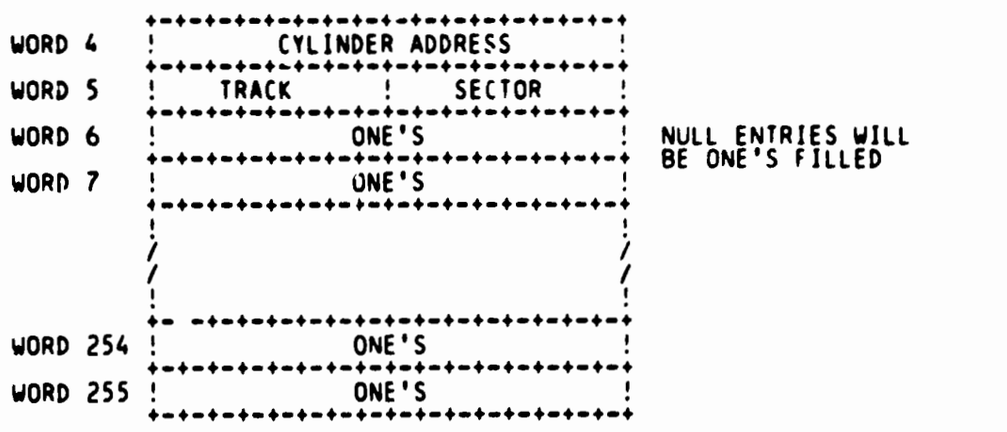
'*' REFERS TO EITHER THE 'C' BIT FOR PDP-11 OPERATION OR BIT 16 FOR MACHINES OTHER THAN THE PDP-11. NOTE THAT THE ROTATE OCCURS AFTER THE ADDITION; THE BIT SHIFTED INTO THE LSB POSITION OF THE ACCUMULATED CHECKSUM IS THE 'CARRY OUT' FROM THE 16 BIT ADDITION OPERATION.

8.2 DEC STD 144 MANUFACTURING AND USER BAD SECTOR FILES

THE DEC STD 144 FILE IS CONTAINED ON TWO TRACKS ON THE RM80. THE FIRST TRACK IS THE PRIMARY COPY OF THE FILE, THE SECOND TRACK IS THE SECONDARY COPY OF THE FILE. THE PRIMARY COPY OF THE DEC STD 144 FILE IS WRITTEN ON THE LAST ADDRESSABLE TRACK BEFORE THE FE CYLINDERS (CYL 558, TRK 13); THE SECONDARY COPY OF THE DEC STD 144 FILE IS WRITTEN AT CYL 559, TRK 1.

BOTH THE PRIMARY AND THE SECONDARY DEC STD 144 TRACKS ARE WRITTEN IN 16 BIT MODE. THE LAYOUT OF THE DEC STD 144 RECORD IS GIVEN BELOW. THE LAYOUT IS THE SAME FOR ALL COPIES OF THE FILE.

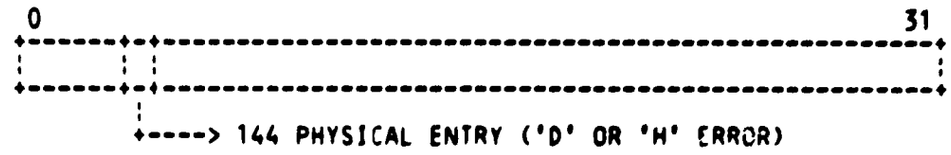
```
          15                              0
          +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
WORD 0    !    HDA SERIAL NUMBER (LSB)    !  SERIAL NUMBER IS A 31
          +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   BIT BINARY NUMBER
WORD 1    !    HDA SERIAL NUMBER (MSB)    !
          +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
WORD 2    !              ZERO             !
          +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
WORD 3    !              ZERO             !
```



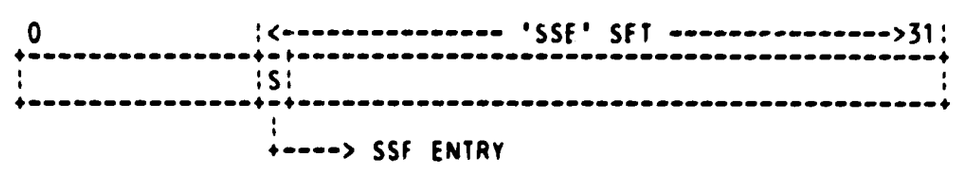
THE 16 BIT MDBSF WILL BE WRITTEN IN SECTORS 0, 2, 4, 6, 8 OF THE PRIMARY AND SECONDARY DEC STD 144 TRACKS; THE 18 BIT MDBSF IS BE WRITTEN IN SECTORS 1,3,5,7,9. THE 16 BIT UDBSF IS WRITTEN IN SECTORS 10,12,14....30; THE 18 BIT UDBSF IS WRITTEN IN SECTORS 11,13,15....29.

9.0 ERROR TYPE DESCRIPTIONS

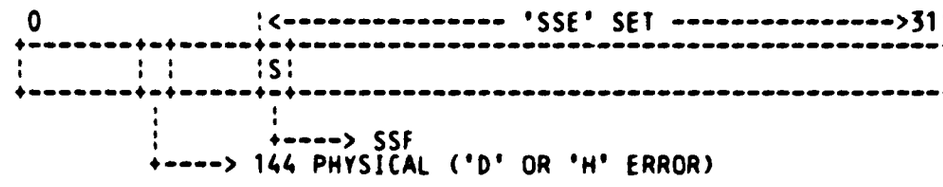
1.



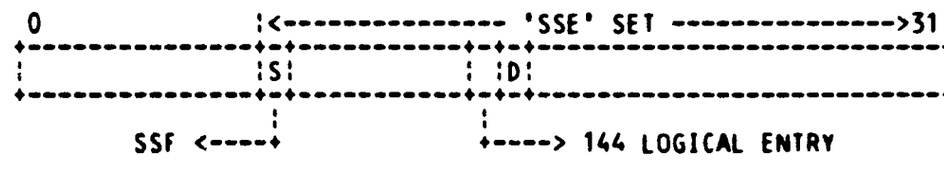
2.



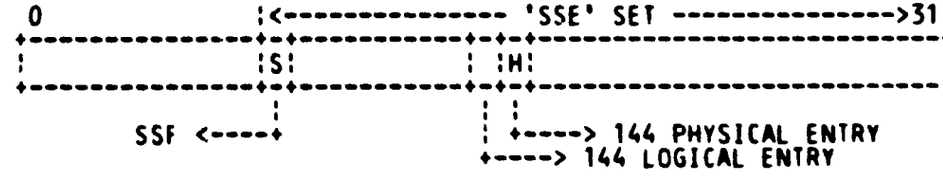
3.



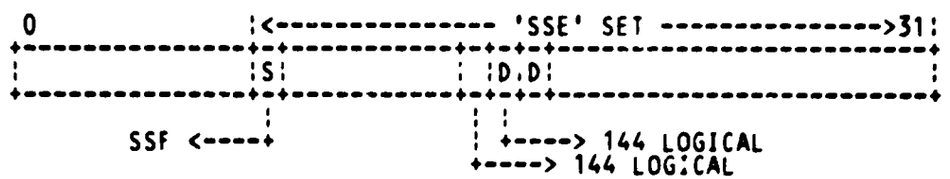
4.



5.



6.


0
'SSE' SET
31
S
D D
SSF
144 LOGICAL
144 LOGICAL

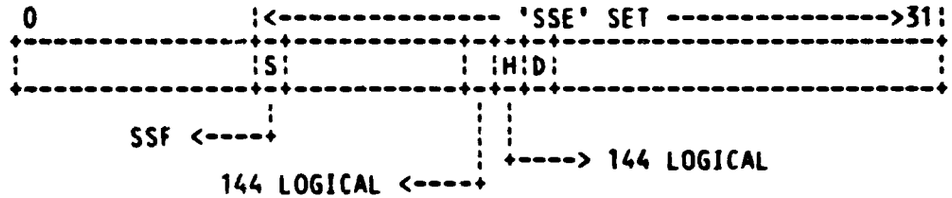
7.


0
'SSE' SET
31
S
H H
SSF
144 PHYSICAL
144 LOGICAL
144 LOGICAL

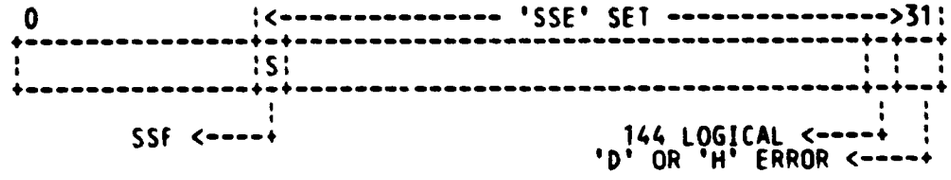
8.


0
'SSE' SET
31
S
D H
SSF
144 PHYSICAL
144 LOGICAL
144 LOGICAL

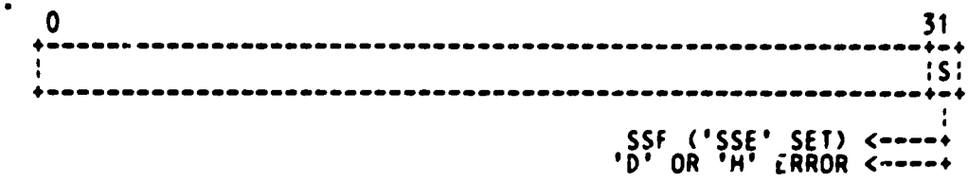
9.


0
'SSE' SET
31
S
H D
SSF
144 LOGICAL
144 LOGICAL

10.


0
'SSE' SET
31
S
SSF
144 LOGICAL
'D' OR 'H' ERROR

11.



NOTE: 'D' IS A DATA FIELD DEFECT; 'H' IS A HEADER FIELD DEFECT;
'S' IS A SKIPPED SECTOR DEFECT

SYMBOL TABLE

```
. ABS.  000000      000
        000000      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  19 WORDS  ( 1 PAGES)
DYNAMIC MEMORY:  8206 WORDS  ( 31 PAGES)
ELAPSED TIME:  00:00:23
,[300,30]CZRNJB=[300,10]BEG.DOC,CZRNJB.DOC,END.DOC
```

```
TASK    NAME    : CZRNJB
PARTITION NAME  : GEN
IDENTIFICATION  : 01.00
TASK  UIC       : [300,10]
TASK  PRIORITY  : 65.
STACK    LIMITS: 026172 026501 000310 00200.
PRG XFR ADDRESS: 026502
TOTAL ADDRESS WINDOWS: 1.
TASK  IMAGE  SIZE   : 14432. WORDS
TASK ADDRESS LIMITS: 026000 116263
R-W DISK BLK LIMITS: 000002 000072 000071 00057.

*** ROOT SEGMENT: FMTROT


R/W MEM  LIMITS: 026000 116263 070264 28852.
DISK BLK LIMITS: 000002 000072 000071 00057.


MEMORY ALLOCATION SYNOPSIS:

SECTION                                          TITLE  IDENT  FILE
-------                                          -----  -----  ----
. BLK.:(RW,I,LCL,REL,CON) 026502 020460 08496.
                          026502 000276 00190.  FMTROT 01.00  FMTROT.OBJ;1
                          027000 000444 00292.  FMTIO  01.00  FMTIO.OBJ;1
                          027444 000152 00106.  FMTDAT 01.00  FMTDAT.OBJ;1
                          027616 001666 00950.  FMTMSG 01.00  FMTMSG.OBJ;1
                          031504 003032 01562.  FMTSUB 02.00  FMTSUB.OBJ;2
                          034536 001026 00534.  FMTPRS 01.00  FMTPRS.OBJ;1
                          035564 001104 00580.  FMTR80 01.00  FMTR80.OBJ;1
                          036670 001246 00678.  FMTR8F 02.00  FMTR8F.OBJ;2
                          040136 001516 00846.  FMTR8V 02.00  FMTR8V.OBJ;2
                          041654 001316 00718.  FMTR8M 02.00  FMTR8M.OBJ;2
                          043172 001270 00696.  FMTR8L 01.00  FMTR8L.OBJ;1
                          044462 000370 00248.  FMTINT 01.00  FMTINT.OBJ;1
IMPURE:(RW,D,LCL,REL,CON) 047162 000016 00014.
$KSTR :(RW,D,LCL,REL,CON) 047200 000031 00025.
                          047200 000031 00025.  FMTPRS 01.00  FMTPRS.OBJ;1
$KTAB :(RW,D,LCL,REL,CON) 047232 000016 00014.
                          047232 000016 00014.  FMTPRS 01.00  FMTPRS.OBJ;1
$STATE:(RW,D,LCL,REL,CON) 047250 000166 00118.
                          047250 000166 00118.  FMTPRS 01.00  FMTPRS.OBJ;1
$$RESL:(RW,I,LCL,REL,CON) 047436 000246 00166.
$$$$BF:(RW,D,LCL,REL,CON) 047704 046356 19694.
                          047704 046356 19694.  FMTDAT 01.00  FMTDAT.OBJ;1


GLOBAL SYMBOLS:

BASYR  000116    IE.BLK 177754    IE.OFL 177677    IO.DET 002000    IO.HMS 004000    IO.RLB 001000    IO.RTD 004120
DPARS  035740-R  IE.DAA 177770    IE.PRI 177760    IO.DGN 004150    IO.LPC 004100    IO.RNF 004060    IO.RVB 010400
DSKLUN 000003    IE.DNR 177775    IO.ATT 001400    IO.FER 004200    IO.OFF 004020    IO.RNR 004070    IO.TDD 004140
GENHDR 034400-R  IE.EOF 177766    IO.BLS 004010    IO.FEW 004210    IO.RDH 004030    IO.RPD 004170    IO.WCH 004110
```

IO.WCK	004050	$DSKWC	027000-R	$MGPRE	037632-R	$MSG32	030714-R	$MXTRK	027566-R	$SRALL	033654-R	$STVFL	035446-R		
IO.WDH	004040	$DSKWH	027014-R	$MOVE	031744-R	$MSG4	027662-R	$NONSI	002416	$SRFES	033726-R	$SWERL	047715-R		
IO.WPD	004160	$DSTRB	042166-R	$MSGR8	031366-R	$MSG40	030742-R	$PARSE	034536-R	$SRMDF	033740-R	$SWFFE	047721-R		
IO.WTD	004130	$DVICE	027446-R	$MSGWR	031414-R	$MSG41	031000-R	$PAT	027462-R	$SRSDF	033764-R	$SWFMT	047720-R		
IO.UMD	000004	$DVPRM	027560-R	$MSG1	027634-R	$MSG42	031026-R	$PHYDV	027606-R	$SRSSF	034046-R	$SWINT	047722-R		
MSGER8	042021-R	$DVUNT	047734-R	$MSG10	030122-R	$MSG43	031056-R	$PRINT	027256-R	$SRTMF	034166-R	$SWLOG	047717-R		
OPTKT2	047232-R	$EROCT	031600-R	$MSG11	030143-R	$MSG44	031103-R	$PRMPT	031754-R	$SRTRK	033612-R	$SWLST	047716-R		
OPTST2	047260-R	$EROUT	031536-R	$MSG12	030167-R	$MSG45	031141-R	$PRNTO	027262-R	$SRTSF	034132-R	$SWMOD	047723-R		
SECSZ	027472-R	$FEBSF	116062-R	$MSG13	030177-R	$MSG5	027707-R	$RDBSF	032322-R	$SRTUF	034222-R	$SYSIZ	003456		
SKPFRE	047724-R	$FLCYL	047744-R	$MSG15	030225-R	$MSG58	031166-R	$RDSSF	033156-R	$SRUDF	033752-R	$TRK	047754-R		
SPEFN	000003	$FLRGS	040024-R	$MSG16	030237-R	$MSG59	031233-R	$READ	027342-R	$SSF	113062-R	$TSTDA	032160-R		
TRKSZ	027470-R	$FLSEC	047750-R	$MSG17	030246-R	$MSG6	027737-R	$RETRY	027466-R	$SSFDT	001365	$TSTDV	035054-R		
UNTKT1	047232-R	$FLTRK	047746-R	$MSG18	030256-R	$MSG6A	027762-R	$REVNM	047712-R	$SSFHD	000013	$TSTMX	032216-R		
UNTST1	047250-R	$FMTEP	026502-R	$MSG19	030301-R	$MSG6B	030004-R	$RGBUF	047760-R	$SSFLB	027454-R	$UBSSZ	027574-R		
$ALFLB	027450-R	$FMTEX	031464-R	$MSG2	027645-R	$MSG7	030025-R	$RM80	035746-R	$SSFLN	000003	$UCB	047704-R		
$BUF	050042-R	$FUNC	027476-R	$MSG20	030324-R	$MSG70	031274-R	$RM80F	037226-R	$SSFSZ	027460-R	$UDBSF	112062-R		
$BUFRX	110242-R	$HDRFG	047742-R	$MSG21	030351-R	$MSG71	031307-R	$RM80I	044550-R	$STARS	027620-R	$UNTID	032266-R		
$BUFX	110242-R	$HOMSK	031710-R	$MSG22	030403-R	$MSG72	031322-R	$RM80L	043554-R	$STATS	027444-R	$UPDMX	032120-R		
$CONPB	027524-R	$INADD	047710-R	$MSG23	030427-R	$MSG73	031344-R	$RM80M	042104-R	$STCSR	035222-R	$VERIF	047725-R		
$CR	027616-R	$IOBUF	027510-R	$MSG24	030474-R	$MSG8	030043-R	$RM80V	040354-R	$STERL	035500-R	$WLKER	027250-R		
$CVTDA	031636-R	$IOST	027554-R	$MSG25	030502-R	$MSG8A	030070-R	$RTCNT	047732-R	$STFFE	035544-R	$WTBSF	032730-R		
$CYL	047752-R	$LBNH	027516-R	$MSG26	030510-R	$MSG9	030076-R	$R80SF	037156-R	$STFOR	035550-R	$WTSSF	033464-R		
$DEVHD	003334	$LBNL	027520-R	$MSG27	030540-R	$MULT	032046-R	$R80TF	037220-R	$STIN	035556-R	$XFRSZ	027512-R		
$DSKIO	027020-R	$LOGDV	027576-R	$MSG28	030574-R	$MVPAT	031504-R	$SCB	047706-R	$STLIS	035536-R	$YESNO	047714-R		
$DSKPB	027474-R	$MBSSZ	027572-R	$MSG29	030626-R	$MXCYL	027564-R	$SEC	047756-R	$STLOG	035532-R				
$DSKRH	027006-R	$MDBSF	111062-R	$MSG30	030650-R	$MXLBN	047736-R	$SMBSF	033774-R	$STUNT	034736-R				
$DSKRT	027030-R	$MESAG	031422-R	$MSG31	030671-R	$MXSEC	027570-R	$SNAME	047726-R	$STVEC	035336-R				

```
                        .TITLE  FMTROT - RM80 FORMATTER ROOT MODULE
                        .IDENT  /01.00/
                        ;
                        ;
                        ; COPYRIGHT (C) 1980
                        ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                        ;
                        ;
                        ; THIS  SOFTWARE  IS  FURNISHED  UNDER A LICENSE FOR USE ONLY ON A
                        ; SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION
                        ; OF  THE  ABOVE COPYRIGHT  NOTICE.  THIS SOFTWARE, OR  ANY  OTHER
                        ; COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE  MADE AVAILABLE
                        ; TO  ANY  OTHER PERSON  EXCEPT  FOR USE ON SUCH SYSTEM AND TO ONE
                        ; WHO AGREES TO THESE LICENSE TERMS.  TITLE  TO  AND  OWNERSHIP OF
                        ; THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DIGITAL.
                        ;
                        ; THE  INFORMATION  IN THIS DOCUMENT IS SUBJECT  TO CHANGE WITHOUT
                        ; NOTICE AND SHOULD NOT BE CONSTRUED AS A  COMMITMENT  BY  DIGITAL
                        ; EQUIPMENT CORPORATION.
                        ;
                        ; DIGITAL ASSUMES NO  RESPONSIBILITY FOR THE USE OR RELIABILITY OF
                        ; ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
                        ;
                        ;
                        ; VERSION 01.00
                        ;
                        ; C. HESS       17-AUG-80
                        ; M. LEAVITT    09-JAN-81
                        ;
                        .MCALL  ALUN$S,UCBDF$
                        
000000                          UCBDF$                  ;CALL MACRO TO DEFINE UCB OFFSETS
```

```
                                ;+
                                ; **-$FMTEP-FORMATTER ENTRY POINT
                                ;
                                ; THIS ROUTINE WILL CALL THE PARSER, SE'UP THE DISK PARAMETERS, CALL THE
                                ; ACTUAL FORMATTING ROUTINE, & CLEAN UP AFTER THE FORMAT OPERATION.
                                ;-

000000  105067  000000G        $FMTEP::CLRB    $SWERL          ; RESET ERROR LIMIT (DEFAULT IS 256.)
000004  105067  000000G                CLRB    $YESNO          ; RESET YES/NO RESPONSE
000010  105067  000000G                CLRB    $SWLST          ; RESET BAD BLOCK FILE LIST SWITCH
000014  105067  000000G                CLRB    $SWLOG          ; RESET LOGICAL LIST SWITCH
000020  105067  000000G                CLRB    $SWFMT          ; RESET FORMAT ENABLE SWITCH
000024  105067  000000G                CLRB    $SWFFE          ; RESET FORMAT FE ENABLE SWITCH
000030  105067  000000G                CLRB    $SWINT          ; RESET MDBSF, UDBSF, SSF GENERATION SWITCH
000034  105067  000000G                CLRB    $SWMOD          ; RESET THE MDBSF, SSF, & FEBSF MODIFY SWITCH
000040  105067  000001G                CLRB    $STATS+1        ; CLEAR STATUS FLAGS
000044  005067  000000G                CLR     $REVNM          ; CLEAR TEMPORARY STORAGE FOR REVISION NUMBER
000050  112767  000001  000000G        MOVB    #1,$VERIF       ; PRESET THE VERIFICATION COUNT
000056  012703  000400                 MOV     #256.,R3        ; LOCATIONS IN THE MDBSF BUFFER TO CLEAR
000062  012700  000000G                MOV     #$MDBSF,R0      ; BUFFER POINTER
000066  012720  177777         1$:     MOV     #-1,(R0)+       ; CLEAR A LOCATION
000072  005303                         DEC     R3              ; COUNT THE LOCATION
000074  001374                         BNE     1$              ; IF NE, NOT FINISHED
000076  012703  000400                 MOV     #256.,R3        ; LOCATIONS TO CLEAR IN THE UDBSF BUFFER
000102  012700  000000G                MOV     #$UDBSF,R0      ; BUFFER POINTER
000106  012720  177777         2$:     MOV     #-1,(R0)+       ; CLEAR A LOCATION
000112  005303                         DEC     R3              ; COUNT THE LOCATION
000114  001374                         BNE     2$              ; IF NE, NOT FINISHED
000116  012703  000100                 MOV     #64.,R3         ; LOCATIONS TO CLEAR IN THE FE BSF BUFFER
000122  012700  000000G                MOV     #$FEBSF,R0      ; BUFFER POINTER
000126  012720  177777         3$:     MOV     #-1,(R0)+       ; CLEAR A LOCATION
000132  005303                         DEC     R3              ; COUNT THE LOCATION
000134  001374                         BNE     3$              ; IF NE, NOT FINISHED
000136  012703  000000C                MOV     #256.*<$SSFLN>,R3       ; NUMBER OF SSF LOCATIONS
000142  012700  000000G                MOV     #$SSF,R0        ; BUFFER POINTER
000146  012720  177777         4$:     MOV     #-1,(R0)+       ; CLEAR A LOCATION
000152  005303                         DEC     R3              ; COUNT THE LOCATION
000154  001374                         BNE     4$              ; IF NE, NOT FINISHED
000156  012706  000000'                MOV     #$FMTEP,SP      ; RESET STACK POINTER
000162  026727  000000G 001400         CMP     $SYSIZ,#40*24.  ; IS THERE AT LEAST 24K?
000170  103001                         BHIS    5$              ; IF HIS, YES
000172                                 CRASH                   ; ELSE CRASH THE SYSTEM!!

000174                         5$:     CALL    $PARSE          ; PROMPT AND PARSE COMMAND LINE
000200                                 ALUN$S  #DSKLUN,$DVICE,R1       ; ASSIGN LUN TO DISK
000220  012700  000000G                MOV     #$MSG4,R0       ; ASSUME DEVICE NOT IN SYSTEM
000224  103002                         BCC     FMTPRM          ; IF CC, DEVICE IN SYSTEM
000226  000167  000000G         6$:     JMP     $MSGWR          ; EXIT WITH WARNING STATUS

                                ;
                                ; ATTACH THE DISK
                                ;

000232  012703  000000C        FMTPRM: MOV     #IO.ATT!IQ.UMD,R3       ; ATTACH DISK FOR FORMATTING
000236                                 CALL    $DSKIO

                                ;
```

```
                                ; PRESET THE DISK PARAMETERS, RETURN THE HEADS HOME AND
                                ; CALL THE ACTUAL FORMAT ROUTINE.
                                ;
000242  005067  000000G               CLR     $SEC            ; RESET DISK PARAMETERS
000246  005067  000000G               CLR     $TRK
000252  005067  000000G               CLR     $CYL
000256                                CALL    $HOMSK          ; DO A RECALIBRATE
000262                                CALL    $RM80           ; CALL THE FORMATTER
000266                                CALL    $HOMSK          ; DO A RECAL
000272  000167  000000G               JMP     $FMTEX          ; EXIT GRACEFULLY

        000000'               .END    $FMTEP
```

SYMBOL TABLE

```
C$$ORE= 002000          S$$YSZ= 001400          UC.ALG= 000200          U.CW3   000014          U2.NEC= 004000
C$$RSH= 177564          S1.BEL= 000400          UC.ATT= 000010          U.CW4   000016          U2.PRV= 000010
C$$TTY= 177564          S1.CTO= 000040          UC.KIL= 000004          U.DCB   000000          U2.RMT= 020000
DSKLUN= ****** GX       S1.CTS= 010000          UC.LGH= 000003          U.FCDE= 000040          U2.R04= 100000
DV.CCL= 000002          S1.DEC= 002000          UC.NPR= 000100          U.FNUM= 000036          U2.SCS= 000004
DV.COM= 020000          S1.DPR= 001000          UC.PWF= 000020          U.FPS = 000044          U2.SLV= 000200
DV.DIR= 000010          S1.DSI= 004000          UC.QUE= 000040          U.KCSR= 000032          U2.VT5= 000002
DV.F11= 040000          S1.ESC= 000004          US.ABO= 000001          U.KCS6= 000034          U2.7C 4= 010000
DV.ISP= 002000          S1.IBF= 100000          US.BSP= 000002          U.LUIC  177774          V$$CTR= 000400
DV.MBC= 000400          S1.IBY= 000200          US.BSY= 000200          U.OWN   177776          V$$RSN= 000032
DV.MNT= 100000          S1.OBY= 000100          US.CRW= 000004          U.RED   000002          X$$DBT= 000000
DV.MXD= 000100          S1.RAL= 000010          US.DSB= 000010          U.RPS = 000042          $CYL  = ****** GX
DV.OSP= 004000          S1.RNE= 000020          US.ECH= 000002          U.SCB   000020          $DSKIO= ****** GX
DV.PSE= 010000          S1.RST= 000001          US.FOR= 000040          U.STS   000005          $DVICE= ****** GX
DV.PEC= 000001          S1.RUB= 000002          US.FRK= 000002          U.ST2   000007          $FEBSF= ****** GX
DV.SDI= 000020          S1.USI= 020000          US.KPF= 000001          U.TCHP  000042          $FMTEP  000000RG
DV.SQD= 000040          S2.ACR= 000001          US.LAB= 000004          U.TCVP  000043          $FMTEX= ****** GX
DV.SWL= 001000          S2.BRQ= 000020          US.MDE= 000002          U.TFLK  000040          $HOMSK= ****** GX
DV.TTY= 000004          S2.CR = 000010          US.MDM= 000020          U.TFRQ  000037          $MDBSF= ****** GX
DV.UMD= 000200          S2.FDX= 100000          US.MNT= 000100          U.TLPP  000036          $MSGWR= ****** GX
D$$IAG= 000000          S2.FLF= 040000          US.OFL= 000001          U.TMTI  000047          $MSG4 = ****** GX
D$$L11= 000001          S2.HFF= 020000          US.OUT= 000001          U.TSTA  000026          $PARSE= ****** GX
FMTPRM  000232R         S2.HFL= 003400          US.PUB= 000004          U.TTAB  000034          $REVNM= ****** GX
H$$RTZ= 000074          S2.HHT= 010000          US.PWF= 000010          U.TTYP  000046          $RM80 = ****** GX
IO.ATT= ****** GX       S2.IRQ= 000200          US.RED= 000002          U.TUX   000024          $SEC  = ****** GX
IQ.UMD= ****** GX       S2.ORQ= 000100          US.SHR= 000001          U.UIC   000044          $SSF  = ****** GX
K$$CNT= 177546          S2.SRQ= 000040          US.SPU= 000002          U.UNIT  000006          $SSFLN= ****** GX
K$$CSR= 177546          S2.VFL= 004000          US.UMD= 000010          U.VCB = 000034          $STATS= ****** GX
K$$IEN= 000115          S2.WRA= 000006          US.VV = 000001          U2.AT.= 000020          $SWERL= ****** GX
K$$LDC= 000001          S2.WRB= 000002          US.WCK= 000010          U2.CRT= 002000          $SWFFE= ****** GX
K$$TPS= 000074          S3.BCC= 020000          U.ACP = 000032          U2.DH1= 100000          $SWFMT= ****** GX
LD$TT = 000000          S3.DAO= 040000          U.ATT   000022          U2.DJ1= 040000          $SWINT= ****** GX
L$$DRV= 000000          S3.PCU= 100000          U.BUF   000024          U2.DZ1= 000100          $SWLOG= ****** GX
M$$CRB= 000124          S3.RAL= 000010          U.CBF = 000032          U2.ESC= 001000          $SWLST= ****** GX
N$$LDV= 000001          S3.RCU= 000400          U.CLI   177772          U2.HFF= 010000          $SWMOD= ****** GX
P$$NIC= 177564          S3.TAB= 000100          U.CNT   000030          U2.HLD= 000040          $SYSIZ= ****** GX
R$$DER= 000000          S3.VER= 010000          U.CTL   000004          U2.LOG= 000400          $TRK  = ****** GX
R$$M11= 000001          S3.WAL= 004000          U.CTYP  000050          U2.LWC= 000001          $UDBSF= ****** GX
R$$11M= 000000          S3.WES= 000040          U.CW1   000010          U2.L3S= 000004          $VERIF= ****** GX
R$$11S= 000000          S3.8BC= 000200          U.CW2   000012          U2.L8S= 010000          $YESNO= ****** GX
S$$WRG= 000000          T$$MIN= 000000

. ABS.  177776      000
        000276      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  3572 WORDS  ( 14 PAGES)
DYNAMIC MEMORY:  7942 WORDS  ( 30 PAGES)
ELAPSED TIME:  00:00:18
[300,20]FMTROT,[300,30]FMTROT=[1,1]EXEMC/ML,[1,60]RSXMC/PA:1,[300,10]FMTROT
```

```
                        .TITLE  FMTIO - I/O ROUTINES
                        .IDENT  /01.00/
                        ;
                        ; COPYRIGHT (C) 1980
                        ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                        ;
                        ;
                        ; THIS  SOFTWARE  IS  FURNISHED  UNDER A LICENSE FOR USE ONLY ON A
                        ; SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION
                        ; OF   THE   ABOVE COPYRIGHT  NOTICE.  THIS SOFTWARE, OR  ANY  OTHER
                        ; COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE  MADE AVAILABLE
                        ; TO  ANY  OTHER PERSON  EXCEPT  FOR USE ON SUCH SYSTEM AND TO ONE
                        ; WHO AGREES TO THESE LICENSE TERMS.  TITLE  TO  AND  OWNERSHIP OF
                        ; THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DIGITAL.
                        ;
                        ; THE  INFORMATION  IN THIS DOCUMENT IS SUBJECT  TO CHANGE WITHOUT
                        ; NOTICE AND SHOULD NOT BE CONSTRUED AS A  COMMITMENT  BY  DIGITAL
                        ; EQUIPMENT CORPORATION.
                        ;
                        ; DIGITAL ASSUMES NO  RESPONSIBILITY FOR THE USE OR RELIABILITY OF
                        ; ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
                        ;
                        ;
                        ; VERSION 01.00
                        ;
                        ; C. HESS        14-AUG-80
                        ; M. LEAVITT     15-OCT-80
                        ;
                        ;
                        .MCALL  DIR$
```

```
                        ;+
                        ; **-$DSKIO-ISSUE QIO TO DISK
                        ; **-$DSKRH-ISSUE QIO TO READ HEADER & DATA
                        ; **-$DSKRT-RETRY QIO TO DISK
                        ; **-$DSKWH-ISSUE QIO TO WRITE HEADER & DATA
                        ; **-$DSKWC-ISSUE QIO TO WRITE CHECK HEADER & DATA
                        ;
                        ; SUBROUTINES TO ISSUE I/O FUNCTIONS TO A DEVICE.
                        ;
                        ; INPUTS:
                        ;       R3 = I/O FUNCTION CODE (IF ENTRY IS AT $DSKIO)
                        ;       R5 = BUFFER ADDRESS
                        ;       ENTRY AT $DSKRT ASSUMES THAT THE DPB IS ALREADY SETUP.
                        ;
                        ; OUTPUTS:
                        ;       THE QIO DIRECTIVE IS ISSUED.  IF THERE WERE NO ERRORS, THE
                        ;       CARRY BIT WILL BE CLEARED.  IF THERE WAS A PARITY ERROR
                        ;       (IE.VER) OR A BAD BLOCK ERROR (IE.BBE) CARRY WILL BE SET.
                        ;       IF WE ARE FORMATTING A "DB" DISK A BLOCK NUMBER VIOLATION
                        ;       (IE.BLK) MAY HAPPEN WHEN WE ARE FORMATTING THE LAST SECTOR.
                        ;       FOR THIS CASE CARRY WILL BE RETURNED CLEAR.  THE SAME APPLIES
                        ;       IF WE ARE RE-WRITING THE LAST TRACK OF "DM" AND "DR" DISKS.
                        ;       ALSO SPECIFIC ERROR MESSAGES WILL BE PRINTED WHEN THE
                        ;       FOLLOWING ERRORS ARE DETECTED:
                        ;
                        ;               1. IE.HWR
                        ;               2. IE.DNR
                        ;               3. IE.DAA
                        ;               4. IE.FHE
                        ;               5. IE.PRI
                        ;               6. IE.OFL
                        ;
                        ;       ANY OTHER ERROR CODE WILL BE REPORTED AS "UNRECOVERABLE".
                        ;
                        ; NOTE: MOST FUNCTIONS ARE ISSUED WITH THE DIAGNOSTIC SUB-FUNCTION
                        ;       BIT (IQ.UMD) SET.  FOR THESE FUNCTIONS THE FUNCTION WILL
                        ;       ALWAYS RETURN SUCCESS (CARRY CLEAR) AND THE USER MUST TEST
                        ;       THE PROPER BIT IN THE REGISTER BUFFER ($RGBUF) TO DETERMINE
                        ;       IF THERE WAS ANY ERRORS.
                        ;-
                        ;
                        .ENABL  LSB
000000  012703  000000C  $DSKWC::MOV     #IO.WCH!IQ.UMD,R3       ; WRITE CHECK HEADER & DATA
000004  000405           $DSKIO          ; GO TO THE COMMON QIO ROUTINE
                        ;
000006  012703  000000C  $DSKRH::MOV     #IO.RDH!IQ.UMD,R3       ;READ DISK HEADER(S)
000012  000402           $DSKIO          ;BR INTO COMMOM DISK ROUTINE
                        ;
000014  012703  000000C  $DSKWH::MOV     #IO.WDH!IQ.UMD,R3       ;WRITE DISK HEADER(S)
000020  010367  000000G  $DSKIO::MOV     R3,$FUNC                ;STORE FUNCTION CODE IN DPB
000024  010567  000000G          MOV     R5,$IOBUF               ;STORE BUFFER ADDRESS IN DPB
000030                   $DSKRT::DIR$    #$DSKPB                 ;ISSUE THE QIOW$
000036  103010                  BCC     1$                      ;IF CC OK
000040  022767  000000G 000000G         CMP     #IE.UPN,$DSW    ;POOL SPACE AVAILABLE?
000046  001770                  BEQ     $DSKRT                  ;IF EQ NO, TRY AGAIN
000050  016767  000000G 000000G         MOV     $DSW,$IOST      ;COPY DSW INTO STATUS BLOCK
000056  000406                  BR      2$
```

```
 90 000060  000241                  1$:     CLC                     ;ASSUME NO ERROR
 91 000062  105767  000000G                  TSTB    $IOST           ;ERROR?
 92 000066  100016                            BPL     6$              ;IF PL NO
 93 000070  105067  000001G                   CLRB    $IOST+1         ;YES, CLEAR THE HIGH BYTE
 94 000074  012700  000204'          2$:     MOV     #9$,R0          ;SET POINTER TO ERROR TABLE
 95 000100  021067  000000G          3$:     CMP     (R0),$IOST      ;SCAN ERROR TABLE TO FIND ERROR TYPE
 96 000104  001403                            BEQ     4$              ;IF EQ MATCH
 97 000106  022020                            CMP     (R0)+,(R0)+     ;STEP TO NEXT ENTRY
 98 000110  005710                            TST     (R0)            ;END OF TABLE?
 99 000112  001372                            BNE     3$              ;IF NE NO
100 000114  016000  000002           4$:     MOV     2(R0),R0        ;GET ADRS OF ERROR MESSAGE
101 000120  001002                            BNE     7$              ;IF NE PRINT MESSAGE
102 000122  000261                  5$:     SEC                     ;ELSE JUST SET CARRY AND EXIT
103 000124  000207                  6$:     RETURN
104
105 000126  020027  000000G          7$:     CMP     R0,#$MSG8       ;"UNRECOVERABLE" MESSAGE?
106 000132  001022                            BNE     8$              ;IF NE NO
107 000134  022767  000000C 000000G          CMP     #IE.BLK&377,$IOST       ;YES, IS THIS A BLOCK # VIOLATION?
108 000142  001767                            BEQ     5$              ;IF EQ YES
109 000144  012700  000000G                   MOV     #$MSG8A,R0      ;GET BUFFER ADDRESS FOR ERROR CODE
110 000150  005467  000000G                   NEG     $IOST           ;MAKE ERROR CODE POSITIVE
111 000154  116701  000000G                   MOVB    $IOST,R1        ;GET ERROR CODE
112 000160  005002                            CLR     R2              ;SUPRESS LEADING 0'S
113 000162  004767  000000G                   CALL    $CBDMG          ;CONVERT TO DECIMAL
114 000166  112720  000056                    MOVB    #'.,(R0)+       ;STUFF DECIMAL INDICATOR
115 000172  105010                            CLRB    (R0)            ;TERMINATE STRING
116 000174  012700  000000G                   MOV     #$MSG8,R0       ;GET ADDRESS OF "UNRECOVERABLE" MESSAGE
117 000200  000167  000000G          8$:     JMP     $MESAG          ;PRINT MESSAGE AND EXIT
118
119                                         ;+
120                                         ; ERROR CODE AND ERROR MESSAGE TABLE.  A ZERO WORD
121                                         ; IN THE MESSAGE FIELD INDICATES THAT THE ERROR WILL
122                                         ; BE HANDLED BY THE CALLING ROUTINE.
123                                         ;-
124
125 000204  000000G 000000G          9$:     .WORD   IE.HWR,         $MSG13  ; HANDLER NOT RESIDENT
126 000210  000000C 000000G                  .WORD   IE.DNR&377,     $MSG10  ; DEVICE NOT READY
127 000214  000000C 000000G                  .WORD   IE.DAA&377,     $MSG5   ; DEVICE ALREADY ATTACHED
128 000220  000000C 000000                   .WORD   IE.VER&377,     0       ; PARITY ERROR
129 000224  000000C 000000                   .WORD   IE.BBE&377,     0       ; BAD BLOCK
130 000230  000000C 000000G                  .WORD   IE.FHE&377,     $MSG20  ; FATAL HARDWARE ERROR
131 000234  000000C 000000G                  .WORD   IE.PRI&377,     $MSG22  ; PRIVLEGE VIOLATION
132 000240  000000C 000000G                  .WORD   IE.OFL&377,     $MSG7   ; DEVICE OFFLINE
133 000244  000000  000000G                  .WORD   0,              $MSG8   ; END OF TABLE
134
135                                         ;+
136                                         ; **-$WLKER-SET WRITE LOCK ERROR
137                                         ;
138                                         ; THIS ROUTINE WILL SET THE WRITE LOCK ERROR MESSAGE AND EXIT
139                                         ; WITH A SEVERE ERROR STATUS SET.
140                                         ;
141                                         ; INPUTS:
142                                         ;       NONE.
143                                         ;
144                                         ; OUTPUTS:
145                                         ;       R0 = WRITE LOCK ERROR MESSAGE ADDRESS
146                                         ;-
```

```
                                        $WLKER::MOV     #$MSG9,R0       ;SET WRITE LOCK MESSAGE ADDRESS
000254  000751                                 BR      8$              ;EXIT WITH SEVERE ERROR STATUS

                                        .DSABL  LSB


                                        ;+
                                        ; **-$PRINT-PRINT MESSAGE ON THE USERS TI:
                                        ; **-$PRNTO-PRINT MESSAGE ON THE USERS TI: (ALTERNATE ENTRY)
                                        ;
                                        ; INPUTS:
                                        ;       R0 = MESSAGE ADDRESS
                                        ;       R1 = CARRIAGE CONTROL (FOR $PRNTO ENTRY ONLY)
                                        ;
                                        ; OUTPUTS:
                                        ;       MESSAGE SENT TO TI:
                                        ;-

000256  012701  000040          $PRINT::MOV     #40,R1          ;SET 1 LINE FEED CARRIAGE CONTROL
000262  012705  000000G         $PRNTO::MOV     #$CONPB,R5      ;GET OUTPUT DPB ADDRESS
000266  012765  000000G 000000G        MOV     #IO.WVB,Q.IOFN(R5)      ;SET FUNCTION CODE
000274  010065  000000G                 MOV     R0,Q.IOPL(R5)   ;STORE MESSAGE ADDRESS IN DPB
000300  005065  000002G                 CLR     Q.IOPL+2(R5)    ;CLEAR MESSAGE LENGTH COUNTER IN DPB
000304  010165  000004G                 MOV     R1,Q.IOPL+4(R5) ;SET CARRIAGE CONTROL
000310  105720                  1$:     TSTB    (R0)+           ;END OF MESSAGE STRING?
000312  001403                          BEQ     2$              ;IF EQ YES
000314  005265  000002G                 INC     Q.IOPL+2(R5)    ;NO, CALCULATE MESSAGE LENGTH
000320  000773                          BR      1$              ;GET NEXT CHARACTER

000322                          2$:     DIR$    R5              ;PRINT THE MESSAGE
000326  103004                          BCC     3$              ;IF CC OK
000330  022767  000000G 000000G        CMP     #IE.UPN,$DSW    ;POOL SPACE AVAILABLE?
000336  001771                          BEQ     2$              ;IF EQ NO, TRY AGAIN
000340  000207                  3$:     RETURN


                                        ;+
                                        ; **-$READ-READ A LINE FROM THE TERMINAL
                                        ;
                                        ; THIS SUBROUTINE WILL READ A LINE FROM THE CONSOLE
                                        ; DEVICE.
                                        ;
                                        ; INPUTS:
                                        ;       NONE.
                                        ;
                                        ; OUTPUTS:
                                        ;       CARRY SET IF ANY ERROR AND
                                        ;       R0 = ERROR MESSAGE ADDRESS
                                        ;
                                        ;       CARRY CLEAR IF READ OK
                                        ;-

000342  012705  000000G         $READ:: MOV     #$CONPB,R5      ;GET INPUT DPB ADDRESS
000346  012765  000000G 000000G        MOV     #IO.RVB,Q.IOFN(R5)      ;LOAD FUNCTION CODE
000354  012765  000000G 000000G        MOV     #$BUFRX,Q.IOPL(R5)      ;LOAD BUFFER ADDRESS
```

```
204 000362  012765  000120  000002G        MOV     #80.,Q.IOPL+2(R5)       ;LOAD MAX NUMBER OF BYTES TO READ
205 000370                                  DIR$    R5                      ;READ IN THE LINE
206 000374  103414                          BCS     2$                      ;IF CS DIRECTIVE ERROR
207 000376  122767  000000C 000000G         CMPB    #IE.EOF&377,$IOST       ;END OF FILE SEEN?
208 000404  001406                          BEQ     1$                      ;IF EQ YES
209 000406  105767  000000G                 TSTB    $IOST                   ;I/O ERROR?
210 000412  002405                          BLT     2$                      ;IF LT YES
211 000414  105767  000001G                 TSTB    $IOST+1                 ;CHECK FOR A TERMINATED LINE
212 000420  001405                          BEQ     3$                      ;IF EQ BUFFER OVERFLOWED
213 000422  000241                  1$:     CLC                             ;INDICATE READ WAS OK
214 000424  000207                          RETURN
215
216 000426  012700  000000G          2$:     MOV     #$MSG29,R0              ;COMMAND I/O ERROR
217 000432  000402                          BR      4$
218 000434  012700  000000G          3$:     MOV     #$MSG30,R0              ;COMMAND TOO LONG
219 000440  000261                  4$:     SEC                             ;SET ERROR RETURN
220 000442  000207                          RETURN
221
222         000001                  .END
```

SYMBOL TABLE

```
IE.BBE= ****** GX       IE.VER= ****** GX       $CBDMG= ****** GX       $IOBUF= ****** GX       $MSG5 = ****** GX
IE.BLK= ****** GX       IO.RDH= ****** GX       $CONPB= ****** GX       $IOST = ****** GX       $MSG7 = ****** GX
IE.DAA= ****** GX       IO.RVB= ****** GX       $DSKIO  000020RG        $MESAG= ****** GX       $MSG8 = ****** GX
IE.DNR= ****** GX       IO.WCH= ****** GX       $DSKPB= ****** GX       $MSG10= ****** GX       $MSG8A= ****** GX
IE.EOF= ****** GX       IO.WDH= ****** GX       $DSKRH  000006RG        $MSG13= ****** GX       $MSG9 = ****** GX
IE.FHE= ****** GX       IO.WVB= ****** GX       $DSKRT  000030RG        $MSG20= ****** GX       $PRINT  000256RG
IE.HWR= ****** GX       IQ.UMD= ****** GX       $DSKWC  000000RG        $MSG22= ****** GX       $PRNTO  000262RG
IE.OFL= ****** GX       Q.IOFN= ****** GX       $DSKWH  000014RG        $MSG29= ****** GX       $READ   000342RG
IE.PRI= ****** GX       Q.IOPL= ****** GX       $DSW  = ****** GX       $MSG30= ****** GX       $WLKER  000250RG
IE.UPN= ****** GX       $BUFRX= ****** GX       $FUNC = ****** GX

. ABS.  000000     000
        000444     001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  481 WORDS  ( 2 PAGES)
DYNAMIC MEMORY:  7942 WORDS  ( 30 PAGES)
ELAPSED TIME:  00:00:10
[300,20]FMTIO,[300,30]FMTIO=[300,10]FMTIO
```

```
                        .TITLE  FMTDAT - FORMAT DATA BASE
                        .IDENT  /01.00/
                        ;
                        ;
                        ; COPYRIGHT (C) 1980
                        ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                        ;
                        ;
                        ; THIS  SOFTWARE  IS  FURNISHED  UNDER A LICENSE FOR USE ONLY ON A
                        ; SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION
                        ; OF  THE  ABOVE COPYRIGHT  NOTICE.  THIS SOFTWARE, OR  ANY  OTHER
                        ; COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE  MADE AVAILABLE
                        ; TO  ANY  OTHER PERSON  EXCEPT  FOR USE ON SUCH SYSTEM AND TO ONE
                        ; WHO AGREES TO THESE LICENSE TERMS.  TITLE  TO  AND  OWNERSHIP OF
                        ; THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DIGITAL.
                        ;
                        ; THE  INFORMATION  IN THIS DOCUMENT IS SUBJECT  TO CHANGE WITHOUT
                        ; NOTICE AND SHOULD NOT BE CONSTRUED AS A  COMMITMENT  BY  DIGITAL
                        ; EQUIPMENT CORPORATION.
                        ;
                        ; DIGITAL ASSUMES NO  RESPONSIBILITY FOR THE USE OR RELIABILITY OF
                        ; ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
                        ;
                        ;
                        ; VERSION 01.00
                        ;
                        ; C. HESS       17-AUG-80
                        ; M. LEAVITT    26-NOV-80
                        ;

                        .MCALL  UMDIO$

000000                          UMDIO$  DEF$G           ;CALL MACRO TO DEFINE DIAGNOSTIC FUNCTION
                                                        ;CODES GLOBALLY

                        ; SPECIAL OPCODES TO BE USED WITH THE FUNCTIONAL 'DR' DRIVER.  NOTE
                        ; THAT THESE FUNCTIONS ARE DEFINED IN THE SPECIAL MANUFACTURING
                        ; 'DR' DRIVER.

004110                  IO.WCH  ==      10*400+110      ; WRITE CHECK HEADER & DATA (DIAGNOSTIC)
004160                  IO.WPD  ==      10*400+160      ; WRITE PHYSICAL BLOCKS (DIAGNOSTIC)
004170                  IO.RPD  ==      10*400+170      ; READ PHYSICAL BLOCKS (DIAGNOSTIC)
004200                  IO.FER  ==      10*400+200      ; READ FE BLOCKS (DIAGNOSTIC)
004210                  IO.FEW  ==      10*400+210      ; WRITE FE BLOCKS (DIAGNOSTIC)
```

```
                                DSKEFN  =       1               ;EVENT FLAG FOR DISK I/O
                                CONEFN  =       2               ;EVENT FLAG FOR CONSOLE I/O
                                SPEFN   ==      3               ;EVENT FLAG FOR SPAWN DIRECTIVE
                                CONLUN  =       2               ;LUN FOR CONSOLE I/O
                                DSKLUN  ==      3               ;LUN FOR DISK I/O
                                DEFSEC  =       31.             ;DEFAULT SECTOR FOR SKIP FLAG
                                $SSFLN  ==      3               ;BLOCK LENGTH OF SKIP SECTOR FILE
                                $SSFHD  ==      11.             ;NUMBER OF WORDS IN HEADER OF SKIP SECTOR FILE
                                $SSFDT  ==      <256.*$SSFLN>-$SSFHD    ;NUMBER OF WORDS IN DATA OF SKIP SECTOR FILE

000000    000     377           $STATS::.BYTE   0,-1            ; FORMAT STATUS, MCR LINE FLAG
000002    104     122           $DVICE::.ASCII  "DR"            ; DISK NAME (ASCII)
000004  000003  131715           $ALFLB::.WORD   3,131715        ; ADDRESS OF ALTERNATE DEC STD 144 TRACK
000010  000003  131656           $SSFLB::.WORD   3,131656        ; ADDRESS OF START OF SKIP SECTOR FILE
000014  000005                   $SSFSZ::.WORD   5               ; NUMBER OF COPIES OF SKIP SECTOR FILE
000016  155555  133333           $PAT::  .WORD   155555,133333   ; DISK FORMAT PATTERN
000022  000024                   $RETRY::.WORD   20.             ; NUMBER OF RE-VERIFICATION ATTEMPTS
000024  040200                   TRKSZ:: .WORD   516.*32.        ; RM80 TRACK SIZE (IN BYTES)
000026  001004                   SECSZ:: .WORD   516.            ; RM80 SECTOR SIZE (INCLUDING HEADER)

                                ;DISK I/O DIRECTIVE PARAMETER BLOCK (QIOW$ DIRECTIVE)

000030    003     014           $DSKPB::.BYTE   3,12.           ; MACRO DIC, DPB SIZE
000032  000000                   $FUNC:: .WORD   0               ; FUNCTION CODE
000034  000003                           .WORD   DSKLUN          ; LUN
000036    001     000                    .BYTE   DSKEFN,0        ; EVENT FLAG, PRIORITY
000040  000110'                          .WORD   $IOST           ; STATUS BLOCK ADDRESS
000042  000000                           .WORD   0               ; AST ADDRESS
000044  000000                   $IOBUF::.WORD   0               ; BUFFER ADDRESS
000046  000000                   $XFRSZ::.WORD   0               ; TRANSFER SIZE IN BYTES
000050  000000                           .WORD   0               ; NOT USED
000052  000000                   $LBNH:: .WORD   0               ; HIGH ORDER LOGICAL BLOCK NUMBER
000054  000000                   $LBNL:: .WORD   0               ; LOW ORDER LOGICAL BLOCK NUMBER
000056  000054'                          .WORD   $RGBUF          ; REGISTER BUFFER ADDRESS

                                ;CONSOLE I/O DIRECTIVE PARAMETER BLOCK (QIOW$ DIRECTIVE)

000060    003     014           $CONPB::.BYTE   3,12.           ; MACRO DIC, DPB SIZE
000062  000000                           .WORD   0               ; FUNCTION CODE
000064  000002                           .WORD   CONLUN          ; CONSOLE LUN
000066    002     000                    .BYTE   CONEFN,0        ; CONSOLE EVENT FLAG, PRIORITY ZERO
000070  000110'                          .WORD   $IOST           ; STATUS BLOCK ADDRESS
000072  000000                           .WORD   0               ; AST ADDRESS
000074  000000                           .WORD   0               ; BUFFER ADDRESS
000076  000000                           .WORD   0               ; MESSAGE BYTE COUNT
000100  000000                           .WORD   0               ; CARRIAGE CONTROL
000102  000000  000000  000000           .WORD   0,0,0           ; NOT USED

000110  000000  000000           $IOST:: .WORD   0,0             ; I/O STATUS BLOCK

                                ;
                                ; DRIVE PARAMETER TABLE
                                ;

000114  000003  131656           $DVPRM::.WORD   3,131656        ; MAXIMUM LBN

                                ; THE ORDER OF THIS TABLE MUST NOT CHANGE
```

```
                                ;
                                ; MACRO FORMAT:  LBN, CYL, TRK, SEC
                                ; WHERE:
                                ;       LBN =   MAXIMUM NUMBER OF BLOCKS ON DEVICE (DOUBLE WORD)
                                ;       CYL =   MAXIMUM NUMBER OF CYLINDERS ON DEVICE
                                ;       TRK =   MAXIMUM NUMBER OF TRACKS ON DEVICE
                                ;       SEC =   MAXIMUM NUMBER OF SECTORS ON DEVICE
                                ;
                                ;-
                                .MACRO  DVPRM   LBN, CYL, TRK, SEC
                                        .WORD   LBN
                                        .WORD   CYL
                                        .BYTE   SEC
                                        .BYTE   TRK
                                .ENDM   DVPRM

                                $LOGDV::DVPRM   <3,131656>, 559., 14., 31.      ; LOGICAL DEVICE GEOMETRY
                                $PHYDV::DVPRM   <3,152700>, 561., 14., 32.      ; PHYSICAL DEVICE GEOMETRY

                                .PSECT  $$$$BF,D

000000  000000                 $UCB::  .WORD   0               ; ADDRESS OF CURRENT UCB
000002  000000                 $SCB::  .WORD   0               ; ADDRESS OF CURRECT SCB
000004  000000                 $INADD::.WORD   0               ; ADDRESS OF INTERRUPT VECTOR
000006  000000                 $REVNM::.WORD   0               ; TEMP. STORAGE FOR SSF REVISION NUMBER ($SSF*10)
000010     000                 $YESNO::.BYTE   0               ; YES/NO CONTINUE RESPONSE
000011     000                 $SWERL::.BYTE   0               ; ERROR LIMIT
000012     000                 $SWLST::.BYTE   0               ; LIST BAD BLOCK FILE CONTENTS
000013     000                 $SWLOG::.BYTE   0               ; LOGICAL LIST SWITCH
000014     000                 $SWFMT::.BYTE   0               ; ENABLE FORMAT FLAG
000015     000                 $SWFFE::.BYTE   0               ; ENABLE FORMAT FE CYLINDERS ONLY
000016     000                 $SWINT::.BYTE   0               ; CREATE THE MDBSF, UDBSF, AND SSF
000017     000                 $SWMOD::.BYTE   0               ; MODIFY THE SSF, UDBSF, & FEBSF
000020     000                 SKPFRE::.BYTE   0               ; SKIP SECTOR FLAG
000021     000                 $VERIF::.BYTE   0               ; NUMBER OF TIMES TO VERIFY THE TRACK

                                .EVEN
000022  000000  000000          $SNAME  .WORD   0,0             ; HDA SERIAL NUMBER
000026  000000                  $RTCNT::.WORD   0               ; RETRY COUNT
000030  000000                  $DVUNT::.WORD   0               ; DISK UNIT NUMBER
000032  000000  000000          $MXLBN::.WORD   0,0             ; MAXIMUM BLOCK NUMBER
000036  000000                  $HDRFG::.WORD   0               ; HEADER DEFECT FLAG
000040  000000                  $FLCYL::.WORD   0               ; FAILING CYLINDER ADDRESS
000042  000000                  $FLTRK::.WORD   0               ;       "       TRACK       "
000044  000000                  $FLSEC::.WORD   0               ;       "       SECTOR      "
000046  000000                  $CYL::  .WORD   0               ; CURRENT CYLINDER
000050  000000                  $TRK::  .WORD   0               ; CURRENT TRACK
000052  000000                  $SEC::  .WORD   0               ; CURRENT SECTOR

                                ;BUFFERS AND DATA STORAGE AREA
```

```
                                ;RMBO REGISTER RETURN BUFFER
000054                          $RGBUF::.BLKW   25.
                                ;BUFFER FOR DISK I/O
000136                          $BUF::  .BLKW   32.*258.
                                ;DIRECTIVE BUFFER
040336                          $BUFRX::.BLKW   200.
040336'         $BUFX==$BUFRX   ;BUFFER FOR DIRECTIVE USE

                                ;BAD SECTOR AND DEFECT STORAGE AREA

                                ;BUFFER FOR 16 BIT MDBSF
041156                          $MDBSF::.BLKW   256.
                                ;BUFFER FOR 16 BIT UDBSF
042156                          $UDBSF::.BLKW   256.
                                ;BUFFER FOR 16 BIT SKIP SECTOR FILE
043156                          $SSF::  .BLKW   256.*<$SSFLN>
                                ;BUFFER FOR FE CYLINDER DEFECTS
046156                          $FEBSF::.BLKW   64.

000001                          .END
```

SYMBOL TABLE

```
CONEFN= 000002          IO.WCH= 004110 G        $DVPRM  000114RG        $MXLBN  000032RG    002 $SSFSZ  000014RG
CONLUN= 000002          IO.WCK= 004050 G        $DVUNT  000030RG    002 $MXSEC  000124RG        $STATS  000000RG
DEFSEC= 000037          IO.WDH= 004040 G        $FEBSF  046156RG    002 $MXTRK  000122RG        $SWERL  000011RG    002
DSKEFN= 000001          IO.WPD= 004160 G        $FLCYL  000040RG    002 $PAT    000016RG        $SWFFE  000015RG    002
DSKLUN= 000003 G        IO.WTD= 004130 G        $FLSEC  000044RG    002 $PHYDV  000142RG        $SWFMT  000014RG    002
IO.BLS= 004010 G        IQ.UMD= 000004 G        $FLTRK  000042RG    002 $RETRY  000022RG        $SWINT  000016RG    002
IO.DGN= 004150 G        SECSZ   000026RG        $FUNC   000032RG        $REVNM  000006RG    002 $SWLOG  000013RG    002
IO.FER= 004200 G        SKPFRE  000020RG    002 $HDRFG  000036RG    002 $RGBUF  000054RG    002 $SWLST  000012RG    002
IO.FEW= 004210 G        SPEFN = 000003 G        $INADD  000004RG    002 $RTCNT  000026RG    002 $SWMOD  000017RG    002
IO.HMS= 004000 G        TRKSZ   000024RG        $IOBUF  000044RG        $SCB    000002RG    002 $TRK    000050RG    002
IO.LPC= 004100 G        $ALFLB  000004RG        $IOST   000110RG        $SEC    000052RG    002 $UBSSZ  000130RG
IO.OFF= 004020 G        $BUF    000136RG    002 $LBNH   000052RG        $SNAME  000022RG    002 $UCB    000000RG    002
IO.RDH= 004030 G        $BUFRX  040336RG    002 $LBNL   000054RG        $SSF    043156RG    002 $UDBSF  042156RG    002
IO.RNF= 004060 G        $BUFX = 040336RG    002 $LOGDV  000132RG        $SSFDT= 001365 G        $VERIF  000021RG    002
IO.RNR= 004070 G        $CONPB  000060RG        $MBSSZ  000126RG        $SSFHD= 000013 G        $XFRSZ  000046RG
IO.RPD= 004170 G        $CYL    000046RG    002 $MDBSF  041156RG    002 $SSFLB  000010RG        $YESNO  000010RG    002
IO.RTD= 004120 G        $DSKPB  000030RG        $MXCYL  000120RG        $SSFLN= 000003 G        ...GBL= 000001
IO.TDD= 004140 G        $DVICE  000002RG

. ABS.  000000      000
        000152      001
$$$$BF  046356      002
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  898 WORDS  ( 4 PAGES)
DYNAMIC MEMORY:  7942 WORDS  ( 30 PAGES)
ELAPSED TIME:  00:00:12
[300,20]FMTDAT,[300,30]FMTDAT=[300,10]FMTDAT
```

```
                        .TITLE  FMTMSG - MESSAGE OUTPUT ROUTINES
                        .IDENT  /01.00/
                        ;
                        ; COPYRIGHT (C) 1980
                        ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                        ;
                        ;
                        ; THIS  SOFTWARE  IS  FURNISHED  UNDER A LICENSE FOR USE ONLY ON A
                        ; SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION
                        ; OF  THE  ABOVE COPYRIGHT  NOTICE.  THIS SOFTWARE, OR  ANY  OTHER
                        ; COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE  MADE AVAILABLE
                        ; TO  ANY  OTHER PERSON  EXCEPT  FOR USE ON SUCH SYSTEM AND TO ONE
                        ; WHO AGREES TO THESE LICENSE TERMS.  TITLE  TO  AND  OWNERSHIP OF
                        ; THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DIGITAL.
                        ;
                        ; THE  INFORMATION  IN THIS DOCUMENT IS SUBJECT  TO CHANGE WITHOUT
                        ; NOTICE AND SHOULD NOT BE CONSTRUED AS A  COMMITMENT  BY  DIGITAL
                        ; EQUIPMENT CORPORATION.
                        ;
                        ; DIGITAL ASSUMES NO  RESPONSIBILITY FOR THE USE OR RELIABILITY OF
                        ; ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
                        ;
                        ;
                        ; VERSION 01.00
                        ;
                        ; C. HESS       15-AUG-80
                        ; M. LEAVITT    23-OCT-80
                        ;
                        .NLIST  CND,BEX
                        .MCALL  DIR$
```

```
 35
 38
 39 000000  015  000       $CR::   .ASCIZ  <CR>
 40 000002  052  052  052  $STARS::.ASCIZ  /**********/<LF>
 41 000016  015  106  115  $MSG1:: .ASCIZ  <CR>/FMT -- /
 42 000027  123  131  116  $MSG2:: .ASCIZ  /SYNTAX ERROR/
 43 000044  104  105  126  $MSG4:: .ASCIZ  /DEVICE NOT IN SYSTEM/
 44 000071  106  101  111  $MSG5:: .ASCIZ  /FAILED TO ATTACH DEVICE/
 45 000121  012  012  123  $MSG6:: .ASCIZ  <LF><LF>/START FORMATTING/
 46 000144  040  055  101  $MSG6A::.ASCIZ  / -ALL CYLINDERS-/<CR>
 47 000166  040  055  106  $MSG6B::.ASCIZ  / -FE CYLINDERS-/<CR>
 48 000207  104  122  111  $MSG7:: .ASCIZ  /DRIVE OFFLINE/
 49 000225  125  116  122  $MSG8:: .ASCII  /UNRECOVERABLE ERROR -/
 50 000252  040  040  040  $MSG8A::.ASCIZ  /     /
 51 000260  104  105  126  $MSG9:: .ASCIZ  /DEVICE WRITE LOCKED/
 52 000304  104  105  126  $MSG10::.ASCIZ  /DEVICE NOT READY/
 53 000325  012  123  124  $MSG11::.ASCIZ  <LF>/START VERIFICATION/
 54 000351  012  104  117  $MSG12::.ASCIZ  <LF>/DONE !/
 55 000361  104  105  126  $MSG13::.ASCIZ  /DEVICE DRIVER MISSING/
 56 000407  103  131  114  $MSG15::.ASCIZ  /CYLINDER=/
 57 000421  124  122  101  $MSG16::.ASCIZ  /TRACK=/
 58 000430  123  105  103  $MSG17::.ASCIZ  /SECTOR=/
 59 000440  115  104  102  $MSG18::.ASCIZ  /MDBSF COPY CORRUPT/
 60 000463  125  104  102  $MSG19::.ASCIZ  /UDBSF COPY CORRUPT/
 61 000506  106  101  124  $MSG20::.ASCIZ  /FATAL HARDWARE ERROR/
 62 000533  106  101  111  $MSG21::.ASCIZ  /FAILED TO READ MDBSF COPY/
 63 000565  120  122  111  $MSG22::.ASCIZ  /PRIVILEGE VIOLATION/
 64 000611  012  007  052  $MSG23::.ASCIZ  <LF><07>/** WARNING - DATA WILL BE LOST ON /
 65 000656  072  040  052  $MSG24::.ASCIZ  /: **/<07>
 66 000664  015  106  115  $MSG25::.ASCIZ  <CR>/FMT>/
 67 000672  125  116  101  $MSG26::.ASCIZ  /UNABLE TO RECOVER MDBSF/
 68 000722  126  105  103  $MSG27::.ASCIZ  /VECTOR NOT MULTIPLE OF FOUR/
 69 000756  103  123  122  $MSG28::.ASCIZ  /CSR ADDRESS NOT IN SYSTEM/
 70 001010  103  117  115  $MSG29::.ASCIZ  @COMMAND I/O ERROR@
 71 001032  103  117  115  $MSG30::.ASCIZ  /COMMAND TOO LONG/
 72 001053  012  103  117  $MSG31::.ASCIZ  <LF>/CONTINUE (L) N ? /
 73 001076  122  105  123  $MSG32::.ASCIZ  /RESPONSE OUT OF RANGE/
 74 001124  111  116  103  $MSG40::.ASCIZ  /INCORRECT MDBSF SERIAL NUMBER/
 75 001162  125  116  101  $MSG41::.ASCIZ  /UNABLE TO RECOVER SSF/
 76 001210  125  116  101  $MSG42::.ASCIZ  /UNABLE TO RECOVER UDBSF/
 77 001240  125  116  101  $MSG43::.ASCIZ  /UNABLE TO READ UDBSF/
 78 001265  111  116  103  $MSG44::.ASCIZ  /INCORRECT UDBSF SERIAL NUMBER/
 79 001323  105  122  122  $MSG45::.ASCIZ  /ERROR LIMIT EXCEEDED/
 80 001350  125  116  101  $MSG58::.ASCIZ  /UNABLE TO READ SKIP SECTOR FILE COPY/
 81 001415  123  113  111  $MSG59::.ASCIZ  /SKIP SECTOR FILE COPY IS CORRUPT/
 82 001456  015  104  122  $MSG70::.ASCIZ  <CR>/DRIVE #: /
 83 001471  015  117  120  $MSG71::.ASCIZ  <CR>/OPTIONS: /
 84 001504  104  122  111  $MSG72::.ASCIZ  /DRIVE NOT PRESENT/
 85 001526  104  122  111  $MSG73::.ASCIZ  /DRIVE NOT AN RM80/
 86 001550  125  120  104  $MSGR8::.ASCIZ  /UPDATE MODE (L) N ? /
 87
 88                       .EVEN
```

```
                                        ;+
                                        ; **-$MESAG-OUTPUT MESSAGE WITH CURRENT EXIT STATUS
                                        ; **-$MSGWR-OUTPUT MESSAGE WITH WARNING EXIT STATUS
                                        ;
                                        ; INPUTS:
                                        ;       R0 = ADDRESS OF "ASCIZ" MESSAGE STRING
                                        ;-
                                        
001576  112767  000000G 000000G $MSGWR::MOVB    #EX$WAR,$STATS  ;SET WARNING EXIT STATUS
001604  010046                  $MESAG::MOV     R0,-(SP)        ;SAVE MESSAGE ADDRESS
001606  012700  000000G                 MOV     #$BUF,R0        ;GET PRINT BUFFER ADDRESS
001612  012703  000016'                MOV     #$MSG1,R3       ;GET PREFIX ADDRESS
001616  004767  000000G                CALL    $MOVE           ;STORE PREFIX
001622  012603                        MOV     (SP)+,R3        ;GET BASIC MESSAGE ADDRESS
001624  004767  000000G                CALL    $MOVE           ;STORE BASIC MESSAGE
001630  112720  000012                MOVB    #12,(R0)+       ;ADD AN ADDITIONAL LINE FEED
001634  105010                        CLRB    (R0)            ;TERMINATE THE BUFFER
001636  012700  000000G                MOV     #$BUF,R0        ;SET MESSAGE BUFFER ADDRESS
001642  004767  000000G                CALL    $PRINT          ;PRINT THE MESSAGE
                                        
001646  012767  000000C 000000G $FMTEX::MOV     #IO.DET!IQ.UMD,$FUNC    ;LOAD DETACH FUNCTION CODE
001654                                  DIR$    #$DSKPB         ;DETACH DISK (WHO CARES IF IT FAILS!!)
001662  000167  000000G                JMP     $FMTEP          ;RESTART
                                        
000001                          .END
```

SEQ 0040

SYMBOL TABLE

```
CR     = 000015          $MSGR8  001550RG        $MSG20  000506RG        $MSG4   000044RG        $MSG6B  000166RG
EX$WAR= ****** GX        $MSGWR  001576RG        $MSG21  000533RG        $MSG40  001124RG        $MSG7   000207RG
IQ.DET= ****** GX        $MSG1   000016RG        $MSG22  000565RG        $MSG41  001162RG        $MSG70  001456RG
IQ.UMD= ****** GX        $MSG10  000304RG        $MSG23  000611RG        $MSG42  001210RG        $MSG71  001471RG
LF     = 000012          $MSG11  000325RG        $MSG24  000656RG        $MSG43  001240RG        $MSG72  001504RG
$BUF   = ****** GX       $MSG12  000351RG        $MSG25  000664RG        $MSG44  001265RG        $MSG73  001526RG
$CR      000000RG        $MSG13  000361RG        $MSG26  000672RG        $MSG45  001323RG        $MSG8   000225RG
$DSKPB= ****** GX        $MSG15  000407RG        $MSG27  000722RG        $MSG5   000071RG        $MSG8A  000252RG
$FMTEP= ****** GX        $MSG16  000421RG        $MSG28  000756RG        $MSG58  001350RG        $MSG9   000260RG
$FMTEX  001646RG         $MSG17  000430RG        $MSG29  001010RG        $MSG59  001415RG        $PRINT= ****** GX
$FUNC  = ****** GX       $MSG18  000440RG        $MSG30  001032RG        $MSG6   000121RG        $STAPS  000002RG
$MESAG  001604RG         $MSG19  000463RG        $MSG31  001053RG        $MSG6A  000144RG        $STATS= ****** GX
$MOVE  = ****** GX       $MSG2   000027RG        $MSG32  001076RG

. ABS.  000000      000
        001666      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  618 WORDS  ( 3 PAGES)
DYNAMIC MEMORY:  7942 WORDS  ( 30 PAGES)
ELAPSED TIME:  00:00:10
[300,20]FMTMSG,[300,30]FMTMSG=[300,10]FMTMSC
```

```
                        .TITLE  FMTSUB - FORMAT SUBROUTINES
                        .IDENT  /02.00/
                        ;
                        ;
                        ; COPYRIGHT (C) 1980,1982
                        ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                        ;
                        ;
                        ; THIS  SOFTWARE  IS  FURNISHED  UNDER A LICENSE FOR USE ONLY ON A
                        ; SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION
                        ; OF  THE  ABOVE COPYRIGHT  NOTICE.  THIS SOFTWARE, OR  ANY  OTHER
                        ; COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE  MADE AVAILABLE
                        ; TO  ANY  OTHER PERSON  EXCEPT  FOR USE ON SUCH SYSTEM AND TO ONE
                        ; WHO AGREES TO THESE LICENSE TERMS.  TITLE  TO  AND  OWNERSHIP OF
                        ; THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DIGITAL.
                        ;
                        ; THE  INFORMATION  IN THIS DOCUMENT IS SUBJECT  TO CHANGE WITHOUT
                        ; NOTICE AND SHOULD NOT BE CONSTRUED AS A  COMMITMENT  BY  DIGITAL
                        ; EQUIPMENT CORPORATION.
                        ;
                        ; DIGITAL ASSUMES NO  RESPONSIBILITY FOR THE USE OR RELIABILITY OF
                        ; ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
                        ;
                        ;
                        ; VERSION 02.00
                        ;
                        ; C. HESS       21-AUG-80
                        ; M. LEAVITT    02-JUL-82
                        ;
```

```
                        ;+
                        ; **-$MVPAT-LOAD THE DATA BUFFER WITH THE DISK DATA PATTERN
                        ;
                        ; INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       '$BUF' LOADED WITH '$PAT' & '$PAT+2'
                        ;
                        ;-
                        ;
000000  016701  000000G $MVPAT::MOV     TRKSZ,R1        ; LOAD THE DISTRIBUTION COUNTER
000004  006201          ASR     R1              ; CONVERT BYTE COUNT TO WORD COUNT
000006  006201          ASR     R1              ; CONVERT TO 2 WORD COUNT
000010  012700  000000G         MOV     #$BUF,R0        ; BUFFER POINTER
000014  016720  000000G 1$:     MOV     $PAT,(R0)+      ; MOVE THE PATTERN
000020  016720  000002G         MOV     $PAT+2,(R0)+    ;
000024  005301          DEC     R1              ; DECREMENT THE DISTRIBUTION COUNTER
000026  100372          BPL     1$              ; IF PL, MORE TO DO
000030  000207          RETURN                  ;
                        ;
                        ;+
                        ; OUTPUT ERROR MESSAGE
                        ;
                        ; INPUTS:
                        ;       R1 = VALUE TO CONVERT
                        ;       R3 = ADDRESS OF BASIC MESSAGE
                        ;
                        ; OUTPUTS:
                        ;       NONE.
                        ;-
                        ;
000032  012700  000000G $EROUT::MOV     #$BUFRX,R0      ; GET OUTPUT BUFFER ADDDRESS
000036  004767  000176          CALL    $MOVE           ;STORE BASIC MESSAGE
000042  005002          CLR     R2              ;SUPRESS LEADING 0'S
000044  004767  000000G         CALL    $CBDMG          ;CONVERT TO DECIMAL
000050  112720  000056 ERROU1: MOVB    #'.,(R0)+       ;STUFF DECIMAL INDICATOR
000054  112720  000040         MOVB    #' ,(R0)+       ;STUFF BLANK IN MESSAGE
000060  105010          CLRB    (R0)            ;TERMINATE STRING
000062  005001          CLR     R1              ;SET NULL CARRIAGE CONTROL
000064  012700  000000G         MOV     #$BUFRX,R0      ; GET OUTPUT BUFFER ADDRESS
000070  000167  000000G         JMP     $PRNTO          ;PRINT MESSAGE
                        ;
                        ;+
                        ; **-$EROCT-CONVERT TO OCTAL AND PRINT
                        ;       INPUTS:
                        ;               R1 = VALUE TO CONVERT
                        ;               R3 = MESSAGE ADDRESS
                        ;       OUTPUTS:
                        ;               MESSAGE PRINTED ON TI:
                        ;-
                        ;
000074  012700  000000G $EROCT::MOV     #$BUFRX,R0      ;
000100  004767  000134          CALL    $MOVE           ;
000104  105202          INCB    R2              ;INCLUDE LEADING ZEROS
000106  004767  000000G         CALL    $CBOMG          ;CONVERT TO OCTAL
000112  112720  000040         MOVB    #' ,(R0)+       ;STUFF BLANK IN MESSAGE
```

```
                        CLRB    (R0)            ;TERMINATE STRING
                        CLR     R1              ;SET NULL CARRIAGE CONTROL
                        MOV     #$BUFRX,R0      ;
                        CALL    $PRNTO          ;SEND MESSAGE


                ;+
                ; **-$CVTDA-CONVERT DISK ADDRESS TO LOGICAL BLOCK NUMBER
                ;
                ; THIS SUBROUTINE CONVERTS A DISK ADDRESS TO A LOGICAL
                ; BLOCK NUMBER.  THE FORMULA USED IS:
                ;       BLK = ($CYL * $MXTRK + $TRK) * $MXSEC + $SEC
                ;
                ; INPUTS:
                ;       $CYL = CYLINDER NUMBER
                ;       $TRK = TRACK NUMBER
                ;       $SEC = SECTOR NUMBER
                ;
                ; OUTPUTS:
                ;       $LBNH = HIGH ORDER BLOCK NUMBER
                ;       $LBNL = LOW ORDER BLOCK NUMBER
                ;-

$CVTDA::MOV     $CYL,R1         ; GET CYLINDER NUMBER
        MOV     $MXTRK,R0       ; GET TRACKS PER CYLINDER
        CALL    $MULT           ; MULTIPLY
        MOV     $TRK,R0         ; GET TRACK NUMBER
        ADD     R1,R0           ; ADD TO PARTIAL TOTAL
        MOV     $MXSEC,R1       ; GET SECTORS PER TRACK
        CALL    $MULT           ; MULTIPLY
        ADD     $SEC,R1         ; ADD IN SECTOR
        ADC     R0              ; DON'T FORGET CARRY
        MOV     R0,$LBNH        ; SAVE HIGH ORDER BLOCK NUMBER
        MOV     R1,$LBNL        ; SAVE LOW ORDER BLOCK NUMBER
        RETURN                  ;


                ;+
                ; **-$HOMSK-ISSUE A HOME SEEK FUNCTION
                ;
                ; THIS SUBROUTINE WILL ISSUE A HOME SEEK FUNCTION.
                ;
                ; INPUTS:
                ;       NONE.
                ;
                ; OUTPUTS:
                ;       HOME SEEK ISSUED TO DISK.
                ;-

$HOMSK::CLR     $LBNH           ;RESET BLOCK NUMBER
        CLR     $LBNL           ;
        MOV     #10.,$XFRSZ     ;DUMMY UP BYTE COUNT
        MOV     #IO.HMS!IQ.UMD,R3       ;SET HOME SEEK FUNCTION CODE
        MOV     #$BUF,R5        ;SETUP DEFAULT BUFFER ADDRESS
        JMP     $DSKIO          ;DO THE FUNCTION AND EXIT
1$:     RETURN
```

```
                                ;+
                                ; **-$MOVE-MOVE ASCIZ STRING INTO BUFFER
                                ;
                                ; THIS SUBROUTINE WILL MOVE AN ASCIZ STRING INTO A BUFFER.
                                ;
                                ; INPUTS:
                                ;       R0 = ADDRESS OF BUFFER.
                                ;       R3 = ADDRESS OF ASCIZ TEXT TO MOVE.
                                ;
                                ; OUTPUTS:
                                ;       R0 = POINTS TO NEXT PLACE IN BUFFER.
                                ;-
                                
000240  112320                 $MOVE:: MOVB    (R3)+,(R0)+     ;STORE TEXT CHARACTER
000242  001376                         BNE     $MOVE           ;IF NE CONTINUE TO STORE
000244  105740                         TSTB    -(R0)           ;POINT TO NEXT BUFFER POSITION
000246  000207                         RETURN                  ;
                                
                                ;+
                                ; **-$PRMPT-PROMPT USER AND SOLICIT A RESPONSE
                                ;
                                ; THIS SUBROUTINE WILL PROMPT THE USER AND THEN WAIT FOR HIS
                                ; RESPONSE.
                                ;
                                ; INPUTS:
                                ;       R0 = ADDRESS OF PROMPT MESSAGE.
                                ;
                                ; OUTPUTS:
                                ;       R3 = BYTE COUNT
                                ;       BUFFER IS TERMINATED
                                ;       Z = 1 IF BYTE COUNT IS ZERO
                                ;       Z = 0 IF BYTE COUNT IS NON-ZERO
                                ;-
                                
000250  012701  000044         $PRMPT::MOV     #'$,R1          ;SET CARRIAGE CONTROL
000254  004767  000000G                 CALL    $PRNTO          ;PROMPT THE USER
000260  005067  000000G                 CLR     $BUFRX          ;CLEAR USER'S RESPONSE BUFFER
000264  004767  000000G                 CALL    $READ           ;GET THE USERS RESPONSE
000270  103422                         BCS     MSG             ;IF CS ERROR
000272  016703  000002G                 MOV     $IOST+2,R3      ;GET BYTE COUNT
000276  105063  000000G                 CLRB    $BUFRX(R3)      ;TERMINATE THE LINE
000302  012700  000000G                 MOV     #$BUFRX,R0      ;GET INPUT BUFFER ADDRESS
000306  121027  000141         1$:     CMPB    (R0),#141       ;LOWER CASE LETTER?
000312  103405                         BLO     2$              ;IF LO NO
000314  121027  000172                 CMPB    (R0),#172       ;LOWER CASE LETTER?
000320  101002                         BHI     2$              ;IF HI NO
000322  142710  000040                 BICB    #40,(R0)        ;YES, CONVERT TO UPPER CASE
000326  105720                 2$:     TSTB    (R0)+           ;END OF INPUT BUFFER?
000330  001366                         BNE     1$              ;IF NE NO
000332  005703                         TST     R3              ;TEST FOR NULL LINE
000334  000207                         RETURN                  ;
000336  000167  000000G         MSG:    JMP     $MSGWR          ;EXIT WITH WARNING STATUS
                                
                                ;+
                                ; **-$MULT-INTEGER MULTIPLY ROUTINE
                                ;
                                ; INPUTS:
```

```
                        ;       R0 = MULTIPLIER
                        ;       R1 = MULTIPLICAND
                        ;
                        ; OUTPUTS:
                        ;       R0 = HIGH ORDER RESULT
                        ;       R1 = LOW ORDER RESULT
                        ;
                        ; ALL REGISTERS ARE PRESERVED
                        ;
                        ;-
000342  010046          $MULT:: MOV     R0,-(SP)        ; SAVE MULTIPLIER FOR ADDS
000344  012746  000021          MOV     #21,-(SP)       ; SET LOOP COUNT
000350  005000                 CLR     R0              ; CLEAR HIGH PART
000352  006000          1$:     ROR     R0              ; DOUBLE RIGHT SHIFT
000354  006001                 ROR     R1              ;
000356  103002                 BCC     2$              ; IF CC, DO NOT ADD
000360  066600  000002         ADD     2(SP),R0        ; ADD MULTIPLIER TO RESULT
000364  005316          2$:     DEC     (SP)            ; DECREMENT THE LOOP COUNT
000366  003371                 BGT     1$              ; IF GT, MORE TO DO
000370  022626                 CMP     (SP)+,(SP)+     ; PRUNE STACK
000372  000207                 RETURN                  ;
                        ;+
                        ; **-SSFCRC-COMPUTE CHECKSUM OVER SSF DATA
                        ;
                        ; THIS ROUTINE WILL COMPUTE THE CHECKSUM OVER THE SPECIFIED
                        ; DATA AREA OF AN SSF COPY.
                        ;
                        ; INPUTS:
                        ;       R0 = STARTING ADDRESS
                        ;       R1 = LENGTH OF DATA AREA
                        ;
                        ; OUTPUTS:
                        ;       R0 =    POINTS TO CHECKSUM WORD
                        ;       R1 =    CHECKSUM
                        ;-
000374  012746  000001  SSFCRC: MOV     #1,-(SP)        ; INITIALIZE CHECKSUM
000400  062016          1$:     ADD     (R0)+,(SP)      ; ADD DATA WORD TO CHECKSUM
000402  006116                 ROL     (SP)            ; ROTATE CHECKSUM
000404  005301                 DEC     R1              ; ALL DONE ?
000406  001374                 BNE     1$              ; IF NE, NO
000410  012601                 MOV     (SP)+,R1        ; GET CHECKSUM RESULT
000412  000207                 RETURN                  ;
                        ;+
                        ; **-$UPDMX-UPDATE MAXIMUM VALUES FOR DEVICE GEOMETRY
                        ;
                        ; THIS ROUTINE WILL REPLACE THE CURRENT MAXIMUM VALUES FOR THE
                        ; RM80'S CURRENT GEOMETRY WITH THE VALUES SUPPLIED IN THE
                        ; SPECIFIED TABLE.
                        ;
                        ; INPUTS:
                        ;       R3 =    POINTER TO SPECIFIED TABLE
                        ;
                        ; OUTPUTS:
```

```
                        ;       R3 =    POINTS PAST END OF SPECIFIED TABLE
                        ;       $MXLBN, $MXLBN+2, $MXCYL, $MXTRK, AND $MXSEC ARE ALL UPDATED
                        ;
                        ;-

000414  012367  000000G          $UPDMX::MOV     (R3)+,$MXLBN    ;SET UP MAX LBN
000420  012367  000002G                 MOV     (R3)+,$MXLBN+2  ;...
000424  012367  000000G                 MOV     (R3)+,$MXCYL    ;SET UP MAX CYLINDER
000430  112367  000000G                 MOVB    (R3)+,$MXSEC    ;SETUP MAX SECTOR
000434  112367  000000G                 MOVB    (R3)+,$MXTRK    ;SETUP MAX TRACK
000440  166767  000000G 000002G        SUB     $MXSEC,$MXLBN+2 ;DO NOT FORMAT LAST TRACK
000446  005667  000000G                 SBC     $MXLBN          ;...
000452  000207                         RETURN                  ;

                        ;+
                        ; **-$TSTDA-TEST FOR A VALID DISK ADDRESS (BLOCK NUMBER)
                        ;
                        ; THIS SUBROUTINE WILL CONVERT THE DISK ADDRESS TO A LBN AND
                        ; CHECK IF THE LOGICAL BLOCK NUMBER IS WITHIN RANGE.
                        ;
                        ; INPUTS:
                        ;       $SEC, $TRK, $CYL SET TO A VALUE.
                        ;
                        ; OUTPUTS:
                        ;       CS      IF LBN OUT OF RANGE
                        ;       CC      IF LBN IN RANGE
                        ;-

000454  004767  177452          $TSTDA::CALL    $CVTDA          ;CONVERT PARAMETERS TO LBN
000460  126767  000000G 000000G         CMPB    $LBNH,$MXLBN    ;HIGH BLOCK # IN RANGE?
000466  101007                         BHI     2$              ;IF HI NO
000470  103404                         BLO     1$              ;IF LO YES
000472  026767  000000G 000002G        CMP     $LBNL,$MXLBN+2  ;LOW BLOCK # IN RANGE?
000500  101002                         BHI     2$              ;IF HI NO
000502  000241                  1$.    CLC                     ;SHOW LBN OK
000504  000207                         RETURN                  ;
000506  000261                  2$:    SEC                     ;SHOW LBN BAD
000510  000207                         RETURN                  ;

                        ;+
                        ; **-$TSTMX-TEST FOR END OF DISK AND UPDATE DISK PARAMETERS
                        ;
                        ; THIS SUBROUTINE WILL UPDATE THE DISK PARAMETERS AND TEST
                        ; IF THE PARAMETERS WILL OVERFLOW THE DISK.
                        ;
                        ; INPUTS:
                        ;       $CYL = CURRENT CYLINDER NUMBER
                        ;       $TRK = CURRENT TRACK NUMBER
                        ;       $SEC = CURRENT SECTOR NUMBER
                        ;
                        ; OUTPUTS:
                        ;       CS      IF UPDATED PARAMETERS EXCEED DISK LIMITS
                        ;       CC      IF UPDATED PARAMETERS ARE STILL VALID.
                        ;       $SEC = 0 IF NORMAL MODE.
                        ;-
```

```
317 000512  005067  000000G          $TSTMX::CLR     $SEC            ; START WITH SECTOR ZERO
318 000516  005267  000000G                   INC     $TRK            ; UPDATE TRACK NUMBER
329 000522  026767  000000G 000000G 10$:     CMP     $TRK,$MXTRK     ; IS TRACK NUMBER EXCEEDED?
330 000530  103404                           BLO     20$             ; IF LO, NO
331 000532  005067  000000G                   CLR     $TRK            ; RESET TRACK NUMBER
332 000536  005267  000000G                   INC     $CYL            ; UPDATE CYLINDER NUMBER
333 000542  026767  000000G 000000G 20$:     CMP     $CYL,$MXCYL     ; IS CYLINDER NUMBER EXCEEDED?
334 000550  001402                           BEQ     30$             ; IF EQ YES
335 000552  000241                           CLC                     ; INDICATE VALID PARAMETERS
336 000554  000207                           RETURN                  ;
337 000556  000261                 30$:      SEC                     ; INDICATE PARAMETERS EXCEEDED
338 000560  000207                           RETURN                  ;
339
340
341                                       ;+
342                                       ; **--$UNTID-PUT MESSAGE AND DEVICE/UNIT INTO BUFFER
343                                       ;
344                                       ; THIS SUBROUTINE WILL STORE THE MESSAGE AND THE DEVICE AND UNIT
345                                       ; NUMBER INTO THE BUFFER.
346                                       ;
347                                       ; INPUTS:
348                                       ;       R3 = ADDRESS OF MESSAGE
349                                       ;
350                                       ; OUTPUTS:
351                                       ;       R0 = NEXT BUFFER POSITION
352                                       ;-
353
354 000562  012700  000144G          $UNTID::MOV     #$BUF+100.,R0   ;SET BUFFER ADDRESS
355 000566  004767  177446                   CALL    $MOVE           ;PUT MESSAGE INTO BUFFER
356 000572  116720  000000G                  MOVB    $DVICE,(R0)+    ;STORE DEVICE NAME
357 000576  116720  000001G                  MOVB    $DVICE+1,(R0)+  ;
358 000602  005001                           CLR     R1              ;GET READY FOR THE BISB
359 000604  156701  000000G                  BISB    $DVUNT,R1       ;GET UNIT NUMBER
360 000610  005002                           CLR     R2              ;SUPPRESS LEADING 0'S
361 000612  000167  000000G                  JMP     $CBOMG          ;CONVERT UNIT NUMBER TO ASCII
362
363
364                                       ;+
365                                       ; **-$RDBSF-READ MANUFACTURER'S & USER BAD SECTOR FILES
366                                       ;
367                                       ; THIS SUBROUTINE WILL READ THE MDBSF AND THE UDBSF.
368                                       ; IF WE ARE RE-WRITING THE LAST TRACK, A EMPTY MDBSF AND UDBSF
369                                       ; IS RETURNED.
370                                       ;
371                                       ; INPUTS:
372                                       ;       $MXLBN = DISK ADDRESS OF START OF MDBSF.
373                                       ;
374                                       ; OUTPUTS:
375                                       ;       A GOOD SECTOR OF THE MDBSF AND THE UDBSF ARE READ INTO
376                                       ;        THEIR BUFFERS
377                                       ;-
378
379 000616  012703  000000G          $RDBSF::MOV     #$LOGDV,R3      ; LOGICAL DEVICE ADDRESS POINDER
380 000622  004767  177566                   CALL    $UPDMX          ; UPDATE THE ADDRESS VALUES
381 000626  016767  000000G 000000G         MOV     $MXLBN,$LBNH    ; SET DISK ADDRESS OF MDBSF
382 000634  016767  000002G 000000G         MOV     $MXLBN+2,$LBNL  ;
383
```

```
                                ; READ THE MDBSF
                                
000642 016701 000000G       1$: MOV     $MBSSZ,R1       ; GET NUMBER OF BLOCKS IN MDBSF
000646 006201                   ASR     R1              ; DIVIDE BY 2
000650 000405                   BR      3$              ;
000652 062767 000002 000000G 2$: ADD     #2,$LBNL        ; TRY NEXT USABLE MDBSF BLOCK
000660 005567 000000G           ADC     $LBNH           ;
000664 012705 000000G       3$: MOV     #$MDBSF,R5      ; GET ADDRESS OF MDBSF BUFFER
000670 004767 000222            CALL    RDBSF           ; READ THE MDBSF
000674 103017                   BCC     6$              ; IF CC OK
000676 012700 000000G           MOV     #$MSG21,R0      ; CAN'T READ MDBSF
000702 005301               4$: DEC     R1              ; MORE MDBSF BLOCKS AVAILABLE?
000704 001006                   BNE     5$              ; IF NE YES
000706 004767 000000G           CALL    $PRINT          ; PRINT THE ERROR MESSAGE
000712 012700 000000G           MOV     #$MSG26,R0      ; UNABLE TO RECOVER THE MDBSF
000716 000167 000000G           JMP     $MSGWR          ; EXIT
000722 010146               5$: MOV     R1,-(SP)        ; SAVE THE MDBSF COUNT
000724 004767 000000G           CALL    $PRINT          ; PRINT THE MESSAGE
000730 012601                   MOV     (SP)+,R1        ; RESTORE R1
000732 000747                   BR      2$              ; TRY AGAIN
000734 004567 000176        6$: JSR     R5,VYBSF        ; VERIFY THE MDBSF COPY
000740 000000G                  .WORD   $MDBSF          ; MDBSF BUFFER POINTER
000742 000000G                  .WORD   $MSG40          ; BAD S/N MSG POINTER
000744 000000G                  .WORD   $MSG18          ; MDBSF CORRUPT MSG POINTER
000746 103755                   BCS     4$              ; IF CS, MDBSF CORRUPT
000750 004767 001506            CALL    $SRTMF          ; SORT THE MDBSF

                                ; READ THE UDBSF

000754 016767 000000G 000000G   MOV     $MXLBN,$LBNH    ; SET DISK ADDRESS OF UDBSF
000762 016767 000002G 000000G   MOV     $MXLBN+2,$LBNL  ;
000770 066767 000000G 000000G   ADD     $MBSSZ,$LBNL    ; ADDRESS OF THE UDBSF
000776 005567 000000G           ADC     $LBNH           ;
001002 016701 000000G           MOV     $UBSSZ,R1       ; GET NUMBER OF BLOCKS IN UDBSF
001006 006201                   ASR     R1              ; DIVIDE BY 2
001010 000405                   BR      8$              ;
001012 062767 000002 000000G 7$: ADD     #2,$LBNL        ; TRY NEXT USABLE UDBSF BLOCK
001020 005567 000000G           ADC     $LBNH           ;
001024 012705 000000G       8$: MOV     #$UDBSF,R5      ; GET ADDRESS OF UDBSF BUFFER
001030 004767 000062            CALL    RDBSF           ; READ THE UDBSF
001034 103017                   BCC     11$             ; IF CC OK
001036 012700 000000G           MOV     #$MSG43,R0      ; CAN'T READ UDBSF
001042 005301               9$: DEC     R1              ; MORE UDBSF BLOCKS AVAILABLE?
001044 001006                   BNE     10$             ; IF NE, YES
001046 004767 000000G           CALL    $PRINT          ; PRINT THE ERROR MESSAGE
001052 012700 000000G           MOV     #$MSG42,R0      ; UNABLE TO RECOVER THE UDBSF
001056 000167 000000G           JMP     $MSGWR          ; EXIT
001062 010146              10$: MOV     R1,-(SP)        ; SAVE THE MDBSF COUNT
001064 004767 000000G           CALL    $PRINT          ; PRINT THE MESSAGE
001070 012601                   MOV     (SP)+,R1        ; RESTORE R1
001072 000747                   BR      7$              ; TRY AGAIN
001074 004567 000036       11$: JSR     R5,VYBSF        ; VERIFY THE UDBSF COPY
001100 000000G                  .WORD   $UDBSF          ; UDBSF BUFFER POINTER
001102 000000G                  .WORD   $MSG44          ; BAD S/N MSG POINTER
001104 000000G                  .WORD   $MSG19          ; UDBSF CORRUPT MSG POINTER
001106 103755                   BCS     9$              ; IF CS, UDBSF CORRUPT
001110 004767 001402            CALL    $SRTUF          ; SORT THE UDBSF
```

```
001114  000207                          RETURN
                                ;+
                                ; **-RDBSF-READ THE SPECIFIED BSF FILE
                                ;
                                ; READ THE BSF (EITHER MDBSF OR UDBSF) SPECIFIED IN R5
                                ;  (ROUTINE USED BY '$RDBSF')
                                ;
                                ; INPUTS:
                                ;       R5 = BUFFER ADDRESS
                                ;       $LBNL,$LBNH LOADED WITH THE TRANSFER LBN
                                ;
                                ; OUTPUTS:
                                ;       CC      IF TRANSFER WAS NOT SUCESSFUL
                                ;-
001116  012767  001000  000000G RDBSF:  MOV     #512.,$XFRSZ    ; SET SIZE OF ONE SECTOR (BYTES)
001124  012703  000000C                 MOV     #IO.RLB!IO.X,R3 ; SET READ FUNCTION (WITHOUT RETRIES)
001130  004767  000000G                 CALL    $DSKIO          ; READ 1 BLOCK FROM THE BSF
001134  000207                          RETURN                  ;
                                ;+
                                ; **-VYBSF-VERIFY THE BSF COPY
                                ;  (ROUTINE USED BY '$RDBSF')
                                ;
                                ; VERIFY THE BSF COPY SPECIFIED BY THE INPUT PARAMETERS
                                ;
                                ; CALL: JSR     R5,VYBSF
                                ;       BSF POINTER
                                ;       POINTER TO 'XXBSF SERIAL NUMBER DOESN'T MATCH' MSG
                                ;       POINTER TO 'XXBSF COPY CORRUPT' MSG
                                ;
                                ; OUTPUTS:
                                ;       R0 = SPECIFIC ERROR MESSAGE POINTER
                                ;       CC      0 IF BSF NOT CORRUPT
                                ;       CS      1 IF BSF CORRUPT
                                ;-
001136  012503                  VYBSF:  MOV     (R5)+,R3        ; GET THE BUFFER POINTER
001140  012500                          MOV     (R5)+,R0        ; BAD S/N MESSAGE POINTER
001142  105767  000000G                 TSTB    $SWLST          ; LISTING THE MDBSF ?
001146  001010                          BNE     1$              ; IF NE, YES
001150  021367  000000G                 CMP     (R3),$SNAME     ; VERIFY THE SERIAL NUMBER
001154  001020                          BNE     4$              ; IF NE, FIRST S/N WORD NOT CORRECT
001156  026367  000002  000002G         CMP     2(R3),$SNAME+2  ; CHECK THE SECOND WORD
001164  001014                          BNE     4$              ; IF NE, SERIAL NUMBER DOESN'T MATCH
001166  000405                          BR      2$              ; CHECK THE ALIGNMENT FLAG
001170  005713                  1$:     TST     (R3)            ; IS SERIAL NUMBER NON-ZERO?
001172  001003                          BNE     2$              ; IF NE YES
001174  005763  000002                  TST     2(R3)           ; MAYBE?
001200  001403                          BEQ     3$              ; IF EQ, NO
001202  005763  000006          2$:     TST     6(R3)           ; TEST THE ALIGNMENT DISK FLAG
001206  001403                          BEQ     4$              ; IF EQ, BSF OK
001210  012500                  3$:     MOV     (R5)+,R0        ; CORRUPT BSF MESSAGE POINTER
001212  000261                          SEC                     ; SET THE ERROR INDICATOR
001214  000402                          BR      5$              ;
001216  005725                  4$:     TST     (R5)+           ; CORRECT THE RETURN
```

```
                        000241                  CLC                     ; SET THE GOOD INDICATOR
                        000205          5$:     RTS     R5              ; RETURN

                                        ;+
                                        ; **-$WTBSF-WRITE THE MANUFACTURER'S AND USER'S BAD SECTOR FILES
                                        ;
                                        ; THIS ROUTINE WILL WRITE THE 16 BIT MDBSF AND THE UDBSF ON THE
                                        ;  PRIMARY DEC STD 144 TRACK (558,13) AND ON THE ALTERNATE TRACK
                                        ;  (559,1).
                                        ;
                                        ; INPUTS:
                                        ;       NONE.
                                        ;
                                        ; OUTPUTS:
                                        ;       NONE.
                                        ;-

001224  004767  001232          $WTBSF::CALL    $SRTMF          ; SORT THE MDBSF
001230  004767  001262                  CALL    $SRTUF          ; SORT THE UDBSF
001234  012703  000000G                 MOV     #$LOGDV,R3      ; LOGICAL DEVICE ADDRESS POINTER
001240  004767  177150                  CALL    $UPDMX          ; UPDATE THE ADDRESS VALUES
001244  012767  001000  000000G         MOV     #512.,$XFRSZ    ; SET ONE BLOCK BYTE COUNT
001252  016767  000000G 000000G         MOV     $MXLBN,$LBNH    ; SETUP LBN
001260  016767  000002G 000000G         MOV     $MXLBN+2,$LBNL  ; ...

                                        ; WRITE THE 16 BIT MDBSF

001266  012705  000000G                 MOV     #$MDBSF,R5      ; SET MDBSF ADDRESS
001272  016701  000000G                 MOV     $MBSSZ,R1       ; GET NUMBER OF BLOCKS IN MDBSF
001276  006201                          ASR     R1              ; DIVIDE BY TWO
001300  012703  000000C                 MOV     #IO.WLT!IQ.X,R3 ; SET WRITE LAST TRACK FUNCTION CODE
001304  004767  000110                  CALL    1$              ; WRITE THE MDBSF

                                        ; WRITE THE 16 BIT UDBSF

001310  012705  000000G                 MOV     #$UDBSF,R5      ; UDBSF ADDRESS
001314  016701  000000G                 MOV     $UBSSZ,R1       ; GET NUMBER OF BLOCKS IN UDBSF
001320  006201                          ASR     R1              ; DIVIDE BY TWO
001322  012703  000000C                 MOV     #IO.WLT!IQ.X,R3 ; SET WRITE LAST TRACK FUNCTION CODE
001326  004767  000066                  CALL    1$              ; WRITE THE UDBSF

                                        ; WRITE THE ALTERNATE DEC STD 144 TRACK (559,1)

001332  012703  000000G                 MOV     #$PHYDV,R3      ; CHANGE TO PHYSICAL ADDRESSING
001336  004767  177052                  CALL    $UPDMX          ; UPDATE THE MAX VALUES
001342  016767  000000G 000000G         MOV     $ALFLB,$LBNH    ; ALTERNATE DEC STD 144 ADDRESS
001350  016767  000002G 000000G         MOV     $ALFLB+2,$LBNL  ;

                                        ; WRITE THE ALTERNATE MDBSF

001356  012705  000000G                 MOV     #$MDBSF,R5      ; SET MDBSF ADDRESS
001362  016701  000000G                 MOV     $MBSSZ,R1       ; NUMBER OF BLOCKS IN THE MDBSF
001366  006201                          ASR     R1              ; DIVIDE BY TWO
001370  012703  000000C                 MOV     #IO.FEW!IQ.UMD,R3       ;SET WRITE FE CYLINDER FUNCTION
001374  004767  000020                  CALL    1$              ;

                                        ; WRITE THE ALTERNATE UDBSF
```

```

    001400  012705  000000G                 MOV     #$UDBSF,R5        ; UDBSF ADDRESS
    001404  016701  000000G                 MOV     $UBSSZ,R1         ; UDBSF SIZE
    001410  006201                          ASR     R1                ; DIVIDE BY TWO
    001412  012746  000000C                 MOV     #IO.FEW!IQ.UMD,-(SP)    ;SET WRITE FE CYLINDER FUNCTION
    001416  000401                          BR      2$                ;
    001420  010346                  1$:     MOV     R3,-(SP)          ; SAVE THE FUNCTION CODE
    001422  011603                  2$:     MOV     (SP),R3           ; LOAD THE FUNCTION CODE
    001424  004767  000000G                 CALL    $DSKIO            ; WRITE THE BLOCK
    001430  062767  000002  000000G         ADD     #2,$LBNL          ; UPDATE BLOCK NUMBER
    001436  005567  000000G                 ADC     $LBNH             ; ...
    001442  005301                          DEC     R1                ; FINISHED WITH THE BSF ?
    001444  001366                          BNE     2$                ; IF NE NO
    001446  005726                          TST     (SP)+             ; CORRECT THE STACK
    001450  000207                          RETURN                    ;

                                        ;+
                                        ; **-$RDSSF-READ RM80 SKIP SECTOR FILE
                                        ;
                                        ; THIS ROUTINE WILL READ THE SKIP SECTOR FILE FOUND ON THE FIRST TRACK
                                        ; OF THE FE CYLINDER AREA ON THE RM80.
                                        ;
                                        ; INPUTS:
                                        ;         $SSFLB = DISK ADDRESS OF START OF SKIPPED SECTOR FILE
                                        ;
                                        ; OUTPUTS:
                                        ;         A GOOD COPY OF THE SSF IS READ INTO SSF BUFFER
                                        ;
                                        ;-

    001452  012703  000000G         $RDSSF::MOV     #$PHYDV,R3        ; PHYSICAL DEVICE ADDRESS POINTER
    001456  004767  176732                  CALL    $UPDMX            ; UPDATE THE ADDRESS VALUES
    001462  012767  000000C 000000G         MOV     #512.*<$SSFLN>,$XFRSZ   ; SET SIZE OF ONE COPY OF SSF (BYTES)
    001470  016767  000000G 000000G         MOV     $SSFLB,$LBNH      ; SET DISK ADDRESS OF SSF
    001476  016767  000002G 000000G         MOV     $SSFLB+2,$LBNL    ;
    001504  016701  000000G                 MOV     $SSFSZ,R1         ; GET NUMBER OF COPIES OF SKIP SECTOR FILE
    001510  010146                          MOV     R1,-(SP)          ; SAVE THE COPY COUNT
    001512  000405                          BR      2$                ; CONTINUE
    001514  062767  000000G 000000G 1$:     ADD     #$SSFLN,$LBNL     ; TRY NEXT USABLE SSF COPY
    001522  105567  000000G                 ADCB    $LBNH             ;
    001526  012705  000000G         2$:     MOV     #$SSF,R5          ; GET ADDRESS OF SKIP SECTOR FILE BUFFER
    001532  012703  000000C                 MOV     #IO.FER!IQ.UMD,R3       ; SET READ FE CYLINDER FUNCTION
    001536  004767  000000G                 CALL    $DSKIO            ; READ 1 COPY OF SSF
    001542  103017                          BCC     5$                ; IF CC, OK
    001544  012700  000000G                 MOV     #$MSG58,R0        ; CAN'T READ SSF
    001550  012601                  3$:     MOV     (SP)+,R1          ; GET THE SSF COPY COUNTER
    001552  005301                          DEC     R1                ; DECREMENT THE COPY COUNTER
    001554  001006                          BNE     4$                ; IF NE, TRY AGAIN
    001556  004767  000000G                 CALL    $PRINT            ; PRINT THE ERROR MESSAGE
    001562  012700  000000G                 MOV     #$MSG41,R0        ; UNABLE TO RECOVER SSF
    001566  000167  000000G                 JMP     $MSGWR            ; EXIT, CAN'T READ SSF OR SSF CORRUPT
    001572  010146                  4$:     MOV     R1,-(SP)          ; SAVE R1
    001574  004767  000000G                 CALL    $PRINT            ; PRINT THE MESSAGE
    001600  000745                          BR      1$                ; TRY AGAIN
    001602  012700  000000G         5$:     MOV     #$SSF,R0          ; START WITH HEADER
    001606  012701  177777G                 MOV     #$SSFHD-1,R1      ; AND ITS LENGTH
    001612  004767  176556                  CALL    SSFCRC            ; COMPUTE HEADER CHECKSUM
```

```
 612 001616  020120                          CMP     R1,(R0)+        ; CHECK THE CHECKSUM WORD
 613 001620  001020                          BNE     6$              ; IF NE, BAD CHECKSUM
 614 001622  012701  177777G                 MOV     #$SSFDT-1,R1    ; DATA FIELD LENGTH
 615 001626  004767  176542                  CALL    SSFCRC          ; COMPUTE CHECKSUM OVER DATA AREA
 616 001632  020110                          CMP     R1,(R0)         ; CHECKSUM OK?
 617 001634  001012                          BNE     6$              ; IF NE, NO
 618 001636  005767  000004G                 TST     $SSF+4          ; CREATION DATE PRESENT
 619 001642  001407                          BEQ     6$              ; IF EQ, NO
 620 001644  005767  000006G                 TST     $SSF+6          ; REVISION DATE PRESENT
 621 001650  001404                          BEQ     6$              ; IF EQ, NO
 622 001652  122767  000001  000012G         CMPB    #1,$SSF+12      ; 16 BIT MODE SSF ?
 623 001660  001403                          BEQ     7$              ; IF EQ, YES
 624 001662  012700  000000G         6$:     MOV     #$MSG59,R0      ; SSF CORRUPT
 625 001666  000730                          BR      3$              ; TRY AGAIN
 626 001670  005726                  7$:     TST     (SP)+           ; CORRECT THE STACK
 627 001672  105767  000000G                 TSTB    $SWLST          ; LISTING THE BSF'S ?
 628 001676  001026                          BNE     10$             ; IF NE, YES
 629 001700  105767  000000G                 TSTB    $SWMOD          ; MODIFYING THE BSF'S
 630 001704  001007                          BNE     8$              ; IF NE, YES
 631 001706  016767  000000G 000000G         MOV     $SSF,$SNAME     ; COPY THE SERIAL NUMBER
 632 001714  016767  000002G 000002G         MOV     $SSF+2,$SNAME+2 ;
 633 001722  000414                          BR      10$             ;
 634 001724  026767  000000G 000000G 8$:     CMP     $SSF,$SNAME     ; CHECK THE SERIAL NUMBER
 635 001732  001004                          BNE     9$              ; IF NE, DOESN'T MATCH
 636 001734  026767  000002G 000002G         CMP     $SSF+2,$SNAME+2 ; CHECK THE SECOND S/N WORD
 637 001742  001404                          BEQ     10$             ; IF EQ, S/N OK
 638 001744  012700  000000G         9$:     MOV     #MSGER8,R0      ; HDA AND ENTERED SERIAL NUMBER DON'T MATCH
 639 001750  000167  000000G                 JMP     $MSGWR          ; EXIT
 640 001754  000167  000446          10$:    JMP     $SRTSF          ; SORT THE SSF AND RETURN
 641
 642                                         ;+
 643                                         ; **-$WTSSF-WRITE RM80 SKIP SECTOR FILE
 644                                         ;
 645                                         ; THIS ROUTINE WILL WRITE THE SKIP SECTOR FILE FIVE TIMES ON THE FIRST
 646                                         ;  TRACK OF THE FE CYLINDER AREA OF THE RM80 (559,0).
 647                                         ;
 648                                         ; INPUTS:
 649                                         ;       $SSFLB = DISK ADDRESS OF START OF SKIP SECTOR FILE
 650                                         ;
 651                                         ; OUTPUTS:
 652                                         ;       WRITE SKIP SECTOR FILE
 653                                         ;
 654                                         ;-
 655
 656 001760  004767  000442          $WTSSF::CALL    $SRTSF          ; SORT THE SSF
 657 001764  012703  000000G                 MOV     #$PHYDV,R3      ; PHYSICAL DEVICE ADDRESS POINDER
 658 001770  004767  176420                  CALL    $UPDMX          ; UPDATE THE ADDRESS VALUES
 659 001774  012700  000000G                 MOV     #$SSF,R0        ;GET POINTER TO SKIP SECTOR FILE
 660 002000  012701  177777G                 MOV     #$SSFHD-1,R1    ;GET LENGTH OF HEADER
 661 002004  004767  176364                  CALL    SSFCRC          ; COMPUTE HEADER CHECKSUM
 662 002010  010120                          MOV     R1,(R0)+        ;STORE CRC AWAY
 663 002012  012701  177777G                 MOV     #$SSFDT-1,R1    ;GET LENGTH OF SSF DATA AREA
 664 002016  004767  176352                  CALL    SSFCRC          ;GO COMPUTE CHECKSUM OVER DATA
 665 002022  010110                          MOV     R1,(R0)         ;STORE THE DATA FIELD CRC
 666 002024  012767  000000C 000000G         MOV     #512.*<$SSFLN>,$XFRSZ   ;SET SIZE OF ONE COPY OF SSF (BYTES)
 667 002032  012705  000000G                 MOV     #$SSF,R5        ;SET SKIP SECTOR FILE BUFFER ADDRESS
 668 002036  012703  000000C                 MOV     #IO.FEW!IQ.UMD,R3       ;SET WRITE FE CYLINDER FUNCTION
```

```
 669 002042  016767  000000G 000000G          MOV     $SSFLB,$LBNH    ;SETUP LBN
 670 002050  016767  000C02G 000000G          MOV     $SSFLB+2,$LBNL  ;...
 671 002056  016701  000000G                  MOV     $SSFSZ,R1       ; GET NUMBER OF COPIES OF SKIP SECTOR FILE
 672 002062  004767  000000G          1$:     CALL    $DSKIO          ; WRITE THE SSF
 673 002066  062767  000000G 000000G          ADD     #$SSFLN,$LBNL   ; UPDATE BLOCK NUMBER
 674 002074  105567  000000G                  ADCB    $LBNH           ;...
 675 002100  005301                           DEC     R1              ; FINISHED ?
 676 002102  001367                           BNE     1$              ; IF NE, NO
 677 002104  000207                   2$:     RETURN                  ;
 678
 679                                          ;+
 680                                          ; **-$SRTRK-SEARCH BSF OR SSF FOR ANY SECTOR ON A TRACK
 681                                          ;
 682                                          ; THIS ROUTINE WILL SEARCH EITHER THE BSF OR THE SSF FOR ANY SECTOR
 683                                          ; ON THE CURRENT TRACK AND CYLINDER.  IF ONE IS FOUND, THEN NO ADDITIONAL
 684                                          ; SECTORS MAY BE SKIPPED ON THE CURRENT TRACK AND CYLINDER.
 685                                          ;
 686                                          ; INPUTS:
 687                                          ;       R1 =    ADDRESS OF EITHER BSF OR SSF
 688                                          ;       R2 =    MAXIMUM ADDRESS OF THE FILE SPECIFIED IN R1
 689                                          ;
 690                                          ; OUTPUTS:
 691                                          ;       CC =    SOME SECTOR ON THE CURRENT TRACK AND CYLINDER
 692                                          ;               R1 =    ADDRESS OF BSF OR SSF ENTRY
 693                                          ;       CS =    SPECIFIED FILE EXHAUSTED
 694                                          ;               R1 =    POINTS TO NEXT ENTRY
 695                                          ;
 696                                          ;-
 697
 698 002106  022121                   $SRTRK::CMP     (R1)+,(R1)+     ; POINT OT NEXT ENTRY IN SPECIFIED FILE
 699 002110  021127  177777                   CMP     (R1),#-1        ; FILE EXHAUSTED
 700 002114  001413                           BEQ     3$              ; IF EQ, YES
 701 002116  021167  000000G                  CMP     (R1),$CYL       ; CYLINDER NUMBER MATCH ?
 702 002122  001006                           BNE     2$              ; IF NE, NO
 703 002124  126167  000003  000000G          CMPB    3(R1),$TRK      ; TRACK NUMBERS MATCH ?
 704 002132  001002                           BNE     2$              ; IF NE, NO
 705 002134  000241                   1$:     CLC                     ; SHOW ENTRY WAS FOUND
 706 002136  000207                           RETURN                  ;
 707 002140  020102                   2$:     CMP     R1,R2           ; IS SPECIFIED FILE EXHAUSTED ?
 708 002142  001361                           BNE     $SRTRK          ; IF NE, NO
 709 002144  000261                   3$:     SEC                     ; SHOW SPECIFIED FILE EXHAUSTED
 710 002146  000207                           RETURN
 711
 712                                          ;+
 713                                          ; **-$SRALL-SEARCH ALL THE BAD SECTOR FILES
 714                                          ;    $SRFES-SEARCH THE FE CYLINDER BUFFER
 715                                          ;    $SRMDF-SEARCH MANUFACTURER'S BAD SECTOR FILE
 716                                          ;    $SRUDF-SEARCH THE USER'S BAD SECTOR FILE
 717                                          ;    $SRSDF-SEARCH THE SKIPPED SECTOR FILE
 718                                          ;    $SMBSF-SEARCH THE FILE SPECIFIED BY R1 & R2
 719                                          ;
 720                                          ; THIS ROUTINE WILL SEARCH THE SPECIFIED FILE FOR A MATCH.
 721                                          ;
 722                                          ; INPUTS:
 723                                          ;       $SEC, $TRK, AND $CYL PRESET
 724                                          ;
 725                                          ; '$SMBSF' INPUTS:
```

```
                        ;       R1 = ADDRESS OF LAST BSF ENTRY (INITIALLY $BSF+4)
                        ;       R2 = LAST ADDRESS OF SPECIFIED FILE
                        ;
                        ; OUTPUTS:
                        ;       IF CC   R1 = ADDRESS OF MDBSF ENTRY.
                        ;               R3 = ISOLATED BAD SECTOR FLAGS.
                        ;       IF CS   R1 = POINTS TO NEXT ENTRY (FILE EXHAUSTED)
                        ;-
                        .ENABL   LSB

002150  016746  000000G        $SRALL::MOV     $SEC,-(SP)      ; SAVE THE SECTOR ADDRESS
002154  004767  000100                  CALL    $SRSDF          ; SEE IF ENTRY IN THE SSF
002160  103015                         BCC     2$              ; IF CC, IT IS
002162  004767  000154                  CALL    $SRSSF          ; SEE IF ERROR SECTOR IS WITHIN A SKIP REGION
002166  103402                         BCS     1$              ; IF CS, NO
002170  105367  000000G                DECB    $SEC            ; DECREMENT THE SECTOR ADDRESS
002174  004767  000034        1$:     CALL    $SRMDF          ; SEARCH THE MDBSF
002200  103005                         BCC     2$              ; IF CC, ADDRESS PRESENT
002202  004767  000040                  CALL    $SRUDF          ; SEARCH THE UDBSF
002206  103002                         BCC     2$              ; IF CC, ADDRESS PRESENT
002210  004767  000006                  CALL    $SRFES          ; SEE IF ENTRY DUPLICATED IN THE FE AREA
002214  012667  000000G        2$:     MOV     (SP)+,$SEC      ; RESTORE THE SECTOR ADDRESS
002220  000207                         RETURN                  ;

002222  012701  177774G        $SRFES::MOV     #$FEBSF-4,R1    ; FEBSF STARTING ADDRESS
002226  012702  000100G                 MOV     #$FEBSF+64.,R2  ; FEBSF LENGTH
002232  000416                         BR      3$              ;

002234  012701  000004G        $SRMDF::MOV     #$MDBSF+4,R1    ; MDBSF STARTING ADDRESS
002240  012702  000774G                 MOV     #$MDBSF+1000-4,R2       ; MDBSF LENGTH
002244  000411                         BR      3$              ;

002246  012701  000004G        $SRUDF::MOV     #$UDBSF+4,R1    ; UDBSF STARTING ADDRESS
002252  012702  000774G                 MOV     #$UDBSF+1000-4,R2       ; UDBSF LENGTH
002256  000404                         BR      3$              ;

002260  012701  000000C        $SRSDF::MOV     #$SSF+<2+$SSFHD>-4,R1   ; SSF STARTING ADDRESS (-4)
002264  012702  000000C                 MOV     #$SSF+<1000*$SSFLN>-2,R2        ; SSF ENDING ADDRESS
002270                          3$:

002270  022121                 $SMBSF::CMP     (R1)+,(R1)+     ; POINT TO NEXT MDBSF ENTRY
002272  021127  177777                  CMP     (R1),#-1        ; MDBSF EXHAUSTED?
002276  001417                         BEQ     5$              ; IF EQ YES
002300  021167  000000G                 CMP     (R1),$CYL       ; CYLINDER NUMBERS MATCH?
002304  001012                         BNE     4$              ; IF NE NO
002306  126167  000003  000000G        CMPB    3(R1),$TRK      ; TRACK NUMBERS MATCH?
002314  001006                         BNE     4$              ; IF NE NO
002316  126167  000002  000000G        CMPB    2(R1),$SEC      ; SECTOR NUMBERS MATCH?
002324  001002                         BNE     4$              ; IF NE NO
002326  000241                         CLC                     ; SHOW ENTRY WAS FOUND
002330  000207                         RETURN                  ;
002332  020102                  4$:     CMP     R1,R2           ; IS SPECIFIED FILE EXHAUSTED?
002334  001355                         BNE     3$              ; IF NE, NO
002336  000261                  5$:     SEC                     ; SHOW MDBSF EXHAUSTED
002340  000207                         RETURN                  ;
                        .DSABL  LSB
```

```
                                        ;+
                                        ; **-$SRSSF-SEARCH SSF FOR AN TRACK ENTRY
                                        ;
                                        ; THIS ROUTINE WILL SEARCH THE SSF FOR AN ENTRY ON THE CURRENT TRACK AND
                                        ; CYLINDER.  IF AN ENTRY IS FOUND WHICH IS LESS THAN THE SECTOR ADDRESS
                                        ; IN '$SEC', THE ROUTINE WILL RETURN WITH THE 'C' BIT CLEAR.
                                        ;
                                        ; INPUTS:
                                        ;       $CYL LOADED WITH THE CYLINDER ADDRESS
                                        ;       $TRK LOADED WITH THE TRACK ADDRESS
                                        ;       $SEC LOADED WITH THE SECTOR ADDRESS
                                        ;
                                        ; OUTPUTS:
                                        ;       CC =    SSF ENTRY .LE. ADDRESS IN $CYL,$TRK,$SEC
                                        ;       CS =    NO SSF ENTRY .LE. ADDRESS IN $CYL,$TRK,$SEC
                                        ;-
002342  010146                  $SRSSF::MOV     R1,-(SP)        ; SAVE R1
002344  012701  000000C                 MOV     #$SSF+<2+$SSFHD>,R1     ; BEGINNING ADDRESS OF SSF DATA
002350  021167  000000G         1$:     CMP     (R1),$CYL       ; CYLINDER NUMBERS MATCH ?
002354  001012                          BNE     2$              ; IF NE, NO
002356  126167  000003  000000G         CMPB    3(R1),$TRK      ; TRACK NUMBERS MATCH ?
002364  001006                          BNE     2$              ; IF NE, NO
002366  126167  000002  000000G         CMPB    2(R1),$SEC      ; CHECK FOR A SECTOR
002374  101011                          BHI     3$              ; IF HI, SSF ENTRY
002376  000241                          CLC                     ; SHOW '$SEC' IS WITHIN SKIP REGION
002400  000410                          BR      4$              ; OK, EXIT

002402  020127  000000C         2$:     CMP     R1,#$SSF+<1000*$SSFLN>-2        ; SSF BUFFER END ?
002406  001404                          BEQ     3$              ; IF EQ, YES
002410  022121                          CMP     (R1)+,(R1)+     ; INCREMENT BUFFER POINTER
002412  021127  177777                  CMP     (R1),#-1        ; SEE IF END OF DATA
002416  001354                          BNE     1$              ; IF NE, NO
002420  000261                  3$:     SEC                     ; SHOW '$SEC' NOT IN SKIP REGION
002422  012601                  4$:     MOV     (SP)+,R1        ; RESTORE R1
002424  000207                          RETURN

                                        ;+
                                        ; **-$SRTSF-SORT THE SSF BUFFER
                                        ;
                                        ; INPUTS:
                                        ;       NONE
                                        ;
                                        ; OUTPUTS:
                                        ;       SSF FILE SORTED IN ASCENDING SEQUENCE
                                        ;
                                        ;-

002426  012767  001061  000000G $SRTSF::MOV     #561.,$CYL      ; LOAD A PHONY CYLINDER ADDRESS
002434  005067  000000G                 CLR     $TRK            ; CLEAR THE TRACK ADDRESS
002440  005067  000000G                 CLR     $SEC            ;       "     "   SECTOR    "
002444  004767  177610                  CALL    $SRSDF          ; FIND THE END OF THE DATA
002450  010103                          MOV     R1,R3           ; COPY THE END OF DATA POINTER
002452  012701  000000C                 MOV     #$SSF+<2+$SSFHD>,R1     ; SSF STARTING ADDRESS
002456  000167  000070                  JMP     $SORT           ; SORT THE SSF
```

```
                        ;+
                        ; **-$SRTMF-SORT THE MDBSF BUFFER
                        ;
                        ; INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       MDBSF SORTED IN ASCENDING SEQUENCE
                        ;
                        ;-

002462  012767  001061  000000G $SRTMF::MOV     #561.,$CYL      ; LOAD A PHONY CYLINDER ADDRESS
002470  005067  000000G                CLR     $TRK            ; CLEAR THE TRACK ADDRESS
002474  005067  000000G                CLR     $SEC            ;       "     "  SECTOR   "
002500  004767  177530                CALL    $SRMDF          ; SEARCH THE MDBSF
002504  010103                        MOV     R1,R3           ; COPY THE END OF DATA POINTER
002506  012701  000010G                MOV     #$MDBSF+10,R1   ; MDBSF STARTING ADDRESS
002512  000167  000034                JMP     $SORT           ; SORT THE MDBSF

                        ;+
                        ; **-$SRTUF-SORT THE UDBSF BUFFER
                        ;
                        ; INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       UDBSF SORTED IN ASCENDING SEQUENCE
                        ;
                        ;-

002516  012767  001061  000000G $SRTUF::MOV     #561.,$CYL      ; LOAD A PHONY CYLINDER ADDRESS
002524  005067  000000G                CLR     $TRK            ; CLEAR THE TRACK ADDRESS
002530  005067  000000G                CLR     $SEC            ;       "     "  SECTOR   "
002534  004767  177506                CALL    $SRUDF          ; SEARCH THE UDBSF
002540  010103                        MOV     R1,R3           ; COPY THE END OF DATA POINTER
002542  012701  000010G                MOV     #$UDBSF+10,R1   ; UDBSF STARTING ADDRESS
002546  000167  000000                JMP     $SORT           ; SORT THE UMDBSF

                        ;+
                        ; **-$SORT-SORT THE INPUT TABLE
                        ; THIS ROUTINE SORTS THE INPUT TABLE IN ASCENDING ORDER
                        ; BASED ON DISK ADDRESSES.
                        ;
                        ; INPUTS:
                        ;       R1 = TABLE START ADDRESS
                        ;       R3 = TABLE UPPER LIMIT ADDRESS
                        ;
                        ; OUTPUTS:
                        ;       SORTED TABLE
                        ;
                        ;-

002552  010302          $SORT:  MOV     R3,R2           ; R2 = CURRENT ENTRY POINTER
002554  160102                  SUB     R1,R2           ; TEST FOR AT LEAST
002556  022702  000004          CMP     #4,R2           ; TWO ENTRIES IN TABLE
002562  103042                  BHIS    4$              ; IF NOT(HIS), NO NEED TO SORT
```

```
897 002564  010102                         MOV     R1,R2           ; INIT R2
898 002566  022222                         CMP     (R2)+,(R2)+     ;
899 002570  010367  000076                 MOV     R3,TEMP         ; FORM AND SAVE
900 002574  162767  000004  000070         SUB     #4,TEMP         ; UPPER LIMIT ON R1
901 002602  021112                 1$:     CMP     (R1),(R2)       ; CYL I = CYL I+1?
902 002604  002417                         BLT     3$              ; IF LT  YES
903 002606  001004                         BNE     2$              ; IF NE  CYL I > CYL I+1
904 002610  026162  000002  000002         CMP     2(R1),2(R2)     ; CYL I = CYL I+1...TEST TRK/SEC
905 002616  003412                         BLE     3$              ; IF LE  NO SWAP NECESSARY
906 002620  011146                 2$:     MOV     (R1),-(SP)      ; SWAP ENTRIES
907 002622  016146  000002                 MOV     2(R1),-(SP)     ;     .
908 002626  011211                         MOV     (R2),(R1)       ;     .
909 002630  016261  000002  000002         MOV     2(R2),2(R1)     ;     .
910 002636  012662  000002                 MOV     (SP)+,2(R2)     ;     .
911 002642  012612                         MOV     (SP)+,(R2)      ;     .
912 002644  022222                 3$:     CMP     (R2)+,(R2)+     ; UPDATE R2
913 002646  020203                         CMP     R2,R3           ; THIS PASS DONE?
914 002650  002754                         BLT     1$              ; IF LT  NO
915 002652  022121                         CMP     (R1)+,(R1)+     ; UPDATE R1
916 002654  020167  000012                 CMP     R1,TEMP         ; ALL DONE?
917 002660  002003                         BGE     4$              ; IF GE  YES
918 002662  010102                         MOV     R1,R2           ; RESET R2
919 002664  022222                         CMP     (R2)+,(R2)+     ;
920 002666  000745                         BR      1$              ;
921 002670  000207                 4$:     RETURN                  ;
922
923 002672  000000                 TEMP:   .WORD   0               ; SORT WORK LOCATION
924
925                                        ;+
926                                        ; **-GENHDR-GENERATE HEADER WORD PATTERN
927                                        ;
928                                        ; THIS SUBROUTINE WILL GENENRATE THE 2-WORD HEADER PATTERN.
929                                        ;
930                                        ; INPUTS:
931                                        ;       R0 = BUFFER ADDRESS
932                                        ;
933                                        ; OUTPTUS:
934                                        ;       R0 = UPDATED BUFFER ADDRESS
935                                        ;-
936
937 002674  016710  000000G         GENHDR::MOV     $CYL,(R0)       ; SET CYLINDER NUMBER
938 002700  116760  000000G 000002         MOVB    $SEC,2(R0)      ; SET SECTOR NUMBER
939 002706  116760  000000G 000003         MOVB    $TRK,3(R0)      ; SET TRACK NUMBER
940 002714  052710  150000                 BIS     #150000,(R0)    ; SET GOOD SECTOR AND 16-BIT FORMAT BITS
941 002720  116746  000000G                MOVB    $SEC,-(SP)      ; SAVE THE CURRENT SECTOR ADDRESS
942
943                                        ; SEARCH THE SKIP SECTOR FILE FOR A SECTOR ON THIS TRACK THE SAME AS
944                                        ; '$SEC'.  IF A SECTOR IS FOUND, THEN SET THE SKIP SECTOR FLAG IN THE
945                                        ; REMAINING HEADERS AND NOTE THAT A SKIPPED SECTOR WAS FOUND.
946
947 002724  004767  177330          1$:     CALL    $SRSDF          ; SEARCH FOR '$SEC' IN SKIP SECTOR FILE
948 002730  103403                         BCS     2$              ; IF CS, NOT IN SSF
949 002732  052710  020000                 BIS     #20000,(R0)     ; SET SKIP SECTOR FLAG (BAD)
950 002736  000430                         BR      6$              ; EXIT
951 002740  004767  177376          2$:     CALL    $SRSSF          ; SEARCH FOR '$SEC' WITHIN SKIP REGION
952 002744  103404                         BCS     3$              ; IF CS, NOT IN SSF
953 002746  052710  020000                 BIS     #20000,(R0)     ; SET SKIP SECTOR FLAG IN HEADER (BAD)
```

```
 954 002752  105367  000000G                 DECB    $SEC            ; DECREMENT THE SECTOR ADDRESS
 955
 956                                 ; SEARCH THE MDBSF, UDBSF, & THE FE BSF BUFFER FOR A MATCH.  IF ONE IS
 957                                 ; FOUND, SET THE BAD SECTOR BITS IN THE APPROPRIATE HEADER.
 958
 959 002756  004767  177252          3$.     CALL    $SRMDF          ; SEARCH THE MDBSF
 960 002762  103403                          BCS     4$              ; IF CS, MDBSF EXHAUSTED
 961 002764  042710  100000                  BIC     #100000,(R0)    ; CLEAR THE MF BIT (BAD)
 962 002770  000413                          BR      6$
 963
 964 002772  004767  177250          4$:     CALL    $SRUDF          ; SEARCH THE UDBSF
 965 002776  103403                          BCS     5$              ; IF CS, UDBSF EXHAUSTED
 966 003000  042710  040000                  BIC     #40000,(R0)     ; CLEAR THE UF BIT (BAD)
 967 003004  000405                          BR      6$
 968
 969 003006  004767  177210          5$:     CALL    $SRFES          ; SEARCH THE FE BSF
 970 003012  103402                          BCS     6$              ; IF CS, FE BSF EXHAUSTED
 971 003014  042710  140000                  BIC     #140000,(R0)    ; CLEAR BOTH THE UF & MF BITS (BAD)
 972 003020  112667  000000G         6$:     MOVB    (SP)+,$SEC      ; RESTORE THE SECTOR ADDRESS
 973 003024  066700  000000G                 ADD     SECSZ,R0        ; UPDATE BUFFER ADDRESS
 974 003030  000207                          RETURN
 975
 976         000001                  .END
```

SYMBOL TABLE

```
ERROU1  000050R        $CBDMG= ****** GX      $MSG18= ****** GX      $PRINT= ****** GX      $SRUDF  002246RG
GENHDR  002674RG       $CBOMG= ****** GX      $MSG19= ****** GX      $PRMPT  000250RG       $SSF   = ****** GX
IO.FER= ****** GX      $CVTDA  000132RG       $MSG21= ****** GX      $PRNTO= ****** GX      $SSFDT= ****** GX
IO.FEW= ****** GX      $CYL   = ****** GX     $MSG26= ****** GX      $RDBSF  000616RG       $SSFHD= ****** GX
IO.HMS= ****** GX      $DSKIO= ****** GX      $MSG40= ****** GX      $RDSSF  001452RG       $SSFLB= ****** GX
IO.RLB= ****** GX      $DVTBE= ****** GX      $MSG41= ****** GX      $READ  = ****** GX     $SSFLN= ****** GX
IO.WLT= ****** GX      $DVUNT= ****** GX      $MSG42= ****** GX      $SEC   = ****** GX     $SSFSZ= ****** GX
IQ.UMD= ****** GX      $EROCT  000074RG       $MSG43= ****** GX      $SMBSF  002270RG       $SWLST= ****** GX
IQ.X   = ****** GX     $EROUT  000032RG       $MSG44= ****** GX      $SNAME= ****** GX      $SWMOD= ****** GX
MSG     000336R        $FEBSF= ****** GX      $MSG58= ****** GX      $SORT   002552R        $TRK   = ****** GX
MSGER8= ****** GX      $HOMSK  000204RG       $MSG59= ****** GX      $SRALL  002150RG       $TSTDA  000454RG
RDBSF   001116R        $IOST  = ****** GX     $MULT   000342RG       $SRFES  002222RG       $TSTMX  000512RG
SECSZ  = ****** GX     $LBNH  = ****** GX     $MVPAT  000000RG       $SRMDF  002234RG       $UBSSZ= ****** GX
SSFCRC  000374R        $LBNL  = ****** GX     $MXCYL= ****** GX      $SRSDF  002260RG       $UDBSF= ****** GX
TEMP    002672R        $LOGDV= ****** GX      $MXLBN= ****** GX      $SRSSF  002342RG       $UNTID  000562RG
TRKSZ  = ****** GX     $MBSSZ= ****** GX      $MXSEC= ****** GX      $SRTMF  002462RG       $UPDMX  000414RG
VYBSF   001136R        $MDBSF= ****** GX      $MXTRK= ****** GX      $SRTRK  002106RG       $WTBSF  001224RG
$ALFLB= ****** GX      $MOVE   000240RG       $PAT   = ****** GX     $SRTSF  002426RG       $WTSSF  001760RG
$BUF   = ****** GX     $MSGWR= ****** GX      $PHYDV= ****** GX      $SRTUF  002516RG       $XFRSZ= ****** GX
$BUFRX= ****** GX

. ABS.  000000      000
        003032      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  908 WORDS  ( 4 PAGES)
DYNAMIC MEMORY:  7942 WORDS  ( 30 PAGES)
ELAPSED TIME:  00:00:31
[300,20]FMTSUB,[300,30]FMTSUB=[300,10]FMTSUB
```

```
                        .TITLE  FMTPRS - GET AND PARSE COMMAND LINE
                        .IDENT  /01.00/
                        ;
                        ; COPYRIGHT (C) 1980
                        ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                        ;
                        ;
                        ; THIS  SOFTWARE  IS  FURNISHED  UNDER A LICENSE FOR USE ONLY ON A
                        ; SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION
                        ; OF   THE   ABOVE COPYRIGHT  NOTICE.  THIS SOFTWARE, OR  ANY  OTHER
                        ; COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE  MADE AVAILABLE
                        ; TO  ANY  OTHER PERSON  EXCEPT  FOR USE ON SUCH SYSTEM AND TO ONE
                        ; WHO AGREES TO THESE LICENSE TERMS.  TITLE  TO  AND  OWNERSHIP OF
                        ; THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DIGITAL.
                        ;
                        ; THE  INFORMATION  IN THIS DOCUMENT IS SUBJECT  TO CHANGE WITHOUT
                        ; NOTICE AND SHOULD NOT BE CONSTRUED AS A  COMMITMENT  BY  DIGITAL
                        ; EQUIPMENT CORPORATION.
                        ;
                        ; DIGITAL ASSUMES NO  RESPONSIBILITY FOR THE USE OR RELIABILITY OF
                        ; ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
                        ;
                        ;
                        ; VERSION 01.00
                        ;
                        ; C. HESS        14-AUG-80
                        ; M. LEAVITT     14-OCT-80

                        .MCALL  DIR$, ISTAT$, STATE$, TRAN$
                        .MCALL  DEVDF$
000000                          DEVDF$
```

```
                                ;+
                                ; **-$PARSE-PARSE COMMAND LINE
                                ;
                                ; THIS ROUTINE WILL GET THE COMMAND LINE AND PARSE THE
                                ; DEVICE NAME AND SWITCHES.
                                ;
                                ; INPUTS:
                                ;       NONE.
                                ;
                                ; OUTPUTS:
                                ;       R1 = DEVICE UNIT NUMBER
                                ;       EXIT IS TO "$MESAG" IF ERROR.
                                ;       EXIT IS TO "$FMTEX" COMMAND LINE TERMINATED WITH "EOF".
                                ;-

000000  005227  177777          $PARSE::INC     #-1             ; FIRST TIME THRU HERE ?
000004  001404                  1$      BEQ     1$              ; BR IF YES
000006  012700  000000G                 MOV     #$STARS,R0      ; GET ADDRESS OF MESSAGE
000012  004767  000000G                 CALL    $PRINT          ; PRINT '**********'<LF>
000016  012700  000000G         1$:     MOV     #$MSG70,R0      ; GET PROMPT MESSAGE ADDRESS
000022  004767  000000G                 CALL    $PRMPT          ; PROMPT USER AND GET RESPONSE
000026  001006                  2$      BNE     2$              ; IF NE COMMAND LINE ENTERED
000030  122767  000000C 000000G         CMPB    #IE.EOF&377,$IOST       ; END OF INPUT?
000036  001367                  1$      BNE     1$              ; IF NE NO
000040  000167  000000G                 JMP     $FMTEX          ; EXIT GRACEFULLY

                                ; SETUP TO CALL THE PARSER

000044  005001                  2$:     CLR     R1              ; CLEAR OPTIONS WORD
000046  012702  000000'                 MOV     #UNTKT1,R2      ; GET KEYWORD TABLE ADDRESS
000052  012704  000000G                 MOV     #$BUFRX,R4      ; GET ADDRESS OF COMMAND LINE
000056  012705  000000'                 MOV     #UNIT,R5        ; GET STARTING STATE ADDRESS
000062  012767  177777  000000G         MOV     #-1,$DVUNT      ; RESET DEVICE NUMBER
000070  004767  000000G                 CALL    .TPARS          ; GO PARSE THE INPUT LINE
000074  103435                  SYNTAX  BCS     SYNTAX          ; IF CS SYNTAX ERROR
000076  005767  000000G                 TST     $DVUNT          ; NULL PARSE?
000102  100745                  1$      BMI     1$              ; IF MI, YES
000104  012700  000000G         3$:     MOV     #$MSG71,R0      ; GET PROMPT MESSAGE ADDRESS
000110  004767  000000G                 CALL    $PRMPT          ; PROMPT USER AND GET RESPONSE
000114  001006                  4$      BNE     4$              ; IF NE COMMAND LINE ENTERED
000116  122767  000000C 000000G         CMPB    #IE.EOF&377,$IOST       ; END OF INPUT?
000124  001002                  4$      BNE     4$              ; IF NE NO
000126  000167  000000G                 JMP     $FMTEX          ; EXIT GRACEFULLY
000132  005001                  4$:     CLR     R1              ; CLEAR OPTIONS WORD
000134  012702  000000'                 MOV     #OPTKT2,R2      ; GET KEYWORD TABLE ADDRESS
000140  012704  000000G                 MOV     #$BUFRX,R4      ; GET ADDRESS OF COMMAND LINE
000144  012705  000010'                 MOV     #OPTION,R5      ; GET STARTING STATE ADDRESS
000150  004767  000000G                 CALL    .TPARS          ; GO PARSE THE INPUT LINE
000154  103405                  SYNTAX  BCS     SYNTAX          ; IF CS SYNTAX ERROR
000156  004767  000134                  CALL    $TSTDV          ; MAKE A FINAL DEVICE TEST
000162  016701  000000G                 MOV     $DVUNT,R1       ; SET SPECIFIED UNIT NUMBER
000166  000207                  4$      RETURN                  ; ALL IS OK

000170  012700  000000G         SYNTAX: MOV     #$MSG2,R0       ; SYNTAX ERROR
000174  000167  000000G                 JMP     $MSGWR          ; EXIT WITH WARNING STATUS
```

```
                        ;+
                        ; STATE TABLE FOR 'TPARS' PARSER
                        ;-

                        .NLIST  MEB

000200                          ISTAT$  UNTST1,UNTKT1

                        ; READ DEVICE AND UNIT NUMBER

000200                          STATE$  UNIT
000200                          TRAN$   $EOS,$EXIT
000200                          TRAN$   $NUMBR,,$STUNT

000200                          STATE$                  ; TERMINATING STATE

000200                          ISTAT$  OPTST2,OPTKT2

                        ; PROCESS SWITCH OPTIONS

000200                          STATE$  OPTION
000200                          TRAN$   $EOS,$EXIT
000200                          TRAN$   "CSR",CSR
000200                          TRAN$   "VEC",VEC
000200                          TRAN$   "VFL",VFL
000200                          TRAN$   "ERL",ERL
000200                          TRAN$   "LI",LI
000200                          TRAN$   "FO",FO
000200                          TRAN$   "IN",IN

                        ; SET NEW CSR ADDRESS

000200                          STATE$  CSR
000200                          TRAN$   '=

000200                          STATE$
000200                          TRAN$   $NUMBR,SWITCH,$STCSR

                        ; SET NEW VECTOR ADDRESS

000200                          STATE$  VEC
000200                          TRAN$   '=

000200                          STATE$
000200                          TRAN$   $NUMBR,SWITCH,$STVEC

                        ; SET VERIFY LIMIT (NUMBER OF TIMES TO VERIFY A SECTOR)

000200                          STATE$  VFL
000200                          TRAN$   '=

000200                          STATE$
000200                          TRAN$   $DNUMB,SWITCH,$STVFL

                        ; SET ERROR LIMIT

```

```
                                        STATE$  ERL
                                        TRAN$   '=

                                        STATE$
                                        TRAN$   $DNUMB,SWITCH,$STERL

                                ; SET BAD BLOCK FILE/SKIP SECTOR FILE LIST

                                        STATE$  LI
                                        TRAN$   ':,LOG
                                        TRAN$   $LAMDA,SWITCH,$STLIS

                                        STATE$  LOG
                                        TRAN$   'L,SWITCH,$STLOG

                                ; SET FORMAT ENABLE

                                        STATE$  FO
                                        TRAN$   ':,FE
                                        TRAN$   $LAMDA,SWITCH,$STFOR

                                        STATE$  FE
                                        TRAN$   'F,SWITCH,$STFFE

                                ; SET BSF/SSF INITIALIZATION

                                        STATE$  IN
                                        TRAN$   $LAMDA,SWITCH,$STIN

                                ; CHECK FOR ANOTHER OPTION

                                        STATE$  SWITCH
                                        TRAN$   $EOS,$EXIT
                                        TRAN$   '/,OPTION

000200                                  STATE$                  ; NULL (TERMINATING) STATE

                                .ENABL  LSB

                                ;
                                ; DEVICE UNIT NUMBER
                                ;

000200  016767  000000G 000000G $STUNT::MOV     .PNUMB,$DVUNT   ; STORE UNIT NUMBER
000206  026727  000000G 000007          CMP     $DVUNT,#7       ; IS DRIVE NUMBER WITHIN RANGE ?
000214  101402                          BLOS    1$              ; IF LOS, YES
000216  000167  177746                  JMP     SYNTAX          ; SYNTAX ERROR

                                ;
                                ; DETERMINE DEVICE VALIDITY
                                ;

000222  016702  000000G         1$:     MOV     $DEVHD,R2       ; ADDRESS OF 1ST DCB IN SYSTEM
000226  026762  000000G 000004  2$:     CMP     $DVICE,D.NAM(R2); IS THIS THE RIGHT DCB?
000234  001407                          BEQ     4$              ; IF EQ YES
000236  016202  000000                  MOV     D.LNK(R2),R2    ; POINT TO NEXT DCB
000242  001371                          BNE     2$              ; LOOP TILL END OF LIST
```

```
205 000244  012700  000000G          3$:     MOV     #$MSG4,R0       ; DEVICE NOT IN SYSTEM
206 000250  000167  000000G                  JMP     $MESAG          ;
207 000254  116762  000000G 000006  4$:     MOVB    $DVUNT,D.UNIT(R2)       ; SET UNIT NUMBER RANGE
208 000262  116762  000000G 000007          MOVB    $DVUNT,D.UNIT+1(R2)
209 000270  016201  000002                  MOV     D.UCB(R2),R1    ; GET UCB ADDRESS
210 000274  116761  000000G 000006          MOVB    $DVUNT,U.UNIT(R1)       ; SAVE SPECIFIED UNIT NUMBER
211 000302  010167  000000G                  MOV     R1,$UCB         ; SAVE UCB ADDRESS
212 000306  016167  000020  000000G         MOV     U.SCB(R1),$SCB  ; SAVE SCB ADDRESS
213 000314  000207                          RETURN
214
215                                 ;
216                                 ; TEST DEVICE CHARACTERISTICS
217                                 ;
218
219 000316  016701  000000G          $TSTDV::MOV     $UCB,R1         ; FETCH UCB ADDRESS
220 000322  016702  000000G                  MOV     $SCB,R2         ; FETCH SCB ADDRESS
221 000326  016202  000012                  MOV     S.CSR(R2),R2    ; GET CSR ADDRESS
222 000332  013746  000004                  MOV     @#4,-(SP)       ; SAVE TRAP VECTOR
223 000336  012737  000570' 000004          MOV     #INTR,@#4       ; PUT OUR OWN IN ITS PLACE
224 000344  005712                          TST     (R2)            ; DOES CSR ADDRESS EXIST?
225 000346  103003                          BCC     5$              ; IF CC YES
226 000350  012637  000004                  MOV     (SP)+,@#4       ; RESTORE TRAP ADDRESS
227 000354  000733                          BR      3$              ; PRINT ERROR MESSAGE
228 000356  012637  000004          5$:     MOV     (SP)+,@#4       ; RESTORE TRAP ADDRESS
229 000362  012762  000040  000010  6$:     MOV     #40,10(R2)      ; CLEAR THE SUBSYSTEM
230 000370  116162  000006  000010          MOVB    U.UNIT(R1),10(R2)       ; SELECT UNIT
231 000376  016200  000026                  MOV     26(R2),R0       ; GET DRIVE TYPE REGISTER
232 000402  032762  010000  000010          BIT     #10000,10(R2)   ; SEE IF DRIVE PRESENT
233 000410  001016                          BNE     9$              ; IF NE, NO - 'NED' SET
234 000412  022700  024026                  CMP     #24026,R0       ; IS IT AN RM80 ?
235 000416  001403                          BEQ     7$              ; IF EQ, YES
236 000420  022700  020026                  CMP     #20026,R0       ; CHECK OTHER DEVICE TYPE CODE
237 000424  001013                          BNE     10$             ; IF NE, NO
238 000426  032762  010000  000012  7$:     BIT     #10000,12(R2)   ; IS DRIVE ONLINE (MOL SET) ?
239 000434  001401                          BEQ     8$              ; IF EQ, NO
240 000436  000207                          RETURN                  ;
241 000440  012700  000000G          8$:     MOV     #$MSG7,R0       ; DRIVE OFFLINE
242 000444  000405                          BR      11$             ;
243 000446  012700  000000G          9$:     MOV     #$MSG72,R0      ; DRIVE NOT PRESENT
244 000452  000402                          BR      11$             ;
245 000454  012700  000000G          10$:    MOV     #$MSG73,R0      ; DRIVE NOT AN RM80
246 000460  000167  000000G          11$:    JMP     $MESAG          ;
247
248                                 .DSABL  LSB
249
250                                 ;
251                                 ; SET NEW CSR ADDRESS
252                                 ;
253
254 000464  016702  000000G          $STCSR::MOV     $UCB,R2         ; FETCH UCB ADDRESS
255 000470  026727  000000G 160000          CMP     .PNUMB,#160000  ; CSR ADDRESS WITHIN RANGE?
256 000476  103003                          BHIS    1$              ; IF HIS YES
257 000500  012700  000000G                 MOV     #$MSG2,R0       ; SYNTAX ERROR
258 000504  000414                          BR      2$              ;
259 000506  013746  000004          1$:     MOV     @#4,-(SP)       ; SAVE INTERRUPT VECTOR
260 000512  012737  000570' 000004          MOV     #INTR,@#4       ; SET UP OUR OWN VECTOR
261 000520  005777  000000G                 TST     @.PNUMB         ; SEE IF IT'S ON THE BUS
```

```
262 000524  103006                         BCC     3$              ; IF CC IT'S THERE
263 000526  012637  000004                 MOV     (SP)+,@#4       ; RESTORE VECTOR
264 000532  012700  000000G                MOV     #$MSG28,R0      ; CSR NOT IN SYSTEM
265 000536  000167  000000G          2$:   JMP     $MESAG          ;
266 000542  142762  000001  000007  3$:   BICB    #US.OFL,U.ST2(R2)       ; SET DEVICE ON-LINE
267 000550  012637  000004                 MOV     (SP)+,@#4       ; RESTORE INTERRUPT VECTOR
268 000554  016702  000000G                MOV     $SCB,R2         ; GET SCB ADDRESS
269 000560  016762  000000G 000012         MOV     .PNUMB,S.CSR(R2)        ; SET CSR ADDRESS
270 000566  000207                         RETURN                  ;
271                                 ;
272                                 ; INTERRUPTS COME HERE ON NON-EXISTANT MEMORY TRAPS
273                                 ;
274 000570  052766  000001  000002  INTR:   BIS     #1,2(SP)        ;; SET CARRY BIT IN SAVED PS
275 000576  000002                         RTI                     ;; RETURN
276
277                                 ;
278                                 ; SET NEW VECTOR ADDRESS
279                                 ;
280
281 000600  016700  000000G         $STVEC::MOV     .PNUMB,R0       ; FETCH NEW VECTOR ADDRESS
282 000604  020027  001000                 CMP     R0,#1000        ; IS THE VECTOR IN A REASONABLE PLACE?
283 000610  103006                         BHIS    2$              ; IF HIS NO GOOD
284 000612  032700  000003                 BIT     #3,R0           ; MULTIPLE OF 4?
285 000616  001407                         BEQ     4$              ; IF EQ YES
286 000620  012700  000000G          1$:   MOV     #$MSG27,R0      ; NOT MULTIPLE OF FOUR
287 000624  000402                         BR      3$              ;
288 000626  012700  000000G          2$:   MOV     #$MSG2,R0       ; SYNTAX ERROR
289 000632  000167  000000G          3$:   JMP     $MESAG          ;
290 000636  016702  000000G          4$:   MOV     $SCB,R2         ; PICK UP SCB ADDRESS
291 000642  116202  000005                 MOVB    S.VCT(R2),R2    ; FETCH OLD VECTOR ADDRESS
292 000646  042702  177400                 BIC     #177400,R2      ; THIS IS CHEAPER THAN ANOTHER WAY
293 000652  006302                         ASL     R2              ; COMPUTE ABSOLUTE ADDRESS
294 000654  006302                         ASL     R2              ;
295 000656  011267  000000G                MOV     (R2),$INADD     ; GET RM80 DRIVER ISR ADDRESS
296 000662  012712  000000G                MOV     #$NONSI,(R2)    ; IGNORE INTERRUPT TO OLD VECTOR
297 000666  016710  000000G                MOV     $INADD,(R0)     ; NEW INTERRUPT VECTOR
298 000672  006200                         ASR     R0              ; FORM NEW VECTOR MOD4
299 000674  006200                         ASR     R0              ;
300 000676  016702  000000G                MOV     $SCB,R2         ; GET SCB ADDRESS
301 000702  110062  000005                 MOVB    R0,S.VCT(R2)    ; SET NEW VECTOR ADDRESS
302 000706  000207                         RETURN                  ;
303
304                                 ;
305                                 ; SET VERIFY COUNT
306                                 ;
307
308 000710  022767  000377  000000G $STVFL::CMP     #377,.PNUMB     ; LIMIT WITHIN RANGE?
309 000716  103407                         BLO     1$              ; IF LO NO
310 000720  005767  000000G                TST     .PNUMH          ; STILL?
311 000724  001004                         BNE     1$              ; IF NE NO
312 000726  116767  000000G 000000G        MOVB    .PNUMB,$VERIF   ; SET VERIFICATION COUNT
313 000734  000207                         RETURN                  ;
314 000736  000167  177226          1$:   JMP     SYNTAX          ; SYNTAX ERROR
315
316                                 ;
317                                 ; SET ERROR LIMIT
318                                 ;
```

```
319
320 000742  022767  000377  000000G $STERL::CMP     #377,.PNUMB     ; LIMIT WITHIN RANGE?
321 000750  103407                  BLO     1$              ; IF LO NO
322 000752  005767  000000G         TST     .PNUMH          ; STILL?
323 000756  001004                  BNE     1$              ; IF NE NO
324 000760  116767  000000G 000000G        MOVB    .PNUMB,$SWERL   ; SET ERROR LIMIT
325 000766  000207                  RETURN                  ;
326 000770  000167  177174          1$:     JMP     SYNTAX          ; SYNTAX ERROR
327
328                                 ;
329                                 ; SET LIST MODE
330                                 ;
331
332 000774  105267  000000G         $STLOG::INCB    $SWLOG          ; SET THE 'LIST LOGICAL' SWITCH
333 001000  105267  000000G         $STLIS::INCB    $SWLST          ; SET THE 'LIST' SWITCH
334 001004  000207                  RETURN                  ;
335
336                                 ;
337                                 ; SET ENABLE FORMAT SWITCH
338                                 ;
339
340 001006  105267  000000G         $STFFE::INCB    $SWFFE          ; SET THE 'FORMAT FE' SWITCH
341 001012  105267  000000G         $STFOR::INCB    $SWFMT          ; SET THE 'FORMAT' SWITCH
342 001016  000207                  RETURN
343
344                                 ;
345                                 ; SET BSF/SSF INITIALIZATION SWITCH
346                                 ;
347
348 001020  105267  000000G         $STIN:: INCB    $SWINT          ; SET THE 'INITIALIZATION' SWITCH
349 001024  000207                  RETURN
350
351         000001                  .END
```

SYMBOL TABLE

```
CSR      000050R     002 OPTKT2  000000RG    003 S3.RAL= 000010          U.FNUM= 000036          $DVUNT= ****** GX
DV.CCL= 000002           OPTST2  000010RG    002 S3.RCU= 000400          U.FPS = 000044          $EOS  = 000312
DV.COM= 020000           SPARE = 000010          S3.TAB= 000100          U.KCSR= 000032          $EXIT = 000000
DV.DIR= 000010           SP.EIP= 000001          S3.VER= 010000          U.KCS6= 000034          $FMTEX= ****** GX
DV.F11= 040000           SP.ENB= 000002          S3.WAL= 004000          U.LUIC  177774          $INADD= ****** GX
DV.ISP= 002000           SP.LUG= 000004          S3.WES= 000040          U.OWN   177776          $IOST = ****** GX
DV.MBC= 000400           SWITCH  000156R     002 S3.8BC= 000200          U.RED   000002          $LAMDA= 000300
DV.MNT= 100000           SYNTAX  000170R         UC.ALG= 000200          U.RPS = 000042          $MESAG= ****** GX
DV.MXD= 000100           S.BMSK  177776          UC.ATT= 000010          U.SCB   000020          $MSGWR= ****** GX
DV.OSP= 004000           S.BMSV  177774          UC.KIL= 000004          U.STS   000005          $MSG2 = ****** GX
DV.PSE= 010000           S.CON   000010          UC.LGH= 000003          U.ST2   000007          $MSG27= ****** GX
DV.REC= 000001           S.CSR   000012          UC.NPR= 000100          U.TCHP  000042          $MSG28= ****** GX
DV.SDI= 000020           S.CTM   000006          UC.PWF= 000020          U.TCVP  000043          $MSG4 = ****** GX
DV.SQD= 000040           S.FRK   000016          UC.QUE= 000040          U.TFLK  000040          $MSG7 = ****** GX
DV.SWL= 001000           S.ITM   000007          UNIT    000000R     002 U.TFRQ  000037          $MSG70= ****** GX
DV.TTY= 000004           S.LHD   000000          UNTKT1  000000RG    003 U.TLPP  000036          $MSG71= ****** GX
DV.UMD= 000200           S.PKT   000014          UNTST1  000000RG    002 U.TMTI  000047          $MSG72= ****** GX
D.DSP    000012          S.PRI   000004          US.ABO= 000001          U.TSTA  000026          $MSG73= ****** GX
D.LNK    000000          S.RCNT  177772          US.BSP= 000002          U.TTAB  000034          $NONSI= ****** GX
D.MSK    000014          S.ROFF  177773          US.BSY= 000200          U.TTYP  000046          $NUMBR= 000302
D.NAM    000004          S.STS   000011          US.CRW= 000004          U.TUX   000024          $PARSE  000000RG
D.PCB    000034          S.VCT   000005          US.DSB= 000010          U.UIC   000044          $PRINT= ****** GX
D.UCB    000002          S1.BEL= 000400          US.ECH= 000002          U.UNIT  000006          $PRMPT= ****** GX
D.UCBL   000010          S1.CTO= 000040          US.FOR= 000040          U.VCB = 000034          $RAD50= 000316
D.UNIT   000006          S1.CTS= 010000          US.FRK= 000002          U2.AT.= 000020          $SCB  = ****** GX
D.VCAN= 000002           S1.DEC= 002000          US.KPF= 000001          U2.CRT= 002000          $STARS= ****** GX
D.VDEB= 177776           S1.DPR= 001000          US.LAB= 000004          U2.DH1= 100000          $STCSR  000464RG
D.VINI= 000000           S1.DSI= 004000          US.MDE= 000002          U2.DJ1= 040000          $STERL  000742RG
D.VOUT= 000004           S1.ESC= 000004          US.MDM= 000020          U2.DZ1= 000100          $STFFE  001006RG
D.VPWF= 000006           S1.IBF= 100000          US.MNT= 000100          U2.ESC= 001000          $STFOR  001012RG
ERL      000100R     002 S1.IBY= 000200          US.OFL= 000001          U2.HFF= 010000          $STIN   001020RG
FE       000142R     002 S1.OBY= 000100          US.OUT= 000001          U2.HLD= 000040          $STLIS  001000RG
FO       000130R     002 S1.RAL= 000010          US.PUB= 000004          U2.LOG= 000400          $STLOG  000774RG
IE.EOF= ****** GX        S1.RNE= 000020          US.PWF= 000010          U2.LWC= 000001          $STRNG= 000304
IN       000150R     002 S1.RST= 000001          US.RED= 000002          U2.L3S= 000004          $STUNT  000200RG
INTR     000570R         S1.RUB= 000002          US.SHR= 000001          U2.L8S= 010000          $STVEC  000600RG
LI       000110R     002 S1.USI= 020000          US.SPU= 000002          U2.NEC= 004000          $STVFL  000710RG
LOG      000122R     002 S2.ACR= 000001          US.UMD= 000010          U2.PRV= 000010          $SUBXP= 000310
L.ASG    000010          S2.BRQ= 000020          US.VV = 000001          U2.RMT= 020000          $SWERL= ****** GX
L.LGTH= 000012           S2.CR = 000010          US.WCK= 000010          U2.RO4= 100000          $SWFFE= ****** GX
L.LNK    000000          S2.FDX= 100000          U.ACP = 000032          U2.SCS= 000004          $SWFMT= ****** GX
L.NAM    000002          S2.FLF= 040000          U.ATT   000022          U2.SLV= 000200          $SWINT= ****** GX
L.TYPE   000005          S2.HFF= 020000          U.BUF   000024          U2.VT5= 000002          $SWLOG= ****** GX
L.UCB    000006          S2.HFL= 003400          U.CBF = 000032          U2.7CH= 010000          $SWLST= ****** GX
L.UNIT   000004          S2.HHT= 010000          U.CLI   177772          VEC      000060R    002 $TSTDV  000316RG
M.BFVH   000011          S2.IRQ= 000200          U.CNT   000030          VFL      000070R    002 $UCB  = ****** GX
M.BFVL   000012          S2.ORQ= 000100          U.CTL   000004          $ALPHA= 000322          $VERIF= ****** GX
M.LGTH= 000014           S2.SRQ= 000040          U.CTYP  000050          $ANY  = 000320          $$$FLG= 177777
M.LNK    000000          S2.VFL= 004000          U.CW1   000010          $BLANK= 000306          $$$KEY= 000006
M.UMRA   000002          S2.WRA= 000006          U.CW2   000012          $BUFRX= ****** GX       $$$STA= 000010R    002
M.UMRN   000004          S2.WRB= 000002          U.CW3   000014          $DEVHD= ****** GX       $$$TMP= 000026R    004
M.UMVH   000010          S3.BCC= 020000          U.CW4   000016          $DIGIT= 000324          .PNUMB= ****** GX
M.UMVL   000006          S3.DAO= 040000          U.DCB   000000          $DNUMB= 000314          .PNUMH= ****** GX
OPTION   000010R     002 S3.PCU= 100000          U.FCDF= 000040          $DVICE= ****** GX       .TPARS= ****** GX

. ABS.   177776      000
         001026      001
```

SYMBOL TABLE

```
$STATE  000166      002
$KTAB   000016      003
$KSTR   000031      004
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  3819 WORDS  ( 15 PAGES)
DYNAMIC MEMORY:  7942 WORDS  ( 30 PAGES)
ELAPSED TIME:  00:01:21
[300,20]FMTPRS,[300,30]FMTPRS=[1,1]EXEMC/ML,[300,10]FMTPRS
```

```
                            .TITLE  FMTR80 - FORMAT/VERIFY RM80 DISKS
                            .IDENT  /01.00/
                            ;
                            ;
                            ; COPYRIGHT (C) 1980
                            ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                            ;
                            ;
                            ; THIS  SOFTWARE  IS  FURNISHED  UNDER A LICENSE FOR USE ONLY ON A
                            ; SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION
                            ; OF  THE  ABOVE COPYRIGHT  NOTICE.  THIS SOFTWARE, OR  ANY  OTHER
                            ; COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE  MADE AVAILABLE
                            ; TO  ANY  OTHER PERSON  EXCEPT  FOR USE ON SUCH SYSTEM AND TO ONE
                            ; WHO AGREES TO THESE LICENSE TERMS.  TITLE  TO  AND  OWNERSHIP OF
                            ; THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DIGITAL.
                            ;
                            ; THE  INFORMATION  IN THIS DOCUMENT IS SUBJECT  TO CHANGE WITHOUT
                            ; NOTICE AND SHOULD NOT BE CONSTRUED AS A  COMMITMENT  BY  DIGITAL
                            ; EQUIPMENT CORPORATION.
                            ;
                            ; DIGITAL ASSUMES NO  RESPONSIBILITY FOR THE USE OR RELIABILITY OF
                            ; ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
                            ;
                            ;
                            ;  VERSION 01.00
                            ;
                            ; C. HESS       18-AUG-80
                            ; M. LEAVITT    24-DEC-80
                            ;
                            .NLIST  CND,BEX


                            ; MESSAGES

000000  012  105  116  MSG1R8: .ASCIZ  <LF>/ENTER DATE (DD-MMM-YY): /
000032  110  104  101  MSG2R8: .ASCIZ      /HDA SERIAL NUMBER: /
                            .EVEN

                            ; DATE FIELDS

000056  112  101  116  DAT:     .ASCIZ  /JAN/
000062  106  105  102           .ASCIZ  /FEB/
000066  115  101  122           .ASCIZ  /MAR/
000072  101  120  122           .ASCIZ  /APR/
000076  115  101  131           .ASCIZ  /MAY/
000102  112  125  116           .ASCIZ  /JUN/
000106  112  125  114           .ASCIZ  /JUL/
000112  101  125  107           .ASCIZ  /AUG/
000116  123  105  120           .ASCIZ  /SEP/
000122  117  103  124           .ASCIZ  /OCT/
000126  116  117  126           .ASCIZ  /NOV/
000132  104  105  103           .ASCIZ  /DEC/
000136  000000                  .WORD   0
                            .EVEN

                            ; VALID LIMITS FOR DATE FIELDS
```

```
 60 000140  000001  000014          DATLMT: .WORD   1,12.           ; FOR MONTH
 61 000144  000001  000037                  .WORD   1,31.           ; FOR DAY
 62 000150  000000  000143                  .WORD   0,99.           ; FOR YEAR
 63 000154                          DPARS:: .BLKW   3               ; STORAGE FOR DATE CONVERSION
 64
 65         000116                  BASYR   == 78.                  ; BASE YEAR = 1978
 66
```

```
                        ;+
                        ; **-$RM80-FORMAT/VERIFY RM80 DISK PACKS
                        ;
                        ; THIS ROUTINE WILL INITIATE THE FORMAT AND/OR VERIFICATION
                        ; OF AN RM80 DISK PACK.
                        ;
                        ; INPUTS:
                        ;       DISK PARAMETERS SET TO ZERO
                        ;
                        ; OUTPUTS:
                        ;       NONE.
                        ;-
                        ;
000162  105767  000000G         $RM80:: TSTB    $SWINT          ; SEE IF SSF/BSF INITIALIZATION MODE
000166  001022                          BNE     1$              ; IF NE, YES
000170  004767  000000G                 CALL    $RDSSF          ; READ SKIP SECTOR FILE
000174  004767  000000G                 CALL    $RDBSF          ; READ BAD SECTOR FILES
000200  016767  000010G 000000G         MOV     $SSF+10,$REVNM  ; GET CURRENT REVISION NUMBER
000206  005767  000310                  TST     8$              ; HAS DATE BEEN ENTERED AT LEAST ONCE ?
000212  001010                          BNE     1$              ; BR IF YES
000214  012700  000000'                 MOV     #MSG1R8,R0      ; 'ENTER DATE (DD-MMM-YY): '
000220  004767  000000G                 CALL    $PRMPT          ; PROMPT THE USER - USER'S RESPONSE MAY BE EITHER:
                                                                ;  'MM/DD/YY' OR 'DD-MMM-YY'
000224  004767  000274                  CALL    DATE            ; CONVERT USER'S ENTRY TO A BINARY DATE
000230  005267  000266                  INC     8$              ; INDICATE THAT DATE HAS BEEN ENTERED ONCE
                        ;
000234  105767  000000G         1$:     TSTB    $SWLST          ; DISPLAY THE DEC STD 144 AND SSF FILES ?
000240  001405                          BEQ     2$              ; IF EQ, NO
000242  004767  000000G                 CALL    $RM80L          ; DISPLAY  HE FILES
000246  105767  000000G                 TSTB    $SWFMT          ; FORMAT THE HDA ALSO ?
000252  001522                          BEQ     7$              ; IF EQ, NO
                        ;
                        ; INFORM THE OPERATOR THAT DATA WILL BE DESTROYED
                        ;
000254  012703  000000G         2$:     MOV     #$MSG23,R3      ; WARNING MESSAGE ADDRESS
000260  004767  000000G                 CALL    $UNTID          ; STORE MESSAGE AND DEVICE/UNIT
000264  012703  000000G                 MOV     #$MSG24,R3      ; ADDRSS OF MESSAGE SUFFIX
000270  004767  000000G                 CALL    $MOVE           ; MOVE THE SUFFIX
000274  012700  000144G                 MOV     #$BUF+100.,R0   ; PRINT WARNING MESSAGE
000300  004767  000000G                 CALL    $PRINT          ;
000304  012700  000000G                 MOV     #$MSG31,R0      ; GET CONFIRMATION ADDRESS
000310  004767  000000G                 CALL    $PRMPT          ; PROMPT AND GET OPERATOR'S RESPONSE
000314  001501                          BEQ     7$              ; IF EQ, RESPONSE WAS 'NO'
000316  122767  000131  000000G         CMPB    #'Y,$BUFRX      ; SEE IF RESPONSE WAS 'YES'
000324  001075                          BNE     7$              ; IF NE, RESPONSE WAS 'NO'
000326  105767  000000G                 TSTB    $SWINT          ; SEE IF SSF/BSF INITIALIZATION MODE
000332  001013                          BNE     3$              ; IF NE, YES
000334  012700  000000G                 MOV     #$MSGR8,R0      ; GET UPDATE MODE MESSAGE
000340  004767  000000G                 CALL    $PRMPT          ; PROMPT AND GET RESPONSE
000344  001451                          BEQ     6$              ; IF EQ, NO
000346  122767  000131  000000G         CMPB    #'Y,$BUFRX      ; WAS RESPONSE "Y"?
000354  001045                          BNE     6$              ; IF NE, NO
000356  105267  000000G                 INCB    $SWMOD          ; SET THE MODIFY FLAG
000362  012700  000000'         3$:     MOV     #MSG1R8,R0      ; 'ENTER DATE (DD-MMM-YY): '
000366  004767  000000G                 CALL    $PRMPT          ; PROMPT THE USER - USER'S RESPONSE MAY BE EITHER:
                                                                ;  'MM/DD/YY' OR 'DD-MMM-YY'
000372  004767  000126                  CALL    DATE            ; CONVERT USER'S ENTRY TO A BINARY DATE
```

```
                                                        INC     8$              ; INDICATE THAT DATE HAS BEEN ENTERED ONCE

000402  012700  000032'          4$:   MOV     #MSG2R8,R0      ; 'HDA SERIAL NUMBER: '
000406  004767  000000G                CALL    $PRMPT          ; PROMPT OPERATOR
000412  010304                        MOV     R3,R4           ; NUMBER OF BYTES ENTERED TO R4
000414  012705  000000G                MOV     #$BUFRX,R5      ; INPUT BUFFER ADDRESS
000420  012703  000000G                MOV     #$SNAME,R3      ; OUTPUT BUFFER ADDRESS
000424  004767  000000G                CALL    .DD2CT          ; CONVERT THE S/N TO BINARY
000430  103764                        BCS     4$              ; IF CS, BAD S/N ENTRY
000432  016746  000000G                MOV     $SNAME,-(SP)    ; CORRECT THE SERIAL NUMBER ORDER
000436  016767  000002G 000000G        MOV     $SNAME+2,$SNAME ; ...
000444  012667  000002G                MOV     (SP)+,$SNAME+2  ; ...
000450  105767  000000G                TSTB    $SWINI          ; INITIALIZE THE BAD SECTOR FILES ?
000454  001403                        BEQ     5$              ; IF EQ, NO
000456  004767  000000G                CALL    $RM80I          ; INIT THE DEC STD 144 AND SKIP SECTOR FILES
000462  000416                        BR      7$              ; EXIT

000464  004767  000000G          5$:   CALL    $RM80M          ; GET UPDATE MODE INFORMATION
000470  004767  000000G          6$:   CALL    $RDSSF          ; READ SKIP SECTOR FILE
000474  004767  000000G                CALL    $RDBSF          ; READ BAD SECTOR FILES
000500  004767  000000G                CALL    $RM80F          ; FORMAT THE DISK
000504  004767  000000G                CALL    $RM80V          ; VERIFY THE DISK FORMAT
000510  012700  000000G                MOV     #$MSG12,R0      ; GET ADDRESS OF MESSAGE
000514  004767  000000G                CALL    $PRINT          ; PRINT 'DONE !' MESSAGE
000520  000207                  7$:   RETURN

000522  000000                  8$:   .WORD   0               ; ONCE DATE FLAG
```

```
                                        ;+
                                        ; **-DATE-CONVERT USER'S RESPONSE TO BINARY DATE
                                        ;
                                        ; THIS ROUTINE WILL CONVERT THE ASCII DATE INPUTTED
                                        ; INTO '$BUFRX' AS EITHER [MM/DD/YY] OR [DD-MMM-YY] AND
                                        ; CONVERT IT TO BINARY AND PUT IT IN DATE PARAMETER
                                        ; STORAGE AREA.
                                        ;
                                        ; INPUTS:
                                        ;       '$BUFRX' CONTAINS ASCII STRING FOR DATE
                                        ;
                                        ; OUTPUTS:
                                        ;       DPARS CONTAINS BINARY REPRESENTATION
                                        ;       OF DATE
                                        ;
                                        ;-
000524  012701  000154'         DATE:  MOV     #DPARS,R1        ; POINT TO PARAMTER AREA
000530  012702  000003                 MOV     #3,R2            ; SET WORD COUNT
000534  005021                 1$:     CLR     (R1)+            ; ZERO NEXT WORD
000536  005302                         DEC     R2               ; DONE ?
000540  001375                         BNE     1$               ; IF NE NO
000542  012700  000000G                MOV     #$BUFRX,R0       ; GET BUFFER FOR USER'S RESPONSE
000546  004767  000000G         2$:    CALL    $CDTB            ; GET A NUMBER
000552  120227  000057                 CMPB    R2,#'/           ; TERMINATOR FOR DATE ?
000556  001466                         BEQ     10$              ; IF EQ YES
000560  120227  000055                 CMPB    R2,#'-           ; POSSIBLY DIFFERENT FORMAT ?
000564  001404                         BEQ     3$               ; IF EQ YES
000566  012700  000000G                MOV     #$MSG2,R0        ; NO, SYNTAX ERROR
000572  000167  000000G                JMP     $MSGWR           ; EXIT WITH WARNING STATUS

                                ; GET DATE FIELD IN FORMAT [DD-MMM-YY]

000576  012705  000144'         3$:    MOV     #DATLMT+4,R5     ; POINT TO DAY LIMITS
000602  004767  000256                 CALL    TMTSTL           ; TEST LIMITS
000606  103446                         BCS     9$               ; IF CS ERROR
000610  010167  177342                 MOV     R1,DPARS+2       ; STORE DAY
000614  010003                         MOV     R0,R3            ; COPY BUFFER POINTER
000616  012701  000056'                MOV     #DAT,R1          ; POINT TO VALID MONTH NAMES
000622  005002                         CLR     R2               ; ZERO MONTH NUMBER
000624  105711                 4$:     TSTB    (R1)             ; END OF TABLE ?
000626  001411                         BEQ     6$               ; IF EQ YES, GOOD MONTH
000630  122021                         CMPB    (R0)+,(R1)+      ; NO, CHECK NEXT CHARACTER
000632  001774                         BEQ     4$               ; AND LOOP IF GOOD MATCH
000634  105721                 5$:     TSTB    (R1)+            ; END OF MONTH STRING ?
000636  001376                         BNE     5$               ; IF NE NO, LOOP
000640  010300                         MOV     R3,R0            ; YES, RESET BUFFER POINTER
000642  005202                         INC     R2               ; UPDATE MONTH INDEX
000644  105711                         TSTB    (R1)             ; END OF TABLE ?
000646  001366                         BNE     4$               ; IF NE NO, LOOP
000650  000425                         BR      9$               ; ZERO IS END OF MONTH LIST
000652  005202                 6$:     INC     R2               ; INCREMENT MONTH INDEX
000654  010267  177274                 MOV     R2,DPARS         ; SAVE IT
000660  112002                         MOVB    (R0)+,R2         ; GET NEXT CHARACTER
000662  120227  000055                 CMPB    R2,#'-           ; LEGAL TERMINATOR ?
000666  001401                         BEQ     7$               ; IF EQ YES
000670  000415                         BR      9$               ; NO, IT'S A SYNTAX ERROR
```

```
211 000672  004767  000000G          7$:     CALL    $CDTB           ; YES, GET YEAR
212 000676  012705  000150'                  MOV     #DATLMT+10,R5   ; GET LIMIT ON YEAR
213 000702  004767  000156                   CALL    TMTSTL          ; TEST LIMIT ON YEAR
214 000706  103406                         BCS     9$              ; IF CS ERROR
215 000710  020127  000116                   CMP     R1,#BASYR       ; CHECK YEAR ENTRY
216 000714  103403                         BLO     9$              ; IF LO, INVALID YEAR ENTRY
217 000716  010167  177236          8$:     MOV     R1,DPARS+4      ; SAVE YEAR
218 000722  000436                         BR      12$             ; EXIT
219 000724  012700  000000G          9$:     MOV     #$MSG2,R0       ; GET SYNTAX ERROR MESSAGE
220 000730  000167  000000G                  JMP     $MSGWR          ; EXIT WITH WARNING STATUS
221
222                                  ; GET DATE FIELD IN FORMAT [MM/DD/YY]
223
224 000734  012703  000154'          10$:    MOV     #DPARS,R3       ; POINT TO DATE STORAGE FIELD
225 000740  012705  000140'                  MOV     #DATLMT,R5      ; GET POINTER TO MONTH LIMITS
226 000744  004767  000114           11$:    CALL    TMTSTL          ; TEST LIMITS
227 000750  103765                         BCS     9$              ; IF CS ERROR, OUT OF BOUNDS
228 000752  010125                         MOV     R1,(R3)+        ; STORE IT
229 000754  120227  000057                   CMPB    R2,#'/          ; GOOD TERMINATOR ?
230 000760  001361                         BNE     9$              ; IF NE NO, SYNTAX ERROR
231 000762  004767  000000G                  CALL    $CDTB           ; GET NEXT TWO DIGITS
232 000766  004767  000072                   CALL    TMTSTL          ; TEST LIMITS
233 000772  103754                         BCS     9$              ; IF CS ERROR, OUT OF BOUNDS
234 000774  010123                         MOV     R1,(R3)+        ; STORE IT
235 000776  120227  000057                   CMPB    R2,#'/          ; GOOD TERMINATOR ?
236 001002  001350                         BNE     9$              ; IF NE NO
237 001004  004767  000000G                  CALL    $CDTB           ; GET NEXT TWO DIGITS
238 001010  004767  000050                   CALL    TMTSTL          ; TEST LIMITS
239 001014  103743                         BCS     9$              ; IF CS ERROR, OUT OF BOUNDS
240 001016  010113                         MOV     R1,(R3)         ; STORE IT
241 001020  012703  000154'          12$:    MOV     #DPARS,R3       ; DATE STORAGE FIELD
242 001024  162763  000116  000004          SUB     #BASYR,4(R3)    ; SUBTRACT THE STARTING YEAR
243 001032  000363  000004                   SWAB    4(R3)           ; POSITION YEAR IN UPPER BYTE
244 001036  006363  000004                   ASL     4(R3)           ; SHIFT INTO BITS 9 - 15
245 001042  000313                         SWAB    (R3)            ; POSITION MONTH
246 001044  006213                         ASR     (R3)            ; SHIFT INTO BITS 5 - 8
247 001046  006213                         ASR     (R3)            ;
248 001050  006213                         ASR     (R3)            ;
249 001052  056313  000004                   BIS     4(R3),(R3)      ; MERGE YEAR WITH MONTH
250 001056  056313  000002                   BIS     2(R3),(R3)      ; MERGE DAY
251 001062  000207                         RETURN                  ; EXIT
252
253                                  ;+
254                                  ; **-TMTSTL - TEST DATE LIMITS
255                                  ;
256                                  ; THIS ROUTINE WILL TEST THE DATE AGAINST LIMITS SPECIFIED
257                                  ; BY A TWO WORD LIMIT ENTRY.
258                                  ;
259                                  ; INPUTS:
260                                  ;       R1 =    CONTAINS VALUE TO BE TESTED
261                                  ;       R5 =    POINTS TO TWO WORD LIMIT ENTRY
262                                  ;               WORD 1 - LOW LIMIT OF VALUE
263                                  ;               WORD 2 - HIGH LIMIT OF VALUE
264                                  ;
265                                  ; OUTPUTS:
266                                  ;       R1 =    UNCHANGED
267                                  ;       R5 =    UPDATED TO NEXT LIMIT ENTRY
```

```
                                ;       CC =    GOOD VALUE, LOW<=X<-HIGH
                                ;       CS =    BAD VALUE,  LOW>X OR X>HIGH
                                ;
                                ;-

001064  020125                  TMTSTL: CMP     R1,(R5)+        ; IS VALUE LOWER THAN LOW LIMIT ?
001066  002404                          BLT     1$              ; IF LT YES
001070  020125                          CMP     R1,(R5)+        ; IS IT HIGHER THAN HIGH LIMIT ?
001072  003002                          BGT     1$              ; IF GT YES
001074  000241                          CLC                     ; NO, GOO VALUE
001076  000207                          RETURN                  ; EXIT
001100  000261                  1$:     SEC                     ; SET ERROR FLAG
001102  000207                          RETURN                  ; EXIT

        000001                  .END
```

SYMBOL TABLE

```
BASYR = 000116 G        MSG2R8  000032R        $MSG12= ****** GX       $RDSSF= ****** GX       $SNAME= ****** GX
CR    = 000015          TMTSTL  001064R        $MSG2 = ****** GX       $REVNM= ****** GX       $SSF  = ****** GX
DAT     000056R         $BUF  = ****** GX       $MSG23= ****** GX       $RM80   000162RG        $SWFMT= ****** GX
DATE    000524R         $BUFRX= ****** GX       $MSG24= ****** GX       $RM80F= ****** GX       $SWINT= ****** GX
DATLMT  000140R         $CDTB = ****** GX       $MSG31= ****** GX       $RM80I= ****** GX       $SWLST= ****** GX
DPARS   000154RG        $MOVE = ****** GX       $PRINT= ****** GX       $RM80L= ****** GX       $SWMOD= ****** GX
LF    = 000012          $MSGR8= ****** GX       $PRMPT= ****** GX       $RM80M= ****** GX       $UNTID= ****** GX
MSG1R8  000000R         $MSGK = ****** GX       $RDBSF= ****** GX       $RM80V= ****** GX       .DD2CT= ****** GX

. ABS.  000000      000
        001104      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  456 WORDS  ( 2 PAGES)
DYNAMIC MEMORY:  7942 WORDS  ( 30 PAGES)
ELAPSED TIME:  00:00:11
[300,20]FMTR80,[300,30]FMTR80=[300,10]FMTR80
```

```
                            .TITLE FMTR8F - RM80 FORMAT MODULE
                            .IDENT /02.00/
                            ;
                            ;
                            ; COPYRIGHT (C) 1980,1982
                            ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                            ;
                            ;
                            ; THIS  SOFTWARE  IS  FURNISHED  UNDER A LICENSE FOR USE ONLY ON A
                            ; SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION
                            ; OF  THE  ABOVE COPYRIGHT  NOTICE.  THIS SOFTWARE, OR  ANY  OTHER
                            ; COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE  MADE AVAILABLE
                            ; TO  ANY  OTHER PERSON  EXCEPT  FOR USE ON SUCH SYSTEM AND TO ONE
                            ; WHO AGREES TO THESE LICENSE TERMS.  TITLE  TO  AND  OWNERSHIP OF
                            ; THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DIGITAL.
                            ;
                            ; THE  INFORMATION  IN THIS DOCUMENT IS SUBJECT  TO CHANGE WITHOUT
                            ; NOTICE AND SHOULD NOT BE CONSTRUED AS A  COMMITMENT  BY  DIGITAL
                            ; EQUIPMENT CORPORATION.
                            ;
                            ; DIGITAL ASSUMES NO  RESPONSIBILITY FOR THE USE OR RELIABILITY OF
                            ; ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
                            ;
                            ;
                            ; VERSION 02.00
                            ;
                            ; C. HESS       17-AUG-80
                            ; M. LEAVITT    02-JUL-82
                            ;
                            .NLIST  BEX
000000   104   122   111    MSG50R: .ASCIZ  /DRIVE ERROR DURING FORMAT/
000032   106   117   122    MSG51R: .ASCIZ  /FORMAT CONTINUING/
000054   106   117   122    MSG53R: .ASCIZ  /FORMAT TERMINATING/
000077   122   115   103    MSG61R: .ASCIZ  /RMCS1=/
000106   122   115   103    MSG62R: .ASCIZ  /RMCS2=/
000115   122   115   105    MSG63R: .ASCIZ  /RMER1=/
000124   122   115   105    MSG64R: .ASCIZ  /RMER2=/
000133   122   115   104    MSG65R: .ASCIZ  /RMDA=/
000141   124   122   101    MSG66R: .ASCIZ  /TRACK RE-WRITE SUCCESSFUL/
000173   124   122   101    MSG67R: .ASCIZ  /TRACK RE-WRITE FAILURE/
000222   123   125   103    MSG68R: .ASCIZ  /SUCCESSIVE SEEK ERRORS/
                            .EVEN
                            ;
                            ; WORKING LOCATIONS
                            ;
000252  000000             $ERCS1: .WORD   0               ; CONTROL STATUS REG 1
000254  000000             $ERCS2: .WORD   0               ; STATUS REG 2
000256  000000             $ERER1: .WORD   0               ; ERROR REGISTER 1
000260  000000             $ERER2: .WORD   0               ; ERROR REGISTER 2
000262  000000             $ERDA:  .WORD   0               ; TRACK/SECTOR ADDRESS
000264  000000             $FLBUF: .WORD   0               ; BUFFER ADDRESS
```

```
                                ;+
                                ; **-$R80TF-FORMAT A SINGLE TRACK
                                ; **-$R80SF-FORMAT BSF, SSF AND ALTERNATE DEC144 CYLINDERS
                                ; **-$RM80F-FORMAT ALL CYLINDERS OR FE CYLINDERS
                                ;
                                ;
                                ; THIS ROUTINE WILL FORMAT AN RM80 DISK PACK.  FULL TRACKS
                                ;  (32. SECTORS) OF HEADERS ARE WRITTEN UNTIL THE MAXIMUM
                                ;  DISK ADDRESS IS REACHED.
                                ;
                                ; INPUTS:
                                ;       DISK PARAMETERS SET TO ZERO
                                ;       MDBSF, UDBSF, & SSF READ, VALIDATED, AND IN MEMORY
                                ;       $CYL,$TRK,$SEC PRESET FOR '$R80TF' ENTRY
                                ;       FORMAT FE CYLINDERS ONLY IF '$SWFFE=1'
                                ;
                                ; OUTPUTS:
                                ;       NONE.
                                ;-
                                ;
                                ; FORMAT BSF(558.,13.), SSF(559.,0) AND ALTERNATE DEC144(559.,1) TRACKS
                                ;
000266  012767  001056  000000G $R80SF::MOV     #558.,$CYL      ; STARTING CYLINDER
000274  012767  000015  000000G         MOV     #13.,$TRK       ;          "     TRACK
000302  004767  000212          .       CALL    FMTDSK          ; FORMAT THE DEC STD 144 TRACK
000306  012767  001057  000000G         MOV     #559.,$CYL      ; STARTING CYLINDER
000314  005067  000000G                 CLR     $TRK            ;          "     TRACK
000320  004767  000174                  CALL    FMTDSK          ; FORMAT THE SSF TRACK
000324  005267  000000G                 INC     $TRK            ; FORMAT THE ALTERNATE DEC144 TRACK
                                ;
                                ; FORMAT A SINGLE TRACK
                                ;
000330  004767  000000G         $R80TF::CALL    $MVPAT          ; LOAD THE DATA PATTERN
000334  000471                  .       BR      FMTDSK          ; FORMAT THE TRACK $CYL,$TRK
                                ;
                                ;
                                ; FORMAT ALL CYLINDERS (0,0 TO 560.,13.) OR BSF/SSF/FE CYLINDERS (558.,13.- 560.,13.)
                                ;
000336  005001                  $RM80F::CLR     R1              ; SET NULL CARRIAGE CONTROL
000340  012700  000000G                 MOV     #$MSG6,R0       ; TELL OPERATOR
000344  004767  000000G                 CALL    $PRNTO          ; 'START FORMATTING'
000350  005001                  .       CLR     R1              ; SET NULL CARRIAGE CONTROL
000352  105767  000000G                 TSTB    $SWFFE          ; FORMAT FE CYLINDERS ONLY ?
000356  001011                  .       BNE     1$              ; BR IF YES
000360  012700  000000G                 MOV     #$MSG6A,R0      ; TELL OPERATOR WHICH CYLINDERS
000364  004767  000000G                 CALL    $PRNTO          ; ' -ALL CYLINDERS- '
000370  005067  000000G                 CLR     $CYL            ; START CYLINDER=0
000374  005067  000000G                 CLR     $TRK            ; START TRACK=0
000400  000412                  .       BR      2$              ; FORMAT THE TRACKS
                                ;
000402  012700  000000G         1$:     MOV     #$MSG6B,R0      ; TELL OPERATOR WHICH CYLINDERS
000406  004767  000000G                 CALL    $PRNTO          ; ' -FE CYLINDERS- '
000412  012767  001056  000000G         MOV     #558.,$CYL      ; START CYLINDER=558. (BSF/SSF/FE CYLINDERS)
000420  012767  000015  000000G         MOV     #13.,$TRK       ; START TRACK=13.
000426  004767  000000G         2$:     CALL    $MVPAT          ; LOAD THE DATA PATTERN
000432  004767  000062          3$:     CALL    FMTDSK          ; FORMAT THE DISK
000436  004767  000000G                 CALL    $TSTMX          ; UPDATE DISK ADDRESSES
```

```
115 000442  103001                          BCC     4$              ; IF CS, NOT FINSIHED, ELSE
116 000444  000207                          RETURN                  ; EXIT FORMAT
117
118 000446  022767  001057  000000G 4$:     CMP     #559.,$CYL      ; WAS SSF JUST WRITTEN ?
119 000454  001366                          BNE     3$              ; BR IF NO
120 000456  022767  000002  0000C0G         CMP     #2,$TRK         ; WERE BSF/SSF TRACKS JUST WRITTEN ?
121 000464  001362                          BNE     3$              ; BR IF NO, ELSE
122 000466  016746  000000G                 MOV     $CYL,-(SP)      ; SAVE CURRENT CYLINDER ADDRESS
123 000472  016746  000000G                 MOV     $TRK,-(SP)      ; SAVE CURRENT TRACK ADDRESS
124 000476  004767  000000G                 CALL    $WTSSF          ; RE-WRITE THE SSF
125 000502  004767  000000G                 CALL    $WTBSF          ; RE-WRITE ALL THE DEC144 TRACKS
126 000506  012667  000000G                 MOV     (SP)+,$TRK      ; RESTORE TRACK ADDRESS
127 000512  012667  000000G                 MOV     (SP)+,$CYL      ; RESTORE CYLINDER ADDRESS
128 000516  000745                          BR      3$              ; WRITE NEXT TRACK
129
130                                 ; ACTUAL FORMAT ROUTINE
131
132 000520  012703  000000G         FMTDSK: MOV     #$PHYDV,R3      ; 'FORMAT' DEVICE PARAMETERS
133 000524  004767  000000G                 CALL    $UPDMX          ; UPDATE THE MAX VALUES
134 000530  005067  000000G                 CLR     $SEC            ; START AT SECTOR ZERO
135 000534  012700  000000G                 MOV     #$BUF,R0        ; RELOAD THE BUFFER ADDRESS
136
137                                 ; CREATE THE HEADER PATTERN
138
139 000540  004767  000000G         1$:     CALL    GENHDR          ; GENERATE HEADER PATTERN
140 000544  005267  000000G                 INC     $SEC            ; UPDATE SECTOR NUMBER
141 000550  026767  000000G 000000G         CMP     $SEC,$MXSEC     ; ALL SECTORS DONE?
142 000556  001370                          BNE     1$              ; IF NE NO
143 000560  005067  000000G                 CLR     $SEC            ; RESET SECTOR NUMBER
144
145                                 ; NOW WRITE THE HEADER AND DATA ONTO THE DISK.
146
147 000564  016767  000000G 000000G 2$:     MOV     TRKSZ,$XFRSZ    ; LOAD BYTE COUNT FOR ONE TRACK
148 000572  004767  000000G                 CALL    $CVTDA          ; CONVERT DISK ADDRESS TO LBN
149 000576  012705  000000G                 MOV     #$BUF,R5        ; SET BUFFER ADDRESS
150 000602  004767  000000G         3$:     CALL    $DSKWH          ; WRITE HEADER/DATA FOR THIS TRACK
151 000606  005767  000000G         4$:     TST     $RGBU           ; ANY DISK ERRORS?
152 000612  100401                          BMI     5$              ; IF MI, YES
153 000614  000207                          RETURN                  ; OTHERWISE, RETURN
154
155                                 ; HANDLE THE ERROR
156
157 000616  032767  004000  000014G 5$:     BIT     #4000,$RGBUF+14 ; WRITE LOCK ERROR?
158 000624  001402                          BEQ     6$              ; IF EQ NO
159 000626  000167  000000G                 JMP     $WLKER          ; INFORM THE USER
160
161                                 ; SEND ERROR MESSAGE
162
163 000632  004767  000276          6$:     CALL    $FLRGS          ; SAVE FAILING DATA
164 000636  012700  000000'                 MOV     #MSG50R,R0      ; 'DRIVE ERROR DURING FORMAT'
165 000642  012701  000060                  MOV     #60,R1          ; CARRIAGE CONTROL CHARACTER
166 000646  004767  000000G                 CALL    $PRNTO          ; SEND MESSAGE
167 000652  004767  000064                  CALL    $MGPRE          ; SEND DISK ADDRESSES
168 000656  105367  000000G                 DECB    $SWERL          ; DECREMENT THE ERROR LIMIT
169 000662  001014                          BNE     7$              ; IF NE, ERROR LIMIT NOT EXCEEDED
170 000664  012700  000000G                 MOV     #$MSG45,R0      ; 'ERROR LIMIT EXCEEDED'
171 000670  012701  000060                  MOV     #60,R1          ; CARRIAGE CONTROL CHARACTER
```

```
172 000674  004767  000000G                 CALL    $PRNTO          ; SEND MESSAGE
173 000700  012700  000054'                 MOV     #MSG53R,R0      ; 'FORMAT TERMINATING'
174 000704  004767  000000G                 CALL    $PRINT          ; PRINT THE MESSAGE
175 000710  000167  000000G                 JMP     $FMTEX          ; EXIT THE FORMATTER
176
177                                 ; IF THE ERROR IS A 'SKI', DO A RECAL
178
179 000714  012700  000032'         7$:     MOV     #MSG51R,R0      ; 'FORMAT CONTINUING'
180 000720  004767  000000G                 CALL    $PRINT          ; SEND MESSAGE
181 000724  032767  040000  000042G         BIT     #40000,$RGBUF+42        ; SEE IF ERROR WAS A SEEK INCOMPLETE
182 000732  001714                          BEQ     2$              ; IF EQ, NO
183 000734  004767  000000G                 CALL    $HOMSK          ; ENTRY POINT FOR RETRY WITH RECAL
184 000740  000711                          BR      2$              ; TRY AGAIN
185
186                                 ;+
187                                 ; **-$MGPRE-PREPARE AND SEND DISK ADDRESSES TO OPERATOR
188                                 ;
189                                 ; INPUTS:
190                                 ;       $FLCYL,$FLTRK,$FLSEC LOADED
191                                 ;
192                                 ; OUTPUTS:
193                                 ;       ERROR MESSAGE
194                                 ;-
195
196 000742  012700  000000G         $MGPRE::MOV     #$CR,R0         ;ADDRESS OF MESSAGE
197 000746  004767  000000G                 CALL    $PRINT          ;CR-LF
198 000752  016701  177274                  MOV     $ERCS1,R1       ;GET RMCS1 AND
199 000756  012703  000077'                 MOV     #MSG61R,R3      ;ADDRESS OF MESSAGE
200 000762  004767  000000G                 CALL    $EROCT          ;"RMCS1 = XXXXX"
201 000766  016701  177262                  MOV     $ERCS2,R1       ;GET RMCS2 AND
202 000772  012703  000106'                 MOV     #MSG62R,R3      ;ADDRESS OF MESSAGE
203 000776  004767  000000G                 CALL    $EROCT          ;"RMCS2 = XXXXX"
204 001002  016701  177250                  MOV     $ERER1,R1       ;GET RMER1 AND
205 001006  012703  000115'                 MOV     #MSG63R,R3      ;ADDRESS OF MESSAGE
206 001012  004767  000000G                 CALL    $EROCT          ;"RMER1 = XXXXX"
207 001016  016701  177236                  MOV     $ERER2,R1       ;GET RMER2 AND
208 001022  012703  000124'                 MOV     #MSG64R,R3      ;ADDRESS OF MESSAGE
209 001026  004767  000000G                 CALL    $EROCT          ;"RMER2 = XXXXX"
210 001032  016701  177224                  MOV     $ERDA,R1        ;GET RMDA AND
211 001036  012703  000133'                 MOV     #MSG65R,R3      ;ADDRESS OF MESSAGE
212 001042  004767  000000G                 CALL    $EROCT          ;"RMDA  = XXXXX"
213 001046  012700  000000G                 MOV     #$CR,R0         ;ADDRESS OF MESSAGE
214 001052  004767  000000G                 CALL    $PRINT          ;CR-LF
215
216                                 ;PRINT ERROR POSITION, CYLINDER, TRACK AND SECTOR IN DECIMAL
217
218 001056  016701  000000G                 MOV     $FLCYL,R1       ;GET FAILING CYLINDER AND
219 001062  012703  000000G                 MOV     #$MSG15,R3      ;ADDRESS OF MESSAGE
220 001066  004767  000000G                 CALL    $EROUT          ;"CYLINDER = XXX."
221 001072  016701  000000G                 MOV     $FLTRK,R1       ;GET FAILING TRACK AND
222 001076  012703  000000G                 MOV     #$MSG16,R3      ;ADDRESS OF MESSAGE
223 001102  004767  000000G                 CALL    $EROUT          ;"TRACK = XX."
224 001106  016701  000000G                 MOV     $FLSEC,R1       ;GET FAILING SECTOR AND
225 001112  012703  000000G                 MOV     #$MSG17,R3      ;ADDRESS OF MESSAGE
226 001116  004767  000000G                 CALL    $EROUT          ;"SECTOR = XX."
227 001122  012700  000000G                 MOV     #$CR,R0         ;GET ADDRESS OF MESSAGE
228 001126  004767  000000G                 CALL    $PRINT          ;CR-LF
```

```
 229 001132  000207                         RETURN
 230
 231                                 ;+
 232                                 ; **-$FLRGS-ROUTINE TO SAVE FAILING DISK ADDRESSES AND REGISTERS
 233                                 ;
 234                                 ; INPUTS:
 235                                 ;       $CYL, $TRK SET TO CURRENT CYLINDER AND TRACK
 236                                 ;       $RGBUF CONTAINS IMAGE OF LATEST H/W REGS (SET BY DRIVER)
 237                                 ;
 238                                 ; OUTPUTS:
 239                                 ;       $ERCS1 - CONTROL STATUS REG 1
 240                                 ;       $ERCS2 - CONTROL STATUS REG 2
 241                                 ;       $ERER1 - ERROR REG 1
 242                                 ;       $ERER2 - ERROR REG 2
 243                                 ;       $ERDA  - TRK/SECTOR REG
 244                                 ;       $FLCYL - FAILING CYLINDER
 245                                 ;       $FLTRK - FAILING TRACK
 246                                 ;       $FLSEC - FAILING SECTOR
 247                                 ;-
 248
 249 001134  016705  000006G         $FLRGS::MOV     $RGBUF+6,R5     ; RECOVER FAILING SECTOR ADDRESS
 250 001140  042705  177740                  BIC     #177740,R5      ; ISOLATE SECTOR NO.
 251 001144  001004                          BNE     1$              ;
 252 001146  012767  000037  000000G         MOV     #31.,$FLSEC     ; MUST HAVE BEEN LAST SECTOR
 253 001154  000403                          BR      2$              ;
 254 001156  005305                  1$:     DEC     R5              ; BACKUP ONE
 255 001160  010567  000000G                 MOV     R5,$FLSEC       ; SET SECTOR NUMBER
 256 001164  016767  000000G 000000G 2$:     MOV     $CYL,$FLCYL     ;  "  CYLINDER
 257 001172  016767  000000G 000000G         MOV     $TRK,$FLTRK     ;  "  TRACK
 258
 259                                 ; SAVE THE RM80 REGISTERS
 260
 261 001200  012705  000000G                 MOV     #$RGBUF,R5      ; REGISTER BUFFER
 262 001204  011567  177042                  MOV     (R5),$ERCS1     ; RMCS1
 263 001210  016567  000010  177036          MOV     10(R5),$ERCS2   ; RMCS2
 264 001216  016567  000014  177032          MOV     14(R5),$ERER1   ; RMER1
 265 001224  016567  000042  177026          MOV     42(R5),$ERER2   ; RMER2
 266 001232  016567  000006  177022          MOV     6(R5),$ERDA     ; RMDA
 267 001240  005067  000000G                 CLR     $RGBUF          ; CLEAR ERROR INDICATOR
 268 001244  000207                          RETURN                  ;
 269
 270         000001                  .END
```

SYMBOL TABLE

```
FMTDSK  000520R         MSG68R  000222R         $EROCT= ****** GX       $MSG17= ****** GX       $R80SF  000266RG
GENHDR= ****** GX       TRKSZ = ****** GX       $EROUT= ****** GX       $MSG45= ****** GX       $R80TF  000330RG
MSG50R  000000R         $BUF  = ****** GX       $FLBUF  000264R         $MSG6 = ****** GX       $SEC  = ****** GX
MSG51R  000032R         $CR   = ****** GX       $FLCYL= ****** GX       $MSG6A= ****** GX       $SWERL= ****** GX
MSG53R  000054R         $CVTDA= ****** GX       $FLRGS  001134RG        $MSG6B= ****** GX       $SWFFE= ****** GX
MSG61R  000077R         $CYL  = ****** GX       $FLSEC= ****** GX       $MVPAT= ****** GX       $TRK  = ****** GX
MSG62R  000106R         $DSKWH= ****** GX       $FLTRK= ****** GX       $MXSEC= ****** GX       $TSTMX= ****** GX
MSG63R  000115R         $ERCS1  000252R         $FMTEX= ****** GX       $PHYDV= ****** GX       $UPDMX= ****** GX
MSG64R  000124R         $ERCS2  000254R         $HOMSK= ****** GX       $PRINT= ****** GX       $WLKER= ****** GX
MSG65R  000133R         $ERDA   000262R         $MGPRE  000742RG        $PRNTO= ****** GX       $WTBSF= ****** GX
MSG66R  000141R         $ERER1  000256R         $MSG15= ****** GX       $RGBUF= ****** GX       $WTSSF= ****** GX
MSG67R  000173R         $ERER2  000260R         $MSG16= ****** GX       $RM80F  000336RG        $XFRSZ= ****** GX

. ABS.  000000      000
        001246      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  517 WORDS  ( 3 PAGES)
DYNAMIC MEMORY:  7942 WORDS  ( 30 PAGES)
ELAPSED TIME:  00:00:11
[300,20]FMTR8F,[300,30]FMTR8F=[300,10]FMTR8F
```

```
                            .TITLE  FMTR8V - VERIFY RM80 DISKS
                            .IDENT  /02.00/
                            ;
                            ;
                            ; COPYRIGHT (C) 1980,1982
                            ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                            ;
                            ;
                            ; THIS  SOFTWARE  IS  FURNISHED  UNDER A LICENSE FOR USE ONLY ON A
                            ; SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION
                            ; OF  THE  ABOVE COPYRIGHT  NOTICE.  THIS SOFTWARE, OR  ANY  OTHER
                            ; COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE  MADE AVAILABLE
                            ; TO  ANY  OTHER PERSON  EXCEPT  FOR USE ON SUCH SYSTEM AND TO ONE
                            ; WHO AGREES TO THESE LICENSE TERMS.  TITLE  TO  AND  OWNERSHIP OF
                            ; THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DIGITAL.
                            ;
                            ; THE  INFORMATION  IN THIS DOCUMENT IS SUBJECT  TO CHANGE WITHOUT
                            ; NOTICE AND SHOULD NOT BE CONSTRUED AS A  COMMITMENT  BY  DIGITAL
                            ; EQUIPMENT CORPORATION.
                            ;
                            ; DIGITAL ASSUMES NO  RESPONSIBILITY FOR THE USE OR RELIABILITY OF
                            ; ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
                            ;
                            ;
                            ;  VERSION 02.00
                            ;
                            ; C. HESS        22-AUG-80
                            ; M. LEAVITT     07-JUL-82
                            ;
                            .NLIST  BEX
000000  126  105  122  MSG2V:  .ASCIZ  /VERIFY TERMINATING/
000023  116  117  116  MSG3V:  .ASCIZ  /NON DATA ERROR DURING VERIFY/
000060  126  105  122  MSG4V:  .ASCIZ  /VERIFY CONTINUING/
000102  110  105  101  MSG5V:  .ASCIZ  @HEADER/DATA ERROR DURING VERIFY@
                            .EVEN
                            ;
                            ; WORKING LOCATIONS
                            ;
000142  000000             TRKER:  .WORD   0               ; TRACK ERROR COUNT
000144                      SECTBL: .BLKB   32.             ; SECTOR ERROR TABLE
000204  000000             LASTER: .WORD   0               ; LAST SECTOR IN ERROR DURING CURRENT VERIFICATION
                                                            ; PASS
000206  000000             XFRSEC: .WORD   0               ; NUMBER OF SECTORS ACTUALLY VERIFIED
000210  000000             ENDSEC: .WORD   0               ; ENDING SECTOR ADDRESS
000212  000000             BEGSEC: .WORD   0               ; STARTING SECTOR ADDRESS
000214  000000             VFYCNT: .WORD   0               ; VERIFICATION PASSES
```

```
                        ;+
                        ; **-$RM80V-VERIFICATION ROUTINE FOR RM80 DISK PACKS
                        ;
                        ; TRACK COMPLETE <= 1
                        ; REPEAT PROCESS UNTIL PHYSICAL DISK ADDRESS .EQ. (561,0,0) ($TSTMX)
                        ;    IF TRACK COMPLETE .NE. 0
                        ;        THEN START SECTOR <= 0
                        ;             END SECTOR <= $MXSEC
                        ;    TRACK ERROR COUNT ('TRKER') <= 0
                        ;    CLEAR SECTOR ERROR TABLE
                        ;    GENERATE TRACK BUFFER
                        ;    WORD COUNT <= ((END SECTOR)-(START SECTOR)+1)*258
                        ;    VERIFICATION PASS COUNT ('VFYCNT') <= $VERIF
                        ;    REPEAT WRITE CHECK UNTIL ('VFYCNT' .EQ. 0 .OR. SECTORS TRANSFERED .EQ. 0)
                        ;        IF ERROR
                        ;           THEN CALL $CKERR
                        ;                IF ERROR IS A NON DATA ERROR
                        ;                   THEN RETRY ERROR
                        ;        'VFYCNT' = 'VFYCNT' - 1
                        ;    ENDREPEAT
                        ;    IF TRKER .NE. 0
                        ;        THEN CALL 'VFYSEC'
                        ;             REGENERATE THE TRACK BUFFER (GENHDR)
                        ;             RE-FORMAT TRACK FROM SECTOR 0 TO SECTOR 31
                        ;    IF ENDSEC .NE. 31
                        ;        THEN START SECTOR <= LAST ERROR SECTOR + 1
                        ;             END SECTOR <= 31
                        ;             ELSE PHYSICAL ADDRESS = PHYSICAL ADDRESS + 32 SECTORS
                        ; ENDREPEAT
                        ; RE-WRITE BSF & SSF
                        ; EOJ
                        ;
                        ;-
                        
000216  012700  000000G          $RM80V::MOV     #$MSG11,R0      ; GET VERIFICATION STARTING MESSAGE
000222  004767  000000G                  CALL    $PRINT          ; PRINT THE MESSAGE
000226  012703  000000G                  MOV     #$PHYDV,R3      ; PHYSICAL DEVICE GEOMETRY POINTER
000232  004767  000000G                  CALL    $UPDMX          ; UPDATE THE LIMITS
000236  004767  000000G                  CALL    $HOMSK          ; ISSUE A RECAL

000242  105767  000000G                  TSTB    $SWFFE          ; VERIFY FE CYLINDERS ONLY ?
000246  001005                          BNE     1$              ; BR IF YES
000250  005067  000000G                  CLR     $CYL            ; STARTING CYLINDER=0
000254  005067  000000G                  CLR     $TRK            ; STARTING TRACK=0
000260  000406                          BR      2$
000262  012767  001056  000000G 1$:      MOV     #558.,$CYL      ; STARTING CYLINDER=558.
000270  012767  000015  000000G          MOV     #13.,$TRK       ; STARTING TRACK=13.

000276  005067  177710          2$:     CLR     BEGSEC          ; START WITH SECTOR 0
000302  005067  177634                  CLR     TRKER           ; CLEAR THE TRACK ERROR COUNT
000306  005067  177672                  CLR     LASTER          ; CLEAR LAST ERRORRED SECTOR
000312  016767  000000G 177670          MOV     $MXSEC,ENDSEC   ; LOAD THE ENDING SECTOR ADDRESS
000320  005367  177664                  DEC     ENDSEC          ; CORRECT THE VALUE
000324  004767  000000G                  CALL    $MVPAT          ; LOAD THE DATA PATTERN

000330  026727  000000G 001056  3$:     CMP     $CYL,#558.      ; IS THIS THE BSF CYLINDER ADDRESS ?
000336  001016                          BNE     4$              ; BR IF NO
```

```
110 000340  026727  000000G 000015          CMP     $TRK,#13.       ; IS THIS THE BSF TRACK ADDRESS ?
111 000346  001012                         BNE     4$              ; BR IF NO
112 000350  016746  000000G                MOV     $CYL,-(SP)      ; SAVE CURRENT CYLINDER ADDRESS
113 000354  016746  000000G                MOV     $TRK,-(SP)      ; SAVE CURRENT TRACK ADDRESS
114 000360  004767  000000G                CALL    $R80SF          ; RE-FORMAT BSF/SSF HEADERS AND DATA
115 000364  012667  000000G                MOV     (SP)+,$TRK      ; RESTORE TRACK ADDRESS
116 000370  012667  000000G                MOV     (SP)+,$CYL      ; RESTORE CYLINDER ADDRESS
117
118 000374  012701  000020          4$:    MOV     #16.,R1         ; LOOP CONSTANT
119 000400  012700  000144'                MOV     #SECTBL,R0      ; TABLE ADDRESS
120 000404  005020                  5$:    CLR     (R0)+           ; CLEAR TWO LOCATIONS
121 000406  005301                         DEC     R1              ; DECREMENT THE LOOP COUNTER
122 000410  001375                         BNE     5$              ; IF NE, NOT FINISHED
123
124 000412  116767  000000G 177574          MOVB    $VERIF,VFYCNT   ; LOAD THE VERIFICATION COUNT
125 000420  016767  177566  000000G 6$:    MOV     BEGSEC,$SEC     ; LOAD THE SECTOR ADDRESS
126 000426  012700  000000G                MOV     #$BUF,R0        ; BUFFER ADDRESS
127 000432  004767  000000G          7$:   CALL    GENHDR          ; GENERATE A HEADER FOR WRITE CHECK HEADER & DATA
128 000436  005267  000000G                INC     $SEC            ; INCREMENT THE SECTOR
129 000442  026767  000000G 177540          CMP     $SEC,ENDSEC     ; SEE IF FINISHED
130 000450  101770                         BLOS    7$              ; IF LOS, NO
131
132 000452  016767  177534  000000G 8$:    MOV     BEGSEC,$SEC     ; RESET THE BEGINNING SECTOR ADDRESS
133 000460  016700  177524                 MOV     ENDSEC,R0       ; ENDING SECTOR ADDRESS
134 000464  166700  177522                 SUB     BEGSEC,R0       ; SUBTRACT STARTING ADDRESS
135 000470  005200                         INC     R0              ; CALCULATE # OF SECTORS TO XFER
136 000472  012701  001004                 MOV     #258.*2.,R1     ; GET ONE SECTOR SIZE (IN BYTES)
137 000476  004767  000000G                CALL    $MULT           ; CALCULATE THE BYTE COUNT
138 000502  010167  000000G                MOV     R1,$XFRSZ       ; LOAD THE BYTE COUNT
139 000506  004767  000000G                CALL    $CVTDA          ; CONVERT DISK ADDRESS TO AN LBN
140 000512  105067  000000G          9$:   CLRB    $HDRFG          ; CLEAR HEADER ERROR FLAG
141 000516  012705  000000G                MOV     #$BUF,R5        ; BUFFER ADDRESS
142 000522  004767  000000G                CALL    $DSKWC          ; DO WRITE CHECK HEADER & DATA
143 000526  005767  000000G                TST     $RGBUF          ; SEE IF ANY ERRORS ?
144 000532  100404                         BMI     10$             ; IF MI, YES
145 000534  005367  177454                 DEC     VFYCNT          ; DONE VERIFY PASS ?
146 000540  001364                         BNE     9$              ; IF NE, NO FINISHED
147 000542  000420                         BR      12$             ; SEE IF FINSIHED WITH THIS TRACK
148
149                                 ; PROCESS THE ERROR
150
151 000544  004767  000326          10$:   CALL    CKERR           ; SEE WHICH TYPE OF ERROR IT IS
152 000550  103740                         BCS     8$              ; IF CS, NON-DATA ERROR - RETRY
153 000552  005767  177430                 TST     XFRSEC          ; SEE IF ANY SECTORS WERE TRANSFERED
154 000556  001404                         BEQ     11$             ; IF EQ, NO
155 000560  005367  177430                 DEC     VFYCNT          ; DECREMENT THE VERIFICATION COUNT FOR THIS PASS
156 000564  001315                         BNE     6$              ; IF NE, MORE TO DO
157 000566  000406                         BR      12$             ; SEE IF FINISHED WITH THE CURRENT TRACK
158 000570  016746  177416          11$:   MOV     BEGSEC,-(SP)    ; SEE IF THE ERROR IS THE LAST SECTOR
159 000574  005216                         INC     (SP)            ; CORRECT VALUE FOR THE COMPARE
160 000576  026726  000000G                CMP     $MXSEC,(SP)+    ; ERROR ON SINGLE SECTOR TRANSFER BEGINNING
161                                                                 ;  WITH SECTOR 31 ?
162 000602  001306                         BNE     6$              ; IF NE, NO
163 000604  005767  177332          12$:   TST     TRKER           ; ANY NEW ERRORS ON THIS TRACK ?
164 000610  001404                         BEQ     13$             ; IF EQ, NO
165 000612  004767  000106                 CALL    VFYSEC          ; VERIFY THE SECTORS IN ERROR
166 000616  004767  000000G                CALL    $R80TF          ; REFORMAT THE TRACK AND MARK NEW DEFECTS
```

```
 167 000622  016700  177362          13$:    MOV     ENDSEC,R0       ; CHECK THE ENDING SECTOR
 168 000626  005200                          INC     R0              ; CORRECT IT
 169 000630  020067  000000G                 CMP     R0,$MXSEC       ; SEE IF FINSIHED
 170 000634  001421                          BEQ     14$             ; IF EQ, YES
 171 000636  016700  177342                  MOV     LASTER,R0       ; GET THE LAST ERROR SECTOR ADDRESS
 172 000642  005200                          INC     R0              ; CORPECT VALUE
 173 000644  020067  000000G                 CMP     R0,$MXSEC       ; SEE IF AT MAX SECTOR
 174 000650  001413                          BEQ     14$             ; IF EQ, YES
 175 000652  016767  177326  177332         MOV     LASTER,BEGSEC   ; RELOAD THE STARTING SECTOR
 176 000660  005267  177326                  INC     BEGSEC          ; START WITH THE FIRST GOOD SECTOR
 177 000664  016767  000000G 177316         MOV     $MXSEC,ENDSEC   ; RELOAD THE ENDING SECTOR
 178 000672  005367  177312                  DEC     ENDSEC          ; CORRECT THE VALUE
 179 000676  000536                          BR      4$              ; CONTINUE
 180
 181 000700  004767  000000G          14$:    CALL    $TSTMX          ; INCREMENT THE PHYSICAL ADDRESS
 182 000704  103402                          BCS     15$             ; IF CS, FINISHED
 183 000706  000167  177364                  JMP     2$              ; TEST NEXT TRACK
 184
 185 000712  004767  000000G          15$:    CALL    $WTSSF          ; RE-WRITE THE SSF
 186 000716  004767  000000G                  CALL    $WTBSF          ; RE-WRITE ALL THE DEC STD 144 TRACKS
 187 000722  000207                          RETURN
 188
 189                                 ;+
 190                                 ; **-VFYSEC-VERIFY THE SECTOR(S) IN ERROR
 191                                 ;
 192                                 ; THIS ROUTINE WILL RE-READ THE SECTORS WHICH GAVE ERRORS DURING THE VERIFY
 193                                 ;  PASS ON THE CURRENT TRACK.  ANY SECTOR FAILING DURING THE RE-READ PASS
 194                                 ;  WILL BE CONSIDERED DEFECTIVE.
 195                                 ;
 196                                 ; REPEAT PROCESS UNTIL TRACK ERROR COUNT ('TRKER') .EQ. 0
 197                                 ;     FIND NEXT SECTOR IN SECTOR ERROR TABLE ('SECTBL')
 198                                 ;     REPEAT WRITE CHECK FOR SECTOR ENTRY IN 'SECTBL'
 199                                 ;     UNTIL ('SECTBL'(SECTOR) .LT. 0) .OR. (HEADER FIELD OR DATA FIELD ERROR FOR SECTOR)
 200                                 ;         IF 'SECTBL'(SECTOR) .GE. 0
 201                                 ;             THEN IF NO SKIP SECTOR ON TRACK .AND. '$HDRFG' = 0
 202                                 ;                       THEN PUT SECTOR IN SSF
 203                                 ;                       ELSE IF SECTOR IS NOT IN THE FE CYLINDERS
 204                                 ;                                THEN PUT SECTOR IN FEBSF
 205                                 ;                                ELSE PUT SECTOR IN UDBSF
 206                                 ;           LAST ERROR SECTOR ON TRACK 'LASTER' <= CURRENT ERROR SECTOR
 207                                 ;           TRKER = TRKER - 1
 208                                 ; ENDREPEAT
 209                                 ;
 210                                 ; INPUTS:
 211                                 ;       SECTBL - CONTAINS ADDRESSES OF SECTORS IN ERROR
 212                                 ;       $CYL,$TRK,$SEC CONTAINS ADDRESS OF CURRENT TRACK
 213                                 ;       TRKER .NE. 0
 214                                 ;
 215                                 ; OUTPUTS:
 216                                 ;       SECTBL CLEARED
 217                                 ;       ENTRIES MADE IN THE MDBSF,UDBSF,& SSF'S
 218                                 ;       'LASTER' CONTAINS THE ADDRESS OF THE LAST SECTOR IN ERROR ON
 219                                 ;         THE TRACK
 220                                 ;
 221                                 ;-
 222
 223 000724  012700  177777          VFYSEC: MOV     #-1,R0          ; USE R0 AS THE INDEX INTO 'SECTBL'
```

```
000730  005200                  1$:     INC     R0              ;INCREMENT THE INDEX
000732  105760  000144'                  TSTB    SECTBL(R0)      ;ENTRY ?
000736  001774                          BEQ     1$              ;IF EQ, NO
000740  010046                  2$:     MOV     R0,-(SP)        ;SAVE THE INDEX
000742  010067  000000G                  MOV     R0,$SEC         ;RELOAD THE SECTOR ADDRESS
000746  016767  000000G 000000G          MOV     $CYL,$BUF       ;SETUP A SECTOR IMAGE
000754  052767  150000  000000G          BIS     #150000,$BUF    ;SET THE MF, UF, AND FMT16 BITS
000762  116767  000000G 000003G          MOVB    $TRK,$BUF+3     ;LOAD THE TRACK ADDRESS
000770  116767  000000G 000002G          MOVB    $SEC,$BUF+2     ;LOAD THE SECTOR ADDRESS
000776  012705  000000G          3$:     MOV     #$BUF,R5        ;BUFFER ADDRESS
001002  012767  001004  000000G          MOV     #516.,$XFRSZ    ;SECTOR - INCLUDING HEADER BYTE COUNT
001010  004767  000000G                  CALL    $CVTDA          ;CONVERT THE PHYSICAL ADDRESS TO LBN
001014  004767  000000G                  CALL    $DSKWC          ;WRITE CHECK THE SECTOR
001020  005767  000000G                  TST     $RGBUF          ;ANY ERRORS ?
001024  100007                          BPL     4$              ;IF PL, NO
001026  004767  000044                  CALL    CKERR           ;PROCESS THE ERROR
001032  103761                          BCS     3$              ;IF CS, NON-DATA ERROR - RETRY
001034  004767  000000G                  CALL    $DSTRB          ;PUT THE SECTOR INTO THE PROPER BSF
001040  012600                          MOV     (SP)+,R0        ;RESTORE THE SECTOR TABLE INDEX
001042  000406                          BR      5$              ;
001044  012600                  4$:     MOV     (SP)+,R0        ;RESTORE THE SECTOR TABLE POINTER
001046  105360  000144'                  DECB    SECTBL(R0)      ;COUNT THE SUCESSFUL RE-READ
001052  100332                          BPL     2$              ;IF PL CONTINUE
001054  105060  000144'                  CLRB    SECTBL(R0)      ;CLEAR THE SECTOR ENTRY, ERROR WAS NOT
                                                                 ; REPEATED
001060  016767  000000G 177116  5$:     MOV     $SEC,LASTER     ;ASSUME CURRENT ERROR IS LAST ERROR ON THIS TRACK
001066  005367  177050                  DEC     TRKER           ;DECREMENT THE ERROR COUNTER
001072  001316                          BNE     1$              ;IF NE, NOT FINISHED
001074  000207                          RETURN                  ;
```

```
;*
;**-CKERR-VERIFY MODULE DEVICE ERROR ANALYSIS ROUTINE
;
;THIS ROUTINE WILL ANALYZE THE DISK ERROR AND REPORT THE ERROR AS REQUIRED.
;
;IF SKI
;    THEN REPORT & RECALIBRATE
;    ELSE REPORT ERROR
;        IF NOT DATA ERROR
;            THEN RETRY UNTIL SUCESSFUL OR RETRY LIMIT EXCEEDED
;            ELSE DETERMINE SECTOR IN ERROR
;                IF TRACK ERROR COUNT ('TRKER') .EQ 0
;                    THEN LAST ERROR SECTOR ('LASTER') <= ERROR SECTOR
;                IF ERROR IS HEADER ERROR
;                    THEN ERROR TYPE ('$HDRFG') <= 1
;                    ELSE 'HDRFG' <= 0
;                IF VERIFY COUNT ('VFYCNT') .NE. 0
;                    THEN IF SECTOR IN ERROR NOT LISTED IN BSF, SSF, OR FEBSF ($SRALL)
;                            THEN LOAD SECTOR RETRY COUNT IN SECTOR ERROR TABLE ('SECTBL')
;                                'TRKER' = 'TRKER' + 1
;                TRANSFERRED SECTOR COUNT ('XFRSEC') = ((ERROR SECTOR)-(START SECTOR))-1
;                IF 'XFRSEC' .NE. 0
;                    THEN END SECTOR <= (ERROR SECTOR-1)
;                        WORD COUNT <= ((END SECTOR)-(STAR  SECTOR +1)+1)*258
;
;INPUTS:
;       $RGBUF - CONTAINS THE RM80 REGISTER CONTENTS
```

```
                                ;       VFYCNT - CONTROLS UPDATING 'SECTBL'
                                ;
                                ; RETURNS:
                                ;       CS - IF THE ERROR WAS A NON DATA ERROR
                                ;
                                ; OUTPUTS:
                                ;       'SECTBL' - UPDATED IF ERROR IS A DATA ERROR AND 'VFYCNT' .NE. 0
                                ;
                                ; NOTE:
                                ;       THIS ROUTINE WILL RETURN TO '$FMTEX' IF THE DRIVE EXCEEDS THE
                                ;        ERROR LIMIT.
                                ;
                                ;-

001076  004767  000000G         CKERR:  CALL    $FLRGS          ; SAVE THE REGISTERS
001102  016746  000006G                 MOV     $RGBUF+6,-(SP)  ; GET THE SECTOR ADDRESS
001106  005316                          DEC     (SP)            ; BACK UP TO FAILING SECTOR
001110  042716  177740                  BIC     #^C37,(SP)      ; JUST IN CASE FAILING SECTOR WAS 31.
001114  011667  177064                  MOV     (SP),LASTER     ; GET LAST ERRORED SECTOR
001120  016746  000000G                 MOV     $RGBUF,-(SP)    ; GET RMCS1
001124  042716  157777                  BIC     #^C20000,(SP)   ; LEAVE 'MCPE'
001130  016746  000010G                 MOV     $RGBUF+10,-(SP) ; GET RMCS2
001134  042716  040000                  BIC     #40000,(SP)     ; CLEAR 'WCE'
001140  156666  000001  000002          BISB    1(SP),2(SP)     ; 'OR' THE ERROR BITS FROM RMCS2
001146  016716  0000426                 MOV     $RGBUF+42,(SP)  ; GET RMER2
001152  042716  100000                  BIC     #100000,(SP)    ; CLEAR 'BSE'
001156  052616                          BIS     (SP)+,(SP)      ; 'OR' WITH 'MCPE' & RMCS2 ERROR BITS
001160  005726                          TST     (SP)+           ; SEE IF ANY ERROR BITS SET
001162  001016                          BNE     1$              ; IF NE, YES - NON DATA ERROR
001164  032767  110400  000014G         BIT     #110400,$RGBUF+14       ; TEST 'DCK!DTE!HCRC'
001172  001056                          BNE     4$              ; IF NE, MUST BE A DATA OR HEADER ERROR
001174  032767  040000  000010G         BIT     #40000,$RGBUF+10        ; TEST 'WCE'
001202  001052                          BNE     4$              ; IF NE, MUST BE DATA ERROR
001204  016746  000014G                 MOV     $RGBUF+14,-(SP) ; COPY RMER1
001210  042716  110400                  BIC     #110400,(SP)    ; CLEAR THE DATA/HEADER ERROR BITS
001214  005726                          TST     (SP)+           ; SEE IF ANY OTHER ERROR BITS ARE SET
001216  001504                          BEQ     6$              ; IF EQ, NO MUST BE 'BSE' ERROR, ELSE
                                                                ; REPORT NON DATA ERROR

                                ; SEND NON DATA ERROR MESSAGE

001220  012700  000023'          1$:     MOV     #MSG3V,R0       ; 'NON DATA ERROR DURING VERIFY'
001224  012701  000060                  MOV     #60,R1          ; CARRIAGE CONTROL CHARACTER
001230  004767  000000G                 CALL    $PRNTO          ; SEND MESSAGE
001234  004767  000000G                 CALL    $MGPRE          ; TYPE THE DEVICE REGISTERS
001240  105367  000000G                 DECB    $SWERL          ; SEE IF ERROR LIMIT EXCEEDED
001244  001014                          BNE     2$              ; IF NE, NO
001246  012700  000000G                 MOV     #$MSG45,R0      ; 'ERROR LIMIT EXCEEDED'
001252  012701  000060                  MOV     #60,R1          ; CARRIAGE CONTROL CHARACTER
001256  004767  000000G                 CALL    $PRNTO          ; SEND MESSAGE
001262  012700  000000'                 MOV     #MSG2V,R0       ; 'VERIFY TERMINATING'
001266  004767  000000G                 CALL    $PRINT          ; PRINT THE MESSAGE
001272  000167  000000G                 JMP     $FMTEX          ; ABORT THE FORMATTER
001276  012700  000060'          2$:     MOV     #MSG4V,R0       ; 'VERIFY CONTINUING'
001302  004767  000000G                 CALL    $PRINT          ; SEND MESSAGE
001306  032767  040000  0000426         BIT     #40000,$RGBUF+42        ; SEE IF 'SKI'
001314  001402                          BEQ     3$              ; IF EQ, NO
```

```
338 001316  004767  000000G                 CALL    $HOMSK          ; DO A RECAL
339 001322  005726                  3$:    TST     (SP)+           ; CORRECT THE STACK
340 001324  000261                         SEC                     ; SHOW NON-DATA ERROR
341 001326  000207                         RETURN
342
343                                 ; PROCESS THE DATA ERROR AND EXIT
344
345 001330  032767  000400  000014G 4$:    BIT     #400,$RGBUF+14  ; SEE IF ERROR IS A HEADER ERROR ('HCRC')
346 001336  001402                         BEQ     5$              ; IF EQ, DATA ERROR
347 001340  105267  000000G                INCB    $HDRFG          ; SET HEADER ERROR FLAG
348 001344  005767  176644          5$:    TST     VFYCNT          ; SEE IF VERIFING THE ENTIRE TRACK OR IF
349                                                                 ; PROCESSING A SINGLE SECTOR
350 001350  001457                         BEQ     9$              ; IF EQ, VERIFYING SINGLE SECTORS
351 001352  011667  000000G                MOV     (SP),$SEC       ; PUT THE BAD SECTOR ADDRESS IN '$SEC'
352 001356  004767  000000G                CALL    $SRALL          ; SEE IF THE ERROR IS IN ANY OF THE BAD SECTOR FILES
353 001362  103022                         BCC     6$              ; IF CC, DEFECT IS ALREADY ACCOUNTED FOR
354 001364  005267  176552                 INC     TRKER           ; INCREMENT THE TRACK ERROR COUNT
355 001370  011600                         MOV     (SP),R0         ; GET THE BAD SECTOR ADDRESS
356 001372  116760  000000G 000144'        MOVB    $RETRY,SECTBL(R0)       ; LOAD RETRY COUNT AND FLAG A BAD SECTOR
357 001400  012700  000102'                MOV     #MSG5V,R0       ; 'HEADER/DATA ERROR DURING VERIFY
358 001404  012701  000060                 MOV     #60,R1          ; CARRIAGE CONTROL CHARACTER
359 001410  004767  000000G                CALL    $PRNTO          ; SEND MESSAGE
360 001414  004767  000000G                CALL    $MGPRE          ; TYPE THE DEVICE REGISTERS
361 001420  012700  000060'                MOV     #MSG4V,R0       ; 'VERIFY CONTINUING'
362 001424  004767  000000G                CALL    $PRINT          ; PRINT IT
363
364 001430  011667  176552          6$:    MOV     (SP),XFRSEC     ; COPY THE ADDRESS OF THE ERROR SECTOR
365 001434  166767  176552  176544        SUB     BEGSEC,XFRSEC   ; CALCULATE THE NUMBER OF SECTORS TRANSFERED
366 001442  001003                         BNE     7$              ; IF NE, AT LEAST ONE SECTOR TRANSFERED
367 001444  005267  176542                 INC     BEGSEC          ; STEP AROUND THE ERROR SECTOR
368 001450  000404                         BR      8$
369 001452  011667  176532          7$:    MOV     (SP),ENDSEC     ; CHANGE THE ENDING SECTOR ADDRESS
370 001456  005367  176526                 DEC     ENDSEC          ; POINT TO LAST ERROR FREE SECTOR
371 001462  016700  176522          8$:    MOV     ENDSEC,R0       ; COPY THE ENDING SECTOR
372 001466  166700  176520                 SUB     BEGSEC,R0       ; CALCULATE THE NEW BYTE COUNT
373 001472  005200                         INC     R0              ; CORRECT THE DIFFERENCE
374 001474  012701  001004                 MOV     #516.,R1        ; NUMBER BYTES/SECTOR
375 001500  004767  000000G                CALL    $MULT           ; MULTIPLY THE SECTORS * SIZE
376 001504  010167  000000G                MOV     R1,$XFRSZ       ; UPDATE THE BYTE COUNT
377 001510  005726                  9$:    TST     (SP)+           ; CORRECT THE STACK
378 001512  000241                         CLC                     ; SET THE DATA ERROR INDICATOR
379 001514  000207                         RETURN                  ;
380
381         000001                  .END
```

FMTR8V - VERIFY RM80 DISKS MACRO M1113 07-JUL-82 15:01 PAGE 2-6 SEQ 0090

SYMBOL TABLE

```
BEGSEC  000212R         TRKER   000142R         $FMTEX= ****** GX       $PRINT= ****** GX       $SWERL= ****** GX
CKERR   001076R         VFYCNT  000214R         $HDRFG= ****** GX       $PRNTO= ****** GX       $SWFFE= ****** GX
ENDSEC  000210R         VFYSEC  000724R         $HOMSK= ****** GX       $RETRY= ****** GX       $TRK  = ****** GX
GENHDR= ****** GX       XFRSEC  000206R         $MGPRE= ****** GX       $RGBUF= ****** GX       $TSTMX= ****** GX
LASTER  000204R         $BUF  = ****** GX       $MSG11= ****** GX       $RM80V  000216RG        $UPDMX= ****** GX
MSG2V   000000R         $CVTDA= ****** GX       $MSG45= ****** GX       $R80SF= ****** GX       $VERIF= ****** GX
MSG3V   000023R         $CYL  = ****** GX       $MULT = ****** GX       $R80TF= ****** GX       $WTBSF= ****** GX
MSG4V   000060R         $DSKWC= ****** GX       $MVPAT= ****** GX       $SEC  = ****** GX       $WTSSF= ****** GX
MSG5V   000102R         $DSTRB= ****** GX       $MXSEC= ****** GX       $SRALL= ****** GX       $XFRSZ= ****** GX
SECTBL  000144R         $FLRGS= ****** GX       $PHYDV= ****** GX

. ABS.  000000      000
        001516      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  567 WORDS  ( 3 PAGES)
DYNAMIC MEMORY:  7942 WORDS  ( 30 PAGES)
ELAPSED TIME:  00:00:15
[300,20]FMTR8V,[300,30]FMTR8V=[300,10]FMTR8V
```

```
                                .TITLE  FMTR8M - DEFECT ENTRY MODULE
                                .IDENT  /02.00/
                                ;
                                ;
                                ; COPYRIGHT (C) 1980,1982
                                ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                                ;
                                ;
                                ; THIS  SOFTWARE  IS  FURNISHED  UNDER A LICENSE FOR USE ONLY ON A
                                ; SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION
                                ; OF   THE   ABOVE COPYRIGHT  NOTICE.  THIS SOFTWARE, OR  ANY  OTHER
                                ; COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE  MADE AVAILABLE
                                ; TO  ANY  OTHER PERSON  EXCEPT  FOR USE ON SUCH SYSTEM AND TO ONE
                                ; WHO AGREES TO THESE LICENSE TERMS.  TITLE  TO  AND  OWNERSHIP OF
                                ; THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DIGITAL.
                                ;
                                ; THE  INFORMATION  IN THIS DOCUMENT IS SUBJECT  TO CHANGE WITHOUT
                                ; NOTICE AND SHOULD NOT BE CONSTRUED AS A  COMMITMENT  BY  DIGITAL
                                ; EQUIPMENT CORPORATION.
                                ;
                                ; DIGITAL ASSUMES NO  RESPONSIBILITY FOR THE USE OR RELIABILITY OF
                                ; ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
                                ;
                                ;
                                ;  VERSION 01.00
                                ;
                                ; C. HESS        18-AUG-80
                                ; M. LEAVITT     02-JUL-82
                                ;
                                ;
                                ;
                                ; MESSAGES
                                ;
                                ;
                                .NLIST  BEX
                                
000000      125     104     102 MSG1R8: .ASCIZ  /UDBSF /
000007      115     123     102 MSG2R8: .ASCIZ  /MSBSF/
000015      123     123     106 MSG3R8: .ASCIZ  /SSF/
000021      106     105     040 MSG4R8: .ASCIZ  /FE BAD SECTORS/
000040      104     125     120 MSG7R8: .ASCIZ  /DUPLICATE ENTRY - /
000063      106     111     114 MSG8R8: .ASCIZ  /FILE FULL - /
000100      105     116     124 MSGCR8: .ASCIZ  /ENTER BAD SECTR ADRS:/
000126      103     131     114 MSGDR8: .ASCIZ  /CYL,TRK,SEC = /
000145      110     104     101 MSGER8::.ASCIZ  @HDA S/N DOESN'T MATCH CURRENT S/N@
                                .EVEN
                                
                                ; LBN CONSTANTS
                                
000210  000003  151040          FELBN:  .WORD   3,151040        ; BEGINNING OF THE DEC144 / FE CYLINDERS
                                                                ; (558.,13,0)
000214  000003  151037          USRLBN: .WORD   3,151037        ; END OF USER SPACE
                                                                ; (558.,12.,31.)
                                
                                ; MINIMUM/MAXIMUM STORAGE FOR INPUT ROUTINE
                                
000220  000000                  MINCYL: .WORD   0               ;STORE MIN. CYLINDER ADDRESS HERE
```

```
 58 000222  000000                MINTRK: .WORD   0               ;       ''   TRACK       ''
 59 000224  000000                MAXCYL: .WORD   0               ;STORE MAX. CYLINDER ADDRESS HERE
 60 000226  000000                MAXTRK: .WORD   0               ;       ''   TRACK       ''
 61
```

```
                        ;+
                        ; **-$RM80M-MANUAL INPUT OF UDBSF OR SSF DATA
                        ;
                        ; THIS ROUTINE WILL ACCEPT MANUALLY INPUTTED DATA FOR EITHER THE
                        ; UDBSF OR SSF FILES AND RE-WRITE THESE FILES TO THE RM80 DISK.
                        ;
                        ; INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       RE-WRITTEN UDBSF OR SSF
                        ;
                        ;-
                        ;
000230  004767  000000G $RM80M::CALL    $RDSSF          ; READ SKIP SECTOR FILE
000234  004767  000000G         CALL    $RDBSF          ; READ THE MDBSF & UDBSF
000240  012701  000060          MOV     #60,R1          ; SET 2 LINE FEED CONTROL
000244  012700  000100'         MOV     #MSGCR8,R0      ; GET ADDRESS OF MESSAGE
000250  004767  000000G         CALL    $PRNTO          ; PRINT 'ENTER BAD SECTR ADRS:'
                        ;
000254  004767  000554  1$:     CALL    INPUT           ; GET SSF ENTRIES
000260  001403                  BEQ     2$              ; IF EQ, FINISHED
000262  004767  000024          CALL    $DSTRB          ; DISTRIBUTE ENTERED ADDRESS INTO CORRECT FILE
000266  000772                  BR      1$              ; CONTINUE
                        ;
000270  004767  000000G 2$:     CALL    $R80SF          ; FORMAT THE BSF/SSF TRACKS
000274  004767  000000G         CALL    $WTSSF          ; RE-WRITE THE SSF
000300  004767  000000G         CALL    $WTBSF          ; RE-WRITE THE BSF
000304  105067  000000G         CLRB    $SWMOD          ; CLEAR THE MODIFY FLAG
000310  000207                  RETURN
                        ;
                        ;+
                        ; **-$DSTRB-ROUTINE WILL DISTRIBUTE THE DEFECT ADDRESSES
                        ;
                        ; THIS ROUTINE DISTRIBUTES THE OPERATOR ENTERED ADDRESS TO THE CORRECT
                        ;  DEFECT FILE AND CHECKS FOR DUPLICATION IN THE SSF.
                        ;
                        ; INPUTS:
                        ;       $CYL = OPERATOR ENTERED CYLINDER ADDRESS
                        ;       $TRK = OPERATOR ENTERED TRACK ADDRESS
                        ;       $SEC = OPERATOR ENTERED SECTOR ADDRESS
                        ;       $HDRFG = 1  IF DEFECT IS A HEADER DEFECT
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;-
                        ;
000312  105067  000000G $DSTRB::CLRB    SKPFRE          ; CLEAR THE SKIP SECTOR FLAG
000316  004767  000000G         CALL    $SRSDF          ; SEE IF ENTRY IN THE SSF
000322  103405                  BCS     1$              ; IF CS, NO
000324  012704  000015'         MOV     #MSG3R8,R4      ; DUPLICATE ENTRY MESSAGE
000330  004767  000410          CALL    DUP             ; TYPE THE MESSAGE
000334  000556                  BR      13$             ; TRY AGAIN
000336  105767  000000G 1$:     TSTB    $HDRFG          ; SEE IF DEFECT IS A HEADER DEFECT
000342  001041                  BNE     4$              ; IF NE, IT IS
000344  012701  000000C         MOV     #$SSF+<2+$SSFHD>-4,R1   ; SSF ADDRESS
000350  012702  000000C         MOV     #$SSF+<1000*$SSFLN>-2,R2        ; SSF LENGTH
```

```
                                        CALL    $SRTRK          ; SEARCH THE SSF
                                        BCC     4$              ; IF CC, TRACK IS IN THE SSF
                                        CMP     R1,#$SSF+<1000*$SSFLN>-2        ; IS SSF FULL?
                                        BNE     2$              ; IF NE NO
                                        MOV     #MSG3R8,R4      ; GET SSF MESSAGE
                                        CALL    FULL            ; ISSUE FILE FULL MESSAGE
                                        BR      13$             ; GET NEXT ENTRY

                                ; LOAD SSF WITH NEW BAD SECTOR ADDRESS, REVISION DATE, REVISION NUMBER
                                ; AND UPDATED ENTRY NUMBER

                        2$:     MOV     $CYL,(R1)+      ; PUT CYLINDER ADDRESS INTO SSF
                                MOVB    $SEC,(R1)+      ; PUT SECTOR ADDRESS INTO SSF
                                MOVB    $TRK,(R1)+      ; PUT TRACK ADDRESS INTO SSF
                                MOV     DPAR$,$SSF+6    ; LOAD SSF REVISION DATE IN TABLE
                                CMP     $REVNM,$SSF+10  ; HAS REVISION NUMBER BEEN UPDATED YET ?
                                BNE     3$              ; BR IF YES
                                INC     $SSF+10         ; UPDATE REVISION NUMBER
                        3$:     INC     $SSF+22         ; INCREMENT NUMBER OF ENTRIES
                                BR      13$             ; CONTINUE

                                ; SEE IF THE DEFECT CAN GO INTO THE UDBSF OR FEBSF

                        4$:     CALL    $SRSSF          ; SEE IF ENTERED SECTOR IS WITHIN A SKIP REGION
                                BCS     5$              ; IF CS, NO
                                DECB    $SEC            ; DECREMENT THE SECTOR ADDRESS
                                INCB    SKPFRE          ; SHOW SECTOR WITHIN A SKIP REGION
                        5$:     MOV     #MSG2R8,R4      ; ASSUME DUPLICATE MDBSF ENTRY
                                CALL    $SRMDF          ; SEARCH THE MDBSF
                                BCC     6$              ; IF CC, ENTRY DUPLICATED
                                MOV     #MSG4R8,R4      ; ASSUME DUPLICATE ENTRY
                                CALL    $SRFES          ; SEE IF ENTRY DUPLICATED IN THE FE AREA
                                BCC     6$              ; IF CC, ENTRY DUPLICATED
                                MOV     #MSG1R8,R4      ; ASSUME DUPLICATE UDBSF
                                CALL    $SRUDF          ; SEARCH THE UDBSF
                                BCS     7$              ; IF CS, ENTRY NOT DUPLICATED
                        6$:     TSTB    SKPFRE          ; SEE IF IN A SKIP REGION
                                BNE     12$             ; IF NE, YES
                                CALL    DUP             ; REPORT IT
                                BR      13$             ; GET ANOTHER ENTRY
                        7$:     CALL    CKUSR           ; SEE IF ENTRY IF FOR THE USER ADDRESSES
                                BCC     9$              ; IF CC, YES
                                CALL    CHKFE           ; SEE IF ENTRY IS FOR FE CYLINDERS
                                BCC     8$              ; IF CC, YES
                                MOV     #$MSG32,R0      ; 'INVALID ENTRY'
                                CALL    $PRINT          ; PRINT THE MESSAGE
                                BR      13$             ;
                        8$:     CALL    $SRFES          ; LOCATE WHERE TO PUT THE ENTRY
                                MOV     #MSG4R8,R4      ; FE BSF MESSAGE SUFFIX
                                CMP     R1,#$FEBSF+64.  ; IF THE FE BSF FULL ?
                                BNE     11$             ; IF NE, NO
                                BR      10$             ; REPORT FILE FULL
                        9$:     CALL    $SRUDF          ; LOCATE LAST ENTRY IN THE UDBSF
                                CMP     R1,#$UDBSF+1000-4       ; IS UDBSF FULL?
                                BNE     11$             ; IF NE, NO
                                MOV     #MSG1R8,R4      ; UDBSF MESSAGE SUFFIX
                        10$:    CALL    FULL            ; ISSUE FILE FULL MESSAGE
```

```
177 000626  000421                         BR      13$             ; TRY AGAIN
178 000630  016721  000000G         11$:    MOV     $CYL,(R1)+      ; PUT CYLINDER ADDRESS INTO UDBSF
179 000634  116721  000000G                 MOVB    $SEC,(R1)+      ; PUT SECTOR ADDRESS INTO UDBSF
180 000640  116721  000000G                 MOVB    $TRK,(R1)+      ; PUT TRACK ADDRESS INTO UDBSF
181 000644  105767  000000G                 TSTB    SKPFRE          ; SEE IF SECTOR WITHIN A SKIP REGION
182 000650  001410                         BEQ     13$             ; IF EQ, NO
183 000652  105767  000000G         12$:    TSTB    $HDRFG          ; WAS ENTRY FOR A HEADER DEFECT ?
184 000656  001405                         BEQ     13$             ; IF EQ, NO - GET NEXT ENTRY
185 000660  105067  000000G                 CLRB    $HDPFG          ; CLEAR THE HEADER DEFECT FLAG
186 000664  105267  000000G                 INCB    $SEC            ; INCREMENT SECTOR ADDRESS
187 000670  000675                         BR      5$              ; CONTINUE
188 000672  000207                 13$:    RETURN                  ;
189
190                                         ;+
191                                         ; **-CHKFE-SEE IF THE DEFECT ADDRESS ENTRY IS FOR THE FE CYLINDERS
192                                         ;
193                                         ; INPUTS:
194                                         ;       $CYL,$TRK,$SEC LOADED
195                                         ;
196                                         ; RETURNS:
197                                         ;       CS      ENTRY IS NOT FOR THE FE CYLINDERS
198                                         ;       CC      ENTRY IS FOR FE CYLINDERS
199                                         ;-
200
201 000674  004767  000000G         CHKFE:  CALL    $CVTDA          ; CONVERT THE PHYSICAL ADDRESS TO AN LBN
202 000700  026767  000000G 177302         CMP     $LBNH,FELBN     ; CHECK HIGH LBN
203 000706  103403                         BLO     1$              ; IF LO, DON'T LOOK ANY MORE
204 000710  026767  000000G 177274         CMP     $LBNL,FELBN+2   ; CHECK THE LOW LBN
205 000716  000207                 1$:     RETURN                  ;
206
207                                         ;+
208                                         ; **-CKUSR-SEE IF THE DEFECT ADDRESS ENTRY IS FOR THE USER CYLINDERS
209                                         ;
210                                         ; INPUTS:
211                                         ;       $CYL,$TRK,$SEC LOADED
212                                         ;
213                                         ; RETURNS:
214                                         ;       CS      ENTRY IS NOT FOR THE USER CYLINDERS
215                                         ;       CC      ENTRY IS FOR USER CYLINDERS
216                                         ;-
217
218 000720  004767  000000G         CKUSR:  CALL    $CVTDA          ; CONVERT THE PHYSICAL ADDRESS TO AN LBN
219 000724  026767  177264  000000G         CMP     USRLBN,$LBNH    ; CHECK HIGH LBN
220 000732  101003                         BHI     1$              ; IF LO, DON'T LOOK ANY MORE
221 000734  026767  177256  000000G         CMP     USRLBN+2,$LBNL  ; CHECK THE LOW LBN
222 000742  000207                 1$:     RETURN                  ;
223
224                                         ;+
225                                         ; **-DUP-DUP ROUTINE WILL ISSUE THE DUPLICATE ENTRY MEESAGE FOR SPECIFIED FILE
226                                         ;
227                                         ; THIS ROUTINE WILL ISSUE THE DUPLICATE ENTRY MESSAGE FOR THE SPECIFIED FILE
228                                         ;
229                                         ; INPUTS:
230                                         ;       R4 = ADDRESS OF SPECIFIED FILE MESSAGE
231                                         ;
232                                         ; OUTPUTS:
233                                         ;       NONE
```

```
                                        ;
                                        ;-
                                        ;
000744  012703  000040'       DUP:     MOV     #MSG7R8,R3      ;GET DUPLICATE ENTRY MESSAGE
000750  012700  000000G                MOV     #$BUFRX,R0      ;GET ADDRESS OF OUTPUT BUFFER
000754  004767  000000G                CALL    $MOVE           ;PUT MESSAGE IN OUTPUT BUFFER
000760  010403                        MOV     R4,R3           ;GET ADDRESS OF SPECIFIED FILE
000762  004767  000000G                CALL    $MOVE           ;PUT THAT IN OUTPUT BUFFER
000766  012700  000000G                MOV     #$BUFRX,R0      ;GET ADDRESS OF MESSAGE BUFFER
000772  004767  000000G                CALL    $PRINT          ;PRINT IT
000776  000207                        RETURN                  ;
                                        ;
                                        ;+
                                        ; **-FULL-FULL ROUTINE WILL ISSUE THE FILE FULL MESSAGE FOR SPECIFIED FILE
                                        ;
                                        ; INPUTS:
                                        ;       R4 = ADDRESS OF FILE NAME MESSAGE
                                        ;
                                        ; OUTPUTS:
                                        ;       NONE
                                        ;
                                        ;-
                                        ;
001000  012703  000063'       FULL:    MOV     #MSG8R8,R3      ;GET FILE FULL MESSAGE
001004  012700  000000G                MOV     #$BUFRX,R0      ;GET ADDRESS OF OUTPUT BUFFER
001010  004767  000000G                CALL    $MOVE           ;PUT MESSAGE IN OUTPUT BUFFER
001014  010403                        MOV     R4,R3           ;GET ADDRESS OF FILE NAME MESSAGE
001016  004767  000000G                CALL    $MOVE           ;PUT THAT IN OUTPUT BUFFER
001022  012700  000000G                MOV     #$BUFRX,R0      ;GET ADDRESS OF MESSAGE BUFFER
001026  004767  000000G                CALL    $PRINT          ;PRINT IT
001032  000207                        RETURN                  ;
                                        ;
                                        ;+
                                        ; **-INPUT-INPUT DATA FOR SPECIFIED FILE
                                        ;
                                        ; THIS ROUTINE WILL PROMPT THE USER FOR THE DESIRED CYLINDER, TRACK, AND SECTOR
                                        ; ADDRESS.
                                        ;
                                        ; INPUTS:
                                        ;       NONE
                                        ;
                                        ; OUTPUTS:
                                        ;       $CYL, $TRK, AND $SEC SET TO SPECIFIED VALUES
                                        ;       $HDRFG = 1 IF THE ENTRY HAD AN 'H' SUFFIX
                                        ;       IF Z = 1        NO MORE DATA TO INPUT
                                        ;       IF Z = 0        MORE DATA TO INPUT
                                        ;
                                        ;-
                                        ;
001034  105067  000000G       INPUT:   CLRB    $HDRFG          ; CLEAR THE HEADER FLAG
001040  012703  000000G                MOV     #$PHYDV,R3      ; POINTER TO PHYSICAL DEVICE GEOMETRY
001044  004767  000000G                CALL    $UPDMX          ; UPDATE MAX VALUES
001050  012700  000126'       1$:      MOV     #MSGDR8,R0      ; OPERATOR PROMPT
001054  004767  000000G                CALL    $PRMPT          ; GET INPUT FROM THE USER
001060  001001                        BNE     2$              ; IF NE, INPUT FROM USER
001062  000207                        RETURN                  ;
                                        ;
```

```
001064  004767  000150          2$:     CALL    LDLMTS          ; INITIALIZE DISK ADDRESS LIMITS
001070  012700  000000G                  MOV     #$BUFRX,R0      ; BUFFER POINTER
001074  122710  000043                  CMPB    #'#,(R0)        ; SEE IF OCTAL INPUT
001100  001452                          BEQ     4$              ; IF EQ, ERROR
001102  004767  000000G                  CALL    $CDTB           ; CONVERT THE CYLINDER ADDRESS
001106  122702  000054                  CMPB    #',,R2          ; CHECK PROPER SEPARATOR
001112  001045                          BNE     4$              ; IF NE, ERROR
001114  004767  000120                  CALL    LDLMTS          ; LOAD NEW TRACK ADDRESS LIMITS
001120  020167  177100                  CMP     R1,MAXCYL       ; LOOK FOR VALID CYLINDER ADDRESS
001124  101040                          BHI     4$              ; IF HI, INVALID CYLINDER ADDRESS
001126  020167  177066                  CMP     R1,MINCYL       ; LOOK FOR VALID CYLINDER ADDRESS
001132  002435                          BLT     4$              ; IF LT, INVALID CYLINDER ADDRESS
001134  010167  000000G                  MOV     R1,$CYL         ; LOAD THE CYLINDER ADDRESS

001140  004767  000000G                  CALL    $CDTB           ; CONVERT THE TRACK ADDRESS
001144  122702  000054                  CMPB    #',,R2          ; CHECK FOR PROPER SEPARATOR
001150  001026                          BNE     4$              ; IF NE, ERROR
001152  020167  177050                  CMP     R1,MAXTRK       ; LOOK FOR VALID TRACK ADDRESS
001156  101023                          BHI     4$              ; IF HI, INVALID TRACK  ADDRESS
001160  020167  177036                  CMP     R1,MINTRK       ; LOOK FOR VALID TRACK ADDRESS
001164  002420                          BLT     4$              ; IF LT, INVALID TRACK  ADDRESS
001166  010167  000000G                  MOV     R1,$TRK         ; LOAD THE TRACK ADDRESS

001172  004767  000000G                  CALL    $CDTB           ; CONVERT THE SECTOR ADDRESS
001176  020127  000037                  CMP     R1,#31.         ; LOOK FOR VALID SECTOR ADDRESS
001202  101011                          BHI     4$              ; IF HI, INVALID SECTOR  ADDRESS
001204  010167  000000G                  MOV     R1,$SEC         ; LOAD THE SECTOR ADDRESS
001210  122702  000110                  CMPB    #'H,R2          ; HEADER ERROR ADDRESS ?
001214  001002                          BNE     3$              ; IF NE, NO
001216  105267  000000G                  INCB    $HDRFG          ; SET THE HEADER ERROR FLAG
001222  000244                  3$:     CLZ                     ; SHOW MORE INPUT
001224  000207                          RETURN                  ; EXIT
001226  012700  000000G          4$:     MOV     #$MSG32,R0      ; INVALID RESPONSE
001232  004767  000000G                  CALL    $PRINT          ; TYPE THE MESSAGE
001236  000704                          BR      1$              ; CONTINUE

                                ;+
                                ; **-LDLMTS-LOAD LIMITS
                                ;
                                ; THIS ROUTINE LOADS THE MINIMUM AND MAXIMUM DISK ADDRESS LIMITS
                                ; USED BY THE 'INPUT' SUBROUTINE.
                                ;
                                ;-

001240  005067  176754          LDLMTS: CLR     MINCYL          ; ASSUME MIN. CYLINDER= 0.
001244  005067  176752                  CLR     MINTRK          ;        "       TRACK= 0.
001250  012767  001060  176746          MOV     #560.,MAXCYL    ; ASSUME MAX. CYLINDER= 560.
001256  012767  000015  176742          MOV     #13.,MAXTRK     ;        "       TRACK= 13.
001264  105767  000000G                  TSTB    $SWFFE          ; FORMATTING FE CYLINDERS ONLY ?
001270  001411                          BEQ     1$              ; BR IF NO
001272  012767  001056  176720          MOV     #558.,MINCYL    ; MIN. CYLINDER= 558.
001300  020127  001056                  CMP     R1,#558.        ; IS CYLINDER INPUT 558. ?
001304  001003                          BNE     1$              ; BR IF NO
001306  012767  000015  176706          MOV     #13.,MINTRK     ; MIN. TRACK= 13.
001314  000207                  1$:     RETURN

000001                          .END
```

SYMBOL TABLE

```
CHKFE   000674R         MSGCR8  000100R         $CDTB = ****** GX       $PRMPT= ****** GX       $SRTRK= ****** GX
CKUSR   000720R         MSGDR8  000126R         $CVTDA= ****** GX       $PRNTQ= ****** GX       $SRUDF= ****** GX
DPARS = ****** GX       MSGER8  000145RG        $CYL  = ****** GX       $RDBSF= ****** GX       $SSF  = ****** GX
DUP     000744R         MSG1R8  000000R         $DSTRB  000312RG        $RDSSF= ****** GX       $SSFHD= ****** GX
FELBN   000210R         MSG2R8  000007R         $FEBSF= ****** GX       $REVNM= ****** GX       $SSFLN= ****** GX
FULL    001000R         MSG3R8  000015R         $HDRFG= ****** GX       $RMBOM  000230RG        $SWFFE= ****** GX
INPUT   001034R         MSG4R8  000021R         $LBNH = ****** GX       $RBOSF= ****** GX       $SWMCD= ****** GX
LDLMTS  001240R         MSG7R8  000040R         $LBNL = ****** GX       $SEC  = ****** GX       $TRK  = ****** GX
MAXCYL  000224R         MSG8R8  000063R         $MOVE = ****** GX       $SRFES= ****** GX       $UDBSF= ****** GX
MAXTRK  000226R         SKPFRE= ****** GX       $MSG32= ****** GX       $SRMDF= ****** GX       $UPDMX= ****** GX
MINCYL  000220R         USRLBN  000214R         $PHYDV= ****** GX       $SRSDF= ****** GX       $WTBSF= ****** GX
MINTRK  000222R         $BUFRX= ****** GX       $PRINT= ****** GX       $SRSSF= ****** GX       $WTSSF= ****** GX

. ABS.  000000      000
        001316      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  602 WORDS  ( 3 PAGES)
DYNAMIC MEMORY:  7942 WORDS  ( 30 PAGES)
ELAPSED TIME:  00:00:13
[300,20]FMTR8M,[300,30]FMTR8M=[300,10]FMTR8M
```

```
                        .TITLE  FMTR8L - DISPLAY BAD SECTOR FILE CONTENTS
                        .IDENT  /01.00/
                        ;
                        ;
                        ; COPYRIGHT (C) 1980
                        ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                        ;
                        ;
                        ; THIS  SOFTWARE  IS  FURNISHED  UNDER A LICENSE FOR USE ONLY ON A
                        ; SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION
                        ; OF  THE  ABOVE COPYRIGHT  NOTICE.  THIS SOFTWARE, OR  ANY  OTHER
                        ; COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE  MADE AVAILABLE
                        ; TO  ANY  OTHER PERSON  EXCEPT  FOR USE ON SUCH SYSTEM AND TO ONE
                        ; WHO AGREES TO THESE LICENSE TERMS.  TITLE  TO  AND  OWNERSHIP OF
                        ; THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DIGITAL.
                        ;
                        ; THE  INFORMATION  IN THIS DOCUMENT IS SUBJECT  TO CHANGE WITHOUT
                        ; NOTICE AND SHOULD NOT BE CONSTRUED AS A  COMMITMENT  BY  DIGITAL
                        ; EQUIPMENT CORPORATION.
                        ;
                        ; DIGITAL ASSUMES NO  RESPONSIBILITY FOR THE USE OR RELIABILITY OF
                        ; ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
                        ;
                        ;
                        ;  VERSION 01.00
                        ;
                        ; C. HESS        9-JUL-80
                        ; M. LEAVITT     22-OCT-80
                        ;
                        .NLIST  BEX
000000  123  123  106   MSG1L:  .ASCIZ  /SSF (SKIPPED SECTOR FILE) CONTENTS/
000043  115  104  102   MSG2L:  .ASCIZ  /MDBSF CONTENTS/
000062  125  104  102   MSG3L:  .ASCIZ  /UDBSF CONTENTS/
000101  103  122  105   MSG4L:  .ASCIZ  @CREATION DATE: @
000121  110  104  101   MSG5L:  .ASCIZ  @HDA S/N: @
000133  103  131  114   MSG6L:  .ASCIZ  /CYL,TRK,SEC/
000147  052  040  111   MSG7L:  .ASCIZ  /* INVALID DATE */
000170  052  040  116   MSG8L:  .ASCIZ  /* NO ENTRIES */
000207  122  105  126   MSG9L:  .ASCIZ  /REVISION DATE: /
000227  122  105  126   MSG10L: .ASCIZ  /REVISION NUMBER: /
000251  115  104  102   MSG11L: .ASCIZ  /MDBSF 'LOGICAL' CONTENTS/
000302  125  104  102   MSG12L: .ASCIZ  /UDBSF 'LOGICAL' CONTENTS/
000333  040  000        SPACE:  .ASCIZ  / /
                        
000335  000            SSFLST: .BYTE   0               ; LIST SSF SWITCH
                        .EVEN
                        
000336                 DPARS:  .BLKW   10.             ; OUTPUT CONVERSION BUFFER
```

```
                        ;+
                        ; **-$RM80L-DISPLAY THE CONTENTS OF THE MDBSF, UDBSF, AND SSF
                        ;
                        ; THIS ROUTINE WILL DISPLAY THE CONTENTS OF THE MDBSF, THE UDBSF
                        ; AND THE SSF FILES.
                        ;
                        ; INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       THE CONTENTS OF THE MDBSF, UDBSF, AND SSF PRINTED ON 'TI:'
                        ;       '$SWLST' CLEARED
                        ;-
                        .ENABL  LSB
                        
000362  004767  000000G $RM80L::CALL    $RDSSF          ; READ THE SKIPPED SECTOR FILE (SSF)
000366  004767  000000G         CALL    $RDBSF          ; READ THE MDBSF AND THE UDBSF
                        
                        ; DISPLAY THE SSF CONTENTS
                        
000372  012700  000000'         MOV     #MSG1L,R0       ; 'SSF (SKIPPED SECTOR FILE) CONTENTS'
000376  012701  000060          MOV     #60,R1          ; CARRIAGE CONTROL CHARACTER
000402  004767  000000G         CALL    $PRNTO          ;
000406  012701  000000G         MOV     #$SSF,R1        ; SERIAL NUMBER POINTER
000412  004767  000600          CALL    HDASN           ; PRINT THE HDA SERIAL NUMBER
000416  012700  000000G         MOV     #$BUFRX,R0      ; BUFFER POINTER
000422  012703  000101'         MOV     #MSG4L,R3       ; 'CREATION DATE: '
000426  004767  000G00G         CALL    $MOVE           ; LOAD THE BUFFER
000432  012703  000004G         MOV     #$SSF+4,R3      ; DATE FIELD POINTER
000436  004767  000240          CALL    SFDATE          ; PRINT THE SSF DATE FIELDS
000442  012700  0G0000G         MOV     #$BUFRX,R0      ; BUFFER POINTER
000446  012703  000207'         MOV     #MSG9L,R3       ; 'REVISION DATE: '
000452  004767  000000G         CALL    $MOVE           ; LOAD THE BUFFER
000456  012703  0000006G        MOV     #$SSF+6,R3      ; DATE FIELD POINTER
000462  004767  000214          CALL    SFDATE          ; PRINT THE SSF DATE FIELDS
000466  012700  000000G         MOV     #$BUFRX,R0      ; BUFFER POINTER
000472  012703  000227'         MOV     #MSG10L,R3      ; 'REVISION NUMBER: '
000476  004767  000000G         CALL    $MOVE           ; LOAD THE BUFFER
000502  016701  0000010G        MOV     $SSF+10,R1      ; REVISION NUMBER
000506  005002                  CLR     R2              ; ENABLE ZERO SUPRESSION
000510  004767  000000G         CALL    $CBDMG          ; CONVERT
000514  105010                  CLRB    (R0)            ; ASCIZ TERMINATOR
000516  012700  000000G         MOV     #$BUFRX,R0      ; OUTPUT POINTER
000522  004767  000000G         CALL    $PRINT          ;
000526  012703  000000C         MOV     #$SSF+<2*$SSFHD>,R3     ; BEGINNING OF SSF DATA FIELD
000532  112767  177777  177575  MOVB    #-1,SSFLST      ; SET SSF LIST FLAG
000540  004767  000270          CALL    BSFDAT          ; PRINT THE SSF FILE CONTENTS
000544  105067  177565          CLRB    SSFLST          ; RESET THE SSF LIST FLAG
                        
                        ; DISPLAY THE MDBSF CONTENTS
                        
000550  012700  000043'         MOV     #MSG2L,R0       ; 'MDBSF CONTENTS'
000554  105767  000000G         TSTB    $SWLOG          ; SEE IF LOGICAL DISPLAY
000560  001402                  BEQ     10$             ; IF EQ, NO
000562  012700  000251'         MOV     #MSG11L,R0      ; CHANGE TO 'MDBSF LOGICAL CONTENTS'
000566  012701  000060  10$:    MOV     #60,R1          ; CARRIAGE CONTROL CHARACTER
```

```
110 000572  004767  000000G                 CALL    $PRNTO          ;
111 000576  012701  000000G                 MOV     #$MDBSF,R1      ; MDBSF S/N POINTER
112 000602  004767  000410                  CALL    HDASN           ; TYPE THE SERIAL NUMBER
113 000606  012703  000010G                 MOV     #$MDBSF+10,R3   ; MDBSF DATA POINTER
114 000612  004767  000216                  CALL    BSFDAT          ; DISPLAY THE MDBSF
115
116                                 ; DISPLAY THE UDBSF CONTENTS
117
118 000616  012700  000062'                MOV     #MSG3L,R0       ; 'UDBSF CONTENTS'
119 000622  105767  000000G                TSTB    $SWLOG          ; SEE IF LOGICAL DISPLAY
120 000626  001402                         BEQ     20$             ; IF EQ, NO
121 000630  012700  000302'                MOV     #MSG12L,R0      ; CHANGE TO 'UDBSF LOGICAL CONTENTS'
122 000634  012701  000060          20$:   MOV     #60,R1          ; CARRIAGE CONTROL CHARACTER
123 000640  004767  000000G                CALL    $PRNTO          ;
124 000644  012701  000000G                MOV     #$UDBSF,R1      ; UDBSF S/N POINTER
125 000650  004767  000342                 CALL    HDASN           ; TYPE THE SERIAL NUMBER
126 000654  012703  000010G                MOV     #$UDBSF+10,R3   ; UDBSF DATA POINTER
127 000660  004767  000150                 CALL    BSFDAT          ; DISPLAY THE UDBSF
128 000664  012700  000333'                MOV     #SPACE,R0       ; BLANK LINE
129 000670  004767  000000G                CALL    $PRINT          ; PRINT THE LINE
130 000674  105067  000000G                CLRB    $SWLST          ; CLEAR THE 'LIST' FLAG
131 000700  000207                         RETURN                  ;
132
133                                 .DSABL  LSB
134
135                                 ;+
136                                 ; **-SFDATE-DISPLAY THE SSF DATE FIELD CONTENTS
137                                 ;
138                                 ; THIS ROUTINE PRINTS THE CREATION AND REVISION DATE FIELDS
139                                 ; FROM THE SSF
140                                 ;
141                                 ; INPUT:
142                                 ;       R0 = BUFFER POINTER
143                                 ;       R3 = DATE FIELD POINTER
144                                 ;
145                                 ; OUTPUT:
146                                 ;       NONE
147                                 ;
148                                 ;-
149
150 000702  011301                  SFDATE: MOV     (R3),R1         ; DATE
151 000704  042701  177037                  BIC     #^C740,R1       ; LEAVE MONTH CODE
152 000710  005701                          TST     R1              ; LOOK FOR VALID MONTH
153 000712  001436                          BEQ     10$             ; IF EQ, MONTH CODE OF ZERO
161 000726  020127  000014                  CMP     R1,#12.         ; CHECK MONTH UPPER LIMIT
162 000732  101026                          BHI     10$             ; IF HI, MONTH CODE ERROR
163 000734  010167  177400                  MOV     R1,DPARS+2      ; MOVE MONTH
164 000740  011301                          MOV     (R3),R1         ; GET THE DATE AGAIN
165 000742  042701  177740                  BIC     #^C37,R1        ; LEAVE THE DAY CODE
166 000746  020127  000037                  CMP     R1,#31.         ; CHECK VALIDITY
167 000752  101016                          BHI     10$             ; IF HI, INVALID DAY
168 000754  010167  177362                  MOV     R1,DPARS+4      ; MOVE DAY
169 000760  116301  000001                  MOVB    1(R3),R1        ; GET THE YEAR CODE
170 000764  006201                          ASR     R1              ; RIGHT JUSTIFY
171 000766  062701  000000G                 ADD     #BASYR,R1       ; ADD THE BASE YEAR
172 000772  010167  177340                  MOV     R1,DPARS        ; YEAR
173 000776  012701  000336'                 MOV     #DPARS,R1       ; INPUT BUFFER POINTER
```

```
                        CALL    $DAT            ; CONVERT THE DATE
                        BR      20$             ; CONTINUE
                10$:    MOV     #MSG7L,R3       ; '** INVALID DATE **'
                        CALL    $MOVE           ; LOAD THE TEXT
                20$:    CLRB    (R0)            ; ASCIZ TERMINATOR
                        MOV     #$BUFRX,R0      ; BUFFER POINTER
                        CALL    $PRINT          ; PRINT THE DATE
                        RETURN                  ;

                ;+
                ; **-BSFDAT-DISPLAY THE BAD SECTOR FILE CONTENTS
                ;
                ; THIS ROUTINE PRINTS THE DEFECT ADDRESS ENTRIES IN THE MDBSF
                ;
                ; INPUT:
                ;       R3 = START OF THE BAD SECTOR FILE DATA AREA
                ;       SSFLST = -1 IF THE SSF IS BEING LISTED
                ;
                ; OUTPUT:
                ;       NONE
                ;
                ;-

                BSFDAT: TST     (R3)            ; LOOK FOR END OF ENTRIES
                        BPL     10$             ; IF PL, FILE ENTRIES PRESENT
                        MOV     #MSG8L,R0       ; '** NO ENTRIES **'
                        CALL    $PRINT          ;
                        BR      40$             ; EXIT
                10$:    MOV     #MSG6L,R0       ; 'CYL,TRK,SEC'
                        CALL    $PRINT          ;
                20$:    MOV     #$BUFRX,R0      ; OUTPUT BUFFER POINTER
                        MOV     (R3),$CYL       ; LOAD THE CYLINDER ADDRESS
                        MOVB    3(R3),$TRK      ; LOAD THE TRACK ADDRESS
                        MOVB    2(R3),$SEC      ; LOAD THE SECTOR ADDRESS
                        MOV     (R3)+,R1        ; GET THE CYLINDER ADDRESS
                        BMI     40$             ; IF MI, NO MORE ENTRIES
                        CLR     R2              ; ENABLE ZERO SUPRESSION
                        CALL    $CBDMG          ; CONVERT
                        MOVB    #',,(R0)+       ; SEPARATOR
                        MOVB    1(R3),R1        ; TRACK ADDRESS
                        CLR     R2              ; ENABLE ZERO SUPRESSION
                        CALL    $CBDMG          ; CONVERT
                        MOVB    #',,(R0)+       ; SEPARATOR
                        MOVB    (R3),R1         ; SECTOR ADDRESS
                        TSTB    SSFLST          ; LISTING THE SSF CONTENTS ?
                        BNE     30$             ; IF NE, YES
                        TSTB    $SWLOG          ; PRINT BSF LOGICAL ADDRESSES ?
                        BNE     30$             ; IF NE, YES
                        CALL    $SRSSF          ; CHECK THE SSF
                        BCS     30$             ; IF CS, SECTOR ENTRY PHYSICAL VALUE
                        INC     R1              ; INCREMENT THE SECTOR ADDRESS
                30$:    TST     (R3)+           ; INCREMENT THE INPUT POINTER
                        CLR     R2              ; ENABLE ZERO SUPRESSION
                        CALL    $CBDMG          ; CONVERT
                        CLRB    (R0)            ; ASCIZ TERMINATOR
                        MOV     #$BUFRX,R0      ; OUTPUT BUFFER POINTER
                        CALL    $PRINT          ; PRINT THE ADDRESS
```

```
                                        BR      20$             ; LOOP
                                40$:    RETURN                  ;
                                ;+
                                ; **-HDASN-DISPLAY THE HDA SERIAL NUMBER
                                ;
                                ; THIS ROUTINE CONVERTS THE HDA SERIAL NUMBER FROM THE DEC STD 144
                                ; FILES OR THE SSF TO ASCII AND PRINTS IT.
                                ;
                                ; INPUTS:
                                ;       R1 = SERIAL NUMBER FIELD POINTER
                                ;
                                ; OUTPUTS:
                                ;       NONE
                                ;
                                ;-
                                HDASN:  MOV     (R1)+,DPARS+2   ; EXCHANGE S/N MSB & LSB
                                        MOV     (R1),DPARS      ;
                                        MOV     #DPARS,R1       ; RELOAD THE SERIAL NUMBER POINTER
                                        MOV     #MSG5L,R3       ; 'HDA S/N: '
                                        MOV     #$BUFRX,R0      ; BUFFER POINTER
                                        CALL    $MOVE           ; LOAD THE BUFFER
                                        CLR     R2              ; SUPRESS LEADING ZEROS
                                        CALL    $CDDMG          ; CONVERT THE SERIAL NUMBER TO ASCII
                                        CLRB    (R0)            ; ASCIZ TERMINATOR
                                        MOV     #$BUFRX,R0      ; BUFFER ADDRESS
                                        CALL    $PRINT          ; PRINT THE SERIAL NUMBER
                                        RETURN                  ;

                                .END
```

SYMBOL TABLE

```
BASYR = ****** GX       MSG2L   000043R       SFDATE  000702R       $MDBSF= ****** GX       $SRSSF= ****** GX
BSFDAT  001034R         MSG3L   000062R       SPACE   000333R       $MOVE = ****** GX       $SSF  = ****** GX
DPARS   000336R         MSG4L   000101R       SSFLST  000335R       $PRINT= ****** GX       $SSFHD= ****** GX
HDASN   001216R         MSG5L   000121R       $BUFRX= ****** GX     $PRNTO= ****** GX       $SWLOG= ****** GX
MSG1L   000000R         MSG6L   000133R       $CBDMG= ****** GX     $RDBS$= ****** GX       $SWLST= ****** GX
MSG10L  000227R         MSG7L   000147R       $CDDMG= ****** GX     $RDSSF= ****** GX       $TRK  = ****** GX
MSG11L  000251R         MSG8L   000170R       $CYL  = ****** GX     $RM80L  000362RG        $UDBSF= ****** GX
MSG12L  000302R         MSG9L   000207R       $DAT  = ****** GX     $SEC  = ****** GX

. ABS.  000000      000
        001270      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  471 WORDS  ( 2 PAGES)
DYNAMIC MEMORY:  7942 WORDS  ( 30 PAGES)
ELAPSED TIME:  00:00:10
[300,20]FMTR8L,[300,30]FMTR8L=[300,10]FMTR8L
```

```
                            .TITLE  FMTINT - INITIALIZE MDBSF, UDBSF, SSF
                            .IDENT  /01.00/
                            ;
                            ;
                            ; COPYRIGHT (C) 1980
                            ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                            ;
                            ;
                            ; THIS  SOFTWARE  IS  FURNISHED  UNDER A LICENSE FOR USE ONLY ON A
                            ; SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION
                            ; OF   THE   ABOVE COPYRIGHT   NOTICE.   THIS SOFTWARE, OR  ANY  OTHER
                            ; COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE  MADE AVAILABLE
                            ; TO  ANY  OTHER PERSON  EXCEPT  FOR USE ON SUCH SYSTEM AND TO ONE
                            ; WHO AGREES TO THESE LICENSE TERMS.  TITLE  TO  AND  OWNERSHIP OF
                            ; THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DIGITAL.
                            ;
                            ; THE  INFORMATION  IN THIS DOCUMENT IS SUBJECT  TO CHANGE WITHOUT
                            ; NOTICE AND SHOULD NOT BE CONSTRUED AS A  COMMITMENT  BY  DIGITAL
                            ; EQUIPMENT CORPORATION.
                            ;
                            ; DIGITAL ASSUMES NO  RESPONSIBILITY FOR THE USE OR RELIABILITY OF
                            ; ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
                            ;
                            ;
                            ;  VERSION 01.00
                            ;
                            ; C. HESS        15-AUG-80
                            ; M. LEAVITT     14-JAN-81
                            ;
                            .NLIST  BEX
000000  052  052  040  MSG1RI: .ASCIZ  /** WARNING - BAD SECTOR FILES WILL BE OVERWRITTEN ON /
                            .EVEN
```

```
                        ;+
                        ; **-$RM80I-INITIALIZE THE RM80'S MDBSF, UDBSF, & SSF TRACKS
                        ;
                        ; THIS ROUTINE WILL FORMAT THE DEC STD 144 TRACK AND THE SSF TRACK
                        ;  AND WRITE NULL 16 BIT DEC STD 144 AND SKIP SECTOR FILES.
                        ;
                        ; INPUTS:
                        ;       NONE
                        ;
                        ; OUTUTS:
                        ;       NONE
                        ;-
                        ;
000066  012703  000000' $RM80I::MOV     #MSG1RI,R3      ; WARNING MESSAGE ADDRESS
000072  004767  000000G        CALL    $UNTID          ; STORE MESSAGE AND DEVICE/UNIT
000076  012703  000000G        MOV     #$MSG24,R3      ; MESSAGE SUFFIX
000102  004767  000000G        CALL    $MOVE           ; LOAD THE SUFFIX INTO THE MESSAGE BUFFER
000106  012700  000144G        MOV     #$BUF+100.,R0   ; PRINT THE WARNING MESSAGE
000112  012701  000060         MOV     #60,R1          ; 2 LINE FEED CARRIAGE CONTROL
000116  004767  000000G        CALL    $PRNTO          ; PRINT WARNING MESSAGE
000122  012700  000000G        MOV     #$MSG31,R0      ; 'YES-NO' MESSAGE POINTER
000126  004767  000000G        CALL    $PRMPT          ; PROMPT AND GET RESPONSE
000132  001404                 BEQ     1$              ; IF EQ, RESPONSE WAS 'NO'
000134  122767  000131  000000G CMPB    #'Y,$BUFRX      ; SEE IF RESPONSE WAS 'YES'
000142  001402                 BEQ     2$              ; IF EQ, RESPONSE WAS 'YES'
000144  000167  000000G  1$:    JMP     $FMTEX          ; EXIT
000150  012703  000374   2$:    MOV     #256.-4,R3      ; GET NUMBER OF ENTRIES TO RESET
000154  012700  000000G        MOV     #$MDBSF,R0      ; GET START OF MDBSF
000160  016720  000000G        MOV     $SNAME,(R0)+    ; STUFF THE NEW SERIAL NUMBER
000164  016720  000002G        MOV     $SNAME+2,(R0)+  ; ...
000170  005020                 CLR     (R0)+           ; BLANK WORD
000172  005020                 CLR     (R0)+           ; SHOW NOT AN ALIGNMENT DISK
000174  012720  177777   3$:    MOV     #-1,(R0)+       ; EMPTY THE MDBSF
000200  005303                 DEC     R3              ; FINISHED?
000202  001374                 BNE     3$              ; IF NE NO
000204  012700  000000G        MOV     #$UDBSF,R0      ; GET START OF UDBSF
000210  016720  000000G        MOV     $SNAME,(R0)+    ; STUFF THE NEW SERIAL NUMBER
000214  016720  000002G        MOV     $SNAME+2,(R0)+  ; ...
000220  005020                 CLR     (R0)+           ; BLANK WORD
000222  005020                 CLR     (R0)+           ; SHOW NOT AN ALIGNMENT DISK
000224  012703  000374         MOV     #256.-4,R3      ; GET NUMBER OF ENTRIES TO RESET
000230  012720  177777   4$:    MOV     #-1,(R0)+       ; EMPTY THE UDBSF
000234  005303                 DEC     R3              ; FINISHED?
000236  001374                 BNE     4$              ; IF NE NO
000240  012703  000000C        MOV     #256.*<$SSFLN>,R3       ; SSF SIZE
000244  012700  000000G        MOV     #$SSF,R0        ; SSF BUFFER POINTER
000250  012720  177777   5$:    MOV     #-1,(R0)+       ; CLEAR A LOCATION
000254  005303                 DEC     R3              ; COUNT THE LOCATION
000256  001374                 BNE     5$              ; IF NE, NOT FINISHED
000260  016767  000000G 000000G MOV     $SNAME,$SSF     ; SERIAL NUMBER LSB
000266  016767  000002G 000002G MOV     $SNAME+2,$SSF+2 ; SERIAL NUMBER MSB
000274  016767  000000G 000004G MOV     DPARS,$SSF+4    ; CREATION DATE
000302  016767  000000G 000006G MOV     DPARS,$SSF+6    ; REVISION DATE
000310  005067  000010G        CLR     $SSF+10         ; REVISION NUMBER
000314  012767  000001  000012G MOV     #1,$SSF+12      ; SSF DATA TYPE INDICATOR (16 BIT)
000336  005067  000022G        CLR     $SSF+22         ; NUMBER OF SSF ENTRIES
000342  005067  000024G        CLR     $SSF+24         ; CHECKSUM
```

```
101 000346  004767  000000G                 CALL    $R80SF          ; CALL THE 'SPECIAL FORMATTER'
102 000352  004767  000000G                 CALL    $WTBSF          ; WRITE THE DEC STD 144 INFORMATION
103 000356  004767  000000G                 CALL    $WTSSF          ; WRITE THE SSF
104 000362  105067  000000G                 CLRB    $SWINT          ; CLEAR THE INITIALIZATION FLAG
105 000366  000207                  6$:     RETURN                  ;
106
107         000001                  .END
```

SYMBOL TABLE

```
DPARS = ****** GX       $MDBSF= ****** GX       $PRMPT= ****** GX       $SNAME= ****** GX       $UDBSF= ****** GX
MSG1RI  000000R         $MOVE = ****** GX       $PRNTO= ****** GX       $SSF  = ****** GX       $UNTID= ****** GX
$BUF  = ****** GX       $MSG24= ****** GX       $RMBOI  000066RG        $SSFLN= ****** GX       $WTBSF= ****** GX
$BUFRX= ****** GX       $MSG31= ****** GX       $RBOSF= ****** GX       $SWINT= ****** GX       $WTSSF= ****** GX
$FMTEX= ****** GX

. ABS.  000000      000
        000370      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  489 WORDS  ( 2 PAGES)
DYNAMIC MEMORY:  7942 WORDS  ( 30 PAGES)
ELAPSED TIME:  00:00:07
[300,20]FMTINT,[300,30]FMTINT=[300,10]FMTINT
```