

RMBO

RMBO FORMATTER  
CZRNJAO

AH-T123A-MC  
FICHE 1 OF 1

JUL 1982  
COPYRIGHT © 1982  
MADE IN USA



The main body of the document is a large, dark rectangular area. It contains a grid of small, illegible text blocks on the left side, which appear to be individual data entries or records. The right side of this area is mostly blank, with some faint, circular patterns or artifacts visible, possibly from the scanning process or the original document's content.

.REM @

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43

IDENTIFICATION

PRODUCT CODE: AC-T122A-MC  
PRODUCT NAME: CZRNJAO RM80 FORMATTER  
PRODUCT DATE: JANUARY 1982  
MAINTAINER: CX DIAGNOSTIC GROUP  
AUTHOR: C. HESS, M. LEAVITT

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

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED UNDER A LICENSE AND MAY ONLY BE USED OR COPIED IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE.

DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.

COPYRIGHT (C) 1982 DIGITAL EQUIPMENT CORPORATION

CONTENTS

45	
46	
47	
48	
49	1. INTRODUCTION
50	1. ABSTRACT
51	
52	2. OPERATING REQUIREMENTS
53	1. HARDWARE REQUIREMENTS
54	2. PREREQUISITE DIAGNOSTIC PROGRAMS
55	
56	3. OPERATING PROCEDURE
57	1. LOADING
58	2. SWITCH OPTIONS
59	3. STARTING
60	4. HALTING
61	5. RESTARTING
62	
63	4. OPERATOR INTERFACE
64	1. PROGRAM IDENTIFICATION
65	2. CONSOLE DIALOGUE
66	3. WARNING MESSAGES
67	4. PROGRESS REPORTS
68	5. PERFORMANCE REPORTS
69	6. PROGRAM HALTS
70	7. ERROR REPORTS
71	8. EXECUTION TIME
72	
73	5. ENVIRONMENTAL SUPPORT
74	1. PROCESSOR COMPATIBILITY
75	2. DUAL PORT CONFIGURATIONS
76	3. MEMORY PARITY HARDWARE
77	4. MEMORY MANAGEMENT HARDWARE
78	5. ACT, APT COMPATIBILITY
79	6. XXDP COMPATIBILITY
80	7. OPERATING SYSTEM COMPATIBILITY
81	
82	6. FORMAT DESCRIPTION
83	1. FORMAT PROCESS
84	
85	7. RECONSTRUCTION DESCRIPTION
86	1. HANDLING OLD DEFECTS
87	2. HANDLING NEW DEFECTS
88	
89	8. BAD SECTOR FILE LAYOUTS
90	1. SKIPPED SECTOR FILE
91	2. DEC STANDARD 144 FILE
92	
93	9. ERROR TYPE DESCRIPTION
94	
95	
96	
97	
98	
99	
100	
101	

103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159

## 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, MBSF AND UBSF, RESPECTIVELY.)

THE RM80 HDA MEDIA IS VERIFIED IN MANUFACTURING USING PARAMETRIC VERIFICATION EQUIPMENT. THE DEFECT LIST GENERATED BY THE MANUFACTURING VERIFICATION 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 INCIDENTAL 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 MANUFACTURING 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

161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216

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  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273

'DRIVE #: '

ENTER THE DRIVE ADDRESS NUMBER (0-7) FOLLOWED BY CARRIAGE RETURN. IF THE SELECTED DRIVE IS NOT AVAILABLE, THE PROGRAM WILL TYPE AN ERROR MESSAGE AND RETURN TO THE THIS PROMPT.

NOTE: ONLY ONE DRIVE CAN BE FORMATTED AT A TIME. THIS PROGRAM DOES NOT ALLOW MULTIPLE DRIVE FORMATTING.

AFTER THE USER HAS SELECTED THE DRIVE TO BE FORMATTED, THE PROGRAM WILL ASK FOR AN OPTION TO BE SELECTED WITH THE FOLLOWING PROMPT.

'OPTIONS: '

THE COMMANDS AVAILABLE FOR THE OPTION PROMPT ARE AS FOLLOWS:

IN INITIALIZE ALL THE BAD SECTOR FILES (DEC STD 144 AND SSF).

NOTE: THE INITIALIZE COMMAND WILL DESTROY THE CONTENTS OF ALL THE BAD SECTOR FILES (DEC STD 144 AND SSF).

FO FORMAT THE DISK. (ALL CYLINDERS 0 - 560.)

<CR> SAME AS 'FO' COMMAND.

FO:F FORMAT THE FE CYLINDERS ONLY. (CYL 559., TRK 2 - CYL 560., TRK 13.)

LI LIST THE PHYSICAL ADDRESSES OF THE DEFECTS IN ALL THE BAD SECTOR FILES (DEC STD 144 FILES AND SSF).

LI:L LIST THE LOGICAL ADDRESSES OF THE DEFECTS IN THE DEC STD 144 FILES. (SSF DOES NOT CONTAIN ANY LOGICAL ADDRESSES)

VFL=N VERIFY SECTOR COUNT, WHERE 'N' IS THE NUMBER OF TIMES TO RE-READ A TRACK DURING VERIFY OPERATION. THE DEFAULT IS 1 (RANGE 1-256.). AFTER COMPLETING THE FORMAT, 'N' WILL RETURN TO ITS DEFAULT VALUE.

ERL=N ERROR LIMIT COUNT, WHERE 'N' IS THE NUMBER OF ERRORS ALLOWED BEFORE THE FORMAT IS ABORTED. THE DEFAULT IS 256 (RANGE 1-256.). AFTER COMPLETING THE FORMAT, 'N' WILL RETURN TO ITS DEFAULT VALUE.

CSR=N RM BASE ADDRESS, WHERE 'N' IS THE ADDRESS OF RMCS1 IN OCTAL. THE DEFAULT IS 176700. AFTER COMPLETING THE FORMAT, 'N' WILL RETAIN ITS CHANGED VALUE.

VEC=N RM VECTOR ADDRESS, WHERE 'N' IS THE ADDRESS OF INTERRUPT ADDRESS IN OCTAL. THE DEFAULT IS 254. AFTER COMPLETING THE FORMAT, 'N' WILL RETAIN ITS CHANGED VALUE.

275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329

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.

331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387

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 TERMINATE 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.



389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444

'\*\* 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 RESPONDING 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 PERTINENT TO THE ERROR. THE LAST LINE CONTAINS THE DECIMAL EQUIVALENT 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

446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502

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 SUBSYSTEM 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

504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560

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

562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618

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 TO 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

620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676

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 ARE AFFECTED BY DEFECTS, THE HEADER DEFECT WILL TAKE REPORTING PRECEDENCE.
5. NEW DEFECTS DETECTED ON PREVIOUSLY ERROR FREE TRACKS 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.,

678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734

CYLINDER ADDRESS FOLLOWED BY SECTOR AND TRACK ADDRESSES. 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.

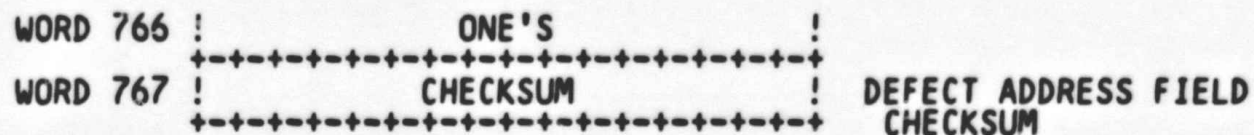
THE SERIAL NUMBER FIELD IN BOTH THE SSF AND THE DEC STD 144 FILE IS IN THE SAME FORMAT.

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:

	15	0		
WORD 0	HDA SERIAL NUMBER (LSB)		SERIAL NUMBER IS A 31 BIT BINARY NUMBER	
WORD 1	HDA SERIAL NUMBER (MSB)			
WORD 2	YEAR	MON	DAY	CREATION DATE (YEARS SINCE 1978)
WORD 3	YEAR	MON	DAY	REVISION DATE (YEARS SINCE 1978)
WORD 4	REVISION NO			
WORD 5	UNUSED	FORMAT TYPE		FORMAT TYPE: 1= 16 BIT 0= 18 BIT
WORD 6	UNUSED			UNUSED HEADER WORDS WILL BE ZERO FILLED
WORD 7	UNUSED			
WORD 8	UNUSED			
WORD 9	NUMBER OF SSF ENTRIES			
WORD 10	CHECKSUM			SSF HEADER CHECKSUM
WORD 11	CYLINDER ADDRESS			PHYSICAL CYL. ADRS
WORD 12	TRACK	SECTOR		PHYSICAL ADDRESSES
WORD 13	ONE'S			NULL ENTRIES WILL BE ONE'S FILLED

736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792

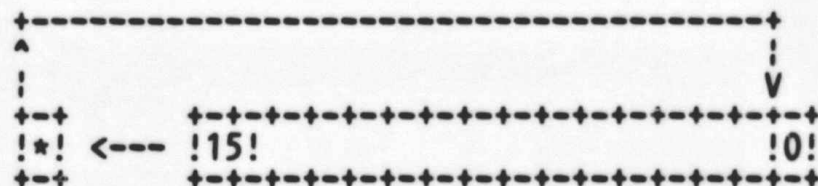


THE CHECKSUM IS CALCULATED USING THE FOLLOWING ALGORITHM. THE CHECKSUM CALCULATION IS BASED ON 16 BIT QUANTITIES (WORDS) AND IS REPEATED ONCE FOR THE SSF HEADER FIELD AND ONCE FOR THE SSF DATA FIELD.

```

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:

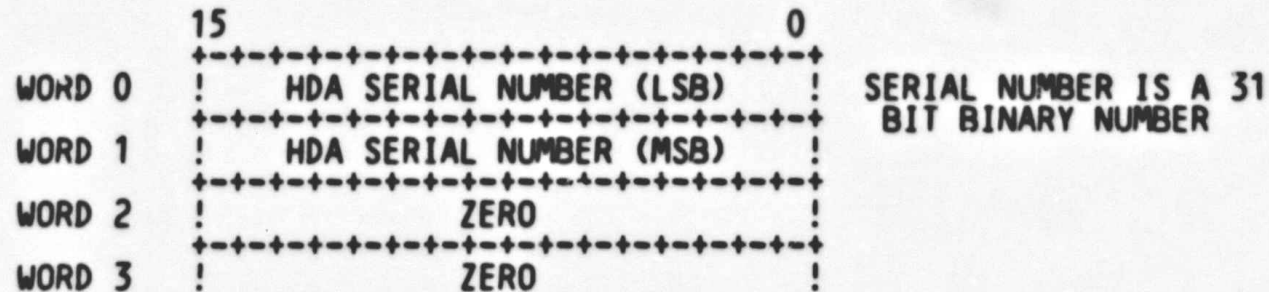


'\*' REFERS TO EITHER THE 'C' BIT FOR PDP-11 OPERATION OR BIT 16 FOR MACHINES OTHER THAN THE PDP-11. NOTE THAT THE ROTATE OCCURS AFTER THE ADDITION; THE BIT SHIFTED INTO THE LSB POSITION OF THE ACCUMULATED CHECKSUM IS THE 'CARRY OUT' FROM THE 16 BIT ADDITION OPERATION.

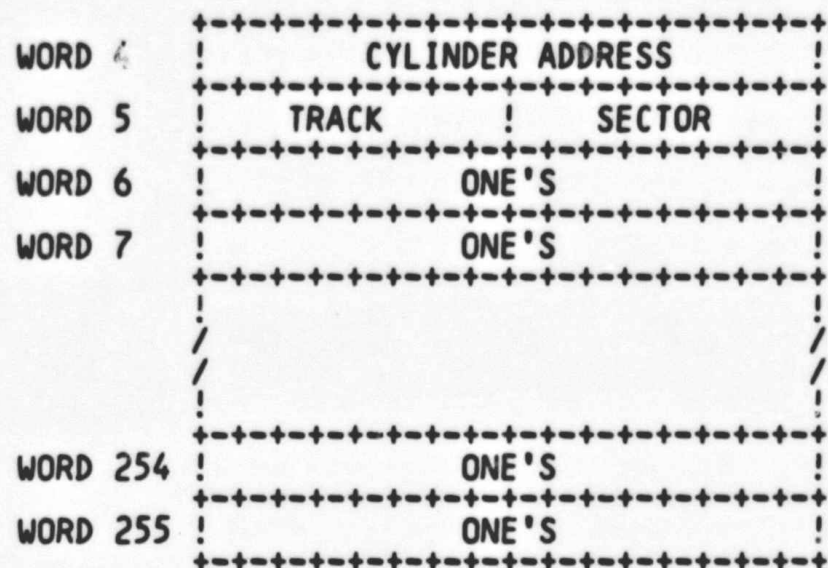
### 8.2 DEC STD 144 MANUFACTURING AND USER BAD SECTOR FILES

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.



794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818



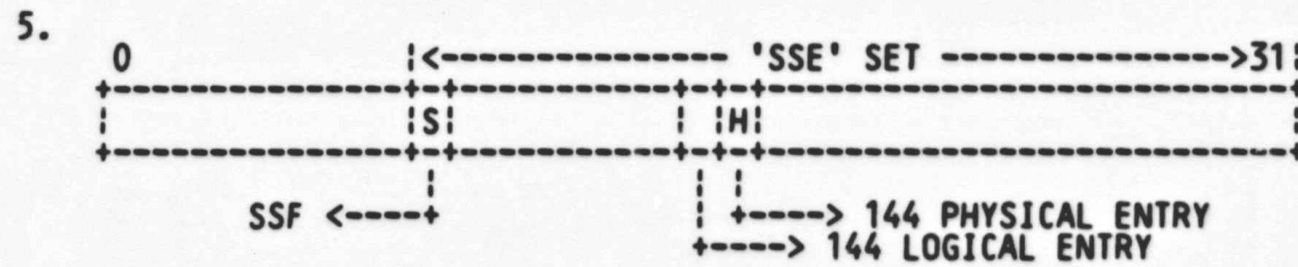
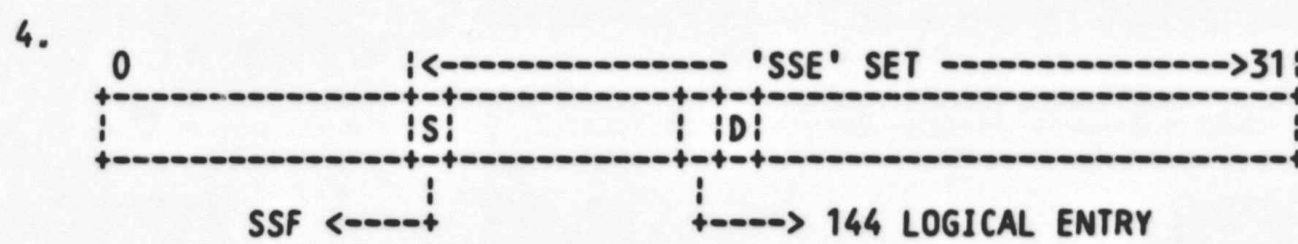
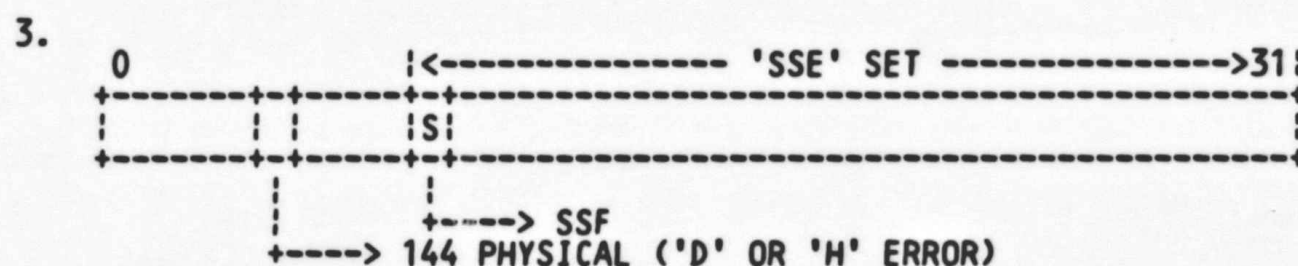
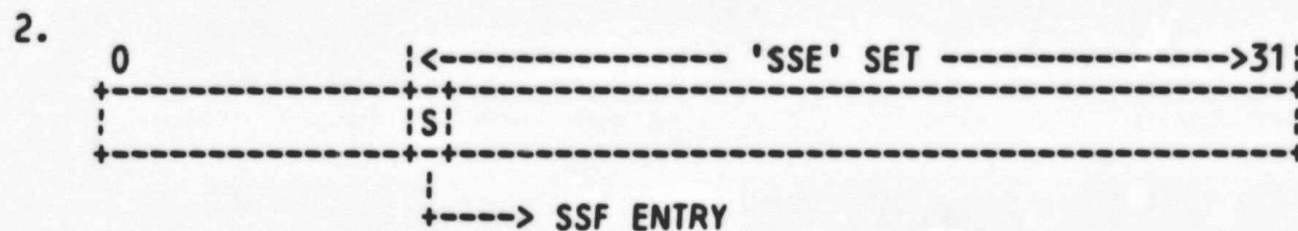
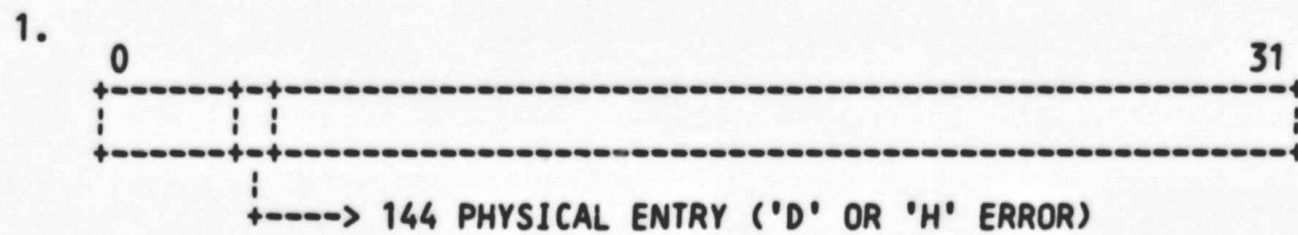
NULL ENTRIES WILL  
BE ONE'S FILLED

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.

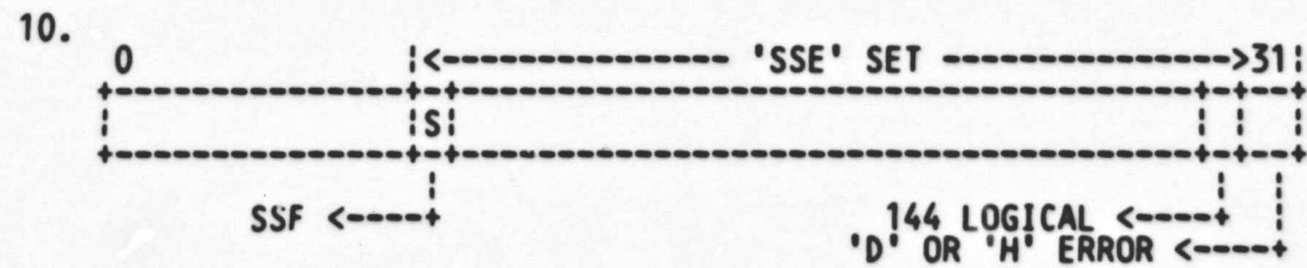
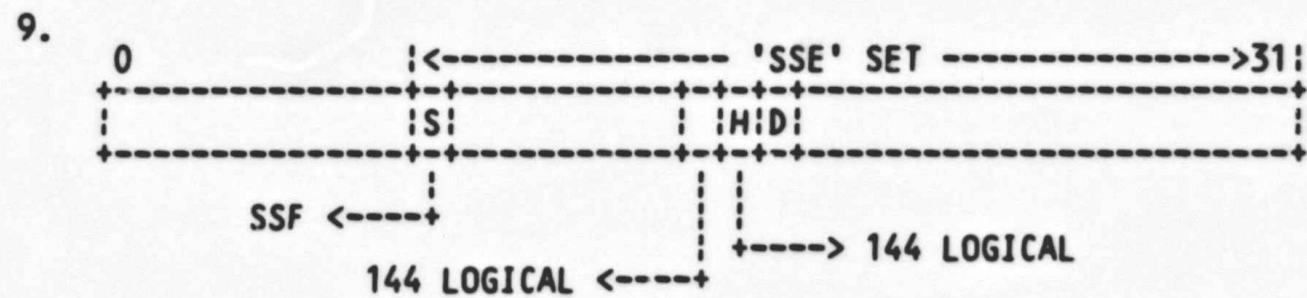
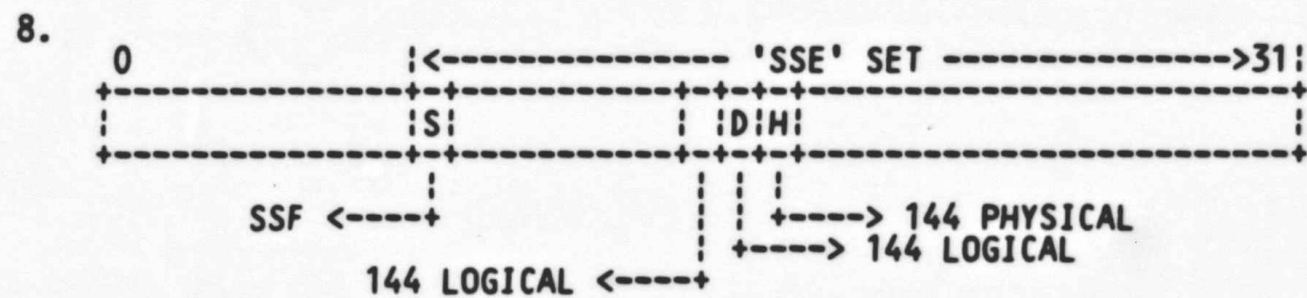
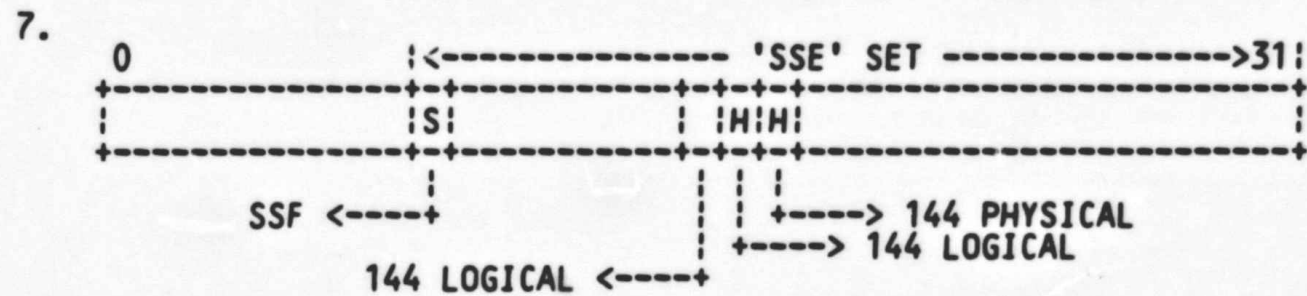
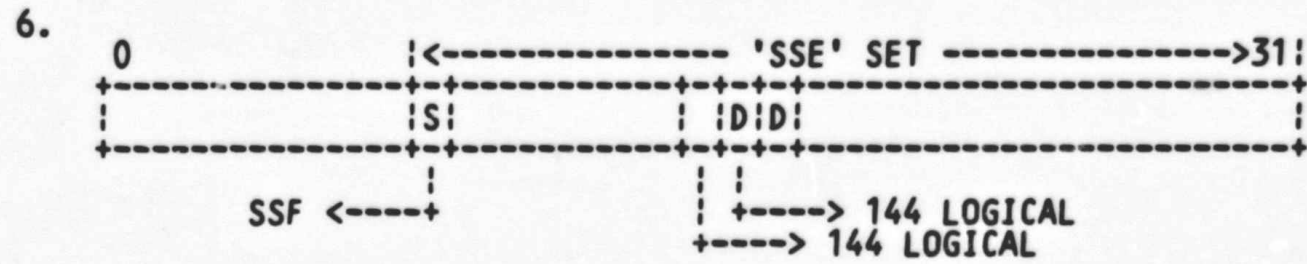


820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871

9.0 ERROR TYPE DESCRIPTIONS

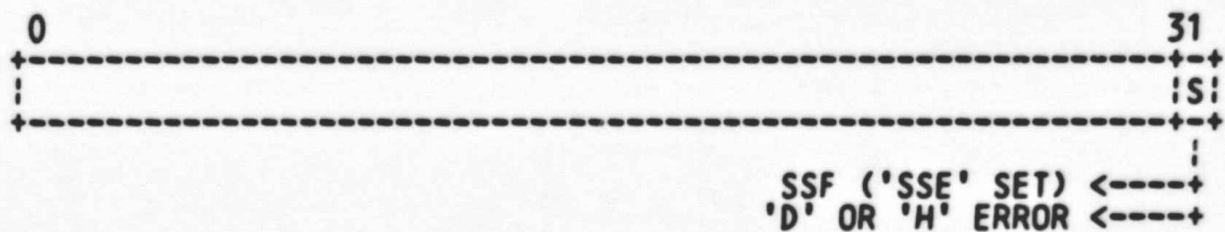


873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929



931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949

11.



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

a

. ABS. 000000 000  
000000 001  
ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 19 WORDS ( 1 PAGES)  
DYNAMIC MEMORY: 8206 WORDS ( 31 PAGES)  
ELAPSED TIME: 00:00:26  
.[300,30]CZRNJA=[300,10]BEG.DOC,CZRNJA.DOC,END.DOC

TASK NAME : CZRNJA  
 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 : 14464. WORDS  
 TASK ADDRESS LIMITS: 026000 116303  
 R-W DISK BLK LIMITS: 000002 000072 000071 00057.

\*\*\* ROOT SEGMENT: FMTROT

R/W MEM LIMITS: 026000 116303 070304 28868.  
 DISK BLK LIMITS: 000002 000072 000071 00057.

MEMORY ALLOCATION SYNOPSIS:

SECTION	TITLE	IDENT	FILE
. BLK.: (RW,I,LCL,REL,CON)	026502 020500 08512.		
	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 003040 01568.	FMTSUB 01.00	FMTSUB.OBJ;1
	034544 001026 00534.	FMTPRS 01.00	FMTPRS.OBJ;1
	035572 001104 00580.	FMTR80 01.00	FMTR80.OBJ;1
	036676 001252 00682.	FMTR8F 01.00	FMTR8F.OBJ;1
	040150 001524 00852.	FMTR8V 01.00	FMTR8V.OBJ;1
	041674 001316 00718.	FMTR8M 01.00	FMTR8M.OBJ;1
	043212 001270 00696.	FMTR8L 01.00	FMTR8L.OBJ;1
	044502 000370 00248.	FMTINT 01.00	FMTINT.OBJ;1
IMPURE: (RW,D,LCL,REL,CON)	047202 000016 00014.		
\$KSTR : (RW,D,LCL,REL,CON)	047220 000031 00025.		
	047220 000031 00025.	FMTPRS 01.00	FMTPRS.OBJ;1
\$KTAB : (RW,D,LCL,REL,CON)	047252 000016 00014.		
	047252 000016 00014.	FMTPRS 01.00	FMTPRS.OBJ;1
\$STATE: (RW,D,LCL,REL,CON)	047270 000166 00118.		
	047270 000166 00118.	FMTPRS 01.00	FMTPRS.OBJ;1
\$\$RESL: (RW,I,LCL,REL,CON)	047456 000246 00166.		
\$\$\$\$BF: (RW,D,LCL,REL,CON)	047724 046356 19694.		
	047724 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 035746-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 037644-R	\$MSG32 030714-R	\$MXTRK 027566-R	\$SRALL 033654-R	\$STVFL 035454-R
IO.WDH 004040	\$DSKWH 027014-R	\$MOVE 031744-R	\$MSG4 027662-R	\$NONSI 002416	\$SRFES 033726-R	\$SWERL 047735-R
IO.WPD 004160	\$DSTRB 042206-R	\$MSG8 031366-R	\$MSG40 030742-R	\$PARSE 034544-R	\$SRMDF 033740-R	\$SWFFE 047741-R
IO.WTD 004130	\$DVCE 027446-R	\$MSGWR 031414-R	\$MSG41 031000-R	\$PAT 027462-R	\$SRSDF 033764-R	\$SWFMT 047740-R
IQ.UMD 000004	\$DVPRM 027560-R	\$MSG1 027634-R	\$MSG42 031026-R	\$PHYDV 027606-R	\$SRSSF 034046-R	\$SWINT 047742-R
MSGER8 042041-R	\$DVUNT 047754-R	\$MSG10 030122-R	\$MSG43 031056-R	\$PRINT 027256-R	\$SRTMF 034166-R	\$SWLOG 047737-R
OPTKT2 047252-R	\$EROCT 031600-R	\$MSG11 030143-R	\$MSG44 031103-R	\$PRMPT 031754-R	\$SRTRK 033612-R	\$SWLST 047736-R
OPTST2 047300-R	\$EROUT 031536-R	\$MSG12 030167-R	\$MSG45 031141-R	\$PRNT0 027262-R	\$SRTSF 034132-R	\$SWMOD 047743-R
SECSZ 027472-R	\$FEBSF 116102-R	\$MSG13 030177-R	\$MSG5 027707-R	\$RDBSF 032322-R	\$SRUUF 034222-R	\$SYSIZ 003456
SKPFRE 047744-R	\$FLCYL 047764-R	\$MSG15 030225-R	\$MSG58 031166-R	\$RDSSF 033156-R	\$SRUDF 033752-R	\$TRK 047774-R
SPEFN 000003	\$FLRGS 040036-R	\$MSG16 030237-R	\$MSG59 031233-R	\$READ 027342-R	\$SSF 113102-R	\$TSTDA 032160-R
TRKSZ 027470-R	\$FLSEC 047770-R	\$MSG17 030246-R	\$MSG6 027737-R	\$RETRY 027466-R	\$SSFDT 001365	\$TSTDV 035062-R
UNTKT1 047252-R	\$FLTRK 047766-R	\$MSG18 030256-R	\$MSG6A 027762-R	\$REVM 047732-R	\$SSFHD 000013	\$TSTMX 032216-R
UNTST1 047270-R	\$FMTEP 026502-R	\$MSG19 030301-R	\$MSG6B 030004-R	\$RBUF 050000-R	\$SSFLB 027454-R	\$SUBSZ 027574-R
\$ALFLB 027450-R	\$FMTEX 031464-R	\$MSG2 027645-R	\$MSG7 030025-R	\$RM80 035754-R	\$SSFLN 000003	\$UCB 047724-R
\$BUF 050062-R	\$FUNC 027476-R	\$MSG20 030324-R	\$MSG70 031274-R	\$RM80F 037234-R	\$SSFSZ 027460-R	\$UDBSF 112102-R
\$BUFRX 110262-R	\$HDRFG 047762-R	\$MSG21 030351-R	\$MSG71 031307-R	\$RM80I 044570-R	\$STARS 027620-R	\$UNTID 032266-R
\$BUFX 110262-R	\$HOMSK 031710-R	\$MSG22 030403-R	\$MSG72 031322-R	\$RM80L 043574-R	\$STATS 027444-R	\$UPDMX 032120-R
\$CONPB 027524-R	\$INADD 047730-R	\$MSG23 030427-R	\$MSG73 031344-R	\$RM80M 042124-R	\$STCSR 035230-R	\$VERIF 047745-R
\$CR 027616-R	\$IOBUF 027510-R	\$MSG24 030474-R	\$MSG8 030043-R	\$RM80V 040366-R	\$STERL 035506-R	\$WLKER 027250-R
\$CVTDA 031636-R	\$IOST 027554-R	\$MSG25 030502-R	\$MSG8A 030070-R	\$RTCNT 047752-R	\$STFFE 035552-R	\$WTBSF 032730-R
\$CYL 047772-R	\$LBNH 027516-R	\$MSG26 030510-R	\$MSG9 030076-R	\$R80SF 037164-R	\$STFOR 035556-R	\$WTSSF 033464-R
\$DEVHD 003334	\$LBNL 027520-R	\$MSG27 030540-R	\$MULT 032046-R	\$R80TF 037226-R	\$STIN 035564-R	\$XFRSZ 027512-R
\$DSKIO 027020-R	\$LOGDV 027576-R	\$MSG28 030574-R	\$MVPAT 031504-R	\$SCB 047726-R	\$STLIS 035544-R	\$YESNO 047734-R
\$DSKPB 027474-R	\$MBSSZ 027572-R	\$MSG29 030626-R	\$MXCYL 027564-R	\$SEC 047776-R	\$STLOG 035540-R	
\$DSKRH 027006-R	\$MDBSF 111102-R	\$MSG30 030650-R	\$MXLBN 047756-R	\$SMBSF 033774-R	\$STUNT 034744-R	
\$DSKRT 027030-R	\$MESAG 031422-R	\$MSG31 030671-R	\$MXSEC 027570-R	\$SNAME 047746-R	\$STVEC 035344-R	

TOTAL WORK FILE REFERENCES: 46347.  
WORK FILE READS: 0.  
WORK FILE WRITES: 0.  
SIZE OF CORE POOL: 8200. WORDS (32. PAGES)  
SIZE OF WORK FILE: 3328. WORDS (13. PAGES)

ELAPSED TIME:00:00:38

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34

000000

.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\$\$,UCBDF\$

UCBDF\$

;CALL MACRO TO DEFINE UCB OFFSETS



36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92

```

: *+
: **-$FMTEP-FORMATTER ENTRY POINT
: THIS ROUTINE WILL CALL THE PARSER, SETUP THE DISK PARAMETERS, CALL THE
: ACTUAL FORMATTING ROUTINE, & CLEAN UP AFTER THE FORMAT OPERATION.
: -
$FMTEP::CLRB  $$WERL      ; RESET ERROR LIMIT (DEFAULT IS 256.)
          CLRB  $YESNO    ; RESET YES/NO RESPONSE
          CLRB  $$WLST    ; RESET BAD BLOCK FILE LIST SWITCH
          CLRB  $$WLOG    ; RESET LOGICAL LIST SWITCH
          CLRB  $$WFMT    ; RESET FORMAT ENABLE SWITCH
          CLRB  $$WFFE    ; RESET FORMAT FE ENABLE SWITCH
          CLRB  $$WINT    ; RESET MDBSF, UDBSF, SSF GENERATION SWITCH
          CLRB  $$WMOD    ; RESET THE MDBSF, SSF, & FEBSF MODIFY SWITCH
          CLRB  $$STATS+1 ; CLEAR STATUS FLAGS
          CLR   $REVM     ; CLEAR TEMPORARY STORAGE FOR REVISION NUMBER
          MOVB  #1,$VERIF ; PRESET THE VERIFICATION COUNT
          MOV   #256.,R3  ; LOCATIONS IN THE MDBSF BUFFER TO CLEAR
          MOV   #MDSF,R0  ; BUFFER POINTER
1$:      MOV   #-1,(R0)+  ; CLEAR A LOCATION
          DEC   R3        ; COUNT THE LOCATION
          BNE  1$        ; IF NE, NOT FINISHED
          MOV   #256.,R3  ; LOCATIONS TO CLEAR IN THE UDBSF BUFFER
          MOV   #UDSF,R0  ; BUFFER POINTER
2$:      MOV   #-1,(R0)+  ; CLEAR A LOCATION
          DEC   R3        ; COUNT THE LOCATION
          BNE  2$        ; IF NE, NOT FINISHED
          MOV   #64.,R3   ; LOCATIONS TO CLEAR IN THE FE BSF BUFFER
          MOV   #FEBSF,R0 ; BUFFER POINTER
3$:      MOV   #-1,(R0)+  ; CLEAR A LOCATION
          DEC   R3        ; COUNT THE LOCATION
          BNE  3$        ; IF NE, NOT FINISHED
          MOV   #256.*<$$SFLN>,R3 ; NUMBER OF SSF LOCATIONS
          MOV   #SSF,R0   ; BUFFER POINTER
4$:      MOV   #-1,(R0)+  ; CLEAR A LOCATION
          DEC   R3        ; COUNT THE LOCATION
          BNE  4$        ; IF NE, NOT FINISHED
          MOV   #FMTEP,SP ; RESET STACK POINTER
          CMP  $$SYSIZ,#40*24. ; IS THERE AT LEAST 24K?
          BHIS 5$        ; IF HIS, YES
          CRASH ; ELSE CRASH THE SYSTEM!!
5$:      CALL  $PARSE     ; PROMPT AND PARSE COMMAND LINE
          ALUN$$ #DSKLUN,$DVICE,R1 ; ASSIGN LUN TO DISK
          MOV  #MSG4,R0   ; ASSUME DEVICE NOT IN SYSTEM
          BCC  FMTPRM    ; IF CC, DEVICE IN SYSTEM
6$:      JMP  $MSGWR     ; EXIT WITH WARNING STATUS

: ATTACH THE DISK
:
FMTPRM: MOV  #IO.ATT!IQ.UMD,R3 ; ATTACH DISK FOR FORMATTING
        CALL $DSKIO
:

```

```
93          ; PRESET THE DISK PARAMETERS, RETURN THE HEADS HOME AND
94          ; CALL THE ACTUAL FORMAT ROUTINE.
95          ;
96          ;
97 000242 005067 000000G          CLR          $SEC          ; RESET DISK PARAMETERS
98 000246 005067 000000G          CLR          $TRK
99 000252 005067 000000G          CLR          $CYL
100 000256          CALL          $HOMSK          ; DO A RECALIBRATE
101 000262          CALL          $RM80          ; CALL THE FORMATTER
102 000266          CALL          $HOMSK          ; DO A RECAL
103 000272 000167 C)0000G          JMP          $FMTEX          ; EXIT GRACEFULLY
104
105          000000'          .END          $FMTEP
```

SYMBOL TABLE

CSSORE= 002000	SS\$YSZ= 001400	UC.ALG= 000200	U.CW3 000014	U2.NEC= 004000
CSSRSH= 177564	S1.BEL= 000400	UC.ATT= 000010	U.CW4 000016	U2.PRIV= 000010
CSSTTY= 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.GUE= 000040	U.KCSR= 000032	U2.VT5= 000002
DV.F11= 040000	S1.ESC= 000004	US.ABO= 000001	U.KCS6= 000034	U2.7CH= 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	SCYL = ***** 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.REC= 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	\$REVM= ***** 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

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31

.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 DIRS

33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89

```

: *+
: **-$DSKIO-ISSUE QIO TO DISK
: **-$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 (!F ENTRY IS AT $DSKIO)
:   R5 = BUFFER ADDRESS
:   ENTRY AT $DSKRT ASSUMES THAT THE DPB IS ALREADY SETUP.
:
: OUTPUTS:
:   THE QIO DIRECTIVE IS ISSUED. IF THERE WERE NO ERRORS, THE
:   CARRY BIT WILL BE CLEARED. IF THERE WAS A PARITY ERROR
:   (IE.VER) OR A BAD BLOCK ERROR (IE.BBE) CARRY WILL BE SET.
:   IF WE ARE FORMATTING A 'DB' DISK A BLOCK NUMBER VIOLATION
:   (IE.BLK) MAY HAPPEN WHEN WE ARE FORMATTING THE LAST SECTOR.
:   FOR THIS CASE CARRY WILL BE RETURNED CLEAR. THE SAME APPLIES
:   IF WE ARE RE-WRITING THE LAST TRACK OF 'DM' AND 'DR' DISKS.
:   ALSO SPECIFIC ERROR MESSAGES WILL BE PRINTED WHEN THE
:   FOLLOWING ERRORS ARE DETECTED:
:
:       1. IE.HWR
:       2. IE.DNR
:       3. IE.DAA
:       4. IE.FHE
:       5. IE.PRI
:       6. IE.OFL
:
:   ANY OTHER ERROR CODE WILL BE REPORTED AS 'UNRECOVERABLE'.
:
: NOTE: MOST FUNCTIONS ARE ISSUED WITH THE DIAGNOSTIC SUB-FUNCTION
: BIT (IQ.UMD) SET. FOR THESE FUNCTIONS THE FUNCTION WILL
: ALWAYS RETURN SUCCESS (CARRY CLEAR) AND THE USER MUST TEST
: THE PROPER BIT IN THE REGISTER BUFFER ($RBUF) TO DETERMINE
: IF THERE WAS ANY ERRORS.
:
:
: .ENABL  LSB
: $DSKWC: :MOV    #IO.WCH!IQ.UMD,R3      ; WRITE CHECK HEADER & DATA
:         BR     $DSKIO                ; GO TO THE COMMON QIO ROUTINE
:
: $DSKRH: :MOV    #IO.RDH!IQ.UMD,R3      ;READ DISK HEADER(S)
:         BR     $DSKIO                ;BR INTO COMMOM DISK ROUTINE
:
: $DSKWH: :MOV    #IO.WDH!IQ.UMD,R3      ;WRITE DISK HEADER(S)
: $DSKIO: :MOV    R3,$FUNC              ;STORE FUNCTION CODE IN DPB
:         MOV    R5,$IOBUF             ;STORE BUFFER ADDRESS IN DPB
: $DSKRT: :DIRS  #$DSKPB               ;ISSUE THE QIOWS
:         BCC   1$                    ;IF CC OK
:         CMP   #IE.UPN,$DSW          ;POOL SPACE AVAILABLE?
:         BEQ   $DSKRT                ;IF EQ NO, TRY AGAIN
:         MOV   $DSW,$IOST            ;COPY DSW INTO STATUS BLOCK
:         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         $MESAG   :PRINT MESSAGE AND EXIT
118
119
120 :+
121 : ERROR CODE AND ERROR MESSAGE TABLE. A ZERO WORD
122 : IN THE MESSAGE FIELD INDICATES THAT THE ERROR WILL
123 : BE HANDLED BY THE CALLING ROUTINE.
124 :-
125 000204 000000G 000000G   9$:   .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,          $MSG8   : END OF TABLE
134
135
136 :+
137 : **-$WAKER-SET WRITE LOCK ERROR
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 :-

```

```

147
148 000250 012700 000000G      SWLKER::MOV      #MSG9,R0      :SET WRITE LOCK MESSAGE ADDRESS
149 000254 000751              BR          8$          :EXIT WITH SEVERE ERROR STATUS
150
151      .DSABL  LSB
152
153
154      :+
155      **-$PRINT-PRINT MESSAGE ON THE USERS TI:
156      **-$PRNTO-PRINT MESSAGE ON THE USERS TI: (ALTERNATE ENTRY)
157
158      INPUTS:
159          RO = MESSAGE ADDRESS
160          R1 = CARRIAGE CONTROL (FOR $PRNTO ENTRY ONLY)
161
162      OUTPUTS:
163          MESSAGE SENT TO TI:
164      :-
165
166 000256 012701 000040      $PRINT::MOV      #40,R1      :SET 1 LINE FEED CARRIAGE CONTROL
167 000262 012705 000000G      $PRNTO::MOV      #CONPB,R5     :GET OUTPUT DPB ADDRESS
168 000266 012765 000000G 000000G      MOV          #IO.WVB,Q.IOFN(R5) :SET FUNCTION CODE
169 000274 010065 000000G      MOV          R0,Q.IOPL(R5)     :STORE MESSAGE ADDRESS IN DPB
170 000300 005065 000002G      CLR          Q.IOPL+2(R5)     :CLEAR MESSAGE LENGTH COUNTER IN DPB
171 000304 010165 000004G      MOV          R1,Q.IOPL+4(R5)   :SET CARRIAGE CONTROL
172 000310 105720      1$:  TSTB      (R0)+          :END OF MESSAGE STRING?
173 000312 001403      BEQ          2$              :IF EQ YES
174 000314 005265 000002G      INC          Q.IOPL+2(R5)     :NO, CALCULATE MESSAGE LENGTH
175 000320 000773      BR          1$              :GET NEXT CHARACTER
176
177 000322      2$:  DIR$      R5              :PRINT THE MESSAGE
178 000326 103004      BCC          3$              :IF CC OK
179 000330 022767 000000G 000000G      CMP          #IE.UPN,$DSW     :POOL SPACE AVAILABLE?
180 000336 001771      BEQ          2$              :IF EQ NO, TRY AGAIN
181 000340 000207      3$:  RETURN
182
183
184
185      :+
186      **-$READ-READ A LINE FROM THE TERMINAL
187
188      THIS SUBROUTINE WILL READ A LINE FROM THE CONSOLE
189      DEVICE.
190
191      INPUTS:
192          NONE.
193
194      OUTPUTS:
195          CARRY SET IF ANY ERROR AND
196          R0 = ERROR MESSAGE ADDRESS
197
198          CARRY CLEAR IF READ OK
199      :-
200
201 000342 012705 000000G      $READ::MOV      #CONPB,R5     :GET INPUT DPB ADDRESS
202 000346 012765 000000G 000000G      MOV          #IO.RVB,Q.IOFN(R5) :LOAD FUNCTION CODE
203 000354 012765 000000G 000000G      MOV          #SBUFEX,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

```



FMTIO - I/O ROUTINES MACRO M1113 12-JAN-82 11:01 PAGE 2-4

## 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	\$PRNT0 000262RG
IE.OFL= ***** GX	Q.IOFN= ***** GX	\$DSKWH 000014RG	\$MSG29= ***** GX	\$READ 000342RG
IE.PRI= ***** GX	Q.IOPL= ***** GX	\$DSW = ***** GX	\$MSG30= ***** GX	\$WALKER 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

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45

.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 UMDIOS

000000

UMDIOS 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)

```

47 000001 DSKEFN = 1 ;EVENT FLAG FOR DISK I/O
48 000002 CONEFN = 2 ;EVENT FLAG FOR CONSOLE I/O
49 000003 SPEFN == 3 ;EVENT FLAG FOR SPAWN DIRECTIVE
50 000002 CONLUN = 2 ;LUN FOR CONSOLE I/O
51 000003 DSKLUN == 3 ;LUN FOR DISK I/O
52 000037 DEFSEC = 31. ;DEFAULT SECTOR FOR SKIP FLAG
53 000003 $$$FLN == 3 ;BLOCK LENGTH OF SKIP SECTOR FILE
54 000013 $$$FHD == 11. ;NUMBER OF WORDS IN HEADER OF SKIP SECTOR FILE
55 001365 $$$FDT == <256.*$$$FLN>-$$$FHD ;NUMBER OF WORDS IN DATA OF SKIP SECTOR FILE
56
57 000000 000 377 $STATS:::BYTE 0,-1 ; FORMAT STATUS, MCR LINE FLAG
58 000002 104 122 $DVICE:::ASCII 'DR' ; DISK NAME (ASCII)
59 000004 000003 131715 $ALFLB:::WORD 3,131715 ; ADDRESS OF ALTERNATE DEC STD 144 TRACK
60 000010 000003 131656 $$$FLB:::WORD 3,131656 ; ADDRESS OF START OF SKIP SECTOR FILE
61 000014 000005 $$$FSZ:::WORD 5 ; NUMBER OF COPIES OF SKIP SECTOR FILE
62 000016 155555 133333 $PAT:::WORD 155555,133333 ; DISK FORMAT PATTERN
63 000022 000024 $RETRY:::WORD 20. ; NUMBER OF RE-VERIFICATION ATTEMPTS
64 000024 040200 TRKSZ:::WORD 516.*32. ; RM80 TRACK SIZE (IN BYTES)
65 000026 001004 SECSZ:::WORD 516. ; RM80 SECTOR SIZE (INCLUDING HEADER)
66
67 ;DISK I/O DIRECTIVE PARAMETER BLOCK (QIOW$ DIRECTIVE)
68
69 000030 003 014 $DSKPB:::BYTE 3,12. ; MACRO DIC, DPB SIZE
70 000032 000000 $FUNC:::WORD 0 ; FUNCTION CODE
71 000034 000003 .WORD DSKLUN ; LUN
72 000036 001 000 .BYTE DSKEFN,0 ; EVENT FLAG, PRIORITY
73 000040 000110 .WORD $IOST ; STATUS BLOCK ADDRESS
74 000042 000000 .WORD 0 ; AST ADDRESS
75 000044 000000 $IOBUF:::WORD 0 ; BUFFER ADDRESS
76 000046 000000 $XFRSZ:::WORD 0 ; TRANSFER SIZE IN BYTES
77 000050 000000 .WORD 0 ; NOT USED
78 000052 000000 $LBNH:::WORD 0 ; HIGH ORDER LOGICAL BLOCK NUMBER
79 000054 000000 $LBNL:::WORD 0 ; LOW ORDER LOGICAL BLOCK NUMBER
80 000056 000054 .WORD $RBUF ; REGISTER BUFFER ADDRESS
81
82 ;CONSOLE I/O DIRECTIVE PARAMETER BLOCK (QIOW$ DIRECTIVE)
83
84 000060 003 014 $CONPB:::BYTE 3,12. ; MACRO DIC, DPB SIZE
85 000062 000000 .WORD 0 ; FUNCTION CODE
86 000064 000002 .WORD CONLUN ; CONSOLE LUN
87 000066 002 000 .BYTE CONEFN,0 ; CONSOLE EVENT FLAG, PRIORITY ZERO
88 000070 000110 .WORD $IOST ; STATUS BLOCK ADDRESS
89 000072 000000 .WORD 0 ; AST ADDRESS
90 000074 000000 .WORD 0 ; BUFFER ADDRESS
91 000076 000000 .WORD 0 ; MESSAGE BYTE COUNT
92 000100 000000 .WORD 0 ; CARRIAGE CONTROL
93 000102 000000 000000 000000 .WORD 0,0,0 ; NOT USED
94
95 000110 000000 000000 $IOST:::WORD 0,0 ; I/O STATUS BLOCK
96
97 ;
98 ; DRIVE PARAMETER TABLE
99 ;
100
101 000114 000003 131656 $DVPRM:::WORD 3,131656 ; MAXIMUM LBN
102
103 ; THE ORDER OF THIS TABLE MUST NOT CHANGE

```

```

104
105 000120 000000 SMXCYL::WORD 0 : # OF CYLINDERS PER DISK
106 000122 000000 SMXTRK::WORD 0 : # OF TRACKS PER CYLINDER
107 000124 000000 SMXSEC::WORD 0 : # OF SECTORS PER TRACK
108 000126 000012 SMBSSZ::WORD 10. : # OF BLOCKS IN MDBSF
109 000130 000024 SUBSSZ::WORD 20. : # OF BLOCKS IN UDBSF
110
111 :+
112 : MACRO FORMAT: LBN, CYL, TRK, SEC
113 : WHERE:
114 : LBN = MAXIMUM NUMBER OF BLOCKS ON DEVICE (DOUBLE WORD)
115 : CYL = MAXIMUM NUMBER OF CYLINDERS ON DEVICE
116 : TRK = MAXIMUM NUMBER OF TRACKS ON DEVICE
117 : SEC = MAXIMUM NUMBER OF SECTORS ON DEVICE
118
119 :-
120 .MACRO DVPRM LBN, CYL, TRK, SEC
121 .WORD LBN
122 .WORD CYL
123 .BYTE SEC
124 .BYTE TRK
125 .ENDM DVPRM
126
127 000132 $LOGDV::DVPRM <3,131656>, 559., 14., 31. : LOGICAL DEVICE GEOMETRY
128 000142 $PHYDV::DVPRM <3,152700>, 561., 14., 32. : PHYSICAL DEVICE GEOMETRY
129
130 000000 .PSECT $$$SBF,D
131
132 000000 000000 $UCB::WORD 0 : ADDRESS OF CURRENT UCB
133 000002 000000 $$SCB::WORD 0 : ADDRESS OF CURRENT SCB
134 000004 000000 $INADD::WORD 0 : ADDRESS OF INTERRUPT VECTOR
135 000006 000000 $REVM::WORD 0 : TEMP. STORAGE FOR SSF REVISION NUMBER ($SSF+10)
136 000010 000 $YESNO::BYTE 0 : YES/NO CONTINUE RESPONSE
137 000011 000 $$SWERL::BYTE 0 : ERROR LIMIT
138 000012 000 $$SWLST::BYTE 0 : LIST BAD BLOCK FILE CONTENTS
139 000013 000 $$SWLOG::BYTE 0 : LOGICAL LIST SWITCH
140 000014 000 $$SWFMT::BYTE 0 : ENABLE FORMAT FLAG
141 000015 000 $$SWFFE::BYTE 0 : ENABLE FORMAT FE CYLINDERS ONLY
142 000016 000 $$SWINT::BYTE 0 : CREATE THE MDBSF, UDBSF, AND SSF
143 000017 000 $$SWMOD::BYTE 0 : MODIFY THE SSF, UDBSF, & FEBSF
144 000020 000 $SKPFRE::BYTE 0 : SKIP SECTOR FLAG
145 000021 000 $SVERIF::BYTE 0 : NUMBER OF TIMES TO VERIFY THE TRACK
146
147 .EVEN
148 000022 000000 000000 $$SNAME::WORD 0,0 : HDA SERIAL NUMBER
149 000026 000000 $SRTCNT::WORD 0 : RETRY COUNT
150 000030 000000 $SDVUNT::WORD 0 : DISK UNIT NUMBER
151 000032 000000 000000 $SMXLBN::WORD 0,0 : MAXIMUM BLOCK NUMBER
152 000036 000000 $SHDRFG::WORD 0 : HEADER DEFECT FLAG
153 000040 000000 $FCLCYL::WORD 0 : FAILING CYLINDER ADDRESS
154 000042 000000 $FLTRK::WORD 0 : " TRACK
155 000044 000000 $FLSEC::WORD 0 : " SECTOR "
156 000046 000000 $CYL::WORD 0 : CURRENT CYLINDER
157 000050 000000 $TRK::WORD 0 : CURRENT TRACK
158 000052 000000 $SEC::WORD 0 : CURRENT SECTOR
159
160 :BUFFERS AND DATA STORAGE AREA

```

```

161
162 000054          $RBUF::.BLKW 25.          ; RM80 REGISTER RETURN BUFFER
163 000136          $BUF::.BLKW 32.*258.      ; BUFFER FOR DISK I/O
164 040336          $BUFRX::.BLKW 200.        ; DIRECTIVE BUFFER
165                040336' $BUF$X=$BUFRX      ; BUFFER FOR DIRECTIVE USE
166
167                ;BAD SECTOR AND DEFECT STORAGE AREA
168
169 041156          $MDBSF::.BLKW 256.        ; BUFFER FOR 16 BIT MDBSF
170 042156          $UDBSF::.BLKW 256.        ; BUFFER FOR 16 BIT UDBSF
171 043156          $SSF::.BLKW 256.*<$SSFLN> ; BUFFER FOR 16 BIT SKIP SECTOR FILE
172 046156          $FEBSF::.BLKW 64.         ; BUFFER FOR FE CYLINDER DEFECTS
173
174                000001 .END

```

CONEFN= 000002	IO.WCH= 004110 G	\$DVPRM 000114RG	\$MXLBN 000032RG	002 \$SSFSZ 000014RG
CONLUN= 000002	IO.WCK= 004050 G	\$DVUNT 000030RG	002 \$MXSEC 000124RG	\$SSTATS 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	\$REVM 000006RG	002 \$SWLOG 000013RG 002
IO.FER= 004200 G	SKPFRE 000020RG	002 \$HDRFG 000036RG	002 \$RBUF 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 \$STRK 000050RG 002
IO.LPC= 004100 G	\$ALFLB 000004RG	\$IOST 000110RG	\$SEC 000052RG	002 \$SUBSZ 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	002 \$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

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33

.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 0C0166 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

```



```

90
91
92      :+
93      : **-$MESAG-OUTPUT MESSAGE WITH CURRENT EXIT STATUS
94      : **-$MSGWR-OUTPUT MESSAGE WITH WARNING EXIT STATUS
95      :
96      : INPUTS:
97      :      RO = ADDRESS OF "ASCIZ" MESSAGE STRING
98      :-
99 001576 112767 000000G 000000G $MSGWR::MOVB #EX$WAR,$STATS :SET WARNING EXIT STATUS
100 001604 010046 $MESAG::MOV RO,-(SP) :SAVE MESSAGE ADDRESS
101 001606 012700 000000G :MOV #SBUF,R0 :GET PRINT BUFFER ADDRESS
102 001612 012703 000016' :MOV #MSG1,R3 :GET PREFIX ADDRESS
103 001616 004767 000000G :CALL $MOVE :STORE PREFIX
104 001622 012603 :MOV (SP)+,R3 :GET BASIC MESSAGE ADDRESS
105 001624 004767 000000G :CALL $MOVE :STORE BASIC MESSAGE
106 001630 112720 000012 :MOVB #12,(R0)+ :ADD AN ADDITIONAL LINE FEED
107 001634 105010 :CLRB (R0) :TERMINATE THE BUFFER
108 001636 012700 000000G :MOV #SBUF,R0 :SET MESSAGE BUFFER ADDRESS
109 001642 004767 000000G :CALL $PRINT :PRINT THE MESSAGE
110
111 001646 012767 000000C 000000G $FMTEX::MOV #IO.DET!IQ.UMD,$FUNC :LOAD DETACH FUNCTION CODE
112 001654 :DIRS #SDSKPB :DETACH DISK (WHO CARES IF IT FAILS!!)
113 001662 000167 000000G :JMP $FMTEP :RESTART
114
115      000001      .END
    
```

CR = 000015	\$MSGR8 001550RG	\$MSG20 000506RG	\$MSG4 000044RG	\$MSG6B 000166RG
EX\$WAR= ***** GX	\$MSGWR 001576RG	\$MSG21 000533RG	\$MSG40 001124RG	\$MSG7 000207RG
IO.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	\$STARS 000002RG
\$MESAG 001604RG	\$MSG19 000463RG	\$MSG31 001053RG	\$MSG6A 001144RG	\$STATS= ***** GX
\$MOVE = ***** GX	\$MSG2 000027RG	\$MSG32 001076RG		

. ABS. 000000 000  
000006 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]FMTMSG

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30

.TITLE FMTSUB - FORMAT SUBROUTINES  
.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 21-AUG-80  
.M. LEAVITT 14-JAN-81

```

32
33
34
35
36
37
38
39
40
41
42
43 000000 016701 000000G
44 000004 006201
45 000006 006201
46 000010 012700 000000G
47 000014 016720 000000G
48 000020 016720 000002G
49 000024 005301
50 000026 100372
51 000030 000207
52
53
54
55
56
57
58
59
60
61
62
63
64 000032 012700 000000G
65 000036 004767 000176
66 000042 005002
67 000044 004767 000000G
68 000050 112720 000056
69 000054 112720 000040
70 000060 105010
71 000062 005001
72 000064 012700 000000G
73 000070 000167 000000G
74
75
76
77
78
79
80
81
82
83
84 000074 012700 000000G
85 000100 004767 000134
86 000104 105202
87 000106 004767 000000G
88 000112 112720 000040

```

```

: +
: **-$MVPAT-LOAD THE DATA BUFFER WITH THE DISK DATA PATTERN
:
: INPUTS:
:     NONE
:
: OUTPUTS:
:     '$BUF' LOADED WITH '$PAT' & '$PAT+2'
: -
:
$MVPAT: :MOV     TRKSZ,R1      ; LOAD THE DISTRIBUTION COUNTER
:         ASR     R1       ; CONVERT BYTE COUNT TO WORD COUNT
:         ASR     R1       ; CONVERT TO 2 WORD COUNT
:         MOV     #SBUF,R0 ; BUFFER POINTER
1$:      :MOV     $PAT,(R0)+  ; MOVE THE PATTERN
:         MOV     $PAT+2,(R0)+
:         DEC     R1       ; DECREMENT THE DISTRIBUTION COUNTER
:         BPL    1$       ; IF PL, MORE TO DO
:         RETURN
:
: +
: OUTPUT ERROR MESSAGE
:
: INPUTS:
:     R1 = VALUE TO CONVERT
:     R3 = ADDRESS OF BASIC MESSAGE
:
: OUTPUTS:
:     NONE.
: -
:
$EROUT: :MOV     #SBUF,RX,R0 ; GET OUTPUT BUFFER ADDRESS
:         CALL    $MOVE   ; STORE BASIC MESSAGE
:         CLR     R2      ; SUPPRESS LEADING 0'S
:         CALL    $CBDMG  ; CONVERT TO DECIMAL
ERROUT1: :MOVB   #'.',(R0)+    ; STUFF DECIMAL INDICATOR
:         MOVB   #' ' ,(R0)+ ; STUFF BLANK IN MESSAGE
:         CLRB   (R0)     ; TERMINATE STRING
:         CLR    R1      ; SET NULL CARRIAGE CONTROL
:         MOV    #SBUF,RX,R0 ; GET OUTPUT BUFFER ADDRESS
:         JMP    $PRNTO   ; PRINT MESSAGE
:
: +
: **-$EROCT-CONVERT TO OCTAL AND PRINT
:
: INPUTS:
:     R1 = VALUE TO CONVERT
:     R3 = MESSAGE ADDRESS
:
: OUTPUTS:
:     MESSAGE PRINTED ON TI:
: -
:
$EROCT: :MOV     #SBUF,RX,R0 ;
:         CALL    $MOVE   ;
:         INCB   R2      ; INCLUDE LEADING ZEROS
:         CALL    $CBOMG  ; CONVERT TO OCTAL
:         MOVB   #' ' ,(R0)+ ; STUFF BLANK IN MESSAGE

```

```

89 000116 105010          CLR      (R0)          ;TERMINATE STRING
90 000120 005001          CLR      R1            ;SET NULL CARRIAGE CONTROL
91 000122 012700 000000G  MOV      #SBUF,R0      ;
92 000126 004767 000000G  CALL     SPRNT0        ;SEND MESSAGE
93
94
95
96      :+
97      : **-$CVTDA-CONVERT DISK ADDRESS TO LOGICAL BLOCK NUMBER
98      : THIS SUBROUTINE CONVERTS A DISK ADDRESS TO A LOGICAL
99      : BLOCK NUMBER. THE FORMULA USED IS:
100     :     BLK = ($CYL * $MXTRK + $TRK) * $MXSEC + $SEC
101
102     : INPUTS:
103     :     $CYL = CYLINDER NUMBER
104     :     $TRK = TRACK NUMBER
105     :     $SEC = SECTOR NUMBER
106
107     : OUTPUTS:
108     :     $LBNH = HIGH ORDER BLOCK NUMBER
109     :     $LBNL = LOW ORDER BLOCK NUMBER
110     : -
111
112 000132 016701 000000G  $CVTDA: MOV      $CYL,R1          ; GET CYLINDER NUMBER
113 000136 016700 000000G  MOV      $MXTRK,R0        ; GET TRACKS PER CYLINDER
114 000142 004767 000174   CALL     $MULT            ; MULTIPLY
115 000146 016700 000000G  MOV      $TRK,R0         ; GET TRACK NUMBER
116 000152 060100          ADD      R1,R0           ; ADD TO PARTIAL TOTAL
117 000154 016701 000000G  MOV      $MXSEC,R1       ; GET SECTORS PER TRACK
118 000160 004767 000156   CALL     $MULT            ; MULTIPLY
119 000164 066701 000000G  ADD      $SEC,R1         ; ADD IN SECTOR
120 000170 005500          ADC      R0              ; DON'T FORGET CARRY
121 000172 010067 000000G  MOV      R0,$LBNH        ; SAVE HIGH ORDER BLOCK NUMBER
122 000176 010167 000000G  MOV      R1,$LBNL        ; SAVE LOW ORDER BLOCK NUMBER
123 000202 000207          RETURN
124
125
126      :+
127      : **-$HOMSK-ISSUE A HOME SEEK FUNCTION
128      : THIS SUBROUTINE WILL ISSUE A HOME SEEK FUNCTION.
129
130     : INPUTS:
131     :     NONE.
132
133     : OUTPUTS:
134     :     HOME SEEK ISSUED TO DISK.
135     : -
136
137 000204 005067 000000G  $HOMSK: CLR      $LBNH          ;RESET BLOCK NUMBER
138 000210 005067 000000G  CLR      $LBNL          ;
139 000214 012767 000010 000000G  MOV      #10,$XFRSZ      ;DUMMY UP BYTE COUNT
140 000222 012703 000000C  MOV      #IO.HMS!IQ.UMD,R3 ;SET HOME SEEK FUNCTION CODE
141 000226 012705 000000G  MOV      #SBUF,R5        ;SETUP DEFAULT BUFFER ADDRESS
142 000232 000167 000000G  JMP      $DSKIO         ;DO THE FUNCTION AND EXIT
143 000236 000207          1$: RETURN
144
145

```

```

146
147
148
149
150
151
152
153
154
155
156
157
158
159 000240 112320
160 000242 001376
161 000244 105740
162 000246 000207
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180 000250 012701 000044
181 000254 004767 000000G
182 000260 005067 000000G
183 000264 004767 000000G
184 000270 103422
185 000272 016703 000002G
186 000276 105063 000000G
187 000302 012700 000000G
188 000306 121027 000141
189 000312 103405
190 000314 121027 000172
191 000320 101002
192 000322 142710 000040
193 000326 105720
194 000330 001366
195 000332 005703
196 000334 000207
197 000336 000167 000000G
198
199
200
201
202
    
```

```

: +
: **-$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.
: -
$MOVE:: MOVB    (R3)+,(R0)+    ;STORE TEXT CHARACTER
        BNE     $MOVE        ;IF NE CONTINUE TO STORE
        TSTB   -(R0)         ;POINT TO NEXT BUFFER POSITION
        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
: -
    
```

```

$PRMPT::MOV    #'$,R1        ;SET CARRIAGE CONTROL
          CALL   $PRMTO      ;PROMPT THE USER
          CLR    $BUFRX     ;CLEAR USER'S RESPONSE BUFFER
          CALL   $READ       ;GET THE USERS RESPONSE
          BCS    MSG        ;IF CS ERROR
          MOV    $IOST+2,R3  ;GET BYTE COUNT
          CLRB   $BUFRX(R3)  ;TERMINATE THE LINE
          MOV    # $BUFRX,R0 ;GET INPUT BUFFER ADDRESS
1$:      CMPB   (R0),#141    ;LOWER CASE LETTER?
          BLO   2$         ;IF LO NO
          CMPB   (R0),#172  ;LOWER CASE LETTER?
          BHI   2$         ;IF HI NO
          BICB   #40,(R0)   ;YES, CONVERT TO UPPER CASE
2$:      TSTB   (R0)+       ;END OF INPUT BUFFER?
          BNE   1$         ;IF NE NO
          TST   R3         ;TEST FOR NULL LINE
          RETURN
MSG:     JMP    $MSGWR      ;EXIT WITH WARNING STATUS
    
```

```

: +
: **-$MULT-INTEGGER MULTIPLY ROUTINE
:
: INPUTS:
    
```

```

203
204
205
206
207
208
209
210
211
212
213
214 000342 010046 000021
215 000344 012746
216 000350 005000
217 000352 006000
218 000354 006001
219 000356 103002
220 000360 066600 000002
221 000364 005316
222 000366 003371
223 000370 022626
224 000372 000207

```

```

:      R0 = MULTIPLIER
:      R1 = MULTIPLICAND
:
: OUTPUTS:
:      R0 = HIGH ORDER RESULT
:      R1 = LOW ORDER RESULT
:
: ALL REGISTERS ARE PRESERVED
:
: -
$MULT:: MOV      R0, -(SP)      : SAVE MULTIPLIER FOR ADDS
:      MOV      #21, -(SP)    : SET LOOP COUNT
:      CLR      R0            : CLEAR HIGH PART
1$:     ROR      R0            : DOUBLE RIGHT SHIFT
:      ROR      R1
:      BCC      2$           : IF CC, DO NOT ADD
:      ADD      2(SP), R0     : ADD MULTIPLIER TO RESULT
2$:     DEC      (SP)         : DECREMENT THE LOOP COUNT
:      BGT      1$           : IF GT, MORE TO DO
:      CMP      (SP)+, (SP)+  : PRUNE STACK
:      RETURN
:
: +
: **-$SFCRC-COMPUTE CHECKSUM OVER SSF DATA
:
: THIS ROUTINE WILL COMPUTE THE CHECKSUM OVER THE SPECIFIED
: DATA AREA OF AN SSF COPY.
:
: INPUTS:
:      R0 = STARTING ADDRESS
:      R1 = LENGTH OF DATA AREA
:
: OUTPUTS:
:      R0 = POINTS TO CHECKSUM WORD
:      R1 = CHECKSUM
:
: -
SSFRC: MOV      #1, -(SP)      : INITIALIZE CHECKSUM
1$:     ADD      (R0)+, (SP)   : ADD DATA WORD TO CHECKSUM
:      ROL      (SP)         : ROTATE CHECKSUM
:      DEC      R1           : ALL DONE ?
:      BNE     1$           : IF NE, NO
:      MOV      (SP)+, R1     : GET CHECKSUM RESULT
:      RETURN
:
: +
: **-$SUPDMX-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:

```

```

225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241 000374 012746 000001
242 000400 062016
243 000402 006116
244 000404 005301
245 000406 001374
246 000410 012601
247 000412 000207
248
249
250
251
252
253
254
255
256
257
258
259

```

```

260      :      R3 = POINTS PAST END OF SPECIFIED TABLE
261      :      $MXLBN, $MXLBN+2, $MXCYL, $MXTRK, AND $MXSEC ARE ALL UPDATED
262      :
263      :--
264
265 000414 012367 000000G      SUPDMX: :MOV      (R3)+,$MXLBN      :SET UP MAX LBN
266 000420 012367 000002G      :MOV      (R3)+,$MXLBN+2  :
267 000424 012367 000000G      :MOV      (R3)+,$MXCYL   :SET UP MAX CYLINDER
268 000430 112367 000000G      :MOVB     (R3)+,$MXSEC   :SETUP MAX SECTOR
269 000434 112367 000000G      :MOVB     (R3)+,$MXTRK   :SETUP MAX TRACK
270 000440 166767 000000G 000002G :SUB      $MXSEC,$MXLBN+2 :DO NOT FORMAT LAST TRACK
271 000446 005667 000000G      :SBC      $MXLBN        :
272 000452 000207 000000G      :RETURN                    :
273

```

```

274      :+
275      : **-$TSTDA-TEST FOR A VALID DISK ADDRESS (BLOCK NUMBER)
276      :
277      : THIS SUBROUTINE WILL CONVERT THE DISK ADDRESS TO A LBN AND
278      : CHECK IF THE LOGICAL BLOCK NUMBER IS WITHIN RANGE.
279      :
280      : INPUTS:
281      :     $SEC, $TRK, $CYL SET TO A VALUE.
282      :
283      : OUTPUTS:
284      :     CS      IF LBN OUT OF RANGE
285      :     CC      IF LBN IN RANGE
286      :--
287

```

```

288 000454 004767 177452      $TSTDA: :CALL      $CVTDA      :CONVERT PARAMETERS TO LBN
289 000460 126767 000000G 000000G :CMPB     $LBNH,$MXLBN  :HIGH BLOCK # IN RANGE?
290 000466 101007 000000G      :BHI      2$           :IF HI NO
291 000470 103404 000000G      :BLO      1$           :IF LO YES
292 000472 026767 000000G 000002G :CMP      $LBNL,$MXLBN+2 :LOW BLOCK # IN RANGE?
293 000500 101002 000000G      :BHI      2$           :IF HI NO
294 000502 000241 000000G      1$:      CLC           :SHOW LBN OK
295 000504 000207 000000G      :RETURN                    :
296 000506 000261 000000G      2$:      SEC           :SHOW LBN BAD
297 000510 000207 000000G      :RETURN                    :
298

```

```

299      :+
300      : **-$TSTMX-TEST FOR END OF DISK AND UPDATE DISK PARAMETERS
301      :
302      : THIS SUBROUTINE WILL UPDATE THE DISK PARAMETERS AND TEST
303      : IF THE PARAMETERS WILL OVERFLOW THE DISK.
304      :
305      : INPUTS:
306      :     $CYL = CURRENT CYLINDER NUMBER
307      :     $TRK = CURRENT TRACK NUMBER
308      :     $SEC = CURRENT SECTOR NUMBER
309      :
310      : OUTPUTS:
311      :     CS      IF UPDATED PARAMETERS EXCEED DISK LIMITS
312      :     CC      IF UPDATED PARAMETERS ARE STILL VALID.
313      :     $SEC = 0 IF NORMAL MODE.
314      :--
315
316

```



```

317 000512 005067 000000G $STMX::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 MOV $DVCE,(R0)+ ;STORE DEVICE NAME
357 000576 116720 000001G MOV $DVCE+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

```

```

384                                     ; READ THE MDBSF
385
386 000642 016701 000000G 1$:  MOV    $MBSSZ,R1      ; GET NUMBER OF BLOCKS IN MDBSF
387 000646 006201                ASR    R1              ; DIVIDE BY 2
388 000650 000405                BR     3$
389 000652 062767 000002 000000G 2$:  ADD    #2,$LBNL      ; TRY NEXT USABLE MDBSF BLOCK
390 000660 005567 000000G        ADC    $LBNH
391 000664 012705 000000G 3$:  MOV    #MDBSF,R5     ; GET ADDRESS OF MDBSF BUFFER
392 000670 004767 000222        CALL   RDBSF         ; READ THE MDBSF
393 000674 103017                BCC    6$           ; IF CC OK
394 000676 012700 000000G        MOV    #MSG21,R0     ; CAN'T READ MDBSF
395 000702 005301                4$:  DEC    R1          ; MORE MDBSF BLOCKS AVAILABLE?
396 000704 001006                BNE    5$           ; IF NE YES
397 000706 004767 000000G        CALL   $SPRINT      ; PRINT THE ERROR MESSAGE
398 000712 012700 000000G        MOV    #MSG26,R0     ; UNABLE TO RECOVER THE MDBSF
399 000716 000167 000000G        JMP    $MSGWR       ; EXIT
400 000722 010146                5$:  MOV    R1,-(SP)    ; SAVE THE MDBSF COUNT
401 000724 004767 000000G        CALL   $SPRINT      ; PRINT THE MESSAGE
402 000730 012601                MOV    (SP)+,R1     ; RESTORE R1
403 000732 000747                BR     2$           ; TRY AGAIN
404 000734 004567 000176        6$:  JSR    R5,VYBSF    ; VERIFY THE MDBSF COPY
405 000740 000000G        .WORD $MDBSF       ; MDBSF BUFFER POINTER
406 000742 000000G        .WORD $MSG40       ; BAD S/N MSG POINTER
407 000744 000000G        .WORD $MSG18       ; MDBSF CORRUPT MSG POINTER
408 000746 103755                BCS    4$           ; IF CS, MDBSF CORRUPT
409 000750 004767 001506        CALL   $SRTMF      ; SORT THE MDBSF
410
411                                     ; READ THE UDBSF
412
413 000754 016767 000000G 000000G  MOV    $MXLBN,$LBNH  ; SET DISK ADDRESS OF UDBSF
414 000762 016767 000002G 000000G  MOV    $MXLBN+2,$LBNL ; ADDRESS OF THE UDBSF
415 000770 066767 000000G 000000G  ADD    $MBSSZ,$LBNL
416 000776 005567 000000G        ADC    $LBNH
417 001002 016701 000000G        MOV    $UBSSZ,R1    ; GET NUMBER OF BLOCKS IN UDBSF
418 001006 006201                ASR    R1              ; DIVIDE BY 2
419 001010 000405                BR     8$
420 001012 062767 000002 000000G 7$:  ADD    #2,$LBNL      ; TRY NEXT USABLE UDBSF BLOCK
421 001020 005567 000000G        ADC    $LBNH
422 001024 012705 000000G 8$:  MOV    #SUDBSF,R5    ; GET ADDRESS OF UDBSF BUFFER
423 001030 004767 000062        CALL   RDBSF         ; READ THE UDBSF
424 001034 103017                BCC    11$          ; IF CC OK
425 001036 012700 000000G        MOV    #MSG43,R0     ; CAN'T READ UDBSF
426 001042 005301                9$:  DEC    R1          ; MORE UDBSF BLOCKS AVAILABLE?
427 001044 001006                BNE    10$          ; IF NE, YES
428 001046 004767 000000G        CALL   $SPRINT      ; PRINT THE ERROR MESSAGE
429 001052 012700 000000G        MOV    #MSG42,R0     ; UNABLE TO RECOVER THE UDBSF
430 001056 000167 000000G        JMP    $MSGWR       ; EXIT
431 001062 010146                10$: MOV    R1,-(SP)    ; SAVE THE UDBSF COUNT
432 001064 004767 000000G        CALL   $SPRINT      ; PRINT THE MESSAGE
433 001070 012601                MOV    (SP)+,R1     ; RESTORE R1
434 001072 000747                BR     7$           ; TRY AGAIN
435 001074 004567 000036        11$: JSR    R5,VYBSF    ; VERIFY THE UDBSF COPY
436 001100 000000G        .WORD $UDBSF       ; UDBSF BUFFER POINTER
437 001102 000000G        .WORD $MSG44       ; BAD S/N MSG POINTER
438 001104 000000G        .WORD $MSG19       ; UDBSF CORRUPT MSG POINTER
439 001106 103755                BCS    9$           ; IF CS, UDBSF CORRUPT
440 001110 004767 001402        CALL   $SRTUF      ; SORT THE UDBSF

```

441 001114 000207

RETURN

442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456

```

: +
: **--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
: -
    
```

```

457 001116 012767 001000 000000G RDBSF:  MOV    #512,$XFRSZ    ; SET SIZE OF ONE SECTOR (BYTES)
458 001124 012703 000000C          MOV    #10.RLB!IQ.X,R3 ; SET READ FUNCTION (WITHOUT RETRIES)
459 001130 004767 000000G          CALL   $SDSKIO      ; READ 1 BLOCK FROM THE BSF
460 001134 000207          RETURN
    
```

461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478

```

: +
: **--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
: -
    
```

```

479 001136 012503          VYBSF:  MOV    (R5)+,R3    ; GET THE BUFFER POINTER
480 001140 012500          MOV    (R5)+,R0    ; BAD S/N MESSAGE POINTER
481 001142 105767 000000G  TSTB   $SWLST      ; LISTING THE MDBSF ?
482 001146 001010          BNE    1$          ; IF NE, YES
483 001150 021367 000000G  CMP    (R3),$SNAME ; VERIFY THE SERIAL NUMBER
484 001154 001020          BNE    4$          ; IF NE, FIRST S/N WORD NOT CORRECT
485 001156 026367 000002 000002G  CMP    2(R3),$SNAME+2 ; CHECK THE SECOND WORD
486 001164 001014          BNE    4$          ; IF NE, SERIAL NUMBER DOESN'T MATCH
487 001166 000405          BR     2$          ; CHECK THE ALIGNMENT FLAG
488 001170 005713          1$:  TST    (R3)      ; IS SERIAL NUMBER NON-ZERO?
489 001172 001003          BNE    2$          ; IF NE YES
490 001174 005763 000002  TST    2(R3)      ; MAYBE?
491 001200 001403          BEQ    3$          ; IF EQ, NO
492 001202 005763 000006  TST    6(R3)      ; TEST THE ALIGNMENT DISK FLAG
493 001206 001403          BEQ    4$          ; IF EQ, BSF OK
494 001210 012500          3$:  MOV    (R5)+,R0  ; CORRUPT BSF MESSAGE POINTER
495 001212 000261          SEC                    ; SET THE ERROR INDICATOR
496 001214 000402          BR     5$
497 001216 005725          4$:  TST    (R5)+    ; CORRECT THE RETURN
    
```

```

498 001220 000241          CLC          ; SET THE GOOD INDICATOR
499 001222 000205          RTS          R5      ; RETURN
500
501
502      :+
503      : **-$WTBSF-WRITE THE MANUFACTURER'S AND USER'S BAD SECTOR FILES
504      : THIS ROUTINE WILL WRITE THE 16 BIT MDBSF AND THE UDBSF ON THE
505      : PRIMARY DEC STD 144 TRACK (558,13) AND ON THE ALTERNATE TRACK
506      : (559,1).
507
508      : INPUTS:
509      :     NONE.
510
511      : OUTPUTS:
512      :     NONE.
513      :-
514
515 001224 004767 001232    $WTBSF::CALL  $SRTMF          ; SORT THE MDBSF
516 001230 004767 001262          CALL  $SRUF           ; SORT THE UDBSF
517 001234 012703 000000G      MOV    #$LOGDV,R3      ; LOGICAL DEVICE ADDRESS POINTER
518 001240 004767 177150          CALL  $UPDMX          ; UPDATE THE ADDRESS VALUES
519 001244 012767 001000 000000G  MOV    #512,$XFRSZ     ; SET ONE BLOCK BYTE COUNT
520 001252 016767 000000G 000000G  MOV    $MXLBN,$LBNH    ; SETUP LBN
521 001260 016767 000002G 000000G  MOV    $MXLBN+2,$LBNL  ; ...
522
523      : WRITE THE 16 BIT MDBSF
524
525 001266 012705 000000G      MOV    #$MDBSF,R5     ; SET MDBSF ADDRESS
526 001272 016701 000000G      MOV    $MBSSZ,R1      ; GET NUMBER OF BLOCKS IN MDBSF
527 001276 006201              ASR    R1                    ; DIVIDE BY TWO
528 001300 012703 000000C      MOV    #IO.WLT!IQ.X,R3 ; SET WRITE LAST TRACK FUNCTION CODE
529 001304 004767 000110          CALL  1$              ; WRITE THE MDBSF
530
531      : WRITE THE 16 BIT UDBSF
532
533 001310 012705 000000G      MOV    #$UDBSF,R5     ; UDBSF ADDRESS
534 001314 016701 000000G      MOV    $UBSSZ,R1      ; GET NUMBER OF BLOCKS IN UDBSF
535 001320 006201              ASR    R1                    ; DIVIDE BY TWO
536 001322 012703 000000C      MOV    #IO.WLT!IQ.X,R3 ; SET WRITE LAST TRACK FUNCTION CODE
537 001326 004767 000066          CALL  1$              ; WRITE THE UDBSF
538
539      : WRITE THE ALTERNATE DEC STD 144 TRACK (559,1)
540
541 001332 012703 000000G      MOV    #$PHYDV,R3     ; CHANGE TO PHYSICAL ADDRESSING
542 001336 004767 177052          CALL  $UPDMX          ; UPDATE THE MAX VALUES
543 001342 016767 000000G 000000G  MOV    $ALFLB,$LBNH    ; ALTERNATE DEC STD 144 ADDRESS
544 001350 016767 000002G 000000G  MOV    $ALFLB+2,$LBNL  ;
545
546      : WRITE THE ALTERNATE MDBSF
547
548 001356 012705 000000G      MOV    #$MDBSF,R5     ; SET MDBSF ADDRESS
549 001362 016701 000000G      MOV    $MBSSZ,R1      ; NUMBER OF BLOCKS IN THE MDBSF
550 001366 006201              ASR    R1                    ; DIVIDE BY TWO
551 001370 012703 000000C      MOV    #IO.FEW!IQ.UMD,R3 ; SET WRITE FE CYLINDER FUNCTION
552 001374 004767 000020          CALL  1$              ;
553
554      : WRITE THE ALTERNATE UDBSF

```

```

555
556 001400 012705 000000G      MOV      #\$UDBSF,R5      ; UDBSF ADDRESS
557 001404 016701 000000G      MOV      \$SUBSSZ,R1     ; UDBSF SIZE
558 001410 006201                ASR      R1              ; DIVIDE BY TWO
559 001412 012746 000000C      MOV      #IO.FEW!IQ.UMD,-(SP) ; SET WRITE FE CYLINDER FUNCTION
560 001416 000401                BR       2\$             ;
561 001420 010346                1\$:    MOV      R3,-(SP)   ; SAVE THE FUNCTION CODE
562 001422 011603                2\$:    MOV      (SP),R3    ; LOAD THE FUNCTION CODE
563 001424 004767 000000G      CALL     \$DSKIO         ; WRITE THE BLOCK
564 001430 062767 000002G 000000G  ADD     #2,\$LBNL      ; UPDATE BLOCK NUMBER
565 001436 005567 000000G      ADC     \$LBNH          ;
566 001442 005301                DEC     R1              ; FINISHED WITH THE BSF ?
567 001444 001366                BNE     2\$             ; IF NE NO
568 001446 005726                TST    (SP)+          ; CORRECT THE STACK
569 001450 000207                RETURN
570
571
572      :+
573      :*-\$RDSSF-READ RM80 SKIP SECTOR FILE
574      : THIS ROUTINE WILL READ THE SKIP SECTOR FILE FOUND ON THE FIRST TRACK
575      : OF THE FE CYLINDER AREA ON THE RM80.
576
577      : INPUTS:
578      :   $$$FLB = DISK ADDRESS OF START OF SKIPPED SECTOR FILE
579
580      : OUTPUTS:
581      :   A GOOD COPY OF THE SSF IS READ INTO SSF BUFFER
582
583      :-
584
585 001452 012703 000000G      \$RDSSF::MOV     #\$PHYDV,R3      ; PHYSICAL DEVICE ADDRESS POINTER
586 001456 004767 176732      CALL     \$UPDMX         ; UPDATE THE ADDRESS VALUES
587 001462 012767 000000C 000000G  MOV     #512.*<$$$FLN>,$XFRSZ ; SET SIZE OF ONE COPY OF SSF (BYTES)
588 001470 016767 000000G 000000G  MOV     $$$FLB,\$LBNH    ; SET DISK ADDRESS OF SSF
589 001476 016767 000002G 000000G  MOV     $$$FLB+2,\$LBNL
590 001504 016701 000000G      MOV     $$$FSZ,R1       ; GET NUMBER OF COPIES OF SKIP SECTOR FILE
591 001510 010146                MOV     R1,-(SP)       ; SAVE THE COPY COUNT
592 001512 000405                BR      2\$             ; CONTINUE
593 001514 062767 000000G 000000G  1\$:    ADD     #$$$FLN,\$LBNL ; TRY NEXT USABLE SSF COPY
594 001522 105567 000000G      ADCB   \$LBNH           ;
595 001526 012705 000000G      2\$:    MOV     #$$$SF,R5     ; GET ADDRESS OF SKIP SECTOR FILE BUFFER
596 001532 012703 000000C      MOV     #IO.FER!IQ.UMD,R3 ; SET READ FE CYLINDER FUNCTION
597 001536 004767 000000G      CALL   \$DSKIO         ; READ 1 COPY OF SSF
598 001542 103017                BCC    5\$             ; IF CC, OK
599 001544 012700 000000G      MOV     #\$MSG58,R0     ; CAN'T READ SSF
600 001550 012601                3\$:    MOV     (SP)+,R1      ; GET THE SSF COPY COUNTER
601 001552 005301                DEC    R1              ; DECREMENT THE COPY COUNTER
602 001554 001006                BNE    4\$             ; IF NE, TRY AGAIN
603 001556 004767 000000G      CALL   \$SPRINT        ; PRINT THE ERROR MESSAGE
604 001562 012700 000000G      MOV     #\$MSG41,R0     ; UNABLE TO RECOVER SSF
605 001566 000167 000000G      JMP    \$MSGWR         ; EXIT, CAN'T READ SSF OR SSF CORRUPT
606 001572 010146                4\$:    MOV     R1,-(SP)     ; SAVE R1
607 001574 004767 000000G      CALL   \$SPRINT        ; PRINT THE MESSAGE
608 001600 000745                BR     1\$             ; TRY AGAIN
609 001602 012700 000000G      5\$:    MOV     #$$$SF,R0     ; START WITH HEADER
610 001606 012701 177777G      MOV     #$$$FHD-1,R1   ; AND ITS LENGTH
611 001612 004767 176556      CALL   \$SF CRC        ; 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      #$$$FDT-1,R1  : DATA FIELD LENGTH
615 001626 004767 176542  CALL     SSFCRC        : COMPUTE CHECKSUM OVER DATA AREA
616 001632 020110          CMP      R1,(R0)      : CHECKSUM OK?
617 001634 001012          BNE      6$           : IF NE, NO
618 001636 005767 000004G  TST     $$$F+4        : CREATION DATE PRESENT
619 001642 001407          BEQ     6$           : IF EQ, NO
620 001644 005767 000006G  TST     $$$F+6        : REVISION DATE PRESENT
621 001650 001404          BEQ     6$           : IF EQ, NO
622 001652 122767 000001 000012G  CMPB   #1,$$$F+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   $$WLST        : LISTING THE BSF'S ?
628 001676 001026          BNE     10$          : IF NE, YES
629 001700 105767 000000G  TSTB   $$WMOD        : MODIFYING THE BSF'S
630 001704 001007          BNE     8$           : IF NE, YES
631 001706 016767 000000G 000000G  MOV     $$$F,$$NAME   : COPY THE SERIAL NUMBER
632 001714 016767 000002G 000002G  MOV     $$$F+2,$$NAME+2
633 001722 000414          BR      10$          :
634 001724 026767 000000G 000000G  8$:    CMP     $$$F,$$NAME  : CHECK THE SERIAL NUMBER
635 001732 001004          BNE     9$           : IF NE, DOESN'T MATCH
636 001734 026767 000002G 000002G  CMP     $$$F+2,$$NAME+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 :     $$$FLB = 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     #SPHYDV,R3   : PHYSICAL DEVICE ADDRESS POINDER
658 001770 004767 176420  CALL    $UPDMX      : UPDATE THE ADDRESS VALUES
659 001774 012700 000000G  MOV     #$$$F,R0    : GET POINTER TO SKIP SECTOR FILE
660 002000 012701 177777G  MOV     #$$$FHD-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     #$$$FDT-1,R1 : GET LENGTH OF SSF DATA AREA
664 002016 004767 176352  CALL    SSFCRC      : GO COMPUTE CHECKSUM OVER DATA
665 002022 010110          MOV     R1,(R0)    : STORE THE DATA FIELD CRC
666 002024 012767 000000C 000000G  MOV     #512.*<$$$FLN>,$XFRSZ : SET SIZE OF ONE COPY OF SSF (BYTES)
667 002032 012705 000000G  MOV     #$$$F,R5    : SET SKIP SECTOR FILE BUFFER ADDRESS
668 002036 012703 000000C  MOV     #IO.FEW!IQ.UMD,R5 : SET WRITE FE CYLINDER FUNCTION

```

```

669 002042 016767 000000G 000000G      MOV      $$$FLB,$LBNH      ;SETUP LBN
670 002050 016767 000002G 000000G      MOV      $$$FLB+2,$LBNL
671 002056 016701 000000G      MOV      $$$FSZ,R1      ;...
672 002062 004767 000000G      1$: CALL  $DSKIO      ; GET NUMBER OF COPIES OF SKIP SECTOR FILE
673 002066 062767 000000G 000000G      ADD      #$$$FLN,$LBNL  ; WRITE THE SSF
674 002074 105567 000000G      ADCB    $LBNH          ; UPDATE BLOCK NUMBER
675 002100 005301      DEC      R1            ; FINISHED ?
676 002102 001367      BNE     1$            ; 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      ; INPUTS:
686      ; R1 = ADDRESS OF EITHER BSF OR SSF
687      ; R2 = MAXIMUM ADDRESS OF THE FILE SPECIFIED IN R1

```

```

688      ; OUTPUTS:
689      ; CC = SOME SECTOR ON THE CURRENT TRACK AND CYLINDER
690      ; R1 = ADDRESS OF BSF OR SSF ENTRY
691      ; CS = SPECIFIED FILE EXHAUSTED
692      ; R1 = POINTS TO NEXT ENTRY
693
694      ;-

```

```

698 002106 022121      $SRTRK::CMP      (R1)+,(R1)+  ; POINT OF NEXT ENTRY IN SPECIFIED FILE
699 002110 021127 177777      CMP      (R1),#-1      ; FILE EXHAUSTED
700 002114 001413      BEQ     3$            ; 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      ; INPUTS:
722      ; $SEC, $TRK, AND $CYL PRESET

```

```

723      ; '$SMBSF' INPUTS:
724
725

```

```

726      :           R1 = ADDRESS OF LAST BSF ENTRY (INITIALLY $BSF+4)
727      :           R2 = LAST ADDRESS OF SPECIFIED FILE
728      :
729      : OUTPUTS:
730      : IF CC   R1 = ADDRESS OF MDBSF ENTRY.
731      :           R3 = ISOLATED BAD SECTOR FLAGS.
732      : IF CS   R1 = POINTS TO NEXT ENTRY (FILE EXHAUSTED)
733      :
734      :
735      .ENABL  LSB
736
737 002150 016746 000000G  $SRALL: :MOV    $SEC,-(SP)      : SAVE THE SECTOR ADDRESS
738 002154 004767 000100    CALL    $SRSDF      : SEE IF ENTRY IN THE SSF
739 002160 103015    BCC    2$           : IF CC, IT IS
740 002162 004767 000154    CALL    $SRSSF      : SEE IF ERROR SECTOR IS WITHIN A SKIP REGION
741 002166 103402    BCS    1$           : IF CS, NO
742 002170 105367 000000G  DEC    $SEC         : DECREMENT THE SECTOR ADDRESS
743 002174 004767 000034  1$:    CALL    $SRMDF      : SEARCH THE MDBSF
744 002200 103005    BCC    2$           : IF CC, ADDRESS PRESENT
745 002202 004767 000040    CALL    $SRUDF      : SEARCH THE UDBSF
746 002206 103002    BCC    2$           : IF CC, ADDRESS PRESENT
747 002210 004767 000006    CALL    $SRFES      : SEE IF ENTRY DUPLICATED IN THE FE AREA
748 002214 012667 000000G  2$:    MOV    (SP)+,$SEC  : RESTORE THE SECTOR ADDRESS
749 002220 000207    RETURN
750
751 002222 012701 177774G  $SRFES: :MOV    #$FBSF-4,R1   : FBSF STARTING ADDRESS
752 002226 012702 000100G  MOV    #$FBSF+64.,R2 : FBSF LENGTH
753 002232 000416    BR     3$           :
754
755 002234 012701 000004G  $SRMDF: :MOV    #$MDBSF+4,R1  : MDBSF STARTING ADDRESS
756 002240 012702 000774G  MOV    #$MDBSF+1000-4,R2 : MDBSF LENGTH
757 002244 000411    BR     3$           :
758
759 002246 012701 000004G  $SRUDF: :MOV    #$UDBSF+4,R1  : UDBSF STARTING ADDRESS
760 002252 012702 000774G  MOV    #$UDBSF+1000-4,R2 : UDBSF LENGTH
761 002256 000404    BR     3$           :
762
763 002260 012701 000000C  $SRSDF: :MOV    #$$SF+<2*$$SFHD>-4,R1 ; SSF STARTING ADDRESS (-4)
764 002264 012702 000000C  MOV    #$$SF+<1000*$$SFLEN>-2,R2 ; SSF ENDING ADDRESS
765 002270 3$:
766
767 002270 022121 177777  $SMBSF: :CMP    (R1)+,(R1)+   : POINT TO NEXT MDBSF ENTRY
768 002272 021127 177777  CMP    (R1),#-1     : MDBSF EXHAUSTED?
769 002276 001417    BEQ    5$           : IF EQ YES
770 002300 021167 000000G  CMP    (R1),%CYL    : CYLINDER NUMBERS MATCH?
771 002304 001012    BNE    4$           : IF NE NO
772 002306 126167 000003 000000G  CMPB   3(R1),%STRK  : TRACK NUMBERS MATCH?
773 002314 001006    BNE    4$           : IF NE NO
774 002316 126167 000002 000000G  CMPB   2(R1),%SEC   : SECTOR NUMBERS MATCH?
775 002324 001002    BNE    4$           : IF NE NO
776 002326 000241    CLC
777 002330 000207    RETURN
778 002332 020102 4$:    CMP    R1,R2
779 002334 001355    BNE    3$
780 002336 000261 5$:    SEC
781 002340 000207    RETURN
782      .DSABL  LSB

```



783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799

```

: +
: **-$SRSSF-SEARCH SSF FOR AN TRACK ENTRY
:
: THIS ROUTINE WILL SEARCH THE SSF FOR AN ENTRY ON THE CURRENT TRACK AND
: CYLINDER. IF AN ENTRY IS FOUND WHICH IS LESS THAN THE SECTOR ADDRESS
: IN '$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
: -
    
```

800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818

```

002342 010146
002344 012701 000000C
002350 021167 000000G
002354 001012
002356 126167 000003 000000G
002364 001006
002366 126167 000002 000000G
002374 101011
002376 000241
002400 000410
002402 020127 000000C
002406 001404
002410 022121
002412 021127 177777
002416 001354
002420 000261
002422 012601
002424 000207
    
```

```

$SRSSF: :MOV R1,-(SP) ; SAVE R1
: MOV #$$$SF+<2*$$$SFHD>,R1 ; BEGINNING ADDRESS OF SSF DATA
1$: :CMP (R1),$CYL ; CYLINDER NUMBERS MATCH
: BNE 2$ ; IF NE, NO
: :CMPB 3(R1),$TRK ; TRACK NUMBERS MATCH ?
: BNE 2$ ; IF NE, NO
: :CMPB 2(R1),$SEC ; CHECK FOR A SECTOR
: BHI 3$ ; IF HI, SSF ENTRY HIGHER THAN $SEC
: CLC ; SHOW ENTRY WAS FOUND
: BR 4$
2$: :CMP R1,#$$$SF+<1000*$$$SFLN>-2 ; SSF BUFFER END ?
: BEQ 3$ ; IF EQ, YES
: :CMP (R1)+,(R1)+ ; INCREMENT BUFFER POINTER
: :CMP (R1),#-1 ; SEE IF END OF DATA
: BNE 1$ ; IF NE, NO
3$: :SEC ; SHOW '$SEC' NOT IN SKIP REGION
4$: :MOV (SP)+,R1 ; RESTORE R1
: RETURN
    
```

819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829

```

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

830  
831  
832  
833  
834  
835  
836  
837  
838  
839

```

002426 012767 001061 000000G
002434 005067 000000G
002440 005067 000000G
002444 004767 177610
002450 010103
002452 012701 000000C
002456 000167 000070
    
```

```

$SRTSF: :MOV #561.,$CYL ; LOAD A PHONY CYLINDER ADDRESS
: CLR $TRK ; CLEAR THE TRACK ADDRESS
: CLR $SEC ; CLEAR THE SECTOR
: CALL $SRSDF ; FIND THE END OF THE DATA
: MOV R1,R3 ; COPY THE END OF DATA POINTER
: MOV #$$$SF+<2*$$$SFHD>,R1 ; SSF STARTING ADDRESS
: JMP $SORT ; SORT THE SSF
    
```

```

840      : **-$SRTMF-SORT THE MDBSF BUFFER
841      :
842      : INPUTS:
843      :     NONE
844      :
845      : OUTPUTS:
846      :     MDBSF SORTED IN ASCENDING SEQUENCE
847      :
848      : -
849
850 002462 012767 001061 000000G $SRTMF::MOV #561.,$CYL : LOAD A PHONY CYLINDER ADDRESS
851 002470 005067 000000G      CLR $TRK : CLEAR THE TRACK ADDRESS
852 002474 005067 000000G      CLR $SEC : " " SECTOR " "
853 002500 004767 177530      CALL $SRMDF : SEARCH THE MDBSF
854 002504 010103      MOV R1,R3 : COPY THE END OF DATA POINTER
855 002506 012701 000010G      MOV #MDBSF+10,R1 : MDBSF STARTING ADDRESS
856 002512 000167 000034      JMP $SORT : SORT THE MDBSF
857
858
859

```

```

860      : +
861      : **-$SRUF-SORT THE UDBSF BUFFER
862      :
863      : INPUTS:
864      :     NONE
865      :
866      : OUTPUTS:
867      :     UDBSF SORTED IN ASCENDING SEQUENCE
868      :
869      : -
870
870 002516 012767 001061 000000G $SRUF::MOV #561.,$CYL : LOAD A PHONY CYLINDER ADDRESS
871 002524 005067 000000G      CLR $TRK : CLEAR THE TRACK ADDRESS
872 002530 005067 000000G      CLR $SEC : " " SECTOR " "
873 002534 004767 177506      CALL $SRUDF : SEARCH THE UDBSF
874 002540 010103      MOV R1,R3 : COPY THE END OF DATA POINTER
875 002542 012701 000010G      MOV #UDBSF+10,R1 : UDBSF STARTING ADDRESS
876 002546 000167 000000      JMP $SORT : SORT THE UDBSF
877
878
879

```

```

880      : +
881      : **-$SORT-SORT THE INPUT TABLE
882      : THIS ROUTINE SORTS THE INPUT TABLE IN ASCENDING ORDER
883      : BASED ON DISK ADDRESSES.
884      :
885      : INPUTS:
886      :     R1 = TABLE START ADDRESS
887      :     R3 = TABLE UPPER LIMIT ADDRESS
888      :
889      : OUTPUTS:
890      :     SORTED TABLE
891      :
892      : -
893
892 002552 010302      $SORT: MOV R3,R2 : R2 = CURRENT ENTRY POINTER
893 002554 160102      SUB R1,R2 : TEST FOR AT LEAST
894 002556 022702 000004      CMP #4,R2 : TWO ENTRIES IN TABLE
895 002562 103042      BHIS 4$ : IF NOT(HIS), NO NEED TO SORT
896 002564 010102      MOV R1,R2 : INIT R2

```

```

897 002566 022222          CMP      (R2)+,(R2)+      :
898 002570 010367 000076  MOV      R3,TEMP        : FORM AND SAVE
899 002574 162767 000004 000070  SUB      #4,TEMP         : UPPER LIMIT ON R1
900 002602 021112          1$:      CMP      (R1),(R2)       : CYL I = CYL I+1?
901 002604 002417          BLT      3$              : IF LT YES
902 002606 001004          BNE      2$              : IF NE CYL I > CYL I+1
903 002610 026162 000002 000002  CMP      2(R1),2(R2)     : CYL I = CYL I+1...TEST TRK/SEC
904 002616 003412          BLE      3$              : IF LE NO SWAP NECESSARY
905 002620 011146          2$:      MOV      (R1),-(SP)     : SWAP ENTRIES
906 002622 016146 000002          MOV      2(R1),-(SP)     :
907 002626 011211          MOV      (R2),(R1)       :
908 002630 016261 000002 000002  MOV      2(R2),2(R1)     :
909 002636 012662 000002          MOV      (SP)+,2(R2)    :
910 002642 012612          MOV      (SP)+,(R2)     :
911 002644 022222          3$:      CMP      (R2)+,(R2)+     : UPDATE R2
912 002646 020203          CMP      R2,R3          : THIS PASS DONE?
913 002650 002754          BLT      1$              : IF LT NO
914 002652 022121          CMP      (R1)+,(R1)+    : UPDATE R1
915 002654 020167 000012          CMP      R1,TEMP        : ALL DONE?
916 002660 002003          BGE      4$              : IF GE YES
917 002662 010102          MOV      R1,R2          : RESET R2
918 002664 022222          CMP      (R2)+,(R2)+    :
919 002666 000745          BR       1$              :
920 002670 000207          4$:      RETURN          :
921
922 002672 000000          TEMP:  .WORD 0          : SORT WORK LOCATION
923
924
925          :+
926          : **-GENHDR-GENERATE HEADER WORD PATTERN
927          : THIS SUBROUTINE WILL GENERATE THE 4-WORD HEADER PATTERN.
928          :
929          : INPUTS:
930          : R0 = BUFFER ADDRESS
931          :
932          : OUTPTUS:
933          : R0 = UPDATED BUFFER ADDRESS
934          : -
935
936 002674 016710 000000G  GENHDR: :MOV      $CYL,(R0)      : SET CYLINDER NUMBER
937 002700 116760 000000G 000002  :MOVB     $SEC,2(R0)      : SET SECTOR NUMBER
938 002706 116760 000000G 000003  :MOVB     $TRK,3(R0)     : SET TRACK NUMBER
939 002714 052710 150000          :BIS      #150000,(R0)   : SET GOOD SECTOR AND 16-BIT FORMAT FLAGS
940 002720 116746 000000G          :MOVB     $SEC,-(SP)     : SAVE THE CURRENT SECTOR ADDRESS
941 002724 105767 000000G          :TSTB     SKPFRE         : LOOK AT THE SKIP SECTOR FLAG
942 002730 001405          :BEQ      1$              : IF EQ, NO SKIP SECTOR ON THIS TRACK
943 002732 052710 020000          :BIS      #20000,(R0)   : SET SKIP SECTOR FLAG IN HEADER (BAD)
944 002736 105367 000000G          :DECB     $SEC           : DECREMENT THE SECTOR ADDRESS
945 002742 000410          :BR       2$              :
946
947          : SEARCH THE SKIP SECTOR FILE FOR A SECTOR ON THIS TRACK THE SAME AS
948          : '$SEC'. IF A SECTOR IS FOUND, THEN SET THE SKIP SECTOR FLAG IN THE
949          : REMAINING HEADERS AND NOTE THAT A SKIPPED SECTOR WAS FOUND.
950
951 002744 004767 177310          1$:      CALL     $SRSDF         : SEARCH THE SKIP SECTOR FILE
952 002750 103405          :BCS     2$              : IF CS NO SECTOR THE SAME
953 002752 052710 020000          :BIS     #20000,(R0)    : SET SKIP SECTOR FLAG (BAD)

```

```

954 002756 105267 000000G          INCB      SKPFRE      : INC SKIPPED SECTOR FLAG
955 002762 000421                   BR        5$          : EXIT
956
957                               : SEARCH THE MDBSF, UDBSF, & THE FE BSF BUFFER FOR A MATCH. IF ONE IS
958                               : FOUND, SET THE BAD SECTOR BITS IN THE APPROPRIATE HEADER.
959
960 002764 004767 177244          2$:  CALL      $SRMDF      : SEARCH THE MDBSF
961 002770 103403                   BCS      3$          : IF CS, MDBSF EXHAUSTED
962 002772 042710 100000          BIC      #100000,(R0) : CLEAR THE MF BIT (BAD)
963 002776 000413                   BR        5$
964
965 003000 004767 177242          3$:  CALL      $SRUDF      : SEARCH THE UDBSF
966 003004 103403                   BCS      4$          : IF CS, UDBSF EXHAUSTED
967 003006 042710 040000          BIC      #40000,(R0)  : CLEAR THE UF BIT (BAD)
968 003012 000405                   BR        5$
969
970 003014 004767 177202          4$:  CALL      $SRFES      : SEARCH THE FE BSF
971 003020 103402                   BCS      5$          : IF CS, FE BSF EXHAUSTED
972 003022 042710 140000          BIC      #140000,(R0) : CLEAR BOTH THE UF & MF BITS (BAD)
973 003026 112667 000000G          5$:  MOVB      (SP)+,$SEC  : RESTORE THE SECTOR ADDRESS
974 003032 066700 000000G          ADD      SECSZ,R0     : UPDATE BUFFER ADDRESS
975 003036 000207                   RETURN
976
977                               .END
000001

```

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

. ABS. 000000 000  
003040 001

ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 908 WORDS ( 4 PAGES)  
DYNAMIC MEMORY: 7942 WORDS ( 30 PAGES)  
ELAPSED TIME: 00:00:32  
[300,20]FMTSUB,[300,30]FMTSUB=[300,10]FMTSUB

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32

000000

.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\$, STATES\$, TRANS  
.MCALL DEVDFS  
DEVDFS

```

34
35      :+
36      **-$PARSE-PARSE COMMAND LINE
37      :
38      THIS ROUTINE WILL GET THE COMMAND LINE AND PARSE THE
39      DEVICE NAME AND SWITCHES.
40      :
41      INPUTS:
42      NONE.
43      :
44      OUTPUTS:
45      R1 = DEVICE UNIT NUMBER
46      EXIT IS TO '$MESAG' IF ERROR.
47      EXIT IS TO '$FMTEX' COMMAND LINE TERMINATED WITH 'EOF'.
48      :-
49 000000 005227 177777 $PARSE::INC # -1 ; FIRST TIME THRU HERE ?
50 000004 001404 BEQ 1$ ; BR IF YES
51 000006 012700 000000G MOV #SSTARS,R0 ; GET ADDRESS OF MESSAGE
52 000012 004767 000000G CALL $SPRINT ; PRINT '*****'<LF>
53 000016 012700 000000G 1$: MOV #MSG70,R0 ; GET PROMPT MESSAGE ADDRESS
54 000022 004767 000000G CALL $PRMPT ; PROMPT USER AND GET RESPONSE
55 000026 001006 BNE 2$ ; IF NE COMMAND LINE ENTERED
56 000030 122767 000000C 000000G CMPB #IE.EOF&377,$IOST ; END OF INPUT?
57 000036 001367 BNE 1$ ; IF NE NO
58 000040 000167 000000G JMP $FMTEX ; EXIT GRACEFULLY
59
60 ; SETUP TO CALL THE PARSER
61
62 000044 005001 2$: CLR R1 ; CLEAR OPTIONS WORD
63 000046 012702 000000' MOV #UNTKT1,R2 ; GET KEYWORD TABLE ADDRESS
64 000052 012704 000000G MOV #SBUFRX,R4 ; GET ADDRESS OF COMMAND LINE
65 000056 012705 000000' MOV #UNIT,R5 ; GET STARTING STATE ADDRESS
66 000062 012767 177777 000000G MOV #-1,$DVUNT ; RESET DEVICE NUMBER
67 000070 004767 000000G CALL .TPARS ; GO PARSE THE INPUT LINE
68 000074 103435 BCS SYNTAX ; IF CS SYNTAX ERROR
69 000076 005767 000000G TST $DVUNT ; NULL PARSE?
70 000102 100745 BMI 1$ ; IF MI, YES
71 000104 012700 000000G 3$: MOV #MSG71,R0 ; GET PROMPT MESSAGE ADDRESS
72 000110 004767 000000G CALL $PRMPT ; PROMPT USER AND GET RESPONSE
73 000114 001006 BNE 4$ ; IF NE COMMAND LINE ENTERED
74 000116 122767 000000C 000000G CMPB #IE.EOF&377,$IOST ; END OF INPUT?
75 000124 001002 BNE 4$ ; IF NE NO
76 000126 000167 000000G JMP $FMTEX ; EXIT GRACEFULLY
77 000132 005001 4$: CLR R1 ; CLEAR OPTIONS WORD
78 000134 012702 000000' MOV #OPTKT2,R2 ; GET KEYWORD TABLE ADDRESS
79 000140 012704 000000G MOV #SBUFRX,R4 ; GET ADDRESS OF COMMAND LINE
80 000144 012705 000010' MOV #OPTION,R5 ; GET STARTING STATE ADDRESS
81 000150 004767 000000G CALL .TPARS ; GO PARSE THE INPUT LINE
82 000154 103405 BCS SYNTAX ; IF CS SYNTAX ERROR
83 000156 004767 000134 CALL $STSDV ; MAKE A FINAL DEVICE TEST
84 000162 016701 000000G MOV $DVUNT,R1 ; SET SPECIFIED UNIT NUMBER
85 000166 000207 RETURN ; ALL IS OK
86
87 000170 012700 000000G SYNTAX: MOV #MSG2,R0 ; SYNTAX ERROR
88 000174 000167 000000G JMP $MSGWR ; EXIT WITH WARNING STATUS
89
90

```

```

91          ;+
92          ; STATE TABLE FOR 'TPARS' PARSER
93          ;-
94
95          .NLIST MEB
96
97 000200          ISTAT$ UNTST1,UNTKT1
98
99          ; READ DEVICE AND UNIT NUMBER
100
101 000200          STATES$ UNIT
102 000200          TRANS$ $EOS,$EXIT
103 000200          TRANS$ $NUMBR,, $STUNT
104
105 000200          STATES$ ; TERMINATING STATE
106
107
108 000200          ISTAT$ OPTST2,OPTKT2
109
110          ; PROCESS SWITCH OPTIONS
111
112 000200          STATES$ OPTION
113 000200          TRANS$ $EOS,$EXIT
114 000200          TRANS$ 'CSR',CSR
115 000200          TRANS$ 'VEC',VEC
116 000200          TRANS$ 'VFL',VFL
117 000200          TRANS$ 'ERL',ERL
118 000200          TRANS$ 'LI',LI
119 000200          TRANS$ 'FO',FO
120 000200          TRANS$ 'IN',IN
121
122          ; SET NEW CSR ADDRESS
123
124 000200          STATES$ CSR
125 000200          TRANS$ '='
126
127 000200          STATES$
128 000200          TRANS$ $NUMBR,SWITCH,$STCSR
129
130          ; SET NEW VECTOR ADDRESS
131
132 000200          STATES$ VEC
133 000200          TRANS$ '='
134
135 000200          STATES$
136 000200          TRANS$ $NUMBR,SWITCH,$STVEC
137
138          ; SET VERIFY LIMIT (NUMBER OF TIMES TO VERIFY A SECTOR)
139
140 000200          STATES$ VFL
141 000200          TRANS$ '='
142
143 000200          STATES$
144 000200          TRANS$ $DNUMB,SWITCH,$STVFL
145
146          ; SET ERROR LIMIT
147

```



```

148 000200          STATES ERL
149 000200          TRANS '='
150
151 000200          STATES
152 000200          TRANS SDNUMB,SWITCH,$STERL
153
154          ; SET BAD BLOCK FILE/SKIP SECTOR FILE LIST
155
156 000200          STATES LI
157 000200          TRANS ':,LOG
158 000200          TRANS $LAMDA,SWITCH,$STLIS
159
160 000200          STATES LOG
161 000200          TRANS 'L,SWITCH,$STLOG
162
163          ; SET FORMAT ENABLE
164
165 000200          STATES FO
166 000200          TRANS ':,FE
167 000200          TRANS $LAMDA,SWITCH,$STFOR
168
169 000200          STATES FE
170 000200          TRANS 'F,SWITCH,$STFFE
171
172          ; SET BSF/SSF INITIALIZATION
173
174 000200          STATES IN
175 000200          TRANS $LAMDA,SWITCH,$STIN
176
177          ; CHECK FOR ANOTHER OPTION
178
179 000200          STATES SWITCH
180 000200          TRANS $EOS,$EXIT
181 000200          TRANS '/,OPTION
182
183 000200          STATES          ; NULL (TERMINATING) STATE
184
185          .ENABL LSB
186
187          ;
188          ; DEVICE UNIT NUMBER
189          ;
190
191 000200 016767 000000G 000000G $STUNT::MOV .PNUMB,$DVUNT ; STORE UNIT NUMBER
192 000206 026727 000000G 000007 CMP $DVUNT,#7 ; IS DRIVE NUMBER WITHIN RANGE ?
193 000214 101402 BLOS 1$ ; IF LOS, YES
194 000216 000167 177746 JMP SYNTAX ; SYNTAX ERROR
195
196          ;
197          ; DETERMINE DEVICE VALIDITY
198          ;
199
200 000222 016702 000000G 1$: MOV $DEVHD,R2 ; ADDRESS OF 1ST DCB IN SYSTEM
201 000226 026762 000000G 000004 2$: CMP $DVICE,D.NAM(R2) ; IS THIS THE RIGHT DCB?
202 000234 001407 BEQ 4$ ; IF EQ YES
203 000236 016202 000000 MOV D.LNK(R2),R2 ; POINT TO NEXT DCB
204 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 $MESAG ;
207 000254 116762 000000G 000006 4$: MOV $DVUNT,D.UNIT(R2) ; SET UNIT NUMBER RANGE
208 000262 116762 000000G 000007 MOV $DVUNT,D.UNIT+1(R2)
209 000270 016201 000002 MOV D.UCB(R2),R1 ; GET UCB ADDRESS
210 000274 116761 000000G 000006 MOV $DVUNT,U.UNIT(R1) ; SAVE SPECIFIED UNIT NUMBER
211 000302 010167 000000G MOV R1,$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 $STSDV::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 MOV $U.UNIT(R1),10(R2) ; SELECT UNIT
231 000376 016200 000026 MOV 26(R2),R0 ; GET DRIVE TYPE REGISTER
232 000402 032762 010000 000010 BIT #10000,10(R2) ; SEE IF DRIVE PRESENT
233 000410 001016 BNE 9$ ; 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 0000C4 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          : SET NEW VECTOR ADDRESS
278          :
279          :
280          :
281 000600 016700 000000G   $STVEC: MOV .PNUMB,R0  : FETCH NEW VECTOR ADDRESS
282 000604 020027 001000    CMP R0,#1000    : IS THE VECTOR IN A REASONABLE PLACE?
283 000610 103006          BHS 2$          : 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          : SET VERIFY COUNT
305          :
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          : SET ERROR LIMIT
317          :
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

```



\$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

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59

.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

37	000000	012	105	116	MSG1R8: .ASCIZ <LF>/ENTER DATE (DD-MMM-YY): /
38	000032	110	104	101	MSG2R8: .ASCIZ /HDA SERIAL NUMBER: /
39					.EVEN

: DATE FIELDS

43	000056	112	101	116	DAT: .ASCIZ /JAN/
44	000062	106	105	102	.ASCIZ /FEB/
45	000066	115	101	122	.ASCIZ /MAR/
46	000072	101	120	122	.ASCIZ /APR/
47	000076	115	101	131	.ASCIZ /MAY/
48	000102	112	125	116	.ASCIZ /JUN/
49	000106	112	125	114	.ASCIZ /JUL/
50	000112	101	125	107	.ASCIZ /AUG/
51	000116	123	105	120	.ASCIZ /SEP/
52	000122	117	103	124	.ASCIZ /OCT/
53	000126	116	117	126	.ASCIZ /NOV/
54	000132	104	105	103	.ASCIZ /DEC/
55	000136	000000			.WORD 0

.EVEN

: VALID LIMITS FOR DATE FIELDS

60 000140 000001 000014  
61 000144 000001 000037  
62 000150 000000 000143  
63 000154  
64  
65 000116  
66

DATLMT: .WORD 1,12.  
.WORD 1,31.  
.WORD 0,99.  
DPARS:: .BLKW 3  
BASYR == 78.

: FOR MONTH  
: FOR DAY  
: FOR YEAR  
: STORAGE FOR DATE CONVERSION  
: BASE YEAR = 1978



```

68
69
70
71
72
73
74
75
76
77
78
79
80
81 000162 105767 000000G $RM80:: TSTB $SWINT : SEE IF SSF/BSF INITIALIZATION MODE
82 000166 001022 BNE 1$ : IF NE, YES
83 000170 004767 000000G CALL $RDSSF : READ SKIP SECTOR FILE
84 000174 004767 000000G CALL $RDBSF : READ BAD SECTOR FILES
85 000200 016767 000010G 000000G MOV $$$F+10,$REVN : GET CURRENT REVISION NUMBER
86 000206 005767 000310 TST 8$ : HAS DATE BEEN ENTERED AT LEAST ONCE ?
87 000212 001010 BNE 1$ : BR IF YES
88 000214 012700 000000' MOV #MSG1R8,R0 : 'ENTER DATE (DD-$$$-YY): '
89 000220 004767 000000G CALL $PRMPT : PROMPT THE USER - USER'S RESPONSE MAY BE EITHER:
90 : 'MM/DD/YY' OR 'DD-$$$-YY'
91 000224 004767 000274 CALL DATE : CONVERT USER'S ENTRY TO A BINARY DATE
92 000230 005267 000266 INC 8$ : INDICATE THAT DATE HAS BEEN ENTERED ONCE
93
94 000234 105767 000000G 1$: TSTB $SWLST : DISPLAY THE DEC STD 144 AND SSF FILES ?
95 000240 001405 BEQ 2$ : IF EQ, NO
96 000242 004767 000000G CALL $RM80L : DISPLAY THE FILES
97 000246 105767 000000G TSTB $SWFMT : FORMAT THE HDA ALSO ?
98 000252 001522 BEQ 7$ : IF EQ, NO
99
100 : INFORM THE OPERATOR THAT DATA WILL BE DESTROYED
101
102 000254 012703 000000G 2$: MOV #MSG23,R3 : WARNING MESSAGE ADDRESS
103 000260 004767 000000G CALL $UNTID : STORE MESSAGE AND DEVICE/UNIT
104 000264 012703 000000G MOV #MSG24,R3 : ADDRSS OF MESSAGE SUFFIX
105 000270 004767 000000G CALL $MOVE : MOVE THE SUFFIX
106 000274 012700 000144G MOV #SBUF+100.,R0 : PRINT WARNING MESSAGE
107 000300 004767 000000G CALL $PRINT :
108 000304 012700 000000G MOV #MSG31,R0 : GET CONFIRMATION ADDRESS
109 000310 004767 000000G CALL $PRMPT : PROMPT AND GET OPERATOR'S RESPONSE
110 000314 001501 BEQ 7$ : IF EQ, RESPONSE WAS 'NO'
111 000316 122767 000131 000000G CMPB #'Y,$BUFRX : SEE IF RESPONSE WAS 'YES'
112 000324 001075 BNE 7$ : IF NE, RESPONSE WAS 'NO'
113 000326 105767 000000G TSTB $SWINT : SEE IF SSF/BSF INITIALIZATION MODE
114 000332 001013 BNE 3$ : IF NE, YES
115 000334 012700 000000G MOV #MSGR8,R0 : GET UPDATE MODE MESSAGE
116 000340 004767 000000G CALL $PRMPT : PROMPT AND GET RESPONSE
117 000344 001451 BEQ 6$ : IF EQ, NO
118 000346 122767 000131 000000G CMPB #'Y,$BUFRX : WAS RESPONSE 'Y'?
119 000354 001045 BNE 6$ : IF NE, NO
120 000356 105267 000000G INCB $SWMOD : SET THE MODIFY FLAG
121 000362 012700 000000' 3$: MOV #MSG1R8,R0 : 'ENTER DATE (DD-$$$-YY): '
122 000366 004767 000000G CALL $PRMPT : PROMPT THE USER - USER'S RESPONSE MAY BE EITHER:
123 : 'MM/DD/YY' OR 'DD-$$$-YY'
124 000372 004767 000126 CALL DATE : CONVERT USER'S ENTRY TO A BINARY DATE

```

```

125 000376 005267 000120          INC      8$          : INDICATE THAT DATE HAS BEEN ENTERED ONCE
126                                     :
127 000402 012700 000032'        4$:  MOV      #MSG2R8,R0      : 'HDA SERIAL NUMBER: '
128 000406 004767 000000G        CALL     $PRMPT          : PROMPT OPERATOR
129 000412 010304                   MOV      R3,R4          : NUMBER OF BYTES ENTERED TO R4
130 000414 012705 000000G        MOV      #SBUFEX,R5     : INPUT BUFFER ADDRESS
131 000420 012703 000000G        MOV      #SSNAME,R3     : OUTPUT BUFFER ADDRESS
132 000424 004767 000000G        CALL     .DD2CT         : CONVERT THE S/N TO BINARY
133 000430 103764                   BCS      4$             : IF CS, BAD S/N ENTRY
134 000432 016746 000000G        MOV      $SNAME,-(SP)   : CORRECT THE SERIAL NUMBER ORDER
135 000436 016767 000002G 000000G  MOV      $SNAME+2,$SNAME : ...
136 000444 012667 000002G        MOV      (SP)+,$SNAME+2 : ...
137 000450 105767 000000G        TSTB    $SWINT         : INITIALIZE THE BAD SECTOR FILES ?
138 000454 001403                   BEQ      5$             : IF EQ, NO
139 000456 004767 000000G        CALL     $RM80I        : INIT THE DEC STD 144 AND SKIP SECTOR FILES
140 000462 000416                   BR       7$             : EXIT
141                                     :
142 000464 004767 000000G        5$:  CALL     $RM80M        : GET UPDATE MODE INFORMATION
143 000470 004767 000000G        6$:  CALL     $RDSSF        : READ SKIP SECTOR FILE
144 000474 004767 000000G        CALL     $RDBSF        : READ BAD SECTOR FILES
145 000500 004767 000000G        CALL     $RM80F        : FORMAT THE DISK
146 000504 004767 000000G        CALL     $RM80V        : VERIFY THE DISK FORMAT
147 000510 012700 000000G        MOV      #MSG12,R0     : GET ADDRESS OF MESSAGE
148 000514 004767 000000G        CALL     $PRINT        : PRINT 'DONE !' MESSAGE
149 000520 000207                   7$:  RETURN
150                                     :
151 000522 000000                   8$:  .WORD   0           : ONCE DATE FLAG
152

```

```

154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171 000524 012701 000154'
172 000530 012702 000003
173 000534 005021
174 000536 005302
175 000540 001375
176 000542 012700 000000G
177 000546 004767 000000G
178 000552 120227 000057
179 000556 001466
180 000560 120227 000055
181 000564 001404
182 000566 012700 000000G
183 000572 000167 000000G
184
185
186
187 000576 012705 000144'
188 000602 004767 000256
189 000606 103446
190 000610 010167 177342
191 000614 010003
192 000616 012701 000056'
193 000622 005002
194 000624 105711
195 000626 001411
196 000630 122021
197 000632 001774
198 000634 105721
199 000636 001376
200 000640 010300
201 000642 005202
202 000644 105711
203 000646 001366
204 000650 000425
205 000652 005202
206 000654 010267 177274
207 000660 112002
208 000662 120227 000055
209 000666 001401
210 000670 000415
    
```

```

: *+
: **--DATE-CONVERT USER'S RESPONSE TO BINARY DATE
:
: THIS ROUTINE WILL CONVERT THE ASCII DATE INPUTTED
: INTO '$$BUFRX' AS EITHER [MM/DD/YY] OR [DD-$$$-YY] AND
: CONVERT IT TO BINARY AND PUT IT IN DATE PARAMETER
: STORAGE AREA.
:
: INPUTS:
:   '$$BUFRX' CONTAINS ASCII STRING FOR DATE
:
: OUTPUTS:
:   DPARS CONTAINS BINARY REPRESENTATION
:   OF DATE
:
: -
DATE:  MOV    #DPARS,R1      : POINT TO PARAMTER AREA
      MOV    #3,R2         : SET WORD COUNT
1$:   CLR    (R1)+         : ZERO NEXT WORD
      DEC    R2            : DONE ?
      BNE   1$            : IF NE NO
      MOV   #$$BUFRX,R0   : GET BUFFER FOR USER'S RESPONSE
2$:   CALL  $CDTB         : GET A NUMBER
      CMPB  R2,#'/        : TERMINATOR FOR DATE ?
      BEQ   10$           : IF EQ YES
      CMPB  R2,#'-        : POSSIBLY DIFFERENT FORMAT ?
      BEQ   3$            : IF EQ YES
      MOV   #$$MSG2,R0    : NO, SYNTAX ERROR
      JMP   $$MSGWR       : EXIT WITH WARNING STATUS
:
: GET DATE FIELD IN FORMAT [DD-$$$-YY]
3$:   MOV    #DATLMT+4,R5  : POINT TO DAY LIMITS
      CALL  TMTSTL        : TEST LIMITS
      BCS   9$            : IF CS ERROR
      MOV   R1,DPARS+2    : STORE DAY
      MOV   R0,R3         : COPY BUFFER POINTER
      MOV   #DAT,R1       : POINT TO VALID MONTH NAMES
      CLR   R2            : ZERO MONTH NUMBER
4$:   TSTB  (R1)           : END OF TABLE ?
      BEQ   6$            : IF EQ YES, GOOD MONTH
      CMPB  (R0)+,(R1)+   : NO, CHECK NEXT CHARACTER
      BEQ   4$            : AND LOOP IF GOOD MATCH
5$:   TSTB  (R1)+         : END OF MONTH STRING ?
      BNE   5$            : IF NE NO, LOOP
      MOV   R3,R0         : YES, RESET BUFFER POINTER
      INC   R2            : UPDATE MONTH INDEX
      TSTB  (R1)           : END OF TABLE ?
      BNE   4$            : IF NE NO, LOOP
6$:   BR    9$            : ZERO IS END OF MONTH LIST
      INC   R2            : INCREMENT MONTH INDEX
      MOV   R2,DPARS      : SAVE IT
      MOVB  (R0)+,R2     : GET NEXT CHARACTER
      CMPB  R2,#'-        : LEGAL TERMINATOR ?
      BEQ   7$            : IF EQ YES
      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,#BASYSR  : 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  #SMMSG2,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 010123      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  #BASYSR,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      :+
255      : **-TMTSTL - TEST DATE LIMITS
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

```

```

268          :          CC =   GOOD VALUE, LOW<=X<-HIGH
269          :          CS =   BAD VALUE,  LOW>X OR X>HIGH
270          :
271          :-
272
273 001064 020125  TMTSTL:  CMP      R1,(R5)+      : IS VALUE LOWER THAN LOW LIMIT ?
274 001066 002404          BLT      1$           : IF LT YES
275 001070 020125          CMP      R1,(R5)+      : IS IT HIGHER THAN HIGH LIMIT ?
276 001072 003002          BGT      1$           : IF GT YES
277 001074 000241          CLC                    : NO, GOOD VALUE
278 001076 000207          RETURN                 : EXIT
279 001100 000261  1$:   SEC                     : SET ERROR FLAG
280 001102 000207          RETURN                 : EXIT
281
282          000001
          .END

```

BASYR = 000116 G	MSG2R8 000032R	\$MSG12= ***** GX	\$RDSSF= ***** GX	\$SNAME= ***** GX
CR = 000015	TMTSTL 001064R	\$MSG2 = ***** GX	\$REVM= ***** 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	\$MSGWR= ***** 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

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56

.TITLE FMTR8F - RM80 FORMAT 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 14-JAN-81

..NLIST BEX

33	000000	104	122	111	MSG50R: .ASCIZ /DRIVE ERROR DURING FORMAT/
34	000032	106	117	122	MSG51R: .ASCIZ /FORMAT CONTINUING/
35	000054	106	117	122	MSG53R: .ASCIZ /FORMAT TERMINATING/
36	000077	122	115	103	MSG61R: .ASCIZ /RMCS1=/
37	000106	122	115	103	MSG62R: .ASCIZ /RMCS2=/
38	000115	122	115	105	MSG63R: .ASCIZ /RMER1=/
39	000124	122	115	105	MSG64R: .ASCIZ /RMER2=/
40	000133	122	115	104	MSG65R: .ASCIZ /RMDA=/
41	000141	124	122	101	MSG66R: .ASCIZ /TRACK RE-WRITE SUCCESSFUL/
42	000173	124	122	101	MSG67R: .ASCIZ /TRACK RE-WRITE FAILURE/
43	000222	123	125	103	MSG68R: .ASCIZ /SUCCESSIVE SEEK ERRORS/

..EVEN

..WORKING LOCATIONS

50	000252	000000	\$ERCS1: .WORD	0	: CONTROL STATUS REG 1
51	000254	000000	\$ERCS2: .WORD	0	: STATUS REG 2
52	000256	000000	\$ERER1: .WORD	0	: ERROR REGISTER 1
53	000260	000000	\$ERER2: .WORD	0	: ERROR REGISTER 2
54	000262	000000	\$ERDA: .WORD	0	: TRACK/SECTOR ADDRESS
55	000264	000000	\$FLBUF: .WORD	0	: BUFFER ADDRESS

```

58      :+
59      : **-$R80TF-FORMAT A SINGLE TRACK
60      : **-$R80SF-FORMAT BSF, SSF AND ALTERNATE DEC144 CYLINDERS
61      : **-$RM80F-FORMAT ALL CYLINDERS OR FE CYLINDERS
62
63
64      : THIS ROUTINE WILL FORMAT AN RM80 DISK PACK. FULL TRACKS
65      : (32. SECTORS) OF HEADERS ARE WRITTEN UNTIL THE MAXIMUM
66      : DISK ADDRESS IS REACHED.
67
68      : INPUTS:
69      :     DISK PARAMETERS SET TO ZERO
70      :     MDBSF, UDBSF, & SSF READ, VALIDATED, AND IN MEMORY
71      :     $CYL,$STRK,$SEC PRESET FOR '$R80TF' ENTRY
72      :     FORMAT FE CYLINDERS ONLY IF '$SWFFE=1'
73
74      : OUTPUTS:
75      :     NONE.
76
77      : -
78      : FORMAT BSF(558.,13.), SSF(559.,0) AND ALTERNATE DEC144(559.,1) TRACKS
79
80 000266 012767 001056 000000G $R80SF::MOV #558.,$CYL ; STARTING CYLINDER
81 000274 012767 000015 000000G      MOV #13.,$STRK ; TRACK
82 000302 004767 000212          CALL FMTDSK ; FORMAT THE DEC STD 144 TRACK
83 000306 012767 001057 000000G      MOV #559.,$CYL ; STARTING CYLINDER
84 000314 005067 000000G      CLR $STRK ; TRACK
85 000320 004767 000174          CALL FMTDSK ; FORMAT THE SSF TRACK
86 000324 005267 000000G      INC $STRK ; FORMAT THE ALTERNATE DEC144 TRACK
87
88      : FORMAT A SINGLE TRACK
89
90 000330 004767 000000G $R80TF::CALL $MVPAT ; LOAD THE DATA PATTERN
91 000334 000471          BR FMTDSK ; FORMAT THE TRACK $CYL,$STRK
92
93
94      : FORMAT ALL CYLINDERS (0,0 TO 560.,13.) OR BSF/SSF/FE CYLINDERS (558.,13.- 560.,13.)
95
96 000336 005001 $RM80F::CLR R1 ; SET NULL CARRIAGE CONTROL
97 000340 012700      MOV #MSG6,RO ; TELL OPERATOR
98 000344 004767      CALL $PRNTO ; 'START FORMATTING'
99 000350 005001      CLR R1 ; SET NULL CARRIAGE CONTROL
100 000352 105767      TSTB $SWFFE ; FORMAT FE CYLINDERS ONLY ?
101 000356 001011      BNE 1$ ; BR IF YES
102 000360 012700      MOV #MSG6A,RO ; TELL OPERATOR WHICH CYLINDERS
103 000364 004767      CALL $PRNTO ; '-ALL CYLINDERS-'
104 000370 005067      CLR $CYL ; START CYLINDER=0
105 000374 005067      CLR $STRK ; START TRACK=0
106 000400 000412      BR 2$ ; FORMAT THE TRACKS
107
108 000402 012700      1$: MOV #MSG6B,RO ; TELL OPERATOR WHICH CYLINDERS
109 000406 004767      CALL $PRNTO ; '-FE CYLINDERS-'
110 000412 012767      MOV #558.,$CYL ; START CYLINDER=558. (BSF/SSF/FE CYLINDERS)
111 000420 012767      MOV #13.,$STRK ; START TRACK=13.
112 000426 004767      2$: CALL $MVPAT ; LOAD THE DATA PATTERN
113 000432 004767      3$: CALL FMTDSK ; FORMAT THE DISK
114 000436 004767      CALL $STMTX ; UPDATE DISK ADDRESSES

```



```

115 000442 103001          BCC      4$          : IF CS, NOT FINISHED, 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 000000G      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    $WTBSF    : RE-WRITE THE DEC144 TRKS (PRIMARY AND SECONDARY)
126 000506 012667 000000G      MOV      (SP)+,$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    $UPDMX    : UPDATE THE MAX VALUES
134 000530 012700 000000G      MOV     #$BUF,R0  : RELOAD THE BUFFER ADDRESS
135 000534 105067 000000G      CLRB   $KPFRE    : CLEAR SKIP SECTOR FLAG
136 000540 005067 000000G      CLR    $SEC      : START AT SECTOR ZERO
137
138          : CREATE THE HEADER PATTERN
139
140 000544 004767 000000G      1$:  CALL    GENHDR    : GENERATE HEADER PATTERN
141 000550 005267 000000G      INC    $SEC      : UPDATE SECTOR NUMBER
142 000554 026767 000000G 000000G      CMP    $SEC,$MXSEC : ALL SECTORS DONE?
143 000562 001370          BNE    1$         : IF NE NO
144 000564 005067 000000G      CLR    $SEC      : RESET SECTOR NUMBER
145
146          : NOW WRITE THE HEADER AND DATA ONTO THE DISK.
147
148 000570 016767 000000G 000000G 2$:  MOV     TRKSZ,$XFRSZ : LOAD BYTE COUNT FOR ONE TRACK
149 000576 004767 000000G      CALL    $CVTDA    : CONVERT DISK ADDRESS TO LBN
150 000602 012705 000000G      MOV     #$BUF,R5  : SET BUFFER ADDRESS
151 000606 004767 000000G      3$:  CALL    $DSKWH    : WRITE HEADER/DATA FOR THIS TRACK
152 000612 005767 000000G      4$:  TST    $RBUF     : ANY DISK ERRORS?
153 000616 100401          BMI    5$         : IF MI, YES
154 000620 000207          RETURN         : OTHERWISE, RETURN
155
156          : HANDLE THE ERROR
157
158 000622 032767 004000 000014G 5$:  BIT     #4000,$RBUF+14 : WRITE LOCK ERROR?
159 000630 001402          BEQ    6$         : IF EQ NO
160 000632 000167 000000G      JMP    $WLKER    : INFORM THE USER
161
162          : SEND ERROR MESSAGE
163
164 000636 004767 000276          6$:  CALL    $FLRGS    : SAVE FAILING DATA
165 000642 012700 000000'      MOV    #MSG50R,R0 : 'DRIVE ERROR DURING FORMAT'
166 000646 012701 000060          MOV    #60,R1     : CARRIAGE CONTROL CHARACTER
167 000652 004767 000000G      CALL    $SPRNT0   : SEND MESSAGE
168 000656 004767 000064          CALL    $MGPRES   : SEND DISK ADDRESSES
169 000662 105367 000000G      DECB   $SWERL    : DECREMENT THE ERROR LIMIT
170 000666 001014          BNE    7$         : IF NE, ERROR LIMIT NOT EXCEEDED
171 000670 012700 000000G      MOV    #MSG45,R0 : 'ERROR LIMIT EXCEEDED'

```

```

172 000674 012701 000060          MOV      #60,R1          ; CARRIAGE CONTROL CHARACTER
173 000700 004767 000000G        CALL     $SPRINT        ; SEND MESSAGE
174 000704 012700 000054'        MOV      #MSG53R,R0    ; 'FORMAT TERMINATING'
175 000710 004767 000000G        CALL     $SPRINT        ; PRINT THE MESSAGE
176 000714 000167 000000G        JMP      $FMTEX        ; EXIT THE FORMATTER
177
178                                ; IF THE ERROR IS A 'SKI', DO A RECAL
179
180 000720 012700 000032'        7$:     MOV      #MSG51R,R0 ; 'FORMAT CONTINUING'
181 000724 004767 000000G        CALL     $SPRINT        ; SEND MESSAGE
182 000730 032767 040000 000042G  BIT      #40000,$RBUF+42 ; SEE IF ERROR WAS A SEEK INCOMPLETE
183 000736 001714 000000G        BEQ     2$              ; IF EQ, NO
184 000740 004767 000000G        CALL     $HOMSK        ; ENTRY POINT FOR RETRY WITH RECAL
185 000744 000711 000000G        BR      2$              ; TRY AGAIN
186
187                                ;+
188                                ;*- $MGPRE-PREPARE AND SEND DISK ADDRESSES TO OPERATOR
189
190                                ; INPUTS:
191                                ; $FLCYL,$FLTRK,$FLSEC LOADED
192
193                                ; OUTPUTS:
194                                ; ERROR MESSAGE
195                                ; -
196
197 000746 012700 000000G        $MGPRE: MOV     # $SCR,R0    ; ADDRESS OF MESSAGE
198 000752 004767 000000G        CALL     $SPRINT        ; CR-LF
199 000756 016701 177270          MOV     $ERCS1,R1      ; GET RMCS1 AND
200 000762 012703 000077'        MOV     #MSG61R,R3    ; ADDRESS OF MESSAGE
201 000766 004767 000000G        CALL     $EROCT        ; 'RMCS1 = XXXXX'
202 000772 016701 177256          MOV     $ERCS2,R1      ; GET RMCS2 AND
203 000776 012703 000106'        MOV     #MSG62R,R3    ; ADDRESS OF MESSAGE
204 001002 004767 000000G        CALL     $EROCT        ; 'RMCS2 = XXXXX'
205 001006 016701 177244          MOV     $ERER1,R1      ; GET RMER1 AND
206 001012 012703 000115'        MOV     #MSG63R,R3    ; ADDRESS OF MESSAGE
207 001016 004767 000000G        CALL     $EROCT        ; 'RMER1 = XXXXX'
208 001022 016701 177232          MOV     $ERER2,R1      ; GET RMER2 AND
209 001026 012703 000124'        MOV     #MSG64R,R3    ; ADDRESS OF MESSAGE
210 001032 004767 000000G        CALL     $EROCT        ; 'RMER2 = XXXXX'
211 001036 016701 177220          MOV     $ERDA,R1       ; GET RMDA AND
212 001042 012703 000133'        MOV     #MSG65R,R3    ; ADDRESS OF MESSAGE
213 001046 004767 000000G        CALL     $EROCT        ; 'RMDA = XXXXX'
214 001052 012700 000000G        MOV     # $SCR,R0     ; ADDRESS OF MESSAGE
215 001056 004767 000000G        CALL     $SPRINT        ; CR-LF
216
217                                ; PRINT ERROR POSITION, CYLINDER, TRACK AND SECTOR IN DECIMAL
218
219 001062 016701 000000G        MOV     $FLCYL,R1     ; GET FAILING CYLINDER AND
220 001066 012703 000000G        MOV     #MSG15,R3     ; ADDRESS OF MESSAGE
221 001072 004767 000000G        CALL     $EROUT        ; 'CYLINDER = XXX.'
222 001076 016701 000000G        MOV     $FLTRK,R1     ; GET FAILING TRACK AND
223 001102 012703 000000G        MOV     #MSG16,R3     ; ADDRESS OF MESSAGE
224 001106 004767 000000G        CALL     $EROUT        ; 'TRACK = XX.'
225 001112 016701 000000G        MOV     $FLSEC,R1     ; GET FAILING SECTOR AND
226 001116 012703 000000G        MOV     #MSG17,R3     ; ADDRESS OF MESSAGE
227 001122 004767 000000G        CALL     $EROUT        ; 'SECTOR = XX.'
228 001126 012700 000000G        MOV     # $SCR,R0     ; GET ADDRESS OF MESSAGE

```

```

229 001132 004767 000000G          CALL $PRINT          ;CR-LF
230 001136 000207                   RETURN
231
232
233      :+
234      **-$FLRGS-ROUTINE TO SAVE FAILING DISK ADDRESSES AND REGISTERS
235      :
236      INPUTS:
237      $CYL, $TRK SET TO CURRENT CYLINDER AND TRACK
238      $RGRBUF CONTAINS IMAGE OF LATEST H/W REGS (SET BY DRIVER)
239      :
240      OUTPUTS:
241      $ERCS1 - CONTROL STATUS REG 1
242      $ERCS2 - CONTROL STATUS REG 2
243      $ERER1 - ERROR REG 1
244      $ERER2 - ERROR REG 2
245      $ERDA  - TRK/SECTOR REG
246      $FLCYL - FAILING CYLINDER
247      $FLTRK - FAILING TRACK
248      $FLSEC - FAILING SECTOR
249      :-
250 001140 016705 000006G          $FLRGS::MOV      $RGRBUF+6,R5      ; RECOVER FAILING SECTOR ADDRESS
251 001144 042705 177740                   BIC      #177740,R5      ; ISOLATE SECTOR NO.
252 001150 001004                   BNE      1$
253 001152 012767 000037 000000G          MOV      #31.,$FLSEC    ; MUST HAVE BEEN LAST SECTOR
254 001160 000403                   BR       2$
255 001162 005305                   1$: DEC     R5           ; BACKUP ONE
256 001164 010567 000000G          MOV      R5,$FLSEC     ; SET SECTOR NUMBER
257 001170 016767 000000G 000000G 2$: MOV      $CYL,$FLCYL   ; " CYLINDER
258 001176 016767 000000G 000000G          MOV      $TRK,$FLTRK   ; " TRACK
259
260      : SAVE THE RM80 REGISTERS
261
262 001204 012705 000000G          MOV      #RGRBUF,R5    ; REGISTER BUFFER
263 001210 011567 177036                   MOV      (R5), $ERCS1  ; RMCS1
264 001214 016567 000010 177032          MOV      10(R5), $ERCS2 ; RMCS2
265 001222 016567 000014 177026          MOV      14(R5), $ERER1 ; RMER1
266 001230 016567 000042 177022          MOV      42(R5), $ERER2 ; RMER2
267 001236 016567 000006 177016          MOV      6(R5), $ERDA  ; RMDA
268 001244 005067 000000G          CLR      $RGRBUF      ; CLEAR ERROR INDICATOR
269 001250 000207                   RETURN
270
271      000001          .END

```

FMTDSK	000520R	SKPFRE=	***** GX	SEROCT=	***** GX	\$MSG17=	***** GX	\$R80SF	000266RG
GENHDR=	***** GX	TRKSZ =	***** GX	SEROUT=	***** 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	001140RG	\$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	\$SUPDMX=	***** GX
MSG64R	000124R	\$ERCS2	000254R	\$HOMSK=	***** GX	\$PRINT=	***** GX	\$WLKER=	***** GX
MSG65R	000133R	\$ERDA	000262R	\$MGPRE	000746RG	\$PRNT0=	***** GX	\$WTBSF=	***** GX
MSG66R	000141R	\$ERER1	000256R	\$MSG15=	***** GX	\$RBUF=	***** GX	\$WTSSF=	***** GX
MSG67R	000173R	\$ERER2	000260R	\$MSG16=	***** GX	\$RM80F	000336RG	\$XFRSZ=	***** GX
MSG68R	000222R								

. ABS. 000000 000  
001252 001  
ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 522 WORDS ( 3 PAGES)  
DYNAMIC MEMORY: 7942 WORDS ( 30 PAGES)  
ELAPSED TIME: 00:00:11  
[300,20]FMTR8F,[300,30]FMTR8F=[300,10]FMTR8F

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51

.TITLE FMTR8V - 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 22-AUG-80  
..M. LEAVITT 14-JAN-81

..NLIST BEX

33	000000	126	105	122	MSG2V: .ASCIZ /VERIFY TERMINATING/
34	000023	116	117	116	MSG3V: .ASCIZ /NON DATA ERROR DURING VERIFY/
35	000060	126	105	122	MSG4V: .ASCIZ /VERIFY CONTINUING/
36	000102	110	105	101	MSG5V: .ASCIZ @HEADER/DATA ERROR DURING VERIFY@
37					.EVEN

..WORKING LOCATIONS

43	000142	000000	TRKER: .WORD	0	: TRACK ERROR COUNT
44	000144		SECTBL: .BLKB	32.	: SECTOR ERROR TABLE
45	000204	000000	LASTER: .WORD	0	: LAST SECTOR IN ERROR DURING CURRENT VERIFICATION
46					: PASS
47	000206	000000	XFRSEC: .WORD	0	: NUMBER OF SECTORS ACTUALLY VERIFIED
48	000210	000000	ENDSEC: .WORD	0	: ENDING SECTOR ADDRESS
49	000212	000000	BEGSEC: .WORD	0	: STARTING SECTOR ADDRESS
50	000214	000000	VFYCNT: .WORD	0	: VERIFICATION PASSES
51					

53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109

```

+
**-$RM80V-VERIFICATION ROUTINE FOR RM80 DISK PACKS

TRACK COMPLETE <= 1
REPEAT PROCESS UNTIL PHYSICAL DISK ADDRESS .EQ. (561,0,0) ($STSTMX)
  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 ('VFCNT') <= $VERIF
  REPEAT WRITE CHECK UNTIL ('VFCNT' .EQ. 0 .OR. SECTORS TRANSFERED .EQ. 0)
    IF ERROR
      THEN CALL $CKERR
      IF ERROR IS A NON DATA ERROR
        THEN RETRY ERROR
    'VFCNT' = 'VFCNT' - 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
-
```

```

$RM80V::MOV    #MSG11,R0    ; GET VERIFICATION STARTING MESSAGE
          CALL    $PRINT    ; PRINT THE MESSAGE
          MOV    #SPHYDV,R3 ; PHYSICAL DEVICE GEOMETRY POINTER
          CALL    $UPDMX    ; UPDATE THE LIMITS
          CALL    $HOMSK    ; ISSUE A RECAL
          TSTB   $SWFFE    ; VERIFY FE CYLINDERS ONLY ?
          BNE    1$        ; BR IF YES
          CLR    $CYL      ; STARTING CYLINDER=0
          CLR    $TRK      ; STARTING TRACK=0
          BR     2$
          MOV    #558,$CYL ; STARTING CYLINDER=558.
          MOV    #13,$TRK  ; STARTING TRACK=13.
          CLR    $SEC      ; STARTING SECTOR=0
          CLR    TRKER     ; CLEAR THE TRACK ERROR COUNT
          CLR    BEGSEC    ; CLEAR THE STARTING SECTOR
          MOV    $MXSEC,ENDSEC ; LOAD THE ENDING SECTOR ADDRESS
          DEC    ENDSEC    ; CORRECT THE VALUE
          CALL    $MVPAT    ; LOAD THE DATA PATTERN
          CMP    $CYL,#558. ; IS THIS THE BSF CYLINDER ADDRESS ?
          BNE    4$        ; BR IF NO
          CMP    $TRK,#13. ; IS THIS THE BSF TRACK ADDRESS ?
          BNE    4$        ; BR IF NO
          MOV    $CYL,-(SP) ; SAVE CURRENT CYLINDER ADDRESS

000216 012700 000000G
000222 004767 000000G
000226 012703 000000G
000232 004767 000000G
000236 004767 000000G
000242 105767 000000G
000246 001005
000250 005067 000000G
000254 005067 000000G
000260 000406
000262 012767 001056 000000G 1$:
000270 012767 000015 000000G
000276 005067 000000G 2$:
000302 005067 177634
000306 005067 177700
000312 016767 000000G 177670
000320 005367 177664
000324 004767 000000G
000330 026727 000000G 001056 3$:
000336 001016
000340 026727 000000G 000015
000346 001012
000350 016746 000000G
```

```

110 000354 016746 000000G      MOV      $TRK,-(SP)      ; SAVE CURRENT TRACK ADDRESS
111 000360 004767 000000G      CALL     $R80SF         ; RE-FORMAT BSF/SSF HEADERS AND DATA
112 000364 012667 000000G      MOV      (SP)+,$TRK     ; RESTORE TRACK ADDRESS
113 000370 012667 000000G      MOV      (SP)+,$CYL     ; RESTORE CYLINDER ADDRESS
114 000374 105067 000000G      4$:     CLR      SKPFRÉ   ; CLEAR SKIPPED SECTOR FLAG
115 000400 012701 000020      5$:     MOV      #16.,R1   ; LOOP CONSTANT
116 000404 012700 000144      MOV      #SECTBL,RO     ; TABLE ADDRESS
117 000410 005020      6$:     CLR      (RO)+     ; CLEAR TWO LOCATIONS
118 000412 005301      DEC      R1             ; DECREMENT THE LOOP COUNTER
119 000414 001375      BNE      6$            ; IF NE, NOT FINISHED
120 000416 116767 000000G 177570      MOV      $VERIF,VFYCNT ; LOAD THE VERIFICATION COUNT
121 000424 012700 000000G      7$:     MOV      #SBUF,RO  ; BUFFER ADDRESS
122 000430 016767 177556 000000G      MOV      BEGSEC,$SEC    ; LOAD THE SECTOR ADDRESS
123 000436 004767 000000G      8$:     CALL     GENHDR     ; GENERATE A HEADER
124 000442 005267 000000G      INC      $SEC           ; INCREMENT THE SECTOR
125 000446 026767 000000G 177534      CMP      $SEC,ENDSEC    ; SEE IF FINISHED
126 000454 101770      BLOS     8$            ; IF LOS, NO
127 000456 016767 177530 000000G 9$:     MOV      BEGSEC,$SEC    ; RESET THE SECTOR ADDRESS
128 000464 016700 177520      MOV      ENDSEC,RO     ; ENDING SECTOR ADDRESS
129 000470 166700 177516      SUB      BEGSEC,RO     ; SUBTRACT STARTING ADDRESS
130 000474 005200      INC      RO             ; ALLOW A SINGLE SECTOR TRANSFER
131 000476 012701 001004      MOV      #516.,R1      ; SECTOR (INCLUDING HEADER) SIZE(IN BYTES)
132 000502 004767 000000G      CALL     $MULT          ; CALCULATE THE BYTE COUNT
133 000506 010167 000000G      MOV      R1,$XFRSZ     ; LOAD THE BYTE COUNT
134 000512 004767 000000G      CALL     $CVTDA        ; CONVERT DISK ADDRESS TO AN LBN
135 000516 012705 000000G      10$:    MOV      #SBUF,R5    ; BUFFER ADDRESS
136 000522 004767 000000G      CALL     $DSKWC        ; WRITE CHECK
137 000526 005767 000000G      TST      $RGBUF        ; SEE IF ANY ERRORS
138 000532 100404      BMI      11$           ; IF MI, YES
139 000534 005367 177454      DEC      VFYCNT        ; DECREMENT THE VERIFY COUNT
140 000540 001366      BNE      10$           ; IF NE, NOT FINISHED WITH THIS
141                                     ; VERIFICATION PASS
142 000542 000420      BR       13$           ; SEE IF FINISHED WITH THIS TRACK
143
144 ; PROCESS THE ERROR
145
146 000544 004767 000344      11$:    CALL     CKERR        ; SEE WHICH TYPE OF ERROR IT IS
147 000550 103742      BCS      9$            ; IF CS, NON-DATA ERROR - RETRY
148 000552 005767 177430      TST      XFRSEC        ; SEE IF ANY SECTORS WERE TRANSFERED
149 000556 001404      BEQ      12$           ; IF EQ, NO
150 000560 005367 177430      DEC      VFYCNT        ; DECREMENT THE VERIFICATION COUNT FOR THIS PASS
151 000564 001317      BNE      7$            ; IF NE, MORE TO DO
152 000566 000406      BR       13$           ; SEE IF FINISHED WITH THE CURRENT TRACK
153 000570 016746 177416      12$:    MOV      BEGSEC,-(SP) ; SEE IF THE ERROR IS THE LAST SECTOR
154 000574 005216      INC      (SP)          ; CORRECT VALUE FOR THE COMPARE
155 000576 026726 000000G      CMP      $MXSEC,(SP)+  ; ERROR ON SINGLE SECTOR TRANSFER BEGINNING
156                                     ; WITH SECTOR 31 ?
157 000602 001310      BNE      7$            ; IF NE, NO
158 000604 005767 177332      13$:    TST      TRKER        ; ANY NEW ERRORS ON THIS TRACK ?
159 000610 001404      BEQ      14$           ; IF EQ, NO
160 000612 004767 000124      CALL     VFYSEC        ; VERIFY THE SECTORS IN ERROR
161 000616 004767 000000G      CALL     $R80TF        ; REFORMAT THE TRACK AND MARK NEW DEFECTS
162 000622 016700 177362      14$:    MOV      ENDSEC,RO   ; CHECK THE ENDING SECTOR
163 000626 005200      INC      RO             ; CORRECT IT
164 000630 020067 000000G      CMP      RO,$MXSEC     ; SEE IF FINISHED
165 000634 001421      BEQ      15$           ; IF EQ, YES
166 000636 016700 177342      MOV      LASTER,RO    ; GET THE LAST ERROR SECTOR ADDRESS

```

```

167 000642 005200          INC      R0          ; CORRECT VALUE
168 000644 020067 000000G  CMP      R0,$MXSEC   ; SEE IF AT MAX SECTOR
169 000650 001413          BEQ      15$         ; IF EQ, YES
170 000652 016767 177326 177332  MOV     LASTER,BEGSEC ; RELOAD THE STARTING SECTOR
171 000660 005267 177326          INC     BEGSEC       ; START WITH THE FIRST GOOD SECTOR
172 000664 016767 000000G 177316  MOV     $MXSEC,ENDSEC ; RELOAD THE ENDING SECTOR
173 000672 005367 177312          DEC     ENDSEC       ; CORRECT THE VALUE
174 000676 000640          BR       5$          ; CONTINUE
175
176 000700 004767 000000G          15$:   CALL    $TSTMX        ; INCREMENT THE PHYSICAL ADDRESS
177 000704 103411          BCS     16$         ; IF CS, FINISHED
178 000706 016767 000000G 177274  MOV     $MXSEC,ENDSEC ; RELOAD ENDING SECTOR FOR THE TRACK
179 000714 005367 177270          DEC     ENDSEC       ; MAKE INTO AN ADDRESS
180 000720 005067 177266          CLR     BEGSEC       ; START WITH SECTOR 0
181 000724 000167 177400          JMP     3$
182
183 000730 004767 000000G          16$:   CALL    $WTSSF        ; RE-WRITE THE SSF
184 000734 004767 000000G          CALL    $WTBSF        ; RE-WRITE THE DEC STD 144 TRACKS (PRIMARY AND
185                                     ; SECONDARY
186 000740 000207          RETURN
187
188
189 :+
190 : **--VFYSEC-VERIFY THE SECTOR(S) IN ERROR
191 : THIS ROUTINE WILL RE-READ THE SECTORS WHICH GAVE ERRORS DURING THE VERIFY
192 : PASS ON THE CURRENT TRACK. ANY SECTOR FAILING DURING THE RE-READ PASS
193 : WILL BE CONSIDERED DEFECTIVE.
194 :
195 : REPEAT PROCESS UNTIL TRACK ERROR COUNT ('TRKER') .EQ. 0
196 : FIND NEXT SECTOR IN SECTOR ERROR TABLE ('SECTBL')
197 : REPEAT WRITE CHECK FOR SECTOR ENTRY IN 'SECTBL'
198 : UNTIL ('SECTBL'(SECTOR) .LT. 0) .OR. (HEADER FIELD OR DATA FIELD ERROR FOR SECTOR)
199 : IF 'SECTBL'(SECTOR) .GE. 0
200 : THEN IF NO SKIP SECTOR ON TRACK .AND. '$HDRFG' = 0
201 : THEN PUT SECTOR IN SSF
202 : ELSE IF SECTOR IS NOT IN THE FE CYLINDERS
203 : THEN PUT SECTOR IN FEBSF
204 : ELSE PUT SECTOR IN UBSF
205 :
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,$SEC CONTAINS ADDRESS OF CURRENT TRACK
213 : TRKER .NE. 0
214 :
215 : OUTPUTS:
216 : SECTBL CLEARED
217 : ENTRIES MADE IN THE MBSF,UBSF,& SSF'S
218 : 'LASTER' CONTAINS THE ADDRESS OF THE LAST SECTOR IN ERROR ON
219 : THE TRACK
220 :
221 :
222 000742 012700 177777  VFYSEC: MOV    #-1,R0      ; USE R0 AS THE INDEX INTO 'SECTBL'
223 000746 005200          1$:   INC     R0          ; INCREMENT THE INDEX

```



```

224 000750 105760 000144'      TSTB   SECTBL(R0)      ; ENTRY ?
225 000754 001774              BEQ    1$              ; IF EQ, NO
226 000756 010046              2$:   MOV    R0,-(SP)      ; SAVE THE INDEX
227 000760 010067 000000G      MOV    R0,$SEC        ; RELOAD THE SECTOR ADDRESS
228 000764 016767 000000G 000000G  MOV    $CYL,$BUF      ; SETUP A SECTOR IMAGE
229 000772 052767 150000 000000G  BIS    #150000,$BUF   ; SET THE MF, UF, AND FMT16 BITS
230 001000 116767 000000G 000003G  MOVB   $TRK,$BUF+3    ; LOAD THE TRACK ADDRESS
231 001006 116767 000000G 000002G  MOVB   $SEC,$BUF+2    ; LOAD THE SECTOR ADDRESS
232 001014 012705 000000G      3$:   MOV    #R5,$BUF      ; BUFFER ADDRESS
233 001020 012767 001004 000000G  MOV    #516,$XFRSZ   ; SECTOR - INCLUDING HEADER BYTE COUNT
234 001026 004767 000000G      CALL   $CVTDA        ; CONVERT THE PHYSICAL ADDRESS TO LBN
235 001032 004767 000000G      CALL   $DSKWC        ; WRITE CHECK THE SECTOR
236 001036 005767 000000G      TST    $RBUF         ; ANY ERRORS ?
237 001042 100007              BPL    4$            ; IF PL, NO
238 001044 004767 000044      CALL   CKERR         ; PROCESS THE ERROR
239 001050 103761              BCS    3$            ; IF CS, NON-DATA ERROR - RETRY
240 001052 004767 000000G      CALL   $DSTRB       ; PUT THE SECTOR INTO THE PROPER BSF
241 001056 012600              MOV    (SP)+,R0      ; RESTORE THE SECTOR TABLE INDEX
242 001060 000406              BR     5$            ;
243 001062 012600              4$:   MOV    (SP)+,R0      ; RESTORE THE SECTOR TABLE POINTER
244 001064 105360 000144'      DECB   SECTBL(R0)    ; COUNT THE SUCCESSFUL RE-READ
245 001070 100332              BPL    2$            ; IF PL CONTINUE
246 001072 105060 000144'      CLRB   SECTBL(R0)    ; CLEAR THE SECTOR ENTRY, ERROR WAS NOT
247                                ; REPEATED
248 001076 016767 000000G 177100 5$:   MOV    $SEC,LASTER   ; ASSUME CURRENT ERROR IS LAST ERROR ON THIS TRACK
249 001104 005367 177032      DEC    TRKER         ; DECREMENT THE ERROR COUNTER
250 001110 001316              BNE    1$            ; IF NE, NOT FINISHED
251 001112 000207              RETURN                ;
252                                ;

```

```

253                                ;+
254                                ; **--CKERR-VERIFY MODULE DEVICE ERROR ANALYSIS ROUTINE
255                                ;
256                                ; THIS ROUTINE WILL ANALYZE THE DISK ERROR AND REPORT THE ERROR AS REQUIRED.
257                                ;
258                                ; IF SKI
259                                ; THEN REPORT & RECALIBRATE
260                                ; ELSE REPORT ERROR
261                                ; IF NOT DATA ERROR
262                                ; THEN RETRY UNTIL SUCCESSFUL OR RETRY LIMIT EXCEEDED
263                                ; ELSE DETERMINE SECTOR IN ERROR
264                                ; IF TRACK ERROR COUNT ('TRKER') .EQ 0
265                                ; THEN LAST ERROR SECTOR ('LASTER') <= ERROR SECTOR
266                                ; IF ERROR IS HEADER ERROR
267                                ; THEN ERROR TYPE ('$HDRFG') <= 1
268                                ; ELSE 'HDRFG' <= 0
269                                ; IF VERIFY COUNT ('VFYCNT') .NE. 0
270                                ; THEN IF SECTOR IN ERROR NOT LISTED IN BSF, SSF, OR FEBSF ($SRALL)
271                                ; THEN LOAD SECTOR RETRY COUNT IN SECTOR ERROR TABLE ('SECTBL')
272                                ; 'TRKER' = 'TRKER' + 1
273                                ; TRANSFERRED SECTOR COUNT ('XFRSEC') = ((ERROR SECTOR)-(START SECTOR))-1
274                                ; IF 'XFRSEC' .NE. 0
275                                ; THEN END SECTOR <= (ERROR SECTOR-1)
276                                ; WORD COUNT <= ((END SECTOR)-(START SECTOR +1)+1)*258
277                                ;
278                                ; INPUTS:
279                                ; $RBUF - CONTAINS THE RM80 REGISTER CONTENTS
280                                ; VFYCNT - CONTROLS UPDATING 'SECTBL'

```

281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337

```

: 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.
:
: -
    
```

```

CKERR: CALL $FLRGS ; SAVE THE REGISTERS
        MOVB $RBUF+6,-(SP) ; GET THE SECTOR ADDRESS
        DEC (SP) ; BACK UP TO FAILING SECTOR
        BIC #^C37,(SP) ; JUST IN CASE FAILING SECTOR WAS 31.
        MOV $RBUF,-(SP) ; GET RMCS1
        BIC #^C20000,(SP) ; LEAVE 'MCPE'
        MOV $RBUF+10,-(SP) ; GET RMCS2
        BIC #40000,(SP) ; CLEAR 'WCE'
        BISB 1(SP),2(SP) ; 'OR' THE ERROR BITS FROM RMCS2
        MOV $RBUF+42,(SP) ; GET RMER2
        BIC #100000,(SP) ; CLEAR 'BSE'
        BIS (SP)+,(SP) ; 'OR' WITH 'MCPE' & RMCS2 ERROR BITS
        TST (SP)+ ; SEE IF ANY ERROR BITS SET
        BNE 1$ ; IF NE, YES - NON DATA ERROR
        BIT #110400,$RBUF+14 ; TEST 'DCK!DTE!HCRC'
        BNE 4$ ; IF NE, MUST BE A DATA OR HEADER ERROR
        MOV $RBUF+14,-(SP) ; COPY RMER1
        BIC #110400,(SP) ; CLEAR THE DATA/HEADER ERROR BITS
        TST (SP)+ ; SEE IF ANY OTHER ERROR BITS ARE SET
        BEQ 6$ ; IF EQ, NO - ERROR MUST BE A 'BSE'

: SEND ERROR MESSAGE
1$: MOV #MSG3V,R0 ; 'NON DATA ERROR DURING VERIFY'
    MOV #60,R1 ; CARRIAGE CONTROL CHARACTER
    CALL $PRNT0 ; SEND MESSAGE
    CALL $MGPRE ; TYPE THE DEVICE REGISTERS
    DECB $SWERL ; SEE IF ERROR LIMIT EXCEEDED
    BNE 2$ ; IF NE, NO
    MOV #MSG45,R0 ; 'ERROR LIMIT EXCEEDED'
    MOV #60,R1 ; CARRIAGE CONTROL CHARACTER
    CALL $PRNT0 ; SEND MESSAGE
    MOV #MSG2V,R0 ; 'VERIFY TERMINATING'
    CALL $PRNT ; PRINT THE MESSAGE
    JMP $FMTEX ; ABORT THE FORMATTER
2$: MOV #MSG4V,R0 ; 'VERIFY CONTINUING'
    CALL $PRNT ; SEND MESSAGE
    BIT #40000,$RBUF+42 ; SEE IF 'SKI'
    BEQ 3$ ; IF EQ, NO
    CALL $HOMSK ; DO A RECAL
    TST (SP)+ ; CORRECT THE STACK
    SEC ; SHOW NON-DATA ERROR
    RETURN
    
```

000002

000014G

000042G

000000G

```

338                                     ; THE ERROR IS A DATA ERROR, PROCESS AND EXIT
339
340 001332 105067 000000G 4$: CLR B $HDRFG ; ASSUME DATA ERROR
341 001336 032767 000400 000014G BIT #400,$RBUF+14 ; SEE IF ERROR IS A HEADER ERROR ('HCRC')
342 001344 001402 BEQ 5$ ; IF EQ, DATA ERROR
343 001346 105267 000000G INCB $HDRFG ; SET HEADER ERROR FLAG
344 001352 005767 176636 5$: TST VFYCNT ; SEE IF VERIFYING THE ENTIRE TRACK OR IF PROCESSING A
345 ; SINGLE SECTOR
346 001356 001457 BEQ 9$ ; IF EQ, VERIFYING SINGLE SECTORS
347 001360 011667 000000G MOV (SP),$SEC ; PUT THE BAD SECTOR ADDRESS IN '$SEC'
348 001364 004767 000000G CALL $SRALL ; SEE IF THE ERROR IS LISTED IN ONE OF THE BAD SECTOR
349 ; FILES
350 001370 103022 BCC 6$ ; IF CC, DEFECT IS ALREADY ACCOUNTED FOR
351 001372 005267 176544 INC TRKER ; INCREMENT THE TRACK ERROR COUNT
352 001376 011600 MOV (SP),R0 ; GET THE BAD SECTOR ADDRESS
353 001400 116760 000000G 000144' MOV B $RETRY,SECTBL(R0) ; LOAD RETRY COUNT AND FLAG A BAD SECTOR
354 001406 012700 000102' MOV #MSG5V,R0 ; 'HEADER/DATA ERROR DURING VERIFY
355 001412 012701 000060 MOV #60,R1 ; CARRIAGE CONTROL CHARACTER
356 001416 004767 000000G CALL $PRNT0 ; SEND MESSAGE
357 001422 004767 000000G CALL $MGPRE ; TYPE THE DEVICE REGISTERS
358 001426 012700 000060' MOV #MSG4V,R0 ; 'VERIFY CONTINUING'
359 001432 004767 000000G CALL $PRINT ; PRINT IT
360 001436 011667 176544 6$: MOV (SP),XFRSEC ; COPY THE ADDRESS OF THE ERROR SECTOR
361 001442 166767 176536 SUB BEGSEC,XFRSEC ; CALCULATE THE NUMBER OF SECTORS TRANSFERED
362 001450 001003 BNE 7$ ; IF NE, AT LEAST ONE SECTOR TRANSFERED
363 001452 005267 176534 INC BEGSEC ; STEP AROUND THE ERROR SECTOR
364 001456 000404 BR 8$
365 001460 011667 176524 7$: MOV (SP),ENDSEC ; CHANGE THE ENDING SECTOR ADDRESS
366 001464 005367 176520 DEC ENDSEC ; POINT TO LAST ERROR FREE SECTOR
367 001470 016700 176514 8$: MOV ENDSEC,R0 ; COPY THE ENDING SECTOR
368 001474 166700 176512 SUB BEGSEC,R0 ; CALCULATE THE NEW BYTE COUNT
369 001500 005200 INC R0 ; CORRECT THE DIFFERENCE
370 001502 012701 001004 MOV #516.,R1 ; NUMBER BYTES/SECTOR
371 001506 004767 000000G CALL $MULT ; MULTIPLY THE SECTORS * SIZE
372 001512 010167 000000G MOV R1,$XFRSZ ; UPDATE THE BYTE COUNT
373 001516 005726 9$: TST (SP)+ ; CORRECT THE STACK
374 001520 000241 CLC ; SET THE DATA ERROR INDICATOR
375 001522 000207 RETURN ;
376
377 000001 .END

```

SYMBOL TABLE

BEGSEC 000212R	SKPFRE= ***** GX	\$FLRGS= ***** GX	\$PHYDV= ***** GX	\$SWERL= ***** GX
CKERR 001114R	TRKER 000142R	\$FMTEX= ***** GX	\$SPRINT= ***** GX	\$SWFFE= ***** GX
ENDSEC 000210R	VFYCNT 000214R	\$HDRFG= ***** GX	\$SPRNT0= ***** GX	\$STRK = ***** GX
GENHDR= ***** GX	VFYSEC 000742R	\$HOMSK= ***** GX	\$RETRY= ***** GX	\$STSTMX= ***** GX
LASTR 000204R	XFRSEC 000206R	\$MGPRE= ***** GX	\$RGRBUF= ***** GX	\$SUPDMX= ***** GX
MSG2V 000000R	\$BUF = ***** GX	\$MSG11= ***** GX	\$RM80V 000216RG	\$VERIF= ***** GX
MSG3V 000023R	\$CVTDA= ***** GX	\$MSG45= ***** GX	\$R80SF= ***** GX	\$WTBSF= ***** GX
MSG4V 000060R	\$CYL = ***** GX	\$MULT = ***** GX	\$R80TF= ***** GX	\$WTSSF= ***** GX
MSG5V 000102R	\$DSKWC= ***** GX	\$MVPAT= ***** GX	\$SEC = ***** GX	\$XFRSZ= ***** GX
SECTBL 000144R	\$DSTRB= ***** GX	\$MXSEC= ***** GX	\$SRALL= ***** GX	

. ABS. 000000 000  
 001524 001  
 ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 577 WORDS ( 3 PAGES)  
 DYNAMIC MEMORY: 7942 WORDS ( 30 PAGES)  
 ELAPSED TIME: 00:00:15  
 [300,20]FMTR8V,[300,30]FMTR8V=[300,10]FMTR8V

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57

.TITLE FMTR8M - DEFECT ENTRY 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 18-AUG-80  
M. LEAVITT 14-JAN-81

MESSAGES

.NLIST BEX

37	000000	125	104	102	MSG1R8: .ASCIZ /UDBSF /
38	000007	115	123	102	MSG2R8: .ASCIZ /MSBSF/
39	000015	123	123	106	MSG3R8: .ASCIZ /SSF/
40	000021	106	105	040	MSG4R8: .ASCIZ /FE BAD SECTORS/
41	000040	104	125	120	MSG7R8: .ASCIZ /DUPLICATE ENTRY - /
42	000063	106	111	114	MSG8R8: .ASCIZ /FILE FULL - /
43	000100	105	116	124	MSGCR8: .ASCIZ /ENTER BAD SECTR ADRS:/
44	000126	103	131	114	MSGDR8: .ASCIZ /CYL,TRK,SEC = /
45	000145	110	104	101	MSGER8: .ASCIZ @HDA S/N DOESN'T MATCH CURRENT S/N@

.EVEN

: LBN CONSTANTS

50	000210	000003	151200	FELBN: .WORD 3,151200	: BEGINNING OF THE FE CYLINDERS
51					: (559..2,0)
52	000214	000003	151040	USRLBN: .WORD 3,151040	: END OF USER SPACE
53					: (558..12.,31.)

: MINIMUM/MAXIMUM STORAGE FOR INPUT ROUTINE

56	000220	000000		MINCYL: .WORD 0	: STORE MIN. CYLINDER ADDRESS HERE
----	--------	--------	--	-----------------	------------------------------------

58 000222 000000  
59 000224 000000  
60 000226 000000  
61

MINTRK: .WORD 0  
MAXCYL: .WORD 0  
MAXTRK: .WORD 0

:" TRACK "  
:STORE MAX. CYLINDER ADDRESS HERE  
: TRACK "

```

63
64
65
66
67
68
69
70
71
72
73
74
75
76
77 000230 004767 000000G
78 000234 004767 000000G
79 000240 012701 000060
80 000244 012700 000100
81 000250 004767 000000G
82
83 000254 004767 000554
84 000260 001403
85 000262 004767 000024
86 000266 000772
87
88 000270 004767 000000G
89 000274 004767 000000G
90 000300 004767 000000G
91 000304 105067 000000G
92 000310 000207
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110 000312 105067 000000G
111 000316 004767 000000G
112 000322 103405
113 000324 012704 000015
114 000330 004767 000410
115 000334 000556
116 000336 105767 000000G
117 000342 001041
118 000344 012701 000000C
119 000350 012702 000000C
    
```

```

: +
: **-$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
: -
$RM80M::CALL    $RDSSF          : READ SKIP SECTOR FILE
          CALL    $RDBSF          : READ THE MDBSF & UDBSF
          MOV     #60,R1          : SET 2 LINE FEED CONTROL
          MOV     #MSGCR8,R0      : GET ADDRESS OF MESSAGE
          CALL    $SPRNT0         : PRINT 'ENTER BAD SECTR ADRS:'
1$:      CALL    INPUT           : GET SSF ENTRIES
          BEQ     2$              : IF EQ, FINISHED
          CALL    $DSTRB          : DISTRIBUTE ENTERED ADDRESS INTO CORRECT FILE
          BR      1$              : CONTINUE
2$:      CALL    $R80SF          : FORMAT THE BSF/SSF TRACKS
          CALL    $WTSSF          : RE-WRITE THE SSF
          CALL    $WTBSF          : RE-WRITE THE BSF
          CLRB   $SSWMOD         : CLEAR THE MODIFY FLAG
          RETURN
: +
: **-$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
: -
$DSTRB::CLRB   SKPFRE           : CLEAR THE SKIP SECTOR FLAG
          CALL   $SRSDF          : SEE IF ENTRY IN THE SSF
          BCS    1$              : IF CS, NO
          MOV    #MSG3R8,R4      : DUPLICATE ENTRY MESSAGE
          CALL   DUP             : TYPE THE MESSAGE
          BR     13$             : TRY AGAIN
1$:      TSTB   $HDRFG          : SEE IF DEFECT IS A HEADER DEFECT
          BNE   4$              : IF NE, IT IS
          MOV   #$$$F+<2*$$$FHD>-4,R1 : SSF ADDRESS
          MOV   #$$$F+<1000*$$$FLN>-2,R2 : SSF LENGTH
    
```

```

120 000354 004767 000000G          CALL  $SRTRK          : SEARCH THE SSF
121 000360 103032                   BCC   4$             : IF CC, TRACK IS IN THE SSF
122 000362 020127 000000C          CMP   R1,#$SSF+<1000*$SSFLN>-2 : IS SSF FULL?
123 000366 001005                   BNE   2$             : IF NE NO
124 000370 012704 000015'          MOV   #MSG3R8,R4    : GET SSF MESSAGE
125 000374 004767 000400          CALL  FULL          : ISSUE FILE FULL MESSAGE
126 000400 000534                   BR    13$           : GET NEXT ENTRY
127
128                                     : LOAD SSF WITH NEW BAD SECTOR ADDRESS, REVISION DATE, REVISION NUMBER
129                                     : AND UPDATED ENTRY NUMBER
130
131 000402 016721 000000G          2$:  MOV   $CYL,(R1)+ : PUT CYLINDER ADDRESS INTO SSF
132 000406 116721 000000G          MOVB  $SEC,(R1)+    : PUT SECTOR ADDRESS INTO SSF
133 000412 116721 000000G          MOVB  $TRK,(R1)+    : PUT TRACK ADDRESS INTO SSF
134 000416 016767 000000G 000006G  MOV   DPARS,$SSF+6  : LOAD SSF REVISION DATE IN TABLE
135 000424 026767 000000G 000010G  CMP   $REVM,$SSF+10 : HAS REVISION NUMBER BEEN UPDATED YET ?
136 000432 001002                   BNE   3$             : BR IF YES
137 000434 005267 000010G          INC   $SSF+10       : UPDATE REVISION NUMBER
138 000440 005267 000022G          3$:  INC   $SSF+22     : INCREMENT NUMBER OF ENTRIES
139 000444 000512                   BR    13$           : CONTINUE
140
141                                     : SEE IF THE DEFECT CAN GO INTO THE UDBSF OR FEBSF
142
143 000446 004767 000000G          4$:  CALL  $SRSSF          : SEE IF ENTERED SECTOR IS WITHIN A SKIP REGION
144 000452 103404                   BCS   5$             : IF CS, NO
145 000454 105367 000000G          DECB  $SEC          : DECREMENT THE SECTOR ADDRESS
146 000460 105267 000000G          INCB  SKPFRE        : SHOW SECTOR WITHIN A SKIP REGION
147 000464 012704 000007'          5$:  MOV   #MSG2R8,R4  : ASSUME DUPLICATE MDBSF ENTRY
148 000470 004767 000000G          CALL  $SRMDF        : SEARCH THE MDBSF
149 000474 103012                   BCC   6$             : IF CC, ENTRY DUPLICATED
150 000476 012704 000021'          MOV   #MSG4R8,R4    : ASSUME DUPLICATE ENTRY
151 000502 004767 000000G          CALL  $SRFES        : SEE IF ENTRY DUPLICATED IN THE FE AREA
152 000506 103005                   BCC   6$             : IF CC, ENTRY DUPLICATED
153 000510 012704 000000'          MOV   #MSG1R8,R4    : ASSUME DUPLICATE UDBSF
154 000514 004767 000000G          CALL  $SRUDF        : SEARCH THE UDBSF
155 000520 103406                   BCS   7$             : IF CS, ENTRY NOT DUPLICATED
156 000522 105767 000000G          6$:  TSTB  SKPFRE        : SEE IF IN A SKIP REGION
157 000526 001051                   BNE   12$            : IF NE, YES
158 000530 004767 000210          CALL  DUP            : REPORT IT
159 000534 000456                   BR    13$           : GET ANOTHER ENTRY
160 000536 004767 000156          7$:  CALL  CKUSR          : SEE IF ENTRY IF FOR THE USER ADDRESSES
161 000542 103020                   BCC   9$             : IF CC, YES
162 000544 004767 000124          CALL  CHKFE         : SEE IF ENTRY IS FOR FE CYLINDERS
163 000550 103005                   BCC   8$             : IF CC, YES
164 000552 012700 000000G          MOV   #MSG32,R0     : 'INVALID ENTRY'
165 000556 004767 000000G          CALL  $SPRINT        : PRINT THE MESSAGE
166 000562 000443                   BR    13$           :
167 000564 004767 000000G          8$:  CALL  $SRFES        : LOCATE WHERE TO PUT THE ENTRY
168 000570 012704 000021'          MOV   #MSG4R8,R4    : FE BSF MESSAGE SUFFIX
169 000574 020127 000100G          CMP   R1,#$FESF+64. : IF THE FE BSF FULL ?
170 000600 001013                   BNE   11$            : IF NE, NO
171 000602 000407                   BR    10$           : REPORT FILE FULL
172 000604 004767 000000G          9$:  CALL  $SRUDF        : LOCATE LAST ENTRY IN THE UDBSF
173 000610 020127 000774G          CMP   R1,#$UDBSF+1000-4 : IS UDBSF FULL?
174 000614 001005                   BNE   11$            : IF NE, NO
175 000616 012704 000000'          MOV   #MSG1R8,R4    : UDBSF MESSAGE SUFFIX
176 000622 004767 000152          10$: CALL  FULL          : ISSUE FILE FULL MESSAGE

```



```

177 000626 000421
178 000630 016721 000000G
179 000634 116721 000000G
180 000640 116721 000000G
181 000644 105767 000000G
182 000650 001410
183 000652 105767 000000G
184 000656 001405
185 000660 105067 000000G
186 000664 105267 000000G
187 000670 000675
188 000672 000207
189
190
191
192
193
194
195
196
197
198
199
200
201 000674 004767 000000G
202 000700 026767 000000G 177302
203 000706 103403
204 000710 026767 000000G 177274
205 000716 000207
206
207
208
209
210
211
212
213
214
215
216
217
218 000720 004767 000000G
219 000724 026767 177264 000000G
220 000732 103403
221 000734 026767 177256 000000G
222 000742 000207
223
224
225
226
227
228
229
230
231
232
233

11$: BR 13$ : TRY AGAIN
MOV $CYL,(R1)+ : PUT CYLINDER ADDRESS INTO UDBSF
MOVB $SEC,(R1)+ : PUT SECTOR ADDRESS INTO UDBSF
MOVB $TRK,(R1)+ : PUT TRACK ADDRESS INTO UDBSF
TSTB SKPFRE : SEE IF SECTOR WITHIN A SKIP REGION
BEQ 13$ : IF EQ, NO
12$: TSTB $HDRFG : WAS ENTRY FOR A HEADER DEFECT ?
BEQ 13$ : IF EQ, NO - GET NEXT ENTRY
CLRB $HDRFG : CLEAR THE HEADER DEFECT FLAG
INCB $SEC : INCREMENT SECTOR ADDRESS
BR 5$ : CONTINUE
13$: RETURN :

: +
: **--CHKFE-SEE IF THE DEFECT ADDRESS ENTRY IS FOR THE FE CYLINDERS
:
: INPUTS:
: $CYL,$TRK,$SEC LOADED
:
: RETURNS:
: CS ENTRY IS NOT FOR THE FE CYLINDERS
: CC ENTRY IS FOR FE CYLINDERS
: -
CHKFE: CALL $CVTDA : CONVERT THE PHYSICAL ADDRESS TO AN LBN
CMP $LBNH,$LBNH : CHECK HIGH LBN
BLO 1$ : IF LO, DON'T LOOK ANY MORE
CMP $LBNL,$LBNL+2 : CHECK THE LOW LBN
1$: RETURN :

: +
: **--CKUSR-SEE IF THE DEFECT ADDRESS ENTRY IS FOR THE USER CYLINDERS
:
: INPUTS:
: $CYL,$TRK,$SEC LOADED
:
: RETURNS:
: CS ENTRY IS NOT FOR THE USER CYLINDERS
: CC ENTRY IS FOR USER CYLINDERS
: -
CKUSR: CALL $CVTDA : CONVERT THE PHYSICAL ADDRESS TO AN LBN
CMP $USRLBN,$LBNH : CHECK HIGH LBN
BLO 1$ : IF LO, DON'T LOOK ANY MORE
CMP $USRLBN+2,$LBNL : CHECK THE LOW LBN
1$: RETURN :

: +
: **--DUP-DUP ROUTINE WILL ISSUE THE DUPLICATE ENTRY MESSAGE FOR SPECIFIED FILE
: THIS ROUTINE WILL ISSUE THE DUPLICATE ENTRY MESSAGE FOR THE SPECIFIED FILE
:
: INPUTS:
: R4 = ADDRESS OF SPECIFIED FILE MESSAGE
:
: OUTPUTS:
: NONE

```

```

234
235
236
237 000744 012703 000040'      DUP:  MOV      #MSG7R8,R3      ;GET DUPLICATE ENTRY MESSAGE
238 000750 012700 000000G      MOV      #SBUFrx,R0      ;GET ADDRESS OF OUTPUT BUFFER
239 000754 004767 000000G      CALL     $MOVE           ;PUT MESSAGE IN OUTPUT BUFFER
240 000760 010403                MOV      R4,R3           ;GET ADDRESS OF SPECIFIED FILE
241 000762 004767 000000G      CALL     $MOVE           ;PUT THAT IN OUTPUT BUFFER
242 000766 012700 000000G      MOV      #SBUFrx,R0      ;GET ADDRESS OF MESSAGE BUFFER
243 000772 004767 000000G      CALL     $SPRINT         ;PRINT IT
244 000776 000207                RETURN                    ;
245
246
247      ;+
248      ;*-FULL-FULL ROUTINE WILL ISSUE THE FILE FULL MESSAGE FOR SPECIFIED FILE
249      ;
250      ;INPUTS:
251      ;      R4 = ADDRESS OF FILE NAME MESSAGE
252      ;
253      ;OUTPUTS:
254      ;      NONE
255      ;
256      ;-
257 001000 012703 000063'      FULL: MOV      #MSG8R8,R3      ;GET FILE FULL MESSAGE
258 001004 012700 000000G      MOV      #SBUFrx,R0      ;GET ADDRESS OF OUTPUT BUFFER
259 001010 004767 000000G      CALL     $MOVE           ;PUT MESSAGE IN OUTPUT BUFFER
260 001014 010403                MOV      R4,R3           ;GET ADDRESS OF FILE NAME MESSAGE
261 001016 004767 000000G      CALL     $MOVE           ;PUT THAT IN OUTPUT BUFFER
262 001022 012700 000000G      MOV      #SBUFrx,R0      ;GET ADDRESS OF MESSAGE BUFFER
263 001026 004767 000000G      CALL     $SPRINT         ;PRINT IT
264 001032 000207                RETURN                    ;
265
266
267      ;+
268      ;*-INPUT-INPUT DATA FOR SPECIFIED FILE
269      ;
270      ;THIS ROUTINE WILL PROMPT THE USER FOR THE DESIRED CYLINDER, TRACK, AND SECTOR
271      ;ADDRESS.
272      ;
273      ;INPUTS:
274      ;      NONE
275      ;
276      ;OUTPUTS:
277      ;      $CYL, $TRK, AND $SEC SET TO SPECIFIED VALUES
278      ;      $HDRFG = 1 IF THE ENTRY HAD AN 'H' SUFFIX
279      ;      IF Z = 1      NO MORE DATA TO INPUT
280      ;      IF Z = 0      MORE DATA TO INPUT
281      ;
282      ;-
283 001034 105067 000000G      INPUT: CLRB     $HDRFG      ; CLEAR THE HEADER FLAG
284 001040 012703 000000G      MOV      #S$PHYDV,R3     ; POINTER TO PHYSICAL DEVICE GEOMETRY
285 001044 004767 000000G      CALL     $SUPDMX         ; UPDATE MAX VALUES
286 001050 012700 000126'      1$:  MOV      #MSGDR8,R0   ; OPERATOR PROMPT
287 001054 004767 000000G      CALL     $SPRMP         ; GET INPUT FROM THE USER
288 001060 001001                BNE      2$              ; IF NE, INPUT FROM USER
289 001062 000207                RETURN                    ;
290

```

```

291 001064 004767 000150      2$: CALL   LDLMTS      : INITIALIZE DISK ADDRESS LIMITS
292 001070 012700 000000G     MOV   #SBUFEX,R0    : BUFFER POINTER
293 001074 122710 000043     CMPB  #'',(R0)      : SEE IF OCTAL INPUT
294 001100 001452              BEQ   4$            : IF EQ, ERROR
295 001102 004767 000000G     CALL  $CDTB         : CONVERT THE CYLINDER ADDRESS
296 001106 122702 000054     CMPB  #'',R2       : CHECK PROPER SEPARATOR
297 001112 001045              BNE   4$            : IF NE, ERROR
298 001114 004767 000120     CALL  LDLMTS       : LOAD NEW TRACK ADDRESS LIMITS
299 001120 020167 177100     CMP   R1,MAXCYL    : LOOK FOR VALID CYLINDER ADDRESS
300 001124 101040              BHI   4$            : IF HI, INVALID CYLINDER ADDRESS
301 001126 020167 177066     CMP   R1,MINCYL    : LOOK FOR VALID CYLINDER ADDRESS
302 001132 002435              BLT   4$            : IF LT, INVALID CYLINDER ADDRESS
303 001134 010167 000000G     MOV   R1,$CYL      : LOAD THE CYLINDER ADDRESS
304
305 001140 004767 000000G     CALL  $CDTB         : CONVERT THE TRACK ADDRESS
306 001144 122702 000054     CMPB  #'',R2       : CHECK FOR PROPER SEPARATOR
307 001150 001026              BNE   4$            : IF NE, ERROR
308 001152 020167 177050     CMP   R1,MAXTRK    : LOOK FOR VALID TRACK ADDRESS
309 001156 101023              BHI   4$            : IF HI, INVALID TRACK ADDRESS
310 001160 020167 177036     CMP   R1,MINTRK    : LOOK FOR VALID TRACK ADDRESS
311 001164 002420              BLT   4$            : IF LT, INVALID TRACK ADDRESS
312 001166 010167 000000G     MOV   R1,$TRK      : LOAD THE TRACK ADDRESS
313
314 001172 004767 000000G     CALL  $CDTB         : CONVERT THE SECTOR ADDRESS
315 001176 020127 000037     CMP   R1,#31.      : LOOK FOR VALID SECTOR ADDRESS
316 001202 101011              BHI   4$            : IF HI, INVALID SECTOR ADDRESS
317 001204 010167 000000G     MOV   R1,$SEC       : LOAD THE SECTOR ADDRESS
318 001210 122702 000110     CMPB  #'H,R2       : HEADER ERROR ADDRESS ?
319 001214 001002              BNE   3$            : IF NE, NO
320 001216 105267 000000G     INCB  $HDRFG       : SET THE HEADER ERROR FLAG
321 001222 000244      3$: CLZ              : SHOW MORE INPUT
322 001224 000207      RETURN          : EXIT
323 001226 012700 000000G     4$: MOV   #MSG32,R0  : INVALID RESPONSE
324 001232 004767 000000G     CALL  $PRINT        : TYPE THE MESSAGE
325 001236 000704      BR    1$           : CONTINUE

```

```

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

```

```

335 001240 005067 176754      LDLMTS: CLR   MINCYL      : ASSUME MIN. CYLINDER= 0.
336 001244 005067 176752      CLR   MINTRK        : TRACK= 0.
337 001250 012767 001060 176746  MOV   #560.,MAXCYL  : ASSUME MAX. CYLINDER= 560.
338 001256 012767 000015 176742  MOV   #13.,MAXTRK   : TRACK= 13.
339 001264 105767 000000G     TSTB  $SWFFE        : FORMATTING FE CYLINDERS ONLY ?
340 001270 001411              BEQ   1$            : BR IF NO
341 001272 012767 001056 176720  MOV   #558.,MINCYL  : MIN. CYLINDER= 558.
342 001300 020127 001056      CMP   R1,#558.      : IS CYLINDER INPUT 558. ?
343 001304 001003              BNE   1$            : BR IF NO
344 001306 012767 000015 176706  MOV   #13.,MINTRK   : MIN. TRACK= 13.
345 001314 000207      1$: RETURN
346
347      000001      .END

```

CHKFE 000674R	MSGCR8 000100R	\$CDTB = ***** GX	\$PRMPT= ***** GX	\$SRTRK= ***** GX
CKUSR 000720R	MSGDR8 000126R	\$CVTDA= ***** GX	\$PRNT0= ***** 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	\$REVM= ***** GX	\$SSFLN= ***** GX
FULL 001000R	MSG3R8 000015R	\$HDRFG= ***** GX	\$RMB0M 000230RG	\$SWFFE= ***** GX
INPUT 001034R	MSG4R8 000021R	\$LBNH = ***** GX	\$R80SF= ***** GX	\$SWMOD= ***** 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

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51

.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

33	000000	123	123	106	MSG1L: .ASCIZ /SSF (SKIPPED SECTOR FILE) CONTENTS/
34	000043	115	104	102	MSG2L: .ASCIZ /MDBSF CONTENTS/
35	000062	125	104	102	MSG3L: .ASCIZ /UDBSF CONTENTS/
36	000101	103	122	105	MSG4L: .ASCIZ @CREATION DATE: @
37	000121	110	104	101	MSG5L: .ASCIZ @HDA S/N: @
38	000133	103	131	114	MSG6L: .ASCIZ /CYL,TRK,SEC/
39	000147	052	040	111	MSG7L: .ASCIZ /* INVALID DATE */
40	000170	052	040	116	MSG8L: .ASCIZ /* NO ENTRIES */
41	000207	122	105	126	MSG9L: .ASCIZ /REVISION DATE: /
42	000227	122	105	126	MSG10L: .ASCIZ /REVISION NUMBER: /
43	000251	115	104	102	MSG11L: .ASCIZ /MDBSF 'LOGICAL' CONTENTS/
44	000302	125	104	102	MSG12L: .ASCIZ /UDBSF 'LOGICAL' CONTENTS/
45	000333	040	000		SPACE: .ASCIZ / /
46					
47	000335	000			SSFLST: .BYTE 0 ; LIST SSF SWITCH
48					.EVEN
49					
50	000336				DPARS: .BLKW 10. ; OUTPUT CONVERSION BUFFER
51					

```

53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69 000362 004767 000000G $RM80L::CALL $RDSSF ; READ THE SKIPPED SECTOR FILE (SSF)
70 000366 004767 000000G CALL $RDBSF ; READ THE MDBSF AND THE UDBSF
71
72 ; DISPLAY THE SSF CONTENTS
73
74 000372 012700 000000' MOV #MSG1L,R0 ; 'SSF (SKIPPED SECTOR FILE) CONTENTS'
75 000376 012701 000060 MOV #60,R1 ; CARRIAGE CONTROL CHARACTER
76 000402 004767 000000G CALL $PRNT0 ;
77 000406 012701 000000G MOV #SSSF,R1 ; SERIAL NUMBER POINTER
78 000412 004767 000600 CALL HDASN ; PRINT THE HDA SERIAL NUMBER
79 000416 012700 000000G MOV #SBUFEX,R0 ; BUFFER POINTER
80 000422 012703 000101' MOV #MSG4L,R3 ; 'CREATION DATE: '
81 000426 004767 000000G CALL $MOVE ; LOAD THE BUFFER
82 000432 012703 000004G MOV #SSSF+4,R3 ; DATE FIELD POINTER
83 000436 004767 000240 CALL SFDATE ; PRINT THE SSF DATE FIELDS
84 000442 012700 000000G MOV #SBUFEX,R0 ; BUFFER POINTER
85 000446 012703 000207' MOV #MSG9L,R3 ; 'REVISION DATE: '
86 000452 004767 000000G CALL $MOVE ; LOAD THE BUFFER
87 000456 012703 000006G MOV #SSSF+6,R3 ; DATE FIELD POINTER
88 000462 004767 000214 CALL SFDATE ; PRINT THE SSF DATE FIELDS
89 000466 012700 000000G MOV #SBUFEX,R0 ; BUFFER POINTER
90 000472 012703 000227' MOV #MSG10L,R3 ; 'REVISION NUMBER: '
91 000476 004767 000000G CALL $MOVE ; LOAD THE BUFFER
92 000502 016701 000010G MOV $SSSF+10,R1 ; REVISION NUMBER
93 000506 005002 CLR R2 ; ENABLE ZERO SUPPRESSION
94 000510 004767 000000G CALL $CBDMG ; CONVERT
95 000514 105010 CLRB (R0) ; ASCIZ TERMINATOR
96 000516 012700 000000G MOV #SBUFEX,R0 ; OUTPUT POINTER
97 000522 004767 000000G CALL $PRINT ;
98 000526 012703 000000C MOV #SSSF+<2*SSSFHD>,R3 ; BEGINNING OF SSF DATA FIELD
99 000532 112767 177777 177575 MOVB #-1,SSFLST ; SET SSF LIST FLAG
100 000540 004767 000270 CALL BSFDAT ; PRINT THE SSF FILE CONTENTS
101 000544 105067 177565 CLRB SSFLST ; RESET THE SSF LIST FLAG
102
103 ; DISPLAY THE MDBSF CONTENTS
104
105 000550 012700 000043' MOV #MSG2L,R0 ; 'MDBSF CONTENTS'
106 000554 105767 000000G TSTB $SWLOG ; SEE IF LOGICAL DISPLAY
107 000560 001402 BEQ 10$ ; IF EQ, NO
108 000562 012700 000251' MOV #MSG11L,R0 ; CHANGE TO 'MDBSF LOGICAL CONTENTS'
109 000566 012701 000060 10$: MOV #60,R1 ; CARRIAGE CONTROL CHARACTER

```

```

+
**-$RM80L-DISPLAY THE CONTENTS OF THE MDBSF, UDBSF, AND SSF
THIS ROUTINE WILL DISPLAY THE CONTENTS OF THE MDBSF, THE UDBSF
AND THE SSF FILES.
INPUTS:
NONE
OUTPUTS:
THE CONTENTS OF THE MDBSF, UDBSF, AND SSF PRINTED ON 'TI:'
'$SWLSI' CLEARED
-
```

.ENABL LSB

```

110 000572 004767 000000G          CALL    SPRNTO          :
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   $$WLOG         : SEE IF LOGICAL DISPLAY
120 000626 001402                    BEQ     20$            : IF EQ, NO
121 000630 012700 000302'          MOV     #MSG12L,R0    : CHANGE TO 'UDBSF LOGICAL CONTENTS'
122 000634 012701 000060          20$:  MOV     #60,R1      : CARRIAGE CONTROL CHARACTER
123 000640 004767 000000G          CALL    SPRNTO          :
124 000644 012701 000000G          MOV     #SUDBSF,R1    : UDBSF S/N POINTER
125 000650 004767 000342          CALL    HDASN          : TYPE THE SERIAL NUMBER
126 000654 012703 000010G          MOV     #SUDBSF+10,R3 : UDBSF DATA POINTER
127 000660 004767 000150          CALL    BSFDAT         : DISPLAY THE UDBSF
128 000664 012700 000333'          MOV     #SPACE,R0     : BLANK LINE
129 000670 004767 000000G          CALL    $PRINT        : PRINT THE LINE
130 000674 105067 000000G          CLRB   $$WLST        : CLEAR THE 'LIST' FLAG
131 000700 000207                    RETURN                :
132

```

.DSABL LSB

```

: *+
: **--SFDATE-DISPLAY THE SSF DATE FIELD CONTENTS
:
: THIS ROUTINE PRINTS THE CREATION AND REVISION DATE FIELDS
: FROM THE SSF
:
: INPUT:
:       R0 = BUFFER POINTER
:       R3 = DATE FIELD POINTER
:
: OUTPUT:
:       NONE
: -

```

```

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          MOV    1(R3),R1     : GET THE YEAR CODE
170 000764 006201                    ASR    R1             : RIGHT JUSTIFY
171 000766 062701 000000G          ADD    #BASYS,R1     : ADD THE BASE YEAR
172 000772 010167 177340          MOV    R1,DPARS     : YEAR
173 000776 012701 000336'          MOV    #DPARS,R1    : INPUT BUFFER POINTER

```

174	001002	004767	000000G		CALL	\$DAT		: CONVERT THE DATE
175	001006	000404			BR	20\$		: CONTINUE
176	001010	012703	000147'	10\$:	MOV	#MSG7L,R3		: '** INVALID DATE **'
177	001014	004767	000000G		CALL	\$MOVE		: LOAD THE TEXT
178	001020	105010		20\$:	CLRB	(R0)		: ASCIZ TERMINATOR
179	001022	012700	000000G		MOV	#\$BUFRX,R0		: BUFFER POINTER
180	001026	004767	000000G		CALL	\$PRINT		: PRINT THE DATE
181	001032	000207			RETURN			: :
182								
183								
184								
185								
186								
187								
188								
189								
190								
191								
192								
193								
194								
195								
196								
197	001034	005713			BSF DAT:	TST	(R3)	: LOOK FOR END OF ENTRIES
198	001036	100005				BPL	10\$	: IF PL, FILE ENTRIES PRESENT
199	001040	012700	000170'			MOV	#MSG8L,R0	: '** NO ENTRIES **'
200	001044	004767	000000G			CALL	\$PRINT	: :
201	001050	000461				BR	40\$	: EXIT
202	001052	012700	000133'	10\$:		MOV	#MSG6L,R0	: 'CYL,TRK,SEC'
203	001056	004767	000000G			CALL	\$PRINT	: :
204	001062	012700	000000G	20\$:		MOV	#\$BUFRX,R0	: OUTPUT BUFFER POINTER
205	001066	011367	000000G			MOV	(R3),\$CYL	: LOAD THE CYLINDER ADDRESS
206	001072	116367	000003 000000G			MOVB	3(R3),\$TRK	: LOAD THE TRACK ADDRESS
207	001100	116367	000002 000000G			MOVB	2(R3),\$SEC	: LOAD THE SECTOR ADDRESS
208	001106	012301				MOV	(R3)+,R1	: GET THE CYLINDER ADDRESS
209	001110	100441				BMI	40\$	: IF MI, NO MORE ENTRIES
210	001112	005002				CLR	R2	: ENABLE ZERO SUPRESSION
211	001114	004767	000000G			CALL	\$CBDMG	: CONVERT
212	001120	112720	000054			MOVB	#'(R0)+	: SEPARATOR
213	001124	116301	000001			MOVB	1(R3),R1	: TRACK ADDRESS
214	001130	005002				CLR	R2	: ENABLE ZERO SUPRESSION
215	001132	004767	000000G			CALL	\$CBDMG	: CONVERT
216	001136	112720	000054			MOVB	#'(R0)+	: SEPARATOR
217	001142	111301				MOVB	(R3),R1	: SECTOR ADDRESS
218	001144	105767	177165			TSTB	SSFLST	: LISTING THE SSF CONTENTS ?
219	001150	001007				BNE	30\$	: IF NE, YES
220	001152	105767	000000G			TSTB	\$SWLOG	: PRINT BSF LOGICAL ADDRESSES ?
221	001156	001004				BNE	30\$	: IF NE, YES
222	001160	004767	000000G			CALL	\$SRSSF	: CHECK THE SSF
223	001164	103401				BCS	30\$	: IF CS, SECTOR ENTRY PHYSICAL VALUE
224	001166	005201				INC	R1	: INCREMENT THE SECTOR ADDRESS
225	001170	005723		30\$:		TST	(R3)+	: INCREMENT THE INPUT POINTER
226	001172	005002				CLR	R2	: ENABLE ZERO SUPRESSION
227	001174	004767	000000G			CALL	\$CBDMG	: CONVERT
228	001200	105010				CLRB	(R0)	: ASCIZ TERMINATOR
229	001202	012700	000000G			MOV	#\$BUFRX,R0	: OUTPUT BUFFER POINTER
230	001206	004767	000000G			CALL	\$PRINT	: PRINT THE ADDRESS





SYMBOL TABLE  
BASR = \*\*\*\*\* GX  
BSFDAT 001034R  
DPARS 000336R  
HDASN 001216R  
MSG1L 000000R  
MSG10L 000227R  
MSG11L 000251R  
MSG12L 000302R

MSG2L 000043R  
MSG3L 000062R  
MSG4L 000101R  
MSG5L 000121R  
MSG6L 000133R  
MSG7L 000147R  
MSG8L 000170R  
MSG9L 000207R

SFDATE 000702R  
SPACE 000333R  
SSFLST 000335R  
\$BUFRX= \*\*\*\*\* GX  
\$CBDMG= \*\*\*\*\* GX  
\$CDDMG= \*\*\*\*\* GX  
\$CYL = \*\*\*\*\* GX  
\$DAT = \*\*\*\*\* GX

\$MDBSF= \*\*\*\*\* GX  
\$MOVE = \*\*\*\*\* GX  
\$PRINT= \*\*\*\*\* GX  
\$PRNT0= \*\*\*\*\* GX  
\$RDBSF= \*\*\*\*\* GX  
\$RDSSF= \*\*\*\*\* GX  
\$RMBOL 000362RG  
\$SEC = \*\*\*\*\* GX

\$SRSSF= \*\*\*\*\* GX  
\$SSF = \*\*\*\*\* GX  
\$SSFHD= \*\*\*\*\* GX  
\$SWLOG= \*\*\*\*\* GX  
\$SWLST= \*\*\*\*\* GX  
\$TRK = \*\*\*\*\* GX  
\$SUBSF= \*\*\*\*\* 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

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35

.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

```

37
38
39
40
41
42
43
44
45
46
47
48
49
50 000066 012703 000000' $RM80I::MOV #MSG1R1,R3 ; WARNING MESSAGE ADDRESS
51 000072 004767 000000G CALL $UNTID ; STORE MESSAGE AND DEVICE/UNIT
52 000076 012703 000000G MOV #MSG24,R3 ; MESSAGE SUFFIX
53 000102 004767 000000G CALL $MOVE ; LOAD THE SUFFIX INTO THE MESSAGE BUFFER
54 000106 012700 000144G MOV #SBUF+100.,R0 ; PRINT THE WARNING MESSAGE
55 000112 012701 000060 MOV #60,R1 ; 2 LINE FEED CARRIAGE CONTROL
56 000116 004767 000000G CALL $PRNT0 ; PRINT WARNING MESSAGE
57 000122 012700 000000G MOV #MSG31,R0 ; 'YES-NO' MESSAGE POINTER
58 000126 004767 000000G CALL $PRMPT ; PROMPT AND GET RESPONSE
59 000132 001404 BEQ 1$ ; IF EQ, RESPONSE WAS 'NO'
60 000134 122767 000131 000000G CMPB #'Y,$BUFRX ; SEE IF RESPONSE WAS 'YES'
61 000142 001402 BEQ 2$ ; IF EQ, RESPONSE WAS 'YES'
62 000144 000167 000000G 1$: JMP $FMTEX ; EXIT
63 000150 012703 000374 2$: MOV #256.-4,R3 ; GET NUMBER OF ENTRIES TO RESET
64 000154 012700 000000G MOV #MDBSF,R0 ; GET START OF MDBSF
65 000160 016720 000000G MOV $$NAME,(R0)+ ; STUFF THE NEW SERIAL NUMBER
66 000164 016720 000002G MOV $$NAME+2,(R0)+ ;
67 000170 005020 CLR (R0)+ ; BLANK WORD
68 000172 005020 CLR (R0)+ ; SHOW NOT AN ALIGNMENT DISK
69 000174 012720 177777 3$: MOV #-1,(R0)+ ; EMPTY THE MDBSF
70 000200 005303 DEC R3 ; FINISHED?
71 000202 001374 BNE 3$ ; IF NE NO
72 000204 012700 000000G MOV #SUDBSF,R0 ; GET START OF UDBSF
73 000210 016720 000000G MOV $$NAME,(R0)+ ; STUFF THE NEW SERIAL NUMBER
74 000214 016720 000002G MOV $$NAME+2,(R0)+ ;
75 000220 005020 CLR (R0)+ ; BLANK WORD
76 000222 005020 CLR (R0)+ ; SHOW NOT AN ALIGNMENT DISK
77 000224 012703 000374 4$: MOV #256.-4,R3 ; GET NUMBER OF ENTRIES TO RESET
78 000230 012720 177777 4$: MOV #-1,(R0)+ ; EMPTY THE UDBSF
79 000234 005303 DEC R3 ; FINISHED?
80 000236 001374 BNE 4$ ; IF NE NO
81 000240 012703 000000C MOV #256.*<$$SFLN>,R3 ; SSF SIZE
82 000244 012700 000000G MOV #$$SF,R0 ; SSF BUFFER POINTER
83 000250 012720 177777 5$: MOV #-1,(R0)+ ; CLEAR A LOCATION
84 000254 005303 DEC R3 ; COUNT THE LOCATION
85 000256 001374 BNE 5$ ; IF NE, NOT FINISHED
86 000260 016767 000000G 000000G MOV $$NAME,$$SF ; SERIAL NUMBER LSB
87 000266 016767 000002G 000002G MOV $$NAME+2,$$SF+2 ; SERIAL NUMBER MSB
88 000274 016767 000000G 000004G MOV DPARS,$$SF+4 ; CREATION DATE
89 000302 016767 000000G 000006G MOV DPARS,$$SF+6 ; REVISION DATE
90 000310 005067 000010G CLR $$SF+10 ; REVISION NUMBER
91 000314 012767 000001 000012G MOV #1,$$SF+12 ; SSF DATA TYPE INDICATOR (16 BIT)
99 000336 005067 000022G CLR $$SF+22 ; NUMBER OF SSF ENTRIES
100 000342 005067 000024G CLR $$SF+24 ; CHECKSUM

```

101 000346 004767 000000G  
102 000352 004767 000000G  
103 000356 004767 000000G  
104 000362 105067 000000G  
105 000366 000207  
106  
107 000001

6\$:

.END

CALL \$R80SF  
CALL \$WTBSF  
CALL \$WTSSF  
CLRB \$SWINT  
RETURN

: CALL THE 'SPECIAL FORMATTER'  
: WRITE THE DEC STD 144 INFORMATION  
: WRITE THE SSF  
: CLEAR THE INITIALIZATION FLAG  
:

DPARS = ***** GX	\$MDBSF= ***** GX	\$PRMPT= ***** GX	\$SNAME= ***** GX	\$UDBSF= ***** GX
MSG1RI 000000R	\$MOVE = ***** GX	\$PRNT0= ***** GX	\$SSF = ***** GX	\$UNTID= ***** GX
\$BUF = ***** GX	\$MSG24= ***** GX	\$RM80I 000066RG	\$SSFLN= ***** GX	\$WTBSF= ***** GX
\$BUFRX= ***** GX	\$MSG31= ***** GX	\$R80SF= ***** 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