

RM80

RM80 PERF EXER
CZRNAAO

AH-T105A-MC
FICHE 1 OF 2

JUL 1982
COPYRIGHT © 1982
MADE IN USA



The main body of the document is a large grid of approximately 15 columns and 25 rows of small, dense tables. Each cell in the grid contains a small table with multiple columns and rows of text, likely representing performance data or exercise parameters. The text is too small to be legible, but the overall structure is a comprehensive data matrix.

RM80

RM80 PERF EXER
CZRNAAO

AH-T105A-MC
FICHE 2 OF 2

JUL 1982
COPYRIGHT © 1982
MADE IN USA



Microfilm frame containing a grid of data. The data is organized into columns and rows, with some sections appearing as vertical bars (possibly representing a barcode or specific data fields) and other sections as dense text or patterns of characters. The content is too small to read accurately.



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

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

1. ABSTRACT
 - 1.1 GENERAL DOCUMENT NOTES
2. REQUIREMENTS
 - 2.1 EQUIPMENT
 - 2.2 MEDIA
 - 2.3 PRELIMINARY PROGRAMS
3. OPERATING PROCEDURE
 - 3.1 LOADING THE PROGRAM
 - 3.2 STARTING ADDRESSES
 - 3.3 PROGRAM CONTROL
 - 3.4 SWITCH OPTIONS
 - 3.5 PASS/TEST TERMINATION
 - 3.5.1 PASS TERMINATION
 - 3.5.2 TEST TERMINATION
 - 3.6 RUN TIME
 - 3.6.1 DATA TRANSFER MODE
 - 3.6.2 SEEK VERIFICATION MODE
 - 3.7 DUAL PORT OPERATION
 - 3.8 XXDP, ACT11, APT11
 - 3.9 APT ENVIRONMENTAL TABLE DEFINITIONS
4. CONTROLLING THE PROGRAM
 - 4.1 PARAMETERS
 - 4.1.1 PROGRAM CONTROL PARAMETERS
 - 4.1.2 CHANGE DEVICE ADDRESSES
 - 4.2 KEYBOARD COMMANDS
 - 4.2.1 'T' COMMAND
 - 4.2.2 'D' COMMAND
 - 4.2.3 'S' COMMAND
 - 4.2.4 'W' COMMAND
 - 4.2.5 'R' COMMAND
 - 4.2.6 'WT' COMMAND
5. PERFORMANCE SUMMARY TYPEOUT
 - 5.1 PERFORMANCE SUMMARY TYPEOUT EXPLANATION
 - 5.2 HARD/SOFT ERROR DEFINITIONS
 - 5.2.1 HARD ERRORS
 - 5.2.2 SOFT ERRORS
6. DATA CHECKING & ERROR RECOVERY
 - 6.1 DATA BUFFER COMPARISON
 - 6.2 VERIFICATION OF DATA WRITTEN
 - 6.3 BAD ADDRESS FLAGGING
7. ERROR MESSAGES

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

58
59
60
61
62
63
64
65
66
67
68

7.1 ERROR DESCRIPTION LINES
7.2 DETAIL ERROR LINES

8. PROGRAM DESCRIPTION

8.1 HOW THE PROGRAM OPERATES
8.2 DUAL PORT OPERATION
8.3 SELECTION OF OPERATION VARIABLES
8.4 DATA PATTERNS

9. RM SOFTWARE DRIVER DOCUMENT

1. ABSTRACT

THE RM80 PERFORMANCE EXERCISER PROGRAM IS DESIGNED TO PERFORM AN INTERACTIVE TEST ON RM DISK DRIVES CONNECTED TO A MASSBUS SUBSYSTEM. THE DRIVES MAY BE CONTROLLED BY AN RH70 CONTROLLER. IN ADDITION TO PERFORMING AN INTERACTIVE TEST OF THE DISK DRIVES ON THE SUBSYSTEM, THE PROGRAM IS INTENDED TO BE USED TO VERIFY THAT THE DRIVES UNDER TEST ARE PERFORMING TO THEIR DATA ERROR RATE AND SEEK ERROR RATE (SEE ERROR RATE SPECIFICATIONS).

THE PERFORMANCE EXERCISER PROGRAM WILL EXERCISE DRIVES CONNECTED AS EITHER SINGLE OR DUAL PORT UNITS. DUAL PORT DRIVES ARE TESTED BY LOADING AND RUNNING THE PROGRAM FROM BOTH CONTROLLING SYSTEMS. THE PROGRAM WILL EXERCISE A MIXED SYSTEM OF DUAL PORT AND SINGLE PORT DRIVES.

TO OBTAIN INTERACTIVE TESTING, OPERATIONS ON THE MULTI-DRIVE CONFIGURATIONS ARE OVERLAPPED (OTHER DRIVES ARE PERFORMING SEEK/SEARCH OPERATIONS WHILE ONE DRIVE IS PERFORMING A DATA TRANSFER). OPERATIONS AMONG THE DRIVES ARE OPTIMIZED SO THAT A HIGH SUBSYSTEM DATA TRANSFER RATE OR A HIGH POSITIONING OPERATION RATE IS MAINTAINED.

THE PERFORMANCE OF EACH DRIVE IS MONITORED BY THE PROGRAM. IF A DRIVE EXCEEDS A PRESET NUMBER OF ERRORS IN ANY OF SEVERAL CATEGORIES, THAT DRIVE IS AUTOMATICALLY DEASSIGNED. (THE OPERATOR MAY OVERRIDE THE AUTOMATIC DEASSIGNMENT FEATURE.) THE PROGRAM REPORTS PERFORMANCE STATISTICS FOR EACH DRIVE BEING EXERCISED ON REQUEST FROM THE OPERATOR OR AUTOMATICALLY AT AN INTERVAL DETERMINED BY THE OPERATOR.

ALL DATA TRANSFER COMMANDS EXCEPT WRITE HEADER & DATA AND WRITE CHECK HEADER & DATA ARE USED. RECALIBRATE AND READ-IN PRESET COMMANDS ARE USED AT STARTUP AND DRIVE INITIALIZATION. RECALIBRATE AND RETURN-TO-CENTERLINE COMMANDS ARE USED DURING ERROR RECOVERY.

THE DATA TRANSFER COMMANDS ARE SELECTED RANDOMLY EXCEPT FOR THE WRITE CHECK COMMANDS. THE WRITE CHECK COMMANDS ARE USED TO VERIFY A PREVIOUS WRITE OPERATION. THUS, WHEN A WRITE COMMAND IS SELECTED, THE DATA WRITTEN IS VERIFIED BY THE APPROPRIATE WRITE CHECK COMMAND.

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

1.1 GENERAL DOCUMENT NOTES

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

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

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
110
111
112
113
114

2. REQUIREMENTS

2.1 EQUIPMENT
PDP-11/70 PROCESSOR
16K MEMORY
KW11-L OR KW11-P CLOCK
PROGRAM LOADING DEVICE
TERMINAL
RH70 CONTROLLER
1 TO 8 RM80 DISK DRIVES

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

2.3 PRELIMINARY PROGRAMS
RM80 DISKLESS TEST, PART 1 & 2
RM80 FUNCTIONAL TEST, PART 1, 2 & 3 (OPTIONAL PART 4)

3. OPERATING PROCEDURE

3.1 LOADING THE PROGRAM
THE PROGRAM MAY BE LOADED BY EITHER OF THE FOLLOWING MEDIA:
.PAPER TAPE, USING THE STANDARD PAPER TAPE PROCEDURE
.XXDP MEDIA, USING ANY XXDP DEVICE

3.2 STARTING ADDRESSES
200 - START ADDRESS, ALL SWITCHES CLEAR (SEE SECTION 3.4)
WHEN THE PROGRAM IS STARTED, A DATA PATTERN WILL BE WRITTEN TO
ALL ON-LINE DRIVES IN A SEQUENTIAL SEEK MODE. UPON COMPLETION OF
THE WRITE, THE PROGRAM GOES INTO A TESTING MODE.
204 - RESTART ADDRESS, THE RESTART ADDRESS PROVIDES THE OPERATOR WITH
THE ABILITY TO CHANGE THE DEFAULT RM/RH ADDRESSES (SEE SECTION
4.1.2), CHANGE ANY PROGRAM PARAMETERS (SEE SECTION 4.1) OR
CHANGE DRIVE LIMIT PARAMETERS (SEE SECTION 4.2).

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

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
160
161
162
163
164
165
166
167
168
169
170
171

SWITCH REGISTER SWITCH SETTINGS.

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

3.4 SWITCH OPTIONS

IF THE PROGRAM IS BEING RUN ON A SWITCHLESS PROCESSOR THE PROGRAM WILL DETERMINE THAT THE HARDWARE SWITCH REGISTER IS NOT PRESENT AND WILL USE A 'SOFTWARE' SWITCH REGISTER. THE 'SOFTWARE' SWITCH REGISTER IS LOCATED AT LOCATION 176. THE SETTINGS OF THE 'SOFTWARE' SWITCHES ARE CONTROLLED THROUGH A KEYBOARD ROUTINE WHICH IS CALLED BY TYPING A 'CONTROL G'. THE PROGRAM WILL RECOGNIZE THE 'CONTROL G' AT ANY TIME EXCEPT WHEN THE PROGRAM IS IN KEYBOARD ENTRY MODE, OR IS AT A HIGHER PRIORITY PROCESSING AN DRIVE INTERRUPT. THE 'SOFTWARE' SWITCH VALUES ARE ENTERED AS AN OCTAL NUMBER IN RESPONSE TO THE PROMPT FROM THE SWITCH ENTRY ROUTINE:

'SWR = NNNNNN NEW ='

EACH TIME SWITCH SETTINGS ARE ENTERED, THE ENTIRE SWITCH REGISTER IMAGE MUST BE ENTERED. LEADING ZEROS ARE NOT REQUIRED. 'RUBOUT' AND 'CONTROL U' FUNCTIONS MAY BE USED TO CORRECT TYPING ERRORS DURING SWITCH ENTRY.

ON PROCESSORS WITH HARDWARE SWITCH REGISTERS, THE 'SOFTWARE' SWITCH REGISTER MAY BE USED, IF THE PROGRAM FINDS ALL 1'S IN THE SWITCHES. ALL SWITCH REGISTER REFERENCES WILL BE TO THE 'SOFTWARE' REGISTER AND THE PROCEDURES DESCRIBED ABOVE MUST BE FOLLOWED.

SW<15>=1	HALT ON ERROR
SW<14>	NOT USED
SW<13>=1	INHIBIT ERROR TYPEOUT
SW<12>	NOT USED
SW<11>	NOT USED
SW<10>=1	BELL ON ERROR
SW<09>	NOT USED
SW<08>=1	INHIBIT END OF PASS MESSAGES
SW<07>=1	DISPLAY ALL DATA COMPARE ERRORS
SW<06>=1	DO NOT ALTER THE CURRENT OPERATION PARAMETERS
SW<05>=1	PARTIAL REGISTER DISPLAY IF ERROR; DO NOT DISPLAY ECC CORRECTION RESULTS
SW<04>=1	INHIBIT MAXIMUM ERROR COUNT CHECK; DO NOT DEASSIGN DRIVES WHEN END OF TEST IS REACHED
SW<03>=1	DISPLAY THE SECTOR IN ERROR (BEFORE RETRY ATTEMPTS) IF 'DCK', 'DTE', OR 'WCF' ERRORS OR AFTER THE 28TH RETRY IF UNCORRECTABLE 'DCK' ERROR
	IF DATA COMPARE ERRORS & SW<07> SET, DISPLAY REST OF BUFFER
SW<02>=1	INHIBIT SUBSYSTEM STATUS TYPEOUT DURING STARTUP
	INHIBIT PERFORMANCE REPORT AFTER SPECIFIED TIME
	PROMPT FE CYLINDER MESSAGE DURING AUTO TEST MODE
SW<01>=1	INHIBIT DATA COMPARE AFTER READ COMMAND, W/O ERROR
SW<00>=1	'READ ONLY' OR LOCKED 'READ ONLY' MODE

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
217
218
219
220
221
222
223
224
225
226
227
228

SW<00>

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

FOR EXAMPLE, IF THE PROGRAM IS STARTED AT ADDRESS 200. A DATA PATTERN WILL BE WRITTEN TO ALL ON-LINE DRIVES IN A SEQUENTIAL SEEK MODE. UPON COMPLETION OF THE WRITE, THE PROGRAM GOES INTO A TESTING MODE. HOWEVER, IF THE OPERATOR SWITCHES TO 'READ ONLY' MODE (SWO=1) JUST PRIOR TO OR DURING THE SEQUENTIAL WRITING OF THE DISK, THE PROGRAM WILL CONTINUE WRITING UNTIL THE SEQUENTIAL WRITE IS COMPLETED. UPON COMPLETION OF THE SEQUENTIAL WRITE, THE PROGRAM WILL SWITCH TO A 'READ ONLY' TESTING MODE UNTIL SWO IS RESET TO ZERO BY THE OPERATOR.

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

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

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

3.5 PASS/TEST TERMINATION

A PASS IN RANDOM 'T' COMMAND MODE OR SEQUENTIAL 'T' COMMAND MODE IS DETERMINED BY EITHER BITS READ OR SEEKS PERFORMED. THE NUMBER OF BITS OR SEEKS REQUIRED FOR A PASS IS DERIVED FROM EITHER THE SOFT ERROR RATE SPECIFICATION OR THE SEEK ERROR RATE SPECIFICATION.

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

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

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

3.5.1 PASS TERMINATION

END OF PASS FOR A SINGLE DRIVE IN THE RANDOM 'T' COMMAND MODE OR

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
274
275
276
277
278
279
280
281
282
283
284
285

SEQUENTIAL 'T' COMMAND MODE, IS DETERMINED BY ONE OF THE FOLLOWING CONDITIONS.

- A. IF PARAMETER 'ENDING' IS 1, END OF PASS OCCURS WHEN THE DRIVE HAS READ 3×10^9 BITS (1.875×10^8 WORDS). IT WILL TAKE APPROXIMATELY 3.33 PASSES TO REACH THE SOFT ERROR RATE OF 1×10^{10} BITS (6.25×10^8 WORDS) READ. HOWEVER, IT WILL TAKE 10. PASSES TO REACH THE 90% CONFIDENCE LEVEL OF 3×10^{10} BITS (1.875×10^9 WORDS) READ.
- B. IF PARAMETER 'ENDING' IS 0, END OF PASS OCCURS WHEN THE DRIVE HAS PERFORMED 1×10^6 SEEKS. IT WILL TAKE 1 PASS TO REACH THE SEEK ERROR RATE OF 1×10^6 SEEKS. HOWEVER, IT WILL TAKE 3 PASSES TO REACH THE 90% CONFIDENCE LEVEL OF 3×10^6 SEEKS.

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

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

3.5.2 TEST TERMINATION

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

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

3.6 RUN TIME

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

3.6.1 DATA TRANSFER MODE (DEFAULT)

1 DRIVE - APPROX. 1.75 HRS. (TO REACH 3×10^9 BITS OR 1.875×10^8 WORDS)
WITH SW<00> =1 AND SW<01> =1, THE PROGRAM WILL RUN APPROX. 20% FASTER

3.6.2 SEEK VERIFICATION MODE

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

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
338
339
340
341
342

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

3.7 DUAL PORT OPERATION

- A. LOAD THE PERFORMANCE EXERCISER PROGRAM INTO BOTH PROCESSORS.
- B. SWITCH THE 'CONTROLLER SELECT' SWITCH TO 'A/B' ON EACH DRIVE WHICH IS TO BE TESTED AS A DUAL PORT DRIVE AND CYCLE THE DRIVES UP.
- C. START THE PROGRAM IN EACH PROCESSOR. RUN THE PROGRAM AS THOUGH EACH PROCESSOR WERE RUNNING INDEPENDENTLY OF THE OTHER.

3.8 XXDP, ACT11, APT11 COMPATIBILITY

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

THIS PROGRAM IS ALSO, COMPATIBLE WITH XXDP IN DUMP AND CHAIN MODES, AND PROVIDES MEDIA PROTECTION IN THE CASE WHERE THE RM80 IS THE XXDP LOADING DEVICE.

AUTOMATIC MODE OR CHAIN MODE (MONITOR)

1. IF SW02 OF THE SWITCH REGISTER IS SET(1) WHEN THE PROGRAM IS STARTED AT 200 OR 204, THE OPERATOR IS ALLOWED TO CHOOSE BETWEEN EXERCISING THE USER PORTION OF THE DISK OR JUST THE FE CYLINDERS (SEE SECTION 4.1). IF SW02 IS CLEAR(0), ALL THE INPUT DIALOGUE IS BYPASSED AND THE TEST IS PERFORMED ON THE FE CYLINDERS ONLY.
2. THE BUS ADDRESS AND CONTROLLER INTERRUPT VECTOR ARE DEFAULTED TO 176700 AND 254 RESPECTIVELY.

DUMP MODE (NO MONITOR)

1. INPUT DIALOGUE PROMPTED AFTER PROGRAM STARTS

3.9 APT ETABLE DEFINITIONS

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

1. SOFTWARE ENVIRONMENT:

= 1 IF APT SCRIPT MODE
= 0 IF STANDLONE MODE

2. ENVIRONMENT MODE:

BIT 7 = 1 ETABLE DOES SIZING
= 0 PROGRAM DOES SIZING

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

BIT 5 = 1 SUPPRESS TTY CONSOLE OUTPUT

343
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
388
389
390
391
392
393
394
395
396
397
398
399

= 0 ALLOW TTY CONSOLE OUTPUT

BIT 4 TO BIT 0 ARE NOT USED

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

4. CONTROLLING THE PROGRAM

THE FOLLOWING KEYBOARD CONVENTIONS ARE USED BY THE KEYBOARD ENTRY
ROUTINES IN THE PROGRAM:

- A. TO DELETE AN INCORRECT CHARACTER FROM AN ENTRY STRING, TYPE A
'RUBOUT'. TYPING A RUBOUT WILL DELETE SUCESSIVE CHARACTERS
FROM THE INPUT.
- B. TO DELETE AN ENTIRE LINE, TYPE A 'CONTROL U' (^U).

- 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
445
446
447
448
449
450
451
452
453
454
455
456
- C. AN ENTRY MUST BE TERMINATED BY EITHER A 'CARRIAGE RETURN' OR A 'PERIOD'. THE 'PERIOD' TERMINATION IS RECOGNIZED BY THE PROGRAM AS A DEFAULT ENTRY REQUEST. WHEN A LINE IS TERMINATED BY A 'PERIOD' INSTEAD OF A 'CARRIAGE RETURN', THE PROGRAM WILL ACCEPT THE ENTERED VALUE AND WILL DEFAULT TO THE PRELOADED VALUES FOR ANY REMAINING ENTRIES.
- D. IF A 'CONTROL C' IS TYPED DURING DRIVE TESTING MODE, THE PROGRAM WILL ENTER THE COMMAND MODE. IF A 'CONTROL C' IS TYPED DURING 'ENTER COMMAND' SEQUENCE, WITH NO DRIVES ASSIGNED, THE PROGRAM WILL BE RESTARTED AT LOCATION 204. OTHERWISE, THE PROGRAM WILL RETURN TO 'ENTER COMMAND' PROMPT AND WAIT FOR A CORRECT SEQUENCE OF CHARACTERS. IF 'CONTROL C' IS TYPED DURING ANY OTHER ENTRY SEQUENCE, THE PROGRAM WILL RETURN TO THE BEGINNING OF THE GROUP SEQUENCE BEING ENTERED.

4.1 PARAMETERS

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

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

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

'! CUSTOMER DATA WILL BE OVERWRITTEN !'

'ARE YOU SURE (L) N ?'

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

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

'* EXERCISER WILL OPERATE ON FE CYLINDERS ONLY *'

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

'LOCKED IN READ ONLY MODE'

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

'CHANGE PARAMETERS (L) N ?'

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

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
503
504
505
506
507
508
509
510
511
512
513

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

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

THE FOLLOWING IS AN EXAMPLE DRIVE STATUS PRINTOUT:

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

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

4.1.1 KEYBOARD ENTRY PARAMETERS

NAME	BASE	DEFAULT VALUE	VALUE RANGE	FUNCTION
WRDCNT	10.	7936. (SEE NOTE)	6 - 7936.	CONTROLS THE MAXIMUM WORD COUNT USED FOR DATA TRANSFERS NOTE: THE PROGRAM WILL SELECT A MAXIMUM WORD COUNT, WHICH IS DETERMINED BY THE MEMORY AVAILABLE. THE MAX. WORD COUNT ASSIGNED BY THE PROGRAM IS 7936. (1 TRK) WORDS. THE OPERATOR MAY SPECIFY ANY OTHER MAX. WORD COUNT AS LONG AS THE VALUE SPECIFIED IS AT LEAST 6 WORDS BUT NO LARGER THAN 7936. WORDS OR MEMORY AVAILABLE. (WHICH EVER VALUE IS SMALLER)
INTRVL	10.	0	0 - 32767.	DETERMINES THE INTERVAL (IN MINUTES) BETWEEN AUTOMATIC PERFORMANCE SUMMARY TYPEOUTS; NO TYPEOUT IF THIS PARAMETER IS 0 OR IF SW<02> =1
PASSES	10.	1	1 - 32767.	NUMBER OF PASSES TO END OF TEST. (THIS PARAMETER IS NOT USED WHEN THE PROGRAM IS OPERATING IN AUTO RUN MODE)

514					
515	PATTERN	10.	0	0 - 15.	IF PARAMETER=0, DATA PATTERN
516					IS RANDOMLY SELECTED.
517					IF PARAMETER>0, SPECIFIES
518					ONE OF THE 15. PATTERNS.
519					THE SELECTED DATA PATTERN
520					IS POINTED BY THE PARAMETER
521					'PATTERN'. (SEE SECTION 8.4)
522					
523	RANDWC	8	000000	0 OR 1	IF PARAMETER = 0, THE WORD
524					COUNT IS RANDOMLY SELECTED
525					BETWEEN 6 AND THE VALUE
526					'WRDCNT'.
527					IF PARAMETER = 1, THE WORD
528					COUNT WILL BE THE VALUE
529					'WRDCNT'.
530					
531	RATIO	8	000002	0 - 7	CONTROLS THE APPROXIMATE
532					RATIO OF READ TO WRITE
533					COMMANDS.
534					
535					VALUE R/W RATIO
536					0 15/1
537					1 7/1
538					2 6/2
539					3 5/3
540					4 4/4
541					5 3/5
542					6 2/6
543					7 1/7
544					
545	ENDING	8	000001	0 OR 1	IF PARAMETER = 1, END OF PASS
546					DETERMINED BY THE 'WORDS READ'
547					COUNT.
548					IF PARAMETER = 0, END OF PASS
549					IS DETERMINED BY THE NUMBER
550					OF SEEKS.
551					
552	WRTCHK	8	000001	0 OR 1	IF EQ 1, DO AN APPROPRIATE
553					WRITE CHECK AFTER EACH WRITE
554					COMMAND. IF EQ 0, SELECT WRITE
555					CHECK COMMAND RANDOMLY.
556					
557	MESSAGE	8	000001	0 OR 1	IF PARAMETER =1, DO NOT PRINT
558					ERROR MESSAGES FOR DATA ERRORS
559					OCCURRING AT LOCATIONS DEFINED
560					BY THE OPERATOR AS BAD DISK
561					LOCATION.
562					IF PARAMETER = 0, PRINT ERROR
563					MESSAGES ASSOCIATED WITH BAD
564					DISK LOCATIONS.
565					
566	RANDOM	8	000000	0 OR 1	IF PARAMETER=0, RANDOM
567					DATA BLOCK ADDRESS IS
568					USED IN 'T' COMMAND
569					IF PARAMETER=1, SEQUENTIAL
570					DATA BLOCK IS USED IN

'T' COMMAND.

BADBLK 8 000000 0 OR 1

IF EQ TO 1, THE BAD SECTOR ENTRY TABLE WILL ALWAYS BE INITIALIZED WHEN ASSIGNING A DRIVE;

IF EQ TO 0, THE BAD SECTOR ENTRY TABLE WILL ONLY BE INITIALIZED IF THE HDA SERIAL NUMBER HAS CHANGED SINCE THE LAST TIME IT WAS READ.

NOTE: IF THE HDA SERIAL NUMBER HAS CHANGED, THIS MOST LIKELY MEANS THAT THE HDA OR DRIVE HAD BEEN REPLACED WHILE THE DRIVE WAS DEASSIGNED.

4.1.2 CHANGE DEVICE ADDRESSES

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

(DEFAULT ADDRESS = 176700, VECTOR = 254)

ADDRESS SELECTION EXAMPLES

EXAMPLE 1

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

EXAMPLE 2

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

4.2 KEYBOARD COMMANDS

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

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

IF THE PROGRAM WAS STARTED AT ADDRESS 204 OR IF NO DRIVES ARE ASSIGNED FOR TESTING, THE FOLLOWING MESSAGE WILL BE TYPE BEFORE

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
619
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
677
678
679
680
681
682
683
684

ENTERING THE COMMAND MODE. HOWEVER, IF A 'CONTROL C' IS TYPED WHILE TESTING IS IN PROGRESS, THE FOLLOWING MESSAGE WILL BE OMITTED AND THE PROGRAM WILL ENTER COMMAND MODE.

'NO DRIVES ASSIGNED'

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

'HH:MM:SS
ENTER COMMANDS:'

THE PROGRAM WILL THEN ACCEPT ANY OF THE VALID COMMANDS. AT THE COMPLETION OF A COMMAND, THE PROGRAM WILL EXIT COMMAND MODE AND TRY TO ASSIGN THE DRIVE(S) THAT WERE REQUESTED. IF THE DRIVE(S) CANNOT BE ASSIGNED, ONE OF THE FOLLOWING ERROR MESSAGES WILL BE REPORTED AND THE PROCESS CONTINUES FOR EACH DRIVE.

RESPONSE	COMMAND(S)
-----	-----
?DRIVE N LOAD DEVICE	T, W, R, WT
?DRIVE N OFFLINE	T, W, R, WT
?DRIVE N NOT ASSIGNED	D, S
?DRIVE N ALREADY ASSIGNED	T, W, R, WT
?DRIVE N NOT PRESENT	T, W, R, WT
?DRIVE N UNSAFE	T, W, R, WT
?DRIVE N NOT AN RM80	T, W, R, WT

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

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

'CHANGE DRIVE PARAMETERS (L) N ?'

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

THE PROGRAM WILL FIRST TELL THE OPERATOR WHICH DRIVE IS BEING REFERENCED FOR CHANGES.

'***** DRIVE # N'

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

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

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

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
735
736
737
738
739
740
741

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

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

'ENTER ADDRESS LIMITS:'

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

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

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

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

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

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

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

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

742
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
793
794
795
796
797
798

TYPEOUT:

'ENTER BAD SECTR ADRS:'

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

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

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

B. LEADING ZEROS ARE NOT REQUIRED.

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

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

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

EXAMPLE 3: CYL,TRK,SEC= C<CR>

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

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

IF CONTROL C (^C) IS TYPED AS AN ENTP, ALL CURRENT BAD SECTOR ENTRIES WILL BE LOST AND THE FOLLOWING MESSAGE WILL BE TYPED.

'* ALL CURRENT ENTRIES LOST *'

AFTER TYPING THE PREVIOUS MESSAGE, THE PROGRAM WILL WAIT FOR THE OPERATOR TO ENTER ANOTHER BAD SECTOR AS IN THE PREVIOUS EXAMPLES.

IF 'L' IS TYPED FOR AN INPUT CHARACTER, THE PROGRAM WILL TYPE A LIST OF DEC144 BAD SECTORS, AND THE MANUALLY ENTERED BAD SECTORS, WHICH ARE STORED IN THE DRIVE PARAMETER TABLE(DPB) FOR THAT PARTICULAR DRIVE.

IF THERE ARE NO BAD SECTORS IN THE DPB TABLE, THE FOLLOWING MESSAGE WILL BE TYPED:

'DEC144 AND MANUAL BAD SECTOR LIST
* NO ENTRIES *'

HOWEVER, IF THERE ARE ENTRIES IN DPB TABLE, THE LIST WILL BE TYPED IN THE FOLLOWING FORMAT:

'DEC144 AND MANUAL BAD SECTOR LIST
8,8,3
16,13
256
500,1,29'

THE ABOVE LIST OF BAD SECTORS, ENTRY 1 INDICATES THAT CYLINDER 8., TRACK 8., SECTOR 3 IS THE BAD SECTOR. ENTRY 2 INDICATES THAT ON

799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
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

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

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

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

4.2.1 'T' COMMAND

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

FORMAT: TN<CR>

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

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

4.2.2 'D' COMMAND

USED TO DEASSIGN A DRIVE(S) BEING EXERCISED.

FORMAT: DN<CR>

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

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

4.2.3 'S' COMMAND

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

FORMAT: SN<CR>

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

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

4.2.4 'W' COMMAND

USED TO PERFORM A SEQUENTIAL WRITE OF THE DISK, WITH DATA ACCEPTABLE TO THE PERFORMANCE EXERCISER PROGRAM.

856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
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

FORMAT: WN<CR>

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

EXAMPLE: WO<CR> - WRITE A DATA PATTERN ON DRIVE 0.
WA<CR> - WRITE A DATA PATTERN ON ALL AVAILABLE DRIVES.

4.2.5 'R' COMMAND

USED TO PERFORM A SEQUENTIAL READ OF THE DISK.

FORMAT: RN<CR>

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

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

4.2.6 'WT' COMMAND

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

FORMAT: WTN<CR>

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

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

5. PERFORMANCE SUMMARY TYPEOUT

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

THE SUMMARY TYPEOUT CONTAINS THE FOLLOWING FIELDS:

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

913 'HARD' NUMBER OF HARD DATA ERRORS
 914 'SKI' NUMBER OF 'SKI' ERRORS
 915 'MISP' NUMBER OF PROGRAM DETECTED POSITIONING ERRORS
 916 'OTHER' TOTAL ERRORS OF OTHER TYPES
 917
 918 ALL DATA TRANSFER COUNTS, SEEK COUNTS AND ERROR COUNTS ARE ACCUMULATIVE
 919 AND WILL NOT BE CLEARED AFTER EACH PASS.
 920
 921 TO CALCULATE THE TOTAL NUMBER WORDS READ OR WRITTEN, TAKE THE OVERFLOW
 922 COUNT (RD OFLOW OR WT OFLOW), MULTIPLY IT BY 2,147,483,647., THEN ADD
 923 THAT NUMBER TO THE WORDS READ OR WRITTEN (WRDS READ OR WRDS WRITN).
 924
 925 NOTE: ERRORS EM1, EM2, & EM5 ARE NOT INCLUDED IN THE 'OTHER' ERROR
 926 TOTAL.
 927
 928 5.2 SOFT/HARD ERROR DEFINITIONS
 929
 930 5.2.1 HARD ERRORS
 931
 932 A. A 'DTE' (DRIVE TIMING ERROR) OR A 'DCK' (DATA CHECK ERROR)
 933 WHICH OCCURS DURING A READ DATA OR A READ HEADER & DATA OPERATION
 934 AND IS NOT CORRECTABLE OR DOES NOT BECOME CORRECTABLE AFTER THE
 935 PROGRAM HAS PERFORMED THE COMPLETE RETRY SEQUENCE ON THE BAD
 936 SECTOR.
 937
 938 THE RETRY SEQUENCE IS 16. RE-READS.
 939
 940
 941 5.2.2 SOFT ERRORS
 942
 943 A. ECC CORRECTABLE 'DCK' ERRORS.
 944 B. 'DCK' & 'ECH' ERRORS WHICH BECOME ECC CORRECTABLE DURING
 945 RETRY OR WHICH ARE READ CORRECTLY DURING RETRY.
 946 C. HEADER READ ERRORS - READ DATA, READ HEADER & DATA, OR
 947 WRITE DATA COMMANDS
 948 D. 'DTE' ERRORS WHICH ARE CORRECTED OR WHICH BECOME ECC CORRECTABLE
 949 'DCK' ERROR DURING THE RETRY SEQUENCE.
 950
 951
 952 6. DATA CHECKING & ERROR RECOVERY
 953 -----
 954
 955 6.1 DATA COMPARISON
 956
 957 DATA COMPARISON OCCURS AFTER EACH 'RDDAT' (READ DATA) OR 'RDHD'
 958 (READ HEADER AND DATA) OPERATION UNDER THE FOLLOWING CONDITIONS:
 959
 960 A. THE COMMAND TERMINATED WITH NO ERRORS AND SW<01>=0
 961
 962 B. THE OPERATION TERMINATED WITH 'DCK' SET AND THE ERROR IS ECC
 963 CORRECTABLE OR THE SECTOR IN ERROR IS READ CORRECTLY AFTER
 964 RETRY ATTEMPTS.
 965
 966 6.2 VERIFICATION OF DATA WRITTEN
 967
 968 DATA VERIFICATION IS DONE EITHER THROUGH READING THE DATA BACK
 969

970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026

AND MATCHING THE DATA WITH ONE OF THE 15. PATTERNS OR THROUGH
ISSUING A WRITE CHECK COMMAND AFTER DOING A WRITE DATA COMMAND.

6.3 BAD ADDRESS FLAGGING

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

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

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

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

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

7. ERROR MESSAGES

ERRORS ARE REPORTED ON THE TTY CONSOLE. THE PROGRAM CONTAINS
NO CODED ERROR HALTS. IF THE PROGRAM HALTS (ASSUMING, OF COURSE,
THAT SW<15> IS NOT SET), AN UNRECOVERABLE PROGRAM CONDITION HAS
OCCURRED OR A CENTRAL PROCESSOR FAILURE HAS OCCURRED.

ERROR MESSAGES ARE MADE UP OF SEVERAL LINES. EACH TYPE OF ERROR
HAS SEVERAL OPTIONAL LINES WHICH MAY APPEAR WITH IT. ALL OF THE
POSSIBLE ERROR MESSAGE LINES WHICH MAY APPEAR ARE GIVEN IN THE
SECTION DESCRIBING THE PARTICULAR ERROR HEADER.

7.1 ERROR DESCRIPTION LINES

(THE MESSAGE TAGS ARE GIVEN FOR REFERENCE.)

MESSAGE

TAG	TEXT
---	----

EM1 RH CONTROLLER INTERRUPT OCCURRED (RMAS=0)

THE RH CONTROLLER INTERRUPTED AND THE ATTENTION SUMMARY
REGISTER (RMAS) WAS CLEARED.

EM2 UNEXPECTED ATTENTION OCCURRED

THE INDICATED DRIVE INTERRUPTED BUT THE DRIVE WAS NOT
PERFORMING AN OPERATION.

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

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

1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197

A WRITE OPERATION WAS ATTEMPTED BUT THE DRIVE WAS WRITE LOCKED.

EM40 RH CONTROLLER OR UNIBUS TRANSFER ERROR

'TRE' IS SET IN THE RH CONTROL REGISTER AND NO DRIVE ERROR HAS OCCURRED. THE OPERATION WILL BE RETRIED 3 TIMES IF THE ERROR WAS CAUSED BY 'DLT', 'UPE', 'MXF', OR 'MDPE'.

EM41 BUS ADDRESS OR WORD COUNT INCORRECT

NO DRIVE ERROR OCCURRED BUT EITHER THE BUS ADDRESS INDICATES THAT AN INCORRECT NUMBER OF WORDS WERE TRANSFERED OR THE WORD COUNT REGISTER IS NOT ZERO.

EM42 DATA COMPARE ERRORS - NO DRIVE ERROR DETECTED

NO SUBSYSTEM ERROR WAS SIGNALLED; HOWEVER, THE DATA DOES NOT COMPARE.

EM43 CAN'T MATCH DATA READ WITH A PATTERN - UNDEFINED DATA PATTERN

THE DATA IN THE BUFFER DOES NOT MATCH ANY OF THE STANDARD PATTERNS.

EM44 ERROR BIT(S) SET, BUT NO ERROR SIGNALLED BY THE RH CONTROLLER

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

EM45 ECC LOGIC FAILURE - POSITION REGISTER VALUE NOT VALID

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

EM46 BUS ADDRESS OR WORD COUNT NOT CONSISTENT

THE PROGRAM WAS PROCESSING AN ERROR AND FOUND THAT THE NUMBER OF WORDS TRANSFERED AS INDICATED BY THE BUS ADDRESS REGISTER DOES NOT AGREE WITH THE TRANSFER COUNT FROM THE WORD COUNT REGISTER.

EM47 ECC LOGIC FAILURE - PATTERN REGISTER IS ZERO

DURING 'DCK' ERROR PROCESSING, THE CONTENTS OF ECC PATTERN REGISTER (RMEC2) WAS NOT VALID. THE PATTERN REGISTER CONTAINED ALL ZEROS.

EM50 SEEK INCOMPLETE ERROR

THE DRIVE SIGNALLED EITHER 'SKI' ERROR.

EM51 NOT USED

1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254

EM52 ECH ERROR - UNCORRECTABLE ECC ERROR

DURING 'DCK' ERROR PROCESSING, THE 'ECH' BIT WAS SET IN RMER1, WHICH INDICATES THAT DATA CHECK CANNOT BE ECC CORRECTED.

EM60 DEVICE UNSAFE

THE INDICATED DRIVE UNSAFE ERROR OCCURRED; THE ERROR WAS CLEARED BY A 'DRIVE CLEAR' INSTRUCTION.

7.2 DETAIL ERROR LINES

THE LINE NUMBERS GIVEN BELOW ARE FOR REFERENCE ONLY.

LINE 1

HH:MM:SS

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

LINE 2

'PRSNT COMMAND= XXXX PREV COMMAND= YYYY'

MNEMONICS USED FOR THE COMMANDS ARE DEFINED BELOW:

SEEK	- SEEK (OCTAL 5)
RECAL	- RECALIBRATE (OCTAL 7)
DRVCLR	- DRIVE CLEAR (OCTAL 11)
RELSE	- RELEASE (OCTAL 13)
OFFSET	- OFFSET (OCTAL 15)
RTC	- RETURN TO CENTERLINE (OCTAL 17)
READIN	- READIN PRESET (OCTAL 21)
PACK	- PACK ACKNOWLEDGE (OCTAL 23)
SEARCH	- SEARCH (OCTAL 31)
*GETREG	- GET REGISTERS (OCTAL 41)
*SETFMT	- SET FORMAT (ECI OR HCI) (OCTAL 43)
*SELDRV	- SELECT DRIVE (OCTAL 45)
WCKD	- WRITE CHECK DATA (OCTAL 51)
WCKHD	- WRITE CHECK HEADER & DATA (OCTAL 53)
WRDAT	- WRITE DATA (OCTAL 61)
WRTHD	- WRITE CHECK HEADER & DATA (OCTAL 63)
RDDAT	- READ DATA (OCTAL 71)
RDHD	- READ HEADER & DATA (OCTAL 73)

* SPECIAL RM DRIVER COMMAND (NOT A CONTROLLER COMMAND)

(DISPLAY OF THE RH/RM REGISTERS IN TWO GROUPS:
RMCS1, RMCS2, RMDS, RMER1, RMER2, RMEC1 AND RMEC2 FORM THE FIRST
GROUP; ALL THE OTHER REGISTERS ARE IN THE SECOND GROUP.
IF SW<05> IS SET, ONLY THE REGISTERS IN THE FIRST GROUP WILL BE

1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311

DISPLAYED.)

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

'* ERROR AT BAD TRACK/SECTOR'

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

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

LINE 3

ERROR AT CXXX TYY SZZ PREV ADDR= CUUU TVV SWW

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

LINE 4

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

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

LINE 5

START CYL= XXX END CYL= YYY

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

LINE 6

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

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

LINE 7

RMBA= XXXX RMWC= YYYY

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

1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425

LINE 14

ECC CORRECTABLE

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

LINE 15

READ CORRECTLY

THE SECTOR IN ERROR WAS READ WITHOUT ERROR.

LINE 16

ECC CORRECTABLE

THE SECTOR IN ERROR BECAME ECC CORRECTABLE

LINE 17

CORRECTED ON X RETRY

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

LINE 18

UNCORRECTABLE AFTER X RETRIES

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

LINE 19

DIFFERENT ERROR DURING RETRY

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

LINE 20

DATA COMPARISON ERRORS

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

LINE 21

1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482

TOTAL COMPARE ERRORS= XXXX

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

LINE 22

THE DATA COMPARED OK

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

LINE 23

ECC CORRECTION RESULTS

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

LINE 24

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

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

LINE 25

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

THE DATA ERROR WAS NOT IN DATA TRANSFERED TO MEMORY.

LINE 26

CONTENTS OF THE ERROR SECTOR (REPORTED ABOVE)

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

LINE 27

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

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

1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539

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

'WOFL' NUMBER OF TIMES 'WRDS WRITN' HAS OVERFLOWED
'WRDS WRITN' IS THE TOTAL NUMBER OF WORDS WRITTEN THE DRIVE.

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

LINE 28

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

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

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

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

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

8. PROGRAM DESCRIPTION -----

8.1 PROGRAM OPERATION

WHEN THE PROGRAM IS STARTED, PROVIDING APT TTY ENABLE BIT IS SET OR DIAGNOSTIC LOADED BY OTHER THAN APT SCRIPT MODE, ALL TABLES AND PARAMETERS ARE CLEARED OR INITIALIZED. THE PARAMETERS WHICH ARE UNDER OPERATOR TTY ENTRY CONTROL ARE CHECKED FOR VALIDITY AND CONSISTENCY. RH CONTROLLER INTERRUPT ENABLE ('IE') IS SET, TTY KEYBOARD INTERRUPT ENABLE IS SET, AND THE KW11-L OR KW11-P CLOCK IS STARTED. COMMAND ENTRIES WILL NOW BE ACCEPTED BY THE PROGRAM.

THE PROGRAM SCANS ITS INTERNAL ASSIGNMENT TABLES, LOOKING FOR:

- 1) DRIVES TO ASSIGN/DEASSIGN
- 2) PERFORMANCE SUMMARY TYPEOUT REQUESTS
- 3) DRIVES REQUIRING COMMAND INITIATION, BUFFER ASSIGNMENT, OR PARAMETER SELECTION.
- 4) DRIVES COMPLETING CURRENT OPERATIONS.

THE PROGRAM CONTINUES SCANNING ITS TABLES UNTIL AN ENTRY IS FOUND. IN THE CASE OF THE PROGRAM AT INITIAL START, THE FIRST ENTRY WILL BE MADE BY THE OPERATOR WHEN A DRIVE IS ASSIGNED ('T' COMMAND).

WHEN A DRIVE IS ASSIGNED, THE KEYBOARD ENTRY ROUTINE VERIFIES THAT THE DRIVE IS PRESENT, IS AN RM80, AND IS ONLINE. THE ASSIGNMENT ROUTINE THEN ISSUES A 'READIN PRESET' INSTRUCTION, SETS 'FMT16', AND ISSUES A 'RECALIBRATE' INSTRUCTION.

1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596

PARAMETERS FOR THE OPERATION ARE SELECTED AND A BUFFER IS ASSIGNED. IF THE OPERATION IS A WRITE OR WRITE CHECK COMMAND, THE ASSIGNED BUFFER WILL BE FILLED WITH THE SELECTED PATTERN. (WRITE CHECK COMMANDS ARE ISSUED AFTER EACH WRITE COMMAND. THE WRITE CHECK COMMAND USES THE PARAMETERS SELECTED FOR THE PRECEDING WRITE COMMAND.) CONTROL IS THEN PASSED TO THE COMMAND INITIATION ROUTINE.

THE COMMAND INITIATION ROUTINE FIRST LOOKS AT THE CYLINDER ADDRESS OF THE REQUESTED OPERATION. IF THE DRIVE MUST SEEK TO ANOTHER CYLINDER TO PERFORM THE OPERATION, THE PROGRAM ISSUES A SEARCH INSTRUCTION TO THE DRIVE WITH A 'TARGET' SECTOR WHICH IS 5 SECTORS EARLIER THAN THE 'TRANSFER' SECTOR. (THIS ALLOWS THE PROGRAM TO INITIATE OPERATIONS ON ANOTHER DRIVE WHILE THE PRESENT DRIVE, OR OTHER DRIVES, ARE SEARCHING FOR 'TARGET' SECTORS. ALL SEEKS ISSUED BY THE PROGRAM ARE IMPLIED SEEK SEARCH OPERATIONS.) WHEN A SEARCHING DRIVE FINDS THE 'TARGET' SECTOR AND INTERRUPTS, THE PROGRAM THEN ISSUES THE REQUESTED COMMAND TO THE DRIVE THAT INTERRUPTED.

WHEN THE DATA TRANSFER OPERATION IS COMPLETE, THE DRIVE REGISTERS ARE STORED AND A DATA TRANSFER IS INITIATED FOR A WAITING DRIVE.

IF THE OPERATION HAS BEEN COMPLETED NORMALLY, THE SAVED DRIVE REGISTERS ARE CHECKED TO VERIFY THAT NO ERROR BITS ARE SET; THE RH CONTROLLER BUS ADDRESS AND WORD COUNT ADDRESS REGISTERS ARE CHECKED TO VERIFY THAT THE CORRECT NUMBER OF WORDS HAVE BEEN TRANSFERED AND THAT THE TWO REGISTERS ARE CONSISTENT WITH EACH OTHER; AND IF THE COMMAND WAS A READ COMMAND, THE DATA BUFFER IS COMPARED. WHEN THIS SEQUENCE IS COMPLETED, THE DRIVE IS RETURNED TO THE ASSIGNED, INACTIVE LIST. THE PROGRAM THEN INITIATES A DATA TRANSFER ON A WAITING DRIVE AND RESELECTS AND REINITIATES ANOTHER OPERATION ON THE RELEASED DRIVE.

ERRORS WHICH OCCUR ARE PROCESSED IN THE FOLLOWING ORDER. MULTIPLE ERRORS WILL BE REPORTED AS THE FIRST ERROR TYPE CHECKED.

A. ERRORS REPORTED FOR OPERATIONS WHICH HAVE NOT COMPLETED NORMALLY.

OPERATION NOT COMPLETED WITHIN TIME LIMIT - EM13
DRIVE WENT OFFLINE - EM14

B. ERRORS REPORTED FOR OPERATIONS WHICH COMPLETE NORMALLY.

CORRECTABLE UNSAFE - EM60
DRIVE TIMING ERROR - EM32
DATA CHECK ERROR - EM21
WRITE CHECK WITH DCK SET - EM22
HEADER CRC ERRORS - EM20
FORMAT ERRORS - EM24, EM26
HEADER COMPARE ERRORS - EM25, EM27
PROGRAM DETECTED POSITIONING ERROR - EM51
SEEK INCOMPLETE ERROR - EM50
WRITE CHECK WITHOUT 'DCK' SET - EM23
RH CONTROLLER OR UNIBUS TRANSFER ERROR - EM40
'OPI' ERROR - EM31
'PAR' ERROR - EM33
'WCF' ERROR - EM34
'IAE' ERROR - EM35
'WLE' ERROR - EM36

1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653

MISCELLANEOUS DRIVE ERROR - EM30

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

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

8.2 DUAL PORT OPERATION

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

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

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

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

8.3 SELECTION OF OPERATION VARIABLES

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

1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710

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

- B. THE WORD COUNT IS RANDOMLY SELECTED BETWEEN 6 AND THE VALUE 'WRDCNT'. THIS IS NECESSARY AS THE PROGRAM REQUIRES 4 LOCATIONS IN THE DATA PORTION OF THE SECTOR TO BE ABLE TO MATCH THE DATA TO A PATTERN FOR DATA COMPARISON PURPOSES AND NEEDS 2 MORE LOCATIONS IF A READ HEADER & DATA COMMAND IS ISSUED.
- C. THE DATA WRITTEN IS RANDOMLY SELECTED AMONG THE 15. STANDARD PATTERNS. THE PARAMETER 'PATTERN' ENABLES THE RANDOM PATTERN SELECTION, IF THIS PARAMETER IS 0.
- D. THE COMMANDS ARE SELECTED RANDOMLY. WRITE CHECK DATA COMMAND IS PERFORMED ONLY IF THE PREVIOUS COMMAND WAS THE APPROPRIATE WRITE DATA COMMAND.

8.4 DATA PATTERNS

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

PAT 1	PAT 2	PAT 3	PAT 4	PAT 5	PAT 6	PAT 7	*PAT 8
000001	177776	000000	133331	052525	155555	026455	155555
000003	177774	000000	133331	052525	155555	026455	133333
000007	177770	000000	133331	052525	155555	026455	155555
000017	177760	177777	133331	125252	155555	151322	133333
000037	177740	177777	133331	125252	155555	151322	155555
000077	177700	177777	133331	125252	155555	151322	133333
000177	177600	000000	133331	052525	155555	026455	155555
000377	177400	000000	133331	052525	155555	026455	133333
000777	177000	177777	133331	125252	155555	151322	155555
001777	176000	177777	133331	125252	155555	151322	133333
003777	174000	000000	133331	052525	155555	026455	155555
007777	170000	177777	133331	125252	155555	151322	133333
017777	160000	000000	133331	052525	155555	026455	155555
037777	140000	177777	133331	125252	155555	151322	133333
077777	100000	000000	133331	052525	155555	026455	155555
177777	000000	177777	133331	125252	155555	151322	133333
PAT 9	PAT 10	PAT 11	PAT 12	PAT 13	PAT 14	PAT 15	
000001	177776	172666	077777	153333	000000	177777	
000002	177775	155555	137777	066667	177777	000000	
000004	177773	172666	157777	153333	177777	000000	
000010	177767	155555	167777	066667	177777	000000	
000020	177757	172666	173777	153333	177777	000000	

1711	000040	177737	155555	175777	066667	177777	000000
1712	000100	177677	172666	176777	153333	177777	000000
1713	000200	177577	155555	177377	066667	177777	000000
1714	000400	177377	172666	177577	153333	177777	000000
1715	001000	176777	155555	177677	066667	177777	000000
1716	002000	175777	172666	177737	153333	177777	000000
1717	004000	173777	155555	177757	066667	177777	000000
1718	010000	167777	172666	177767	153333	177777	000000
1719	020000	157777	155555	177773	066667	177777	000000
1720	040000	137777	172666	177775	153333	177777	000000
1721	100000	077777	155555	177776	066667	177777	000000

* WORST CASE PATTERN

9.1 RH/RM DRIVER

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

9.2 TO INITIALIZE THE DRIVER:

```
JSR   PC RMINIT
RETURN
```

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

DRVSTA	DRIVE STATE
-----	-----
>0	ONLINE
=0	OFFLINE, DRIVE IS NOT AN RM80, OR NONEXISTENT DRIVE
<0	UNSAFE

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

DRVTYP	CONDITION
-----	-----
0	NONEXISTENT DRIVE
1	RM80
-1	NOT AN RM80

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

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

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

1767

```

1768 RETURN1 ;RETURN IF QUEUE IS FULL
1769 RETURN2 ;RETURN IF REQUEST IS IN
1770 ;QUEUE OR THERE IS AN
1771 ;ERROR CONDITION
1772
1773 *DPB (DATA PARAMETER BLOCK)
1774
1775 PNTDPB: .BYTE 0 ;(0) DRIVE NUMBER
1776 .BYTE 0 ;(1) OFFSET VALUE OR FMT16, ECT, AND HCI
1777 .BYTE 0 ;(2) COMMAND
1778 .BYTE 0 ;(3) PSEL AND A17 AND A16
1779 .WORD 0 ;(4) WORD COUNT (MUST BE NEG.)
1780 .WORD 0 ;(6) BUFFER ADDRESS OR
1781 ;REGISTER TABLE POINTER
1782 .BYTE 0 ;(10) SECTOR ADDRESS OR
1783 ;FIRST REG. INDEX
1784 .BYTE 0 ;(11) TRACK ADDRESS OR
1785 ;LAST REG. INDEX
1786 .WORD 0 ;(12) CYLINDER ADDRESS
1787 .WORD 0 ;(14) ERROR TABLE POINTER
1788 ;POINTS TO THE FIRST OF TWENTY
1789 ;LOCATIONS OF WHERE THE DRIVER
1790 ;IS TO STORE THE RH/RM
1791 ;REGISTERS ON AN ERROR. IF LEFT
1792 ;ZERO REGISTERS ARE NOT SAVED.
1793 .WORD 0 ;(16) STATUS/ERROR INDICATOR
1794 ;BIT15=1=ERROR OCCURRED
1795 ;BIT07=1=DONE
1796 ;BIT14-BIT09 AND BIT06-BIT03
1797 ;INDICATE TYPE OF ERROR
1798
1799

```

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

```

1800 MOV #16.,-(SP) ;16. MILLISECONDS BETWEEN
1801 ;CLOCK TICKS
1802 JSR PC,RMTMR ;CALL THE TIMER ROUTINE
1803
1804
1805
1806

```

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

9.4.1 EXAMPLE - WRITE 1000. WORDS

```

1810
1811 1$: JSR R0,RM80 ;CALL THE DRIVER
1812 WRTDPB ;DPB ADDRESS
1813 BR 1$ ;WAIT FOR QUEUE IF FULL
1814 2$: TST WRTDPB+16 ;WAIT FOR COMMAND TO COMPLETE
1815 BEQ 2$
1816 BMI ERROR1 ;ERROR OCCURRED
1817 .
1818 .
1819 .
1820
1821 WRTDPB: .BYTE 5 ;DRIVE #5
1822 .BYTE 0 ;
1823
1824

```

```

1825 .BYTE 161 ;WRITE COMMAND
1826 .BYTE 0 ;
1827 .WORD -1000. ;WORD COUNT
1828 .WORD WRIBUF ;BUFFER ADDRESS
1829 .BYTE 3 ;SECTOR
1830 .BYTE 5 ;TRACK
1831 .WORD 400 ;CYLINDER
1832 .WORD ERRTB5 ;ERROR TABLE
1833 .WORD 0 ;STATUS/ERROR INDICATOR
    
```

ALTERNATE DPB SETUP

```

1834
1835
1836
1837 WRTDPB: .WORD 5 ;THIS SETUP ACHIEVED
1838 .WORD WRITE ;EVERYTHING THE
1839 .WORD -1000. ;ABOVE TABLE DID, BUT
1840 .WORD WRIBUF ;IN A CLEANER FORMAT
1841 .BYTE 3,5
1842 .WORD 400,ERRTB5,0
1843
    
```

9.5 RH/RM REGISTERS

	MNEMONIC	INDEX
1844		
1845		
1846		
1847		
1848	RMCS1	0
1849	RMWC	2
1850	RMBA	4
1851	RMDA	6
1852	RMCS2	10
1853	RMDS	12
1854	RMER1	14
1855	RMAS	16
1856	RMLA	20
1857	RMDB	22
1858	RMMR1	24
1859	RMDT	26
1860	RMSN	30
1861	RMOF	32
1862	RMDC	34
1863	RMHR	36
1864	RMMR2	40
1865	RMER2	42
1866	RMEC1	44
1867	RMEC2	46
1868	*RMBAE	50
1869	*RMCS3	52

* RH70 CONTROLLER REGISTERS

9.6 COMMANDS PERFORMED BY THE DRIVER

	COMMAND	CODE	COMMAND TYPE
1870			
1871			
1872			
1873			
1874			
1875			
1876			
1877			
1878	SEEK	105	P
1879	RECALIRATE	107	P
1880	DRIVE CLEAR	111	N
1881	RELEASE	113	N

1882	OFFSET	115	
1883	RETURN TO CENTER	117	P
1884	READIN PRESET	121	P
1885	PACK ACKNOWLEDGE	123	N
1886	SEARCH	131	P
1887	GET REGISTER(S)	141	S
1888	SET FORMAT	143	S
1889	SELECT DRIVE	145	S
1890	WRITE CHECK DATA	151	D
1891	WRITE CHK HEADER & DATA	153	D
1892	WRITE DATA	161	D
1893	WRITE HEADER & DATA	163	D
1894	READ DATA	171	D
1895	READ HEADER & DATA	173	D

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

9.7 DPB STATUS/ERROR INDICATOR WORD

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

BIT NO.	MEANING IF ON A '1'
-----	-----
15	ERROR OCCURRED DONE (BIT07=0); BITS 14-9 SPECIFIES TYPE DONE (BIT07=1); BITS 6-3 SPECIFIES TYPE
14(1)	USER MADE A REQUEST FOR A FUNCTION TO BE PERFORMED ON AN OFFLINE OR UNSAFE DRIVE
9(3)(4)	SOFTWARE TIMEOUT OCCURRED ON THIS DRIVE
7	DONE
6(2)	ERROR OCCURRED DURING AN I/O OPERATION
5(2)	ERROR OCCURRED DURING AN OPERATION OTHER THAN I/O.
4(2)	CORRECTABLE UNSAFE CONDITION OCCURRED
2	PORT REQUEST TIMEOUT. THE DRIVER REQUESTED THE DRIVE BUT THE OPPOSITE PORT DID NOT RELEASE THE DRIVE WITHIN 15. SECONDS.
1	NON-EXISTENT DRIVE REQUESTED. USER MADE A REQUEST FOR A NON-EXISTENT DRIVE.

NOTES FOR ABOVE

1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938

1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994

-
- (1) = REQUEST WASN'T PUT IN QUEUE. (RH/RM REGISTERS WERE NOT SAVED)
 - (2) = REQUEST QUEUE HAS BEEN EMPTIED. THE DRIVER ISSUED A 'DRIVE CLEAR' TO THE DRIVE. NOTE: ALL RH/RM REGISTERS ARE SAVED AS PER DPB+14 BEFORE THE 'DRIVE CLEAR'.
 - (3) = REQUEST QUEUE HAS BEEN EMPTIED. THE DRIVER ISSUED A MASSBUS INIT. ALL RH/RM REGISTERS FOR THE DRIVE WERE SAVED AS PER DPB+14 BEFORE THE INIT.
 - (4) = A 'RECALIBRATE' SHOULD BE ISSUED BEFORE ANY OTHER COMMAND.

9.8 ERROR CALLS MADE BY THE DRIVER.

THERE ARE A FEW ERRORS THAT CAN OCCUR THAT CAN NOT BE INDICATED IN A DPB. WHEN THIS TYPE OF ERROR IS DETECTED BY THE DRIVER IT WILL MAKE AN ERROR CALL OF THE FORM 'ERROR N', WHERE 'N' IS THE ERROR NUMBER AND THE ERROR WILL BE AN EMT INSTRUCTION.

N	TYPE	DATA AVAILABLE
-	----	-----
1	RH70 INTERRUPT OCCURRED (RHAS=0)	*R4= RMCS1'S ADDRESS
2	UNEXPECTED ATTENTION OCCURRED	R1= DRIVE NUMBER R3= ATA BIT *R4= RMCS1'S ADDRESS R5= (RMAS) RMERRS =RMDS RMERRS+2=RMER1 RMERRS+4=RMER2 RMERRS+6=RMMR2
3	NOT USED	
4	NOT USED	
5	ADDRESS PLUG CHANGE BIT SET ('OPE' ERROR)	R1= DRIVE NUMBER R3= ATA BIT *R4= RMCS1'S ADDRESS R5= (RMAS) RMERRS =RMDS RMERRS+2=RMER1 RMERRS+4=RMER2

* THIS IS THE ACTUAL UNIBUS ADDRESS (176700)

1
56
57

58

59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76

```

:*LAST REVISION 23-DEC-81

.TITLE CZRMAAO RM80 PERF EXER
:*COPYRIGHT (C) 1982
:*DIGITAL EQUIPMENT CORPORATION
:*COLORADO SPGS., CO. 80919
:*
:*PROGRAM BY MIKE LEAVITT
:*
:*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
:*PACKAGE (MAINDEC-11-DZQAC-C5), 18-MAR-81
:*
.SBTTL OPERATIONAL SWITCH SETTINGS
:*
:*      SWITCH      USE
:*      -----      -----
:*      15          HALT ON ERROR
:*      13          INHIBIT ERROR TYPEOUTS
:*      10          BELL ON ERROR
:*      8           INHIBIT END OF PASS MESSAGES
:*      7           DISPLAY ALL DATA COMPARE ERRORS
:*      6           DON'T CHANGE PARAMETERS (LOOP ON PRESENT VALUES)
:*      5           A. PARTIAL REGISTER DISPLAY IF ERROR
:*                  B. NO ECC CORRECTION RESULTS DISPLAYED IF ERROR
:*      4           A. DO NOT CHECK FOR MAXIMUM ERROR COUNTS
:*                  B. DO NOT DROP DRIVE AT END OF TEST
:*      3           A. DISPLAY ERROR SECTOR IF 'DCK', 'DTE', OR 'WCF' ERROR
:*                  B. DISPLAY SECTOR IF 'DCK' ERR UNCORRECTABLE AFTER
:*                     28TH RETRY
:*                  C. IF DATA COMPARE ERROR & SW07 SET, DISPLAY
:*                     REMAINDER OF BUFFER
:*      2           A. DO NOT TYPE DRIVE STATUS AT PROGRAM START
:*                  B. DO NOT TYPE PERFORMANCE REPORT AFTER SPECIFIED TIME
:*                  C. PROMPT 'FE CYLINDER' MESSAGE IN AUTO(CHAIN) RUN MODE
:*      1           INHIBIT DATA COMPARE AFTER READ COMMAND, W/O ERROR
:*      0           READ ONLY MODE

.SBTTL BASIC DEFINITIONS

:*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
STACK = 1100
ERROR = EMT      ::BASIC DEFINITION OF ERROR CALL
SCOPE = IOT      ::BASIC DEFINITION OF SCOPE CALL

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

:*GENERAL PURPOSE REGISTER DEFINITIONS

```

001100
104000
000004

000011
000012
000015
000200
177776
177776
177774
177772
177570
177570

000000	R0	=	%0	::GENERAL REGISTER
000001	R1	=	%1	::GENERAL REGISTER
000002	R2	=	%2	::GENERAL REGISTER
000003	R3	=	%3	::GENERAL REGISTER
000004	R4	=	%4	::GENERAL REGISTER
000005	R5	=	%5	::GENERAL REGISTER
000006	R6	=	%6	::GENERAL REGISTER
000007	R7	=	%7	::GENERAL REGISTER
000006	SP	=	%6	::STACK POINTER
000007	PC	=	%7	::PROGRAM COUNTER

.*PRIORITY LEVEL DEFINITIONS

000000	PP0	=	0	::PRIORITY LEVEL 0
000040	PR1	=	40	::PRIORITY LEVEL 1
000100	PR2	=	100	::PRIORITY LEVEL 2
000140	PR3	=	140	::PRIORITY LEVEL 3
000200	PR4	=	200	::PRIORITY LEVEL 4
000240	PR5	=	240	::PRIORITY LEVEL 5
000300	PR6	=	300	::PRIORITY LEVEL 6
000340	PR7	=	340	::PRIORITY LEVEL 7

.*"SWITCH REGISTER" SWITCH DEFINITIONS

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

.*DATA BIT DEFINITIONS (BIT00 TO BIT15)

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

```

000400 BIT08 = 400
000200 BIT07 = 200
000100 BIT06 = 100
000040 BIT05 = 40
000020 BIT04 = 20
000010 BIT03 = 10
000004 BIT02 = 4
000002 BIT01 = 2
000001 BIT00 = 1
001000 BIT9=BIT09
000400 BIT8=BIT08
000200 BIT7=BIT07
000100 BIT6=BIT06
000040 BIT5=BIT05
000020 BIT4=BIT04
000010 BIT3=BIT03
000004 BIT2=BIT02
000002 BIT1=BIT01
000001 BIT0=BIT00
    
```

```

;*BASIC "CPU" TRAP VECTOR ADDRESSES
000004 ERRVEC = 4 ;:TIME OUT AND OTHER ERRORS
000010 RESVEC = 10 ;:RESERVED AND ILLEGAL INSTRUCTIONS
000014 TBITVEC = 14 ;: 'T' BIT
000014 TRIVEC = 14 ;:TRACE TRAP
000014 BPTVEC = 14 ;:BREAKPOINT TRAP (BPT)
000020 IOTVEC = 20 ;:INPUT/OUTPUT TRAP (IOT) **SCOPE**
000024 PWRVEC = 24 ;:POWER FAIL
000030 EMTVEC = 30 ;:EMULATOR TRAP (EMT) **ERROR**
000034 TRAPVEC = 34 ;: 'TRAP' TRAP
000060 TKVEC = 60 ;:TTY KEYBOARD VECTOR
000064 TPVEC = 64 ;:TTY PRINTER VECTOR
000240 PIRQVEC = 240 ;:PROGRAM INTERRUPT REQUEST VECTOR
    
```

.SBTTL RH CONTROLLER REGISTERS

:CONTROL AND STATUS REGISTER 1 (RMCS1)

```

82 000100 IE = 100 ;:INTERRUPT ENABLE (BIT #6)
83 000200 RDY = 200 ;:READY (BIT #7)
84 000400 A16 = 400 ;:HIGH ORDER BUS ADDRESS BIT (BIT #8)
85 001000 A17 = 1000 ;:HIGH ORDER BUS ADDRESS BIT (BIT #9)
86 002000 PSEL = 2000 ;:PORT SELECT (BIT #10)
87 020000 MCPE = 20000 ;:MASSBUSS PARITY ERROR (BIT #13)
88 040000 TRE = 40000 ;:TRANSFER ERROR (BIT #14)
89 ;SC = 100000 ;:SPECIAL CONDITION (BIT #15)
    
```

:WORD COUNT REGISTER (RMWC)
 :(EACH BIT IS CALLED BY BIT NUMBER)

:BUS ADDRESS REGISTER (RMBA)
 :(EACH BIT IS CALLED BY BIT NUMBER)

:CONTROL AND STATUS REGISTER 2 (RMCS2)

```

98 000001 US1 = 1 ;:UNIT SELECT (BIT #0)
99 000002 US2 = 2 ;:UNIT SELECT (BIT #1)
100
    
```

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	000004	US4	= 4	:UNIT SELECT (BIT #2)
102	000010	BAI	= 10	:BUS ADDRESS INCREMENT INHIBIT (BIT #3)
103	000020	PAT	= 20	:MASSBUS PARITY TEST (BIT #4)
104	000040	CLR	= 40	:CLEAR (BIT #5)
105	000100	IR	= 100	:INPUT READY (BIT #6)
106	000200	OR	= 200	:OUTPUT READY (BIT #7)
107	000400	MDPE	= 400	:MASS BUS PARITY ERROR (BIT #8)
108	001000	MXF	= 1000	:MISSED TRANSFER ERROR (BIT #9)
109	002000	PGE	= 2000	:PROGRAM ERROR (BIT #10)
110	004000	NEM	= 4000	:NON EXISTENT MEMORY (BIT #11)
111	010000	NED	= 10000	:NON EXISTENT DRIVE (BIT #12)
112	020000	UPE	= 20000	:UNIBUS PARITY ERROR (BIT #13)
113	040000	WCE	= 40000	:WRITE CHECK ERROR (BIT #14)
114	100000	DLT	= 100000	:DATA LATE (BIT #15)
115				
116				:DATA BUFFER REGISTER (RMDB)
117				: (EACH BIT IS CALLED BY BIT NUMBER)
118				
119				.SBTTL RM REGISTERS
120				
121				:CONTROL AND STATUS 1 REGISTER. (#00)
122				
123	000001	GO	= 1	:GO BIT (BIT #0)
124	000002	F0	= 2	:FUNCTION CODE BIT #1
125	000004	F1	= 4	:FUNCTION CODE BIT #2
126	000010	F2	= 10	:FUNCTION CODE BIT #3
127	000020	F3	= 20	:FUNCTION CODE BIT #4
128	000040	F4	= 40	:FUNCTION CODE BIT #5
129	004000	DVA	= 4000	:DEVICE AVAILABLE (BIT #11)
130				
131				:DRIVE STATUS REGISTER (RMD51) (#01)
132				
133	000001	OFFON	= 1	:OFFSET ON (BIT #0)
134	000100	VV	= 100	:VOLUME VALID (BIT #6)
135	000200	DRY	= 200	:DRIVE READY (BIT #7)
136	000400	DPR	= 400	:DRIVE PRESENT (BIT #8)
137	001000	PGM	= 1000	:PROGRAMABLE (BIT #9)
138	002000	LBT	= 2000	:LAST BLOCK TRANSFERRED (BIT #10)
139	004000	WRL	= 4000	:WRITE LOCK (BIT #11)
140	010000	MOL	= 10000	:MEDIUM ON-LINE (BIT #12)
141	020000	PIP	= 20000	:POSITIONING OPERATION IN PROGRESS (BIT #13)
142	040000	ERR	= 40000	:COMPOSITE ERROR (BIT #14)
143	100000	ATA	= 100000	:ATTENTION ACTIVE (BIT #15)
144				
145				:ERROR REGISTER #01 (RMER1) (#02)
146				
147	000001	ILF	= 1	:ILLEGAL FUNCTION (BIT #0)
148	000002	ILR	= 2	:ILLEGAL REGISTER (BIT #1)
149	000004	RMR	= 4	:REGISTER MODIFICATION REFUSED (BIT #2)
150	000010	PAR	= 10	:PARITY ERROR (BIT #3)
151	000020	FER	= 20	:FORMAT ERROR (BIT #4)
152	000040	WCF	= 40	:WRITE CLOCK FAIL (BIT #5)
153	000100	ECH	= 100	:ECC HARD ERROR (BIT #6)
154	000200	HCE	= 200	:HEADER COMPARE ERROR (BIT #7)
155	000400	HCRC	= 400	:HEADER CRC ERROR (BIT #8)
156	001000	AOE	= 1000	:ADDRESS OVERFLOW ERROR (BIT #9)
157	002000	IAE	= 2000	:INVALID ADDRESS ERROR (BIT #10)

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

```

215
216      ;DESIRED CYLINDER ADDRESS (RMDC) (#12)
217      ;(EACH BIT IS CALLED BY BIT NUMBER)
218
219      ;CURRENT CYLINDER ADDRESS (RMCC) (#13)
220      ;(REGISTER CURRENTLY NOT USED)
221
222      ;RM ERROR REGISTER #02 (RMER2) (#15)
223
224      000010      DPE      = 10      ;DATA PARITY ERROR (BIT #3)
225      000040      SSE      = 40      ;SKIP SECTOR ERROR (BIT #5)
226      000200      DVC      = 200     ;DEVICE CHECK (BIT #7)
227      002000      LBC      = 2000    ;LOSS OF BIT CLOCK (BIT #10)
228      004000      LSC      = 4000    ;LOSS OF SYSTEM CLOCK (BIT #11)
229      010000      IVC      = 10000   ;INVLAID COMMAND ERROR (BIT #12)
230      020000      OPE      = 20000   ;OPERATOR PLUG ERROR (BIT #13)
231      040000      SKI      = 40000   ;SEEK INCOMPLETE (BIT #14)
232      100000      BSE      = 100000  ;BAD SECTOR ERROR (BIT #15)
233
234      ;ECC POSITION REGISTER (RMEC1) (#16)
235      ;(EACH BIT IS CALLED BY BIT NUMBER)
236
237      ;ECC PATTERN REGISTER (RMEC2) (#17)
238      ;(EACH BIT IS CALLED BY BIT NUMBER)
239
240      .SBTTL  RM DRIVER COMMANDS
241
242      000101      RNOP     = 101      ;NO OPERATION
243      000105      SEEK     = 105      ;SEEK
244      000107      RECAL    = 107      ;RECALIBRATE
245      000111      DRVCLR   = 111      ;DRIVE CLEAR
246      000113      RELSE    = 113      ;RELEASE
247      000117      RTC      = 117      ;RETURN TO CENTER LINE
248      000121      READIN   = 121      ;READ IN PRESET
249      000123      ACK      = 123      ;PACK ACKNOWLEDGE
250      000131      SEARCH   = 131      ;SEARCH
251      000141      GETREG   = 141      ;GET REGISTERS
252      000143      SETFMT   = 143      ;SET FORMAT (& ECI OR HCI)
253      000145      SELDRV   = 145      ;SELECT DRIVE
254      000151      WCKD     = 151      ;WRITE CHECK DATA
255      000153      WCKHD    = 153      ;WRITE CHECK HEADER & DATA
256      000161      WRTDAT   = 161      ;WRITE DATA
257      000163      WRTHD    = 163      ;WRITE HEADER & DATA
258      000171      RDDAT    = 171      ;READ DATA
259      000173      RDHD     = 173      ;READ HEADER & DATA
260
261
262
263      176700      ABASE    = 176700
264      000254      AVECT1   = 254
265
266
  
```

```

1          .SBTTL TRAP CATCHER
          000000
          .=0
          :*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
          :*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
          :*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
          000174 000174
          000174 000000
          000176 000000
          .=174
          DISPREG: .WORD 0          ;;SOFTWARE DISPLAY REGISTER
          SWREG:   .WORD 0          ;;SOFTWARE SWITCH REGISTER

          .SBTTL STARTING ADDRESS(ES)
          000200 000137 003542      JMP      @#START1          ;;JUMP TO STARTING ADDRESS OF PROGRAM
          2
          3 000204 000137 003532      JMP      @#START          ;CHANGE THE RH ADDRESS
          4
          5          .SBTTL ACT11 HOOKS
          ;;*****
          ;HOOKS REQUIRED BY ACT11
          000210 000210              $SVPC=.          ;SAVE PC
          000046 000046              .=46
          031752 031752              $ENDAD          ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
          000052 000052              .=52
          040000 040000              .WORD 40000      ;;2)SET LOC.52 TO 40000
          000210 000210              .=$SVPC          ;; RESTORE PC

          6          .=1100
          7          .SBTTL APT PARAMETER BLOCK
          8          ;;*****
          ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
          ;;*****
          000110 001100              .$X=.          ;;SAVE CURRENT LOCATION
          000024 000024              .=24          ;;SET POWER FAIL TO POINT TO START OF PROGRAM
          000200 000200              200          ;;FOR APT START UP
          000044 000044              .=44          ;;POINT TO APT INDIRECT ADDRESS PNTR.
          000110 001100              $APTHDR      ;;POINT TO APT HEADER BLOCK
          001100 001100              .=$X          ;;RESET LOCATION COUNTER
          ;;*****
          ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
          ;INTERFACE SPEC.

          001100 $APTHD:
          001100 000000 $HIBTS: .WORD 0          ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
          001102 001206 $MBADR: .WORD $MAIL      ;;ADDRESS OF APT MAILBOX (BITS 0-15)
          001104 014234 $TSTM:  .WORD 6300.      ;;RUN TIM OF LONGEST TEST
          001106 014234 $PASTM: .WORD 6300.      ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
          001110 014234 $UNITM: .WORD 6300.      ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDED UNIT
          001112 000032          .WORD $ETEND-$MAIL/2  ;;LENGTH MAILBOX-ETABLE(WORDS)
          001114          001114          TAB.XY=.          ;CMTAGSTARING ADDRESS
          9
          10
  
```

0

.SBTTL COMMON TAGS

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

001114	001114			SCMTAG: .=TAB.XY	::START OF COMMON TAGS
001114	000000			.WORD 0	::CONTAINS THE TEST NUMBER
001116	000			\$TSTNM: .BYTE 0	::CONTAINS ERROR FLAG
001117	000			\$ERFLG: .BYTE 0	::CONTAINS SUBTEST ITERATION COUNT
001120	000000			\$ICNT: .WORD 0	::CONTAINS SCOPE LOOP ADDRESS
001122	000000			\$LPADR: .WORD 0	::CONTAINS SCOPE RETURN FOR ERRORS
001124	000000			\$LPERR: .WORD 0	::CONTAINS TOTAL ERRORS DETECTED
001126	000000			\$ERTTL: .WORD 0	::CONTAINS ITEM CONTROL BYTE
001130	000			\$ITEMB: .BYTE 0	::CONTAINS MAX. ERRORS PER TEST
001131	001			\$ERMAX: .BYTE 1	::CONTAINS PC OF LAST ERROR INSTRUCTION
001132	000000			\$ERRPC: .WORD 0	::CONTAINS ADDRESS OF 'GOOD' DATA
001134	000000			\$GDADR: .WORD 0	::CONTAINS ADDRESS OF 'BAD' DATA
001136	000000			\$BDADR: .WORD 0	::CONTAINS 'GOOD' DATA
001140	000000			\$GDDAT: .WORD 0	::CONTAINS 'BAD' DATA
001142	000000			\$BDDAT: .WORD 0	::RESERVED--NOT TO BE USED
001144	000000			.WORD 0	
001146	000000			.WORD 0	
001150	000			\$AUTOB: .BYTE 0	::AUTOMATIC MODE INDICATOR
001151	000			\$INTAG: .BYTE 0	::INTERRUPT MODE INDICATOR
001152	000000			.WORD 0	
001154	177570			\$SWR: .WORD DSWR	::ADDRESS OF SWITCH REGISTER
001156	177570			\$DISPLAY: .WORD DDISP	::ADDRESS OF DISPLAY REGISTER
001160	177560			\$TKS: 177560	::TTY KBD STATUS
001162	177562			\$TKB: 177562	::TTY KBD BUFFER
001164	177564			\$TPS: 177564	::TTY PRINTER STATUS REG. ADDRESS
001166	177566			\$TPB: 177566	::TTY PRINTER BUFFER REG. ADDRESS
001170	000			\$NULL: .BYTE 0	::CONTAINS NULL CHARACTER FOR FILLS
001171	002			\$FILLS: .BYTE 2	::CONTAINS # OF FILLER CHARACTERS REQUIRED
001172	012			\$FILLC: .BYTE 12	::INSERT FILL CHARS. AFTER A 'LINE FEED'
001173	000			\$TPFLG: .BYTE 0	::'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
001174	000000			\$TMP0: .WORD 0	::USER DEFINED
001176	207	377	377	\$BELL: .ASCIZ <207><377><377>	::CODE FOR BELL
001202	077			\$QUES: .ASCII /?/	::QUESTION MARK
001203	015			\$CRLF: .ASCII <15>	::CARRIAGE RETURN
001204	012	000		\$LF: .ASCIZ <12>	::LINE FEED

.SBTTL APT MAILBOX-ETABLE

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

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

.SBTTL USER DEFINED TAGS

001272	176700	\$RMADR:	.WORD	176700	:FIRST ADDRESS OF RH/RM REGISTERS
001274	000254	\$RMVEC:	.WORD	254	:VECTOR ADDRESS
001276	172540	\$LKCSR:	.WORD	172540	:ADDR OF KW11-P STATUS REGISTER
001300	172542	\$LKCSB:	.WORD	172542	:ADDR OF KW11-P COUNTER BUFFER
001302	000104	\$LPVEC:	.WORD	104	:ADDR OF KW11-P VECTOR
001304	177546	\$LKS:	.WORD	177546	:ADDR OF KW11-L STATUS REGISTER
001306	000100	\$LLVEC:	.WORD	100	:ADDR OF KW11-L VECTOR
001310	177777	PCLOCK:	.WORD	-1	: '0' IF KW11-P IS ON SYSTEM
001312	177777	CLKFLG:	.WORD	-1	: '0' IF A CLOCK IS AVAILABLE
001314	000074	HZ:	.WORD	60.	:60. IF 60HZ SYSTEM, 50. IF 50HZ SYSTEM
001316	000000	STATIN:	.WORD	0	: 'TYPE STATISTICS' INDICATOR
001320	000000	PACK:	.WORD	0	: 'W' COMMAND INDICATOR
	001220	DRIVE	=SUNIT		:DRIVE # STORAGE: ERRORS 1-5 & 10
					:SAME AS USED IN APT
001322	000000	ATTN:	.WORD	0	:ATTN REG STORAGE: ERRORS 1-5 & 10
001324	000000	DRVNO:	.WORD	0	:DRIVE # STORAGE FOR PRINTOUT
001326	000000	MASK:	.WORD	0	:ERROR RETRY REGISTER MASK
001330	000	RETRY:	.BYTE	0,0	:ERROR RETRY LIMIT IN THE LOWER BYTE
					:RETRY COUNT IN THE UPPER BYTE
001332	000003	FAIRNS:	.WORD	3	:MAXIMUM TIME IN QUEUE VALUE
001334	000000	LSTAD:	.WORD	0	:STORE LAST MEMORY ADDRESS HERE
001336	000000	CHGADR:	.WORD	0	:CHANGE RH/RM UNIBUS ADDRESS FLAG
001340	000000	CFLAG:	.WORD	0	: 'CONTROL C' FLAG
001342	000000	BADSEC:	.WORD	0	:BAD TRACK/SECTOR FLAG
001344	000000	HOUR:	.WORD	0	:HOUR COUNT STORED HERE
001346	000000	MINUTE:	.WORD	0	:MINUTE'S COUNT STORED HERE
001350	000000	SECOND:	.WORD	0	:SECOND'S COUNT STORED HERE
001352	000000	ONESEC:	.WORD	0	:TIMER ROUTINE COUNTER (FOR ONE SECOND)
001354	177777	ZROIND:	.WORD	-1	:ZERO INDICATOR FOR THE DATA COMPARE ROUTINE
001356	000	FRSTER:	.BYTE	0	:DATA COMPARE ERROR FLAG
					:IF > 0, PROCESSING 'DCKER' OR CAN'T MATCH PATTERN
					:IF < 0, MISCOMPARISON FOUND
001357	000		.BYTE	0	:MISCOMPARISON OR CAN'T MATCH PATTERN FLAG
					:IF < 0, ERROR IN BUFFER
001360	000000	SAVER1:	.WORD	0	:SAVE R1 HERE
001362	000000	SAVER5:	.WORD	0	:SAVE R5 HERE
001364	000000	ERCTR:	.WORD	0	:NUMBER OF ERRORS
001366	000000	LIMIT:	.WORD	0	:DISPLAY LIMIT
001370	000000	CMCNT:	.WORD	0	:WORD COUNT
001372	000000	CMCYL:	.WORD	0	:CYLINDER ADDRESS
001374	000	CMSEC:	.BYTE	0	:SECTOR ADDRESS
001375	000	CMTRK:	.BYTE	0	:TRACK ADDRESS
001376	000000	ECBIT:	.WORD	0	:ERROR BURST BIT OFFSET
001400	000000	ECSEC:	.WORD	0	:ERROR BURST WORD OFFSET (RELATIVE TO SECTOR)
001402	000000	ECMSK0:	.WORD	0	:CORRECTION MASK FOR FIRST ERROR WORD
001404	000000	ECMSK1:	.WORD	0	:CORRECTION MASK FOR SECOND ERROR WORD
001406	000000	ECWRD:	.WORD	0	:LOCATION OF FIRST ERROR WORD
001410	000000	ECGD:	.WORD	0	:GOOD DATA, FIRST WORD
001412	000000	ECBADO:	.WORD	0	:BAD DATA, FIRST WORD
001414	000000	ECWRD1:	.WORD	0	:LOCATION OF SECOND ERROR WORD
001416	000000	ECGD1:	.WORD	0	:GOOD DATA, SECOND WORD
001420	000000	ECBAD1:	.WORD	0	:BAD DATA, SECOND WORD
001422	001056	CYLIMT:	.WORD	558.	:CYLINDER ADDRESS LIMIT
001424	000036	SECLMT:	.WORD	30.	:SECTOR ADDRESS LIMIT

001426	000015	TRKLMT:	.WORD	13.	:TRACK ADDRESS LIMIT
001430	001057	FE1:	.WORD	559.	:1ST FE CYLINDER
001432	001060	FE2:	.WORD	560.	:2ND FE CYLINDER
001434	000000	FEFLAG:	.WORD	0	:EXERCISE FE CYLINDERS ONLY=0, EXERCISE ENTIRE DISK=1
001436	000000	DEC2:	.WORD	0	:DECREMENT TRK/SEC ONCE=0, DECREMENT TRK/SEC TWICE=1
001440	000000	RONLY:	.WORD	0	:NOT READ ONLY=0, READ ONLY=1
001442	000000	DRVPAR:	.WORD	0	:WHEN DRIVES ARE BEING ASSIGNED, :0=CHANGE DRIVE PARAMETERS :1=DO NOT CHANGE DRIVE PARAMETERS
001444	000000	XXDP:	.WORD	0	:THE LOW BYTE CONTAINS THE DRIVE NUMBER FROM WHICH :THE PROGRAM WAS LOADED. THE HIGH BYTE CONTAINS THE : 'XXDP' DEVICE CODE FOR THE RM80.

.SBTTL COMMON PARAMETERS

:THE FOLLOWING TWO LOCATIONS CONTAIN THE SOFT ERROR RATE WORDS USED TO
 :DETERMINE END OF PASS WHEN THE PROGRAM IS DATA BIASED.
 :IT WILL TAKE APPROXIMATELY 3.33 PASSES TO REACH THE SOFT ERROR RATE OF
 :1 X 10¹⁰ BITS (6.25 X 10⁸ WORDS) READ OR 10. PASSES TO REACH THE 90%
 :CONFIDENCE LEVEL OF 3 X 10¹⁰ BITS (1.875 X 10⁹ WORDS) READ.
 :ENDCON= LSB AND ENDCON+2= MSB

001446	002740	ENDCON:	.WORD	002740	;(1.875 X 10 ⁸ WORDS) OR (3 X 10 ⁹ BITS) READ
001450	005455		.WORD	005455	

:THE FOLLOWING TWO LOCATIONS CONTAIN THE SEEK ERROR RATE WORDS USED TO
 :DETERMINE END OF PASS WHEN THE PROGRAM IS SEEK BIASED.
 :IT WILL TAKE 1 PASS TO REACH A SEEK ERROR RATE OF 1 X 10⁶ SEEKS OR 3
 :PASSES TO REACH A 90% CONFIDENCE LEVEL OF 3 X 10⁶ SEEKS.
 :ENDSEK= LSB AND ENDSEK+2= MSB

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

001456	000031	MAXER:	.WORD	25.	:MAXIMUM ERRORS ALLOWED PER DRIVE
001460	000004	CMPLMT:	.WORD	4	:NUMBER OF COMPARE ERRORS TYPED OUT
001462	017400	WRDCNT:	.WORD	7936.	:MAXIMUM WORD COUNT (31. SECTORS)
001464	000000	INTRVL:	.WORD	0,0	:FIRST WORD IS THE PERFORMANCE TIMEOUT INTERVAL :(IN MINUTES). SECOND WORD IS THE INTERVAL COUNTER.
001470	000001	PASSES:	.WORD	1	:NUMBER OF PASSES TO END OF TEST [THIS PARAMETER IS :NOT USED WHEN PROGRAM IS OPERATING IN AUTO RUN(CHAIN) :MODE].

001472	000000	PATTERN:	.WORD	0	:IF EQ 0, RANDOMLY SELECT DATA PATTERN :IF NOT EQ 0, SELECT ONE SET OF PATTERN :POINTED BY THE 'PATTERN'.
--------	--------	----------	-------	---	---

001474	000000	RANDWC:	.WORD	0	:IF EQ TO 0, GENERATE A RANDOM WORD COUNT :FOR THE OPERATION. :IF NOT EQ TO 0, USE THE VALUE IN 'WRDCNT' FOR :THE WORD COUNT
--------	--------	---------	-------	---	---

001476	000003	RATIO:	.WORD	3	:READ/WRITE RATIO [RANGE 0 - 7] :0 - 15/1 (READ/WRITE) :1 - 7/1 :2 - 6/2 :3 - 5/3 :4 - 4/4 :5 - 3/5 :6 - 2/6
--------	--------	--------	-------	---	---

```

001500 000001      ENDING: .WORD 1      ;7 - 1/7
                                     ;IF NOT EQ 0, END OF PASS DETERMINED
                                     ;BY THE 'WORDS READ' COUNT. (2.5 X 10^8 WORDS)
001502 000001      WRTCHK: .WORD 1      ;IF EQ 0, END OF PASS DETERMINED
                                     ;BY THE SEEK COUNT. (4 X 10^5 SEEKS)
001504 000001      MESSAGE: .WORD 1      ;IF NOT EQ 0, DO AN APPROPRIATE WRITE
                                     ;CHECK AFTER EACH WRITE COMMAND.
001506 000000      RANDOM: .WORD 0      ;IF EQ 0, SELECT WRITE CHECK COMMANDS
001510 000000      BADBLK: .WORD 0      ;RANDOMLY.
                                     ;IF EQ 1, DO NOT PRINT DATA ERROR MESSAGES
                                     ;ASSOCIATED WITH OPERATOR SPECIFIED
                                     ;BAD SECTOR AREAS.
                                     ;IF NOT EQ 0, PRINT ERROR MESSAGES RELATING TO
                                     ;THESE AREAS.
                                     ;IF EQ TO 0, RANDOMLY SELECT DATA BLOCK
                                     ;ADDRESS. IF NOT EQU 0, SEQUENTIALLY
                                     ;SELECT DATA BLOCK ADDRESS
                                     ;IF EQ TO 1, THE BAD SECTOR ENTRY TABLE WILL ALWAYS
                                     ;BE INITIALIZED WHEN ASSIGNING A DRIVE; IF EQ TO 0,
                                     ;THE BAD SECTOR ENTRY TABLE WILL ONLY BE INITIALIZED
                                     ;IF THE HDA SERIAL NUMBER HAS CHANGED SINCE THE
                                     ;LAST TIME IT WAS READ. (NOTE: IF THE SERIAL NO. HAS
                                     ;CHANGED, THIS MOST LIKELY MEANS THAT THE HDA OR DRIVE
                                     ;HAD BEEN REPLACED WHILE THE DRIVE WAS DEASSIGNED)

```

.SBTTL VALUES FOR FIRST OPERATION

```

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

```

.SBTTL TABLES, CONSTANTS, AND VARIABLE LOCATIONS

```

;LIST OF DRIVES PERFORMING COMMANDS
001520 000000      ORDERQ: .WORD 0
001522 000000      .WORD 0
001524 000000      .WCRD 0
001526 000000      .WORD 0
001530 000000      .WORD 0
001532 000000      .WORD 0
001534 000000      .WORD 0
001536 000000      .WORD 0
001540 000000      .WORD 0
001542 000000      ASNLST: .WORD 0      ;A BIT SET IS AN ASSIGNED DRIVE
;ADDRESSES OF DRIVES TO BE DEASSIGNED
001544 000000      DDRVS: .WORD 0
001546 000000      .WORD 0
001550 000000      .WORD 0
001552 000000      .WORD 0

```

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

;ADDRESSES OF NEWLY ASSIGNED DRIVES

001566	000000	NEWUNT: .WORD	0
001570	000000	.WORD	0
001572	000000	.WORD	0
001574	000000	.WORD	0
001576	000000	.WORD	0
001600	000000	.WORD	0
001602	000000	.WORD	0
001604	000000	.WORD	0
001606	000000	.WORD	0

;LIST OF DRIVES WAITING FOR BUFFERS/PARAMETERS

001610	000000	AVAIL: .WORD	0
001612	000000	.WORD	0
001614	000000	.WORD	0
001616	000000	.WORD	0
001620	000000	.WORD	0
001622	000000	.WORD	0
001624	000000	.WORD	0
001626	000000	.WORD	0
001630	000000	.WORD	0

;LIST OF DRIVES WAITING FOR BUFFERS

001632	000000	WAIT: .WORD	0
001634	000000	.WORD	0
001636	000000	.WORD	0
001640	000000	.WORD	0
001642	000000	.WORD	0
001644	000000	.WORD	0
001646	000000	.WORD	0
001650	000000	.WORD	0
001652	000000	.WORD	0

;BUFFER ALLOCATION TABLE ENTRY COUNT

001654	000000	BUFTBL: .WORD	0
001656	000000	.WORD	0.0
001662	000000	.WORD	0.0
001666	000000	.WORD	0.0
001672	000000	.WORD	0.0
001676	000000	.WORD	0.0
001702	000000	.WORD	0.0
001706	000000	.WORD	0.0
001712	000000	.WORD	0.0
001716	000000	.WORD	0.0
001722	000000	.WORD	0.0
001726	000000	.WORD	0.0
001732	000000	.WORD	0.0
001736	000000	.WORD	0.0
001742	000000	.WORD	0.0
001746	000000	.WORD	0.0

001757	000000	000000	.WORD	0.0
001756	000000	000000	.WORD	0.0
001762	000000	000000	.WORD	0.0
001766	000000	000000	.WORD	0.0
001772	000000	000000	.WORD	0.0
001776	000000	000000	.WORD	0.0
002002	000000	000000	.WORD	0.0
002006	000000	000000	.WORD	0.0
002012	000000	000000	.WORD	0.0
002016	000000	000000	.WORD	0.0
002022	000000	000000	.WORD	0.0
002026	000000	000000	.WORD	0.0
002032	000000	000000	.WORD	0.0
002036	000000	000000	.WORD	0.0
002042	000000	000000	.WORD	0.0
002046	000000	000000	.WORD	0.0
002052	000000	000000	.WORD	0.0

```

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

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

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

.EVEN

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

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

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

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

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

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

002630	052525	DATA5: .WORD	052525	; STANDARD PATTERN 5
002632	052525	.WORD	052525	
002634	052525	.WORD	052525	
002636	125252	.WORD	125252	
002640	125252	.WORD	125252	
002642	125252	.WORD	125252	
002644	052525	.WORD	052525	
002646	052525	.WORD	052525	
002650	125252	.WORD	125252	
002652	125252	.WORD	125252	
002654	052525	.WORD	052525	
002656	125252	.WORD	125252	
002660	052525	.WORD	052525	
002662	125252	.WORD	125252	
002664	052525	.WORD	052525	
002666	125252	.WORD	125252	

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

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

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

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

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

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

003130 172666 DATA11: .WORD 172666 ;STANDARD PATTERN 11

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

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

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

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

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

003330	177777	DATA15: .WORD	177777	:STANDARD PATTERN 15
003332	000000	.WORD	000000	
003334	000000	.WORD	000000	
003336	000000	.WORD	000000	
003340	000000	.WORD	000000	
003342	000000	.WORD	000000	
003344	000000	.WORD	000000	
003346	000000	.WORD	000000	
003350	000000	.WORD	000000	
003352	000000	.WORD	000000	
003354	000000	.WORD	000000	
003356	000000	.WORD	000000	
003360	000000	.WORD	000000	
003362	000000	.WORD	000000	
003364	000000	.WORD	000000	
003366	000000	.WORD	000000	

.SBTTL ERROR POINTER TABLE

.*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
 .*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
 .*LOCATION \$ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
 .*NOTE1: IF \$ITEMB IS 0 THE ONLY PERTINENT DATA IS (\$ERRPC).
 .*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

.* EM ::POINTS TO THE ERROR MESSAGE
 .* DH ::POINTS TO THE DATA HEADER
 .* DT ::POINTS TO THE DATA
 .* DF ::POINTS TO THE DATA FORMAT

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

```

1          :THIS ROUTINE HANDLES UNEXPECTED TIMEOUTS
2
3 003450 011600 BADTMO: MOV      (SP),R0          :SAVE PC WHERE THE TIME OUT OCCURED
4 003452 005740      TST      -(R0)           :ADJUST PC -2
5 003454 022626      CMP      (SP)+,(SP)+     :RESTORE STACK POINTER
6 003456 104401 003464      TYPE     ,65$      :TYPE ASCIZ STRING
      003462 000417      BR       64$          :GET OVER THE ASCIZ
      ::65$: .ASCIZ <CRLF>/UNEXPECTED BUS TIMEOUT, PC=/
      64$:
7 003522 010046      MOV      R0,-(SP)        :SETUP FOR TYPING OUT PC
8 003524 104402      TYPOC
9 003526 000240      NOP
      :PUT 'HALT(0)' INSTRUCTION HERE IF YOU WISH
      :TO STOP ON UNEXPECTED TIMEOUT.
10
11 003530 000404      BR       START1         :BRANCH TO START1
12
13          .SBTTL  START OF PROGRAM
14
15 003532 012737 177777 001336 START: MOV      #-1,CHGADR      :SET RH/RM ADDRESS CHANGE FLAG
16 003540 000407      BR       START2         :START THE PROGRAM
17
18 003542 012737 000400 001336 START1: MOV      #400,CHGADR     :CLEAR THE RH/RM ADDRESS CHANGE FLAG
19          ::*****
      003550 (.00240      TST1:  NOP
      003552 012737 000001 001212      MOV      #1,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
20
21 003560 005227 000000      START2: INC      #0          :TTY LOOP, WAIT FOR INCREMENT
22 003564 001375      BNE     .-4          :OF WORD
23 003566 000905      RESET
      :CLEAR THE WORLD
24
25          .SBTTL  INITIALIZE THE COMMON TAGS
      ::CLEAR THE COMMON TAGS ($CMTAG) AREA
      MOV      #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
      CLR      (R6)+          :CLEAR MEMORY LOCATION
      CMP      #SWR,R6        :DONE?
      BNE     .-6            :LOOP BACK IF NO
      MOV      #STACK,SP     :SETUP THE STACK POINTER
      ::INITIALIZE A FEW VECTORS
      MOV      #ERROR,@EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
      MOV      #340,@EMTVEC+2 ;;LEVEL 7
      MOV      #STRAP,@TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
      MOV      #340,@TRAPVEC+2;LEVEL 7
      MOV      #SPURDN,@PWRVEC ;;POWER FAILURE VECTOR
      MOV      #340,@PWRVEC+2 ;;LEVEL 7
      MOV      #176543,$HINUM  ;;PRIME THE RANDOM NUMBER GENERATOR
      MOV      #123456,$LONUM  ;;BOTH HIGH AND LOW WORDS
      ::SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
      ::EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
      MOV      @ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
      MOV      #64$,@ERRVEC  ;;SET UP ERROR VECTOR
      MOV      #DSWR,SWR     ;;SETUP FOR A HARDWARE SWICH REGISTER
      MOV      #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
      CMP      #-1,@SWR      ;;TRY TO REFERENCE HARDWARE SWR
      BNE     66$           ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
      :AND THE HARDWARE SWR IS NOT = -1
      BR       65$          :BRANCH IF NO TIMEOUT
      64$:  MOV      #65$, (SP)  ;;SET UP FOR TRAP RETURN
      RTI
    
```

```

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

003756 005037 001214 CLR $PASS ;;CLEAR PASS COUNT
003762 132737 000200 001227 BITB #APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
003770 001403 BEQ 67$ ;;YES,USE NON-APT SWITCH
003772 012737 001230 001154 MOV #$$SWREG,SWR ;;NO,USE APT SWITCH REGISTER
004000 67$:
26 :SETUP 'TIMEOUT' TRAP VECTOR FOR UNEXPECTED BUS TIMEOUTS
27 004000 012737 003450 000004 MOV #BADTMO,ERRVEC ;;SETUP FOR UNEXPECTED TIMEOUT
28 004006 012737 000300 000006 MOV #PR6,ERRVEC+2 ;;LEVEL 6
29
30 .SBTTL TYPE PROGRAM NAME
   ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
004014 005227 177777 INC #-1 ;;FIRST TIME?
004020 001027 BNE 68$ ;;BRANCH IF NO
004022 104401 004030 TYPE ,69$ ;;TYPE ASCIZ STRING
004026 000424 BR 68$ ;;GET OVER THE ASCIZ
   ;;69$: .ASCIZ <CRLF>@CZRMAO - RM80 PERFORMANCE EXERCISE@<CRLF>
004100 68$:
   .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
004100 005737 000042 TST @#42 ;;ARE WE RUNNING UNDER XXDP/ACT?
004104 001012 BNE 70$ ;;BRANCH IF YES
004106 123727 001226 000001 CMPB $ENV,#1 ;;ARE WE RUNNING UNDER APT?
004114 001406 BEQ 70$ ;;BRANCH IF YES
004116 023727 001154 000176 CMP SWR,#SWREG ;;SOFTWARE SWITCH REG SELECTED?
004124 001005 BNE 71$ ;;BRANCH IF NO
004126 104406 GTSWR ;;GET SOFT-SWR SETTINGS
004130 000403 BR 71$
004132 112737 000001 001150 70$: MOV #1,$AUTOB ;;SET AUTO-MODE INDICATOR
004140 71$:

31 :THE FOLLOWING FINDS OUT THE PROGRAM CONTROL MODE:
32 :PAPER TAPE (MANUAL), ACT11, XXDP CHAIN OR DUMP
33
34
35 004140 005037 001444 CLR XXDP ;;CLEAR 'XXDP' LOAD DEVICE STORAGE
36 004144 122737 000016 000041 CMPB #16,@#41 ;;LOADED FROM AN RM80 ?
37 004152 001121 BNE 3$ ;;BR IF NOT
38 004154 013737 000040 001444 MOV @#40,XXDP ;;GET DEVICE INDICATOR AND NUMBER
39 004162 122737 000007 001444 CMPB #7,XXDP ;;IS IT A VALID NUMBER ?
40 004170 103002 BHIS 1$ ;;YES
41 004172 105037 001444 CLRB XXDP ;;NO, DEFAULT TO DRIVE 0
42 004176 005737 000042 1$: TST @#42 ;;CHAIN MODE OR ACT11 AUTO ACCEPT ?
43 004202 001425 BEQ 2$ ;;BR IF NEITHER
44 004204 104401 0042.2 TYPE ,73$ ;;TYPE ASCIZ STRING
   004210 000412 BR 72$ ;;GET OVER THE ASCIZ
   ;;73$: .ASCIZ <CRLF>/NOT TESTING DRIVE /
   72$:
45 004236 005046 CLR -(SP) ;;CLEAR WORD ON STACK
46 004240 113716 001444 MOV #XXDP,(SP) ;;GET DRIVE ADDRESS
47 004244 104403 TYPOS ;;TYPE THE ADDRESS
48 004246 001 .BYTE 1 ;;ONLY 1 CHARACTER
49 004247 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
50 004250 104401 001203 TYPE ,$CRLF ;;CR-LF
51 004254 000460 BR 3$ ;;GET NUMBER OF DRIVES
52

```



```

53 004256 005227 177777      2$:      INC      #-1      :FIRST TIME THRU HERE ?
54 004262 001055              BNE      3$      :NO
55 004264 104401 004272      TYPE     75$     :TYPE ASCIZ STRING
    004270 000410              BR       74$     :GET OVER THE ASCIZ
    :75$: .ASCIZ <CRLF>/TO TEST DRIVE /
    74$:
56 004312 005046              CLR      -(SP)   :CLEAR WORD ON STACK
57 004314 113716 001444      MOVB    XXDP,(SP) :GET DRIVE ADDRESS
58 004320 104403              TYPOS   :TYPE DRIVE ADDRESS
59 004322      001          .BYTE   1       :ONLY 1 CHARACTER
60 004323      000          .BYTE   0       :SUPRESS LEADING ZEROS
61 004324 104401 004352      TYPE     76$     :TYPE ASCIZ STRING
    004330 000432              BR       3$      :GET OVER THE ASCIZ
    :76$: .ASCIZ /, HALT PROGRAM/, CLEAR LOC. 40 AND RESTART PROGRAM./<CRLF>
    3$:
65 004416 004737 033326      JSR     PC,$TKINT :TURN ON THE KEYBOARD INTERRUPT
66 004422 105737 001226      TSTB   $ENV      :RUN UNDER APT MODE
67 004426 001415              BEQ     5$      :NO,DO NOT BOTHER
68 004430 105737 001256      TSTB   $VECT1   :NEW VECTOR ?
69 004434 001403              BEQ     4$      :NOT LOAD IF = 0
70 004436 113737 001256 001274      MOVB    $VECT1,$RMVEC :NEW VECTOR
71 004444 005737 001262      TST    $BASE    :NEW BASE ADDRESS ?
72 004450 001411              BEQ     6$      :NO
73 004452 013737 001262 001272      MOV     $BASE,$RMADR :NEW BASE ADDRESS
74 004460 000405              BR      6$
75
76 004462 105737 001150      5$:      TSTB   $AUTOB   :RUNNING IN AUTO MODE ?
77 004466 001002              BNE     6$      :YES
78 004470 004737 100674      JSR     PC,$BUSADR :CHECK RH/RM BUS ADDRESS
84
85 004474 013737 001272 040310 6$:      MOV     $RMADR,$RMADR :LOAD ADDRESS INTO DRIVER
86 004502 013737 001274 040312      MOV     $RMVEC,$RMVEC :LOAD VECTOR INTO DRIVER
87 004510 005037 001316              CLR     STATIN   :CLEAR PERFORMANCE SUMMARY TYPEOUT FLAG
88 004514 012705 001520              MOV     #ORDERQ,R5 :START OF AREA TO CLEAR
89 004520 005025              7$:      CLR     (R5)+
90 004522 022705 002056      CMP     #BLKADR,R5 :LOOK FOR END OF CLEAR AREA
91 004526 001374              BNE     7$      :BR IF NOT FINISHED
92 004530 012706 001100      MOV     #STACK,SP :SETUP THE STACK POINTER
93 004534 005037 177776              CLR     PS       :CLEAR THE PROCESSOR STATUS WORD
94 004540 013737 001314 001352      MOV     HZ,ONESEC :RESTORE ONE SECOND COUNTER VALUE
95 004546 005037 001344              CLR     HOUR     :CLEAR THE HOUR'S COUNTER
96 004552 005037 001346              CLR     MINUTE   :CLEAR THE MINUTE'S COUNTER
97 004556 005037 001350              CLR     SECOND   :CLEAR THE SECOND'S COUNTER
98 004562 005037 001466              CLR     INTRVL+2 :CLEAR INTERVAL COUNTER
99 004566 005037 001320              CLR     PACK     :SET 'T' & CLEAR 'R' OR 'W' COMMAND FLAG
100 004572 005037 001340              CLR     CFLAG   :CLEAR THE 'CONTROL C' FLAG
103 004576 005037 045542              CLR     DRIVE0+$FIRST :RESET $FIRST FLAG FOR DRIVE 0
    004602 005037 047756              CLR     DRIVE1+$FIRST :RESET $FIRST FLAG FOR DRIVE 1
    004606 005037 052172              CLR     DRIVE2+$FIRST :RESET $FIRST FLAG FOR DRIVE 2
    004612 005037 054406              CLR     DRIVE3+$FIRST :RESET $FIRST FLAG FOR DRIVE 3
    004616 005037 056622              CLR     DRIVE4+$FIRST :RESET $FIRST FLAG FOR DRIVE 4
    004622 005037 061036              CLR     DRIVE5+$FIRST :RESET $FIRST FLAG FOR DRIVE 5
    004626 005037 063252              CLR     DRIVE6+$FIRST :RESET $FIRST FLAG FOR DRIVE 6
    004632 005037 065466              CLR     DRIVE7+$FIRST :RESET $FIRST FLAG FOR DRIVE 7
104 004636 005037 001440              CLR     RONLY   :ASSUME READ/WRITE CONDITION
105 004642 032777 000001 174304      BIT     #SW0,$SWR :IS EXERCISER IN 'READ ONLY' MODE ?
106 004650 001402              BEQ     8$      :BR IF NO
    
```

```

107 004652 005237 001440          INC      RONLY          ;LOCK PROGRAM IN 'READ ONLY' MODE
108 004656          8$:
109
111          ;AUTO SIZE FOR RH70 CONTROLLER AND DETERMINE IF IT IS
112          ;JUMPERED FOR 22 OR 32 REGISTERS
113
114 004656 005037 040316          SIZE70: CLR      RHEXT          ;CLEAR RMBAE OFFSET
115 004662 042737 174000 001234      BIC      #174000,$CPUOP    ;CLEAR CPU TYPE REGISTER
116 004670 013746 000004          MOV      ERRVEC,-(SP)     ;SAVE CONTENTS OF ERROR VECTOR
117 004674 012737 004746 000004      MOV      #2$,ERRVEC      ;SETUP 'TRAP' RETURN ADDRESS
118 004702 013700 001272          MOV      $RMADR,R0       ;GET RMCS1 ADDRESS
119 004706 062700 000050          ADD      #50,R0          ;GET REGISTER OFFSET FOR RH70
120 004712 012701 000012          MOV      #10.,R1        ;GET NUMBER OF REGISTERS TO CHECK
121 004716 005720          TST      (R0)+          ;TRAP IF NOT A VALID RMBAE
122 004720 005720          TST      (R0)+          ;TRAP IF NOT A VALID RMCS3
123 004722 012737 000050 040316      MOV      #50,RHEXT       ;LOAD OFFSET FOR RMBAE (22 REGISTER RH)
124 004730 005720          1$: TST      (R0)+          ;TRAP IF NOT A VALID REGISTER
125 004732 005301          DEC      R1              ;DONE WITH ALL 32 REGISTERS ?
126 004734 001375          BNE      1$             ;BR IF NO
127 004736 012737 000074 040316      MOV      #74,RHEXT       ;LOAD OFFSET FOR RMBAE (32 REGISTER RH)
128 004744 000403          BR       3$
129 004746 012716 004754          2$: MOV      #3$, (SP)     ;SETUP RETURN ADDRESS
130 004752 000002          RTI
131
132 004754 013700 001272          3$: MOV      $RMADR,R0     ;GET RMCS1 REGISTER
133 004760 013701 040316          MOV      RHEXT,R1        ;GET RMBAE REGISTER OFFSET
134 004764 001415          BEQ      4$             ;BR IF NONE
135 004766 060001          ADD      R0,R1          ;GET RMBAE REGISTER
136 004770 052710 001400          BIS      #A17!A16,(R0)   ;SET EXTENDED ADDRESS BITS IN RMCS1
137 004774 022711 000003          CMP      #3,(R1)        ;ARE THE EXTENDED BITS SET IN RMBAE ?
138 005000 001007          BNE      4$             ;BR IF NO
139 005002 005011          CLR      (R1)           ;CLEAR EXTENDED ADDRESS BITS IN RMBAE
140 005004 033710 001400          BIT      A17!A16,(R0)   ;ARE THE EXTEND BITS CLEAR IN RMCS1 ?
141 005010 001003          BNE      4$             ;BR IF NO
142 005012 052737 030000 001234      BIS      #BIT13!BIT12,$CPUOP ;SET THE 11/70 CPU TYPE CODE
143 005020 012637 000004          4$: MOV      (SP)+,ERRVEC ;RESTORE CONTENTS OF ERROR VECTOR
145
146          ;ROUTINE TO DETERMINE BUFFER MAX WORD COUNT AND FUDGE HDA SERIAL NUMBER
147          ;TO ALLOW BAD SECTOR FILE(DEC144) TO BE READ FROM EACH DRIVE AT
148          ;LEAST ONE TIME.
149
150 005024 005227 177777          SIZMEM: INC      #-1          ;FIRST TIME THRU HERE ?
151 005030 001027          BNE      1$             ;BR IF NO
152 005032 012700 177777          MOV      #-1,R0         ;FUDGE MSB'S FOR INITIALIZING HDA S/N
155 005036 010037 045562          MOV      R0,DRIVE0+$HSNM ;INIT. S/N FOR DRIVE 0
156 005042 010037 047776          MOV      R0,DRIVE1+$HSNM ;INIT. S/N FOR DRIVE 1
157 005046 010037 052212          MOV      R0,DRIVE2+$HSNM ;INIT. S/N FOR DRIVE 2
158 005052 010037 054426          MOV      R0,DRIVE3+$HSNM ;INIT. S/N FOR DRIVE 3
159 005056 010037 056642          MOV      R0,DRIVE4+$HSNM ;INIT. S/N FOR DRIVE 4
160 005062 010037 061056          MOV      R0,DRIVE5+$HSNM ;INIT. S/N FOR DRIVE 5
161 005066 010037 063272          MOV      R0,DRIVE6+$HSNM ;INIT. S/N FOR DRIVE 6
162 005072 010037 065506          MOV      R0,DRIVE7+$HSNM ;INIT. S/N FOR DRIVE 7
163 005076 004737 100542          JSR      PC,$SIZE       ;SEE HOW MUCH MEMORY ON SYSTEM
164 005102 013737 100672 001334      MOV      $LSTAD,LSTAD    ;SAVE THE LAST ADDRESS
165 005110 012737 000001 001654          1$: MOV      #1,BUFTBL    ;LOAD NUMBER OF BUFFERS
166 005116 012737 102200 001656          MOV      #ENDPGM,BUFTBL+2 ;STARTING ADDRESS OF BUFFER
167 005124 013737 001334 001660          MOV      LSTAD,BUFTBL+4 ;LAST ADDR TO BUFFER ALLOCATION TABLE
    
```

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

221	005414	001406		BEQ	6\$:BR IF YES
222	005416	122710	000116	CMPB	#'N,(R0)				:WAS IT A 'N' RESPONSE ?
223	005422	001406		BEQ	7\$:BR IF YES
224	005424	104401	076341	5\$:	TYPE	,BADENT			:TYPE BAD ENTRY MESSAGE
225	005430	000746		BR	3\$:TRY AGAIN
226	005432	005237	001434	6\$:	INC	FEFLAG			:EXERCISE THE ENTIRE DISK
227	005436	000402		BR	8\$				
228	005440	104401	076674	7\$:	TYPE	,FEONLY			:TYPE FE CYLINDER ONLY MESSAGE
229									
230	005444	005737	001440	8\$:	TST	RONLY			:IS PROGRAM LOCKED IN 'READ MODE' ?
231	005450	001402		BEQ	9\$:BR IF NO
232	005452	104401	077131		TYPE	,MREAD			:TYPE READ ONLY MESSAGE
233									
234	005456	005037	001340	9\$:	CLR	CFLAG			:CLEAR CONTROL C FLAG
235	005462	105737	001150		TSTB	\$AUTOB			:RUNNING IN AUTO MODE ?
236	005466	001040		BNE	SETVEC				:BR IF YES
237	005470	104401	077576		TYPE	,ASKPAR			:TYPE 'CHANGE PARAMETERS ?'
238	005474	104411		RDLIN					:READ THE ENTRY
239	005476	012600		MOV	(SP)+,R0				:SAVE ADDRESS OF RESPONSE
240	005500	005737	001340	TST	CFLAG				:WAS IT CONTROL C ?
241	005504	001364		BNE	9\$:BR IF YES
242	005506	105710		TSTB	(R0)				:WAS RESPONSE A CARRIAGE RETURN (DEFAULT 'N')?
243	005510	001427		BEQ	SETVEC				:BR IF YES
244	005512	105760	000001	TSTB	1(R0)				:WAS IT TERMINATED WITH CARRIAGE RETURN ?
245	005516	001006		BNE	10\$:BR IF NO
246	005520	122710	000131	CMPB	#'Y,(R0)				:WAS IT A 'Y' RESPONSE ?
247	005524	001406		BEQ	ENTPR				:BR IF YES
248	005526	122710	000116	CMPB	#'N,(R0)				:WAS IT A 'N' RESPONSE ?
249	005532	001416		BEQ	SETVEC				:BR IF YES
250	005534	104401	076341	10\$:	TYPE	,BADENT			:TYPE BAD ENTRY MESSAGE
251	005540	000746		BR	9\$:TRY AGAIN
252									
253	005542	012703	077466	ENTPR:	MOV	#PARLST,R3			:PARAMETER TABLE ADDRESS
254	005546	004737	031030		JSR	PC,PARENT			:GET THE PARAMETER ENTRY
255	005552	023727	001462	0C0006	CMP	WRDCNT,#6			:IS THE 'WRDCNT' VALUE OK ?
256	005560	103003		BHIS	SETVEC				:BR IF IT IS
257	005562	012737	000006	001462	MOV	#6,WRDCNT			:SET 'WRDCNT' TO THE MINIMUM VALUE
258									
259									:DISPLAY DRIVE STATUS AND SET UP THE OTHER SYSTEM DEVICES THAT
260									:THE PROGRAM WILL USE. PROGRAM RETURN HERE ON POWER FAIL
261									
262	005570	004737	023422	SETVEC:	JSR	PC,CKCLK			:START THE CLOCK
263	005574	004737	040322		JSR	PC,RMINIT			:INITIALIZE THE RM DRIVER
264	005600	012737	177777	040252	MOV	#-1,SAVEFG			:SET THE SAVE REGISTERS FLAG
265	005606	005227	177777		INC	#-1			:FIRST TIME THRU ?
266	005612	001407		BEQ	1\$:BR IF YES
267	005614	005737	037716		TST	PWRFLG			:RETURNING FROM POWER FAIL ?
268	005620	001004		BNE	1\$:BRANCH IF YES
269	005622	032777	000004	173324	BIT	#SW02,@SWR			:TYPEOUT THE DRIVE STATUS TABLE ?
270	005630	001066		BNE	12\$:BR IF NOT
271	005632	005004		1\$:	CLR	R4			:DRIVE TABLE POINTER
272	005634	104401	075501		TYPE	,SYSTAT			:TYPE STATUS HEADING
273	005640	104401	001203	2\$:	TYPE	,\$CRLF			:CR-LF
274	005644	010446			MOV	R4,-(SP)			::SAVE R4 FOR TYPEOUT
	005646	104403			TYPOS				::TYPE DRIVE NUMBER
	005650	002			.BYTE	2			::GO TYPE--OCTAL ASCII
									::TYPE 2 DIGIT(S)

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

```

1          ;INITIALIZE PROGRAM PARAMETERS FOR STARTUP
2
3 006006 004737 033326 STA: JSR PC,$TKINT ;INITIALIZE THE KEYBOARD INTERRUPT HANDLER
4 006012 012737 002740 001446 MOV #002740,ENDCON ;INITIALIZE XFER COUNT(LSB)
5 006020 012737 005455 001450 MOV #005455,ENDCON+2
6 006026 105737 001226 TSTB $ENV ;APT SCRIPT MODE, THEN MAKE IT RUN 2 MIN.
7 006032 001411 BEQ 1$ ;NO
8 006034 012737 000001 001476 MOV #1,RATIO ;SPEED UP TEST
9 006042 012737 077777 001446 MOV #77777,ENDCON ;INITIALIZE QUICK XFER COUNT(LSB)
10 006050 012737 000027 001450 MOV #27,ENDCON+2
11
12 006056 105737 001150 1$: TSTB $AUTOB ;RUNNING IN AUTO MODE ?
13 006062 001003 BNE 2$ ;BR IF YES
14 006064 005737 001336 TST CHGADR ;START AT 200 ?
15 006070 003456 BLE 8$ ;NO
16
17 006072 005001 2$: CLR R1 ;DRIVE #
18 006074 005002 CLR R2 ;AVAIL TABLE INDEX
19 006076 005003 CLR R3 ;DRIVE# * 2
20 006100 005737 001444 3$: TST XXDP ;LOADED FROM THIS DEVICE ?
21 006104 001403 BEQ 4$ ;BR IF NO
22 006106 123701 001444 CMPB XXDP,R1 ;LOADED FROM THIS DRIVE ?
23 006112 001435 BEQ 7$ ;BR IF YES
24 006114 105761 040164 4$: TSTB DRVSTA(R1) ;DRIVE ON LINE ?
25 006120 003432 BLE 7$ ;NO
26 006122 110137 067556 MOVB R1,GENDPB ;GET DRIVE NUMBER
27 006126 016300 002056 MOV BLKADR(R3),R0 ;LOAD DPB ADDRESS
28 006132 004737 015624 JSR PC,RECALO ;RECALIBRATE DRIVE
29 006136 004737 026630 JSR PC,CLRDPB ;CLEAR DPB BLOCK
30 006142 004737 027550 JSR PC,GETID ;GET DRIVE (MBA) SERIAL NUMBER
31 006146 004537 027650 JSR R5,GETADR ;RETRIEVE BAD SECTOR FILE
32 006152 010062 001566 MOV R0,NEWUNT(R2) ;LOAD DPB ADDRESS TO ABAIL QUEUE
33 006156 004737 027054 JSR PC,DRVPRM ;SETUP DRIVE PARAMETER LIMITS
34 006162 005060 000124 CLR $FIRST(R0) ;RESET $FIRST FLAG FOR FIRST 204 START
35 006166 005737 037716 TST PWRFLG ;RETURNING FROM POWER FAIL ?
36 006172 001005 BNE 7$ ;BRANCH IF YES
37 006174 112760 177776 000026 MOVB #-2,$SPACK(R0) ;SETUP COMMAND 'WT' (WRITE DATA AND TEST)
38 006202 004737 017000 JSR PC,WRTPK ;SETUP INITIAL PARAMETERS
39
40 006206 022322 7$: CMP (R3)+,(R2)+ ;INCREMENT INDEX
41 006210 005201 INC R1 ;NEXT DRIVE
42 006212 020127 000007 CMP R1,#7 ;ALL DRIVES ASSIGNED ?
43 006216 003730 BLE 3$ ;NO
44 006220 005037 001336 CLR CHGADR ;CLEAR START FLAG
45 006224 000403 BR 9$
46
47 006226 012737 000001 001340 8$: MOV #1,CFLAG ;DUMMY 'CONTROL C' FLAG
48 006234 005037 037716 9$: CLR PWRFLG ;CLEAR POWER FAIL FLAG
    
```

```

1          .SBTTL  MAIN PROGRAM
2
3 006240 005737 001340  MAIN:  TST      CFLAG      ;KEYBOARD INTERRUPTED ?
4 006244 001407          BEQ      3$          ;BR IF NOT
5 006246 005737 001520  1$:   TST      ORDERQ     ;ANY DRIVES IN ORDER QUE ?
6 006252 001402          BEQ      2$          ;BR IF NO, ELSE
7 006254 000137 007104  JMP      IDLE        ;LET ALL DRIVES FINISH ORDER
8 006260 004737 025254  2$:   JSR      PC,KSR    ;SERVICE THE KEYBOARD
9 006264 000240          3$:   NOP          ; !! FOR DEBUGGING !!
10
11          ;CHECK FOR DRIVES TO BE DROPPED
12
13 006266 012703 000010  MAINDA: MOV      #8.,R3      ;DRIVE COUNTER
14 006272 012705 001544  MOV      #DDRVS,R5     ;ADDRESS OF 'DROP DRIVE' TABLE
15 006276 005715          1$:   TST      (R5)        ;SEE IF ENTRY AT PRESENT POSITION
16 006300 001004          BNE     3$          ;BR IF THERE IS ONE
17 006302 005725          2$:   TST      (R5)+      ;INCREMENT TO NEXT TABLE POSITION
18 006304 005303          DEC     R3          ;DECREMENT DRIVE COUNTER
19 006306 001373          BNE     1$          ;BR IF MORE TO CHECK
20 006310 000435          BR      MAIN1       ;GO CHECK FOR NEW ASSIGNED DRIVES
21
22 006312 012701 001610  3$:   MOV      #AVAIL,R1   ;ADDRESS OF 'AVAILABLE DRIVES' TABLE
23 006316 005711          4$:   TST      (R1)        ;IF AT END OF 'AVAIL' TABLE ?
24 006320 001404          BEQ     5$          ;BR IF YES
25 006322 021115          CMP     (R1),(R5)    ;IS DRIVE IN 'AVAIL' THE TABLE ?
26 006324 001412          BEQ     7$          ;BR IF YES
27 006326 005721          TST     (R1)+      ;NO, INCREMENT 'AVAIL' TABLE ADDRESS
28 006330 000772          BR      4$          ;AND CONTINUE LOOKING
29
30 006332 012701 001632  5$:   MOV      #WAIT,R1    ;ADDRESS OF THE 'WAIT' BUFFER TABLE
31 006336 005711          6$:   TST      (R1)        ;AT THE END OF 'WAIT' TABLE ?
32 006340 001760          BEQ     2$          ;BR IF YES
33 006342 021115          CMP     (R1),(R5)    ;IS DRIVE IN THE 'WAIT' TABLE ?
34 006344 001402          BEQ     7$          ;BR IF YES
35 006346 005721          TST     (R1)+      ;NO, INCREMENT 'WAIT' TABLE ADDRESS
36 006350 000772          BR      6$          ;AND CONTINUE LOOKING
37
38 006352 011100          7$:   MOV      (R1),R0     ;PUT THE DRIVE'S BLOCK ADDRESS IN R0
39 006354 104401 001203  TYPE    ,%CRLF        ;CR-LF
40 006360 104401 076003  TYPE    ,DEASSG       ;TYPE 'DRIVE DEASSIGNED'
41 006364 004737 023714  JSR     PC,SUMARY     ;TYPE THE DRIVE'S PERFORMANCE SUMMARY
42 006370 104401 075617  TYPE    ,STARS        ;TYPE '****...ETC'
43 006374 005015          CLR     (R5)        ;CLEAR THE 'DROP DRIVE' TABLE ENTRY
44 006376 004737 020750  JSR     PC,CMPRES     ;COMPRESS THE RESPECTIVE TABLE
45 006402 000737          BR      2$          ;SEE IF ANY MORE DRIVES
46
47          ;LOOK FOR DRIVES TO BE ASSIGNED
48
49 006404 012703 000010  MAIN1: MOV      #8.,R3      ;DRIVE COUNT
50 006410 005001          CLR     R1          ;ASSIGN LIST INDEX
51 006412 005002          CLR     R2          ;'AVAIL' INDEX
52 006414 005005          CLR     R5          ;NEW DRIVE INDEX
53 006416 005765 001566  1$:   TST      NEWUNT(R5)   ;NEW DRIVE IN THIS POSITION
54 006422 001005          BNE     3$          ;BR IF THERE IS
55 006424 005725          2$:   TST      (R5)+      ;INCREMENT R5
56 006426 005201          INC     R1          ;INCREMENT ASSIGN INDEX
57 006430 005303          DEC     R3          ;DECREMENT DRIVE COUNT
    
```



```

58 006432 001371          BNE      1$          :BR IF MORE DRIVES
59 006434 000432          BR       MAIN2         :START OPERATIONS FOR THE AVAILABLE DRIVES
60
61 006436 104401 001203   3$:      TYPE      , $CRLF      :CR-LF
62 006442 104401 075304   TYPE      , UNTMSG     :'DRIVE'
63 006446 010146          MOV      R1, -(SP)     :SAVE R1 FOR TYPEOUT
                                :TYPE DRIVE NUMBER
                                :GO TYPE--OCTAL ASCII
                                :TYPE 2 DIGIT(S)
                                :SUPPRESS LEADING ZEROS
                                :'STARTED'
        006450 104403          TYPOS
        006452      002      .BYTE      2          :AT END OF AVAILABLE TABLE
        006453      000      .BYTE      0          :BR IF YES
64 006454 104401 076053   TYPE      , ASGND      :INCREMENT AVAILABLE TABLE INDEX
65 006460 005762 001610   4$:      TST      AVAIL(R2) :CONTINUE LOOKING FOR END OF TABLE
66 006464 001402          BEQ      5$           :MOVE ADDR OF DRIVE INTO AVAIL LST
67 006466 005722          TST      (R2)+        :TAKE DRIVE OUT OF NEW DRIVE TABLE
68 006470 000773          BR       4$           :SET DRIVE ASSIGNED INDICATOR
69 006472 016562 001566 001610 5$:      MOV      NEWUNT(R5),AVAIL(R2)
70 006500 005065 001566          CLR      NEWUNT(R5)   :CLEAR AUTO ASSIGN
71 006504 156137 040300 001542   BISB    ATABIT(R1),ASNLST
72 006512 005037 032012          CLR      AUTLST      :INCREMENT AVAILABLE TABLE POINTER
73 006516 005722          TST      (R2)+        :LOOK FOR MORE DRIVES
74 006520 000741          BR       2$
75
76                          :GET PARAMETERS, BUFFER SPACE, AND START ORDERS FOR DRIVES IN
77                          :THE 'AVAILABLE' QUEUE
78
79 006522 005002          MAIN2:  CLR      R2          :START FROM THE FIRST LOCATION
80 006524 105737 001542   TSTB    ASNLST        :ANY DRIVES ACTIVE ?
81 006530 001025          BNE      2$           :BR IF YES
82 006532 105737 001150   TSTB    $AUTOB       :RUNNING IN AUTO MODE ?
83 006536 001020          BNE      1$           :BR IF YES
84 006540 012737 000001 001340   MOV      #1,CFLAG     :DUMMY 'CONTROL C' FLAG
85 006546 013737 001314 001352   MOV      HZ,ONESEC    :RESTORE ONE SECOND COUNTER VALUE
86 006554 005037 001344          CLR      HOUR        :CLEAR THE HOUR'S COUNTER
87 006560 005037 001346          CLR      MINUTE      :CLEAR THE MINUTE'S COUNTER
88 006564 005037 001350          CLR      SECOND      :CLEAR THE SECOND'S COUNTER
89 006570 005037 001466          CLR      INTRVL+2    :CLEAR INTERVAL COUNTER
90 006574 104401 077103   TYPE    , NODRVS      :TYPE 'NO DRIVES ASSIGNED'
91 006600 000137 031742   1$:      JMP      $GET42   :GIVE CONTROL TO MONITOR
92
93 006604 005762 001632   2$:      TST      WAIT(R2) :ANY DRIVES WAITING FOR THE BUFFER ?
94 006610 001435          BEQ      5$           :BR IF NO
95 006612 016200 001632   MOV      WAIT(R2),R0  :LOAD R0 WITH THE DPB ADDRESS
96 006616 005046          CLR      -(SP)       :CLEAR THE STACK FOR BUFFER REQ
97 006620 004737 016242   JSR     PC,GETBUF     :CALL TO GET THE BUFFER RT.
98 006624 012660 000006   MOV      (SP)+,$BUF(R0) :IF 0,BUFFER IS STILL NOT AVAILABLE
99 006630 001423          BEQ      4$           :BRANCH IF NO BUFFER AVAILABLE
100 006632 005060 000122   CLR     $NEXT(R0)    :CLEAR PARAMETER SELECT FLAG
101 006636 005060 000106   CLR     $FAIR(R0)    :CLEAR THE FAIR FLAG
102 006642 004737 016632   JSR     PC,FILBUF    :FILL THE BUFFER
103 006646 004737 016710   JSR     PC,GODRIV    :SET COMMAND AND GO
104 006652 012705 001520   MOV     #ORDERQ,R5  :PUT THE WAIT QUE INTO ORDER QUE
105 006656 005725          3$:      TST      (R5)+        :QUE AVAILABLE ?
106 006660 001376          BNE      3$           :BR IF NO
107 006662 010045          MOV     R0,-(R5)     :LOAD THE DPB ADDRESS INTO THE ORDER QUE
108 006664 012701 001632   MOV     #WAIT,R1     :REMOVE THE DRIVE FROM THE 'WAIT' QUE
109 006670 060201          ADD     R2,R1        :OFFSET THE QUE POSITION
110 006672 004737 020750   JSR     PC,COMPRES   :COMPRESS THE QUE
    
```

```

111 006676 000742          BR      2$      :BRANCH IF DONE
112 006700 005722      4$:  TST      (R2)+    :CHECK THE NEXT QUE
113 006702 000740          BR      2$      :LOOPING BACK
114
115 006704 005737 001520  5$:  TST      ORDERQ    :ANY OUTSTANDING ORDERS ?
116 006710 001075          BNE      IDLE      :BR IF YES
117
118 006712 005002          CLR      R2        :CLEAR DRIVE TABLE POINTER
119 006714 005762 001610  6$:  TST      AVAIL(R2)  :ANY DRIVES WAITING FOR PARAMETERS
120 006720 001002          BNE      7$        :BRANCH IF ANY
121 006722 000137 007104  JMP      IDLE      :BRANCH IF NONE
122 006726 016200 001610  7$:  MOV      AVAIL(R2),R0 :CONTROL BLOCK ADDR IN R0
123 006732 005760 000122  TST      $NEXT(R0)  :PARAMETERS BEEN SELECTED ?
124 006736 001010          BNE      9$        :BR IF THEY HAVE
125 006740 105760 000026  TSTB     $PACK(R0)  :'R' OR 'W' COMMAND FOR THIS DRIVE ?
126 006744 001403          BEQ      8$        :BR IF NO
127 006746 004737 017000  JSR      PC,WRTPK   :GET DATA PARAMETERS
128 006752 000404          BR      10$       :GET THE BUFFER
129
130 006754 004737 017370  8$:  JSR      PC,GENPAR :GO GENERATE THE PARAMETERS
131 006760 004737 020500  9$:  JSR      PC,LODPAR  :LOAD THE PARAMETERS JUST GENERATED
132 006764 005046          CLR      -(SP)     :MAKE ROOM ON THE STACK FOR THE BUFFER ADDR
133 006766 004737 016242  JSR      PC,GETBUF  :GET BUFFER
134 006772 012660 000006  MOV      (SP)+,$BUF(R0) :MOVE BUFFER ADDR TO DPB
135 006776 001416          BEQ      12$       :BR IF '0' ADDR (NO BUFFER)
136 007000 004737 016632  JSR      PC,FILBUF  :FILL THE BUFFER
137 007004 005060 000122  CLR      $NEXT(R0)  :CLEAR PARAMETER SELECT FLAG
138 007010 005060 000106  CLR      $FAIR(R0)  :CLEAR THE 'FAIRNESS' COUNT
139 007014 004737 016710  JSR      PC,GODRIV  :PUT CURRENT DPB IN DRIVER
140 007020 012705 001520  MOV      #ORDERQ,R5 :ADDRESS OF ORDER QUE IN R5
141 007024 005725          TST      (R5)+    :END OF QUE ?
142 007026 001376          BNE      11$       :BR IF NOT
143 007030 010045          MOV      R0,-(R5)  :PUT BLOCK ADDRESS INTO QUE
144 007032 000416          BR      15$       :CONTINUE LOOKING
145
146 007034 005260 000106  12$:  INC      $FAIR(R0)  :INCREMENT THE FAIR COUNT
147 007040 022760 000003 000106  CMP      #3,$FAIR(R0) :THREE TIMES,BUFFER IS NOT AVAILABLE?
148 007046 101006          BHI      14$        :BRANCH IF NOT OVER THREE TIMES
149 007050 012705 001632  MOV      #WAIT,R5   :LOAD INTO THE WAIT QUE
150 007054 005725          TST      (R5)+    :AN AVAILABLE LOCATION ?
151 007056 001376          BNE      13$       :BRANCH IF NOT
152 007060 010045          MOV      R0,-(R5)  :LOAD INTO WAIT QUE
153 007062 000402          BR      15$       :REMOVE THE DPB FROM AVAILABLE QUE
154
155 007064 005722          TST      (R2)+    :INCREMENT INDEX
156 007066 000712          BR      6$        :BRANCH BACK TO FIRE NEXT DRIVE
157 007070 012701 001610  15$:  MOV      #AVAIL,R1  :'AVAILABLE' TABLE ADDRESS
158 007074 060201          ADD      R2,R1     :FORM ADDRESS OF LAST ENTRY
159 007076 004737 020750  JSR      PC,COMPRES :COMPRESS THE TABLE
160 007102 000704          BR      6$        :CONTINUE LOOKING
161
162          :WAIT FOR A COMMAND TO FINISH
163
164 007104 012701 001520  IDLE:  MOV      #ORDERQ,R1 :ADDRESS OF THE ORDER QUE IN R1
165 007110 012100          1$:  MOV      (R1)+,R0  :PUT BLOCK ADDRESS INTO R0
166 007112 001433          BEQ      4$        :BR IF END OF QUE
167 007114 005760 000016  2$:  TST      $STATUS(R0) :SEE IF DRIVE FINISHED
    
```

168	007120	001775			BEQ	2\$:BR IF DRIVE NOT FINISHED	
169	007122	005741			TST	-(R1)	:BACKUP THE QUE POINTER	
170	007124	010146			MOV	R1,-(SP)	:SAVE THE QUE ADDRESS	
171	007126	004737	016072		JSR	PC,STATIS	:ACCUMULATE STATISTICS FOR DRIVE IN R0	
172	007132	004737	007252		JSR	PC,PROCES	:PROCESS END OF COMMAND	
173	007136	005060	000112		CLR	\$SENBR(R0)	:CLEAR SKIP SECTORING ENABLED	
174	007142	005037	001342		CLR	BADSEC	:CLEAR THE BAD TRK/SEC ERROR INDICATOR	
175	007146	004737	031264		JSR	PC,ABNRML	:SEE IF ANY DRIVES HAVE TOO MANY ERRORS	
176	007152	004737	031312		JSR	PC,\$EOP	:IS IT END OF PASS ?	
177	007156	012601			MOV	(SP)+,R1	:RESTORE THE ORDER TABLE INDEX	
178	007160	012705	001610		MOV	#AVAIL,R5	:ADDRESS OF THE 'AVAIL' TABLE	
179	007164	005725		3\$:	TST	(R5)+	:IS IT THE END OF THE 'AVAIL' TABLE ?	
180	007166	001376			BNE	3\$:BR IF NO	
181	007170	011145			MOV	(R1),-(R5)	:MOVE THE DPB INTO THE 'AVAIL' TABLE	
182	007172	004737	020750		JSR	PC,CMPRES	:COMPRESS THE ORDER QUE	
183	007176	004737	016376		JSR	PC,RELBUF	:RESTORE BUFFER	
184								
185	007202	032777	000004	171744	4\$:	BIT	#SW02,@SWR	:TYPE PERFORMANCE SUMMARY
186	007210	001016			BNE	5\$:BR IF NOT	
187	007212	005737	001316		TST	STATIN	:TIME TO TYPE THE PERFORMANCE SUMMARY ?	
188	007216	001413			BEQ	5\$:BR IF NOT	
189	007220	005037	001316		CLR	STATIN	:CLEAR THE INDICATOR	
190	007224	005737	001542		TST	ASNLST	:ANY DRIVES ASSIGNED ?	
191	007230	001406			BEQ	5\$:BR IF NO	
192	007232	104401	075525		TYPE	,REPHD	:TYPE PERFORMANCE REPORT HEADING	
193	007236	004737	023626		JSR	PC,STATPR	:TYPE THE SUMMARY	
194	007242	104401	075566		TYPE	,STAR30	:TYPE '****...ETC'	
195	007246	000137	006240		5\$:	JMP	MAIN	:CONTINUE THE LOOP

```

1          ;PROCESS THE COMMAND TERMINATION
2
3 007252 111037 001324 PROCES: MOVB (R0),DRVNO ;DRIVE NUMBER FOR ANY ERROR MESSAGES
4 007256 005760 000016 TST STATUS(R0) ;SEE IF DRIVER SIGNALLED AN ERROR
5 007262 100427 BMI ERPROC ;BR IF ERROR
6 007264 032760 100000 002140 BIT #BIT15,$RMCS1(R0) ;SEE IF 'SC' SET
7 007272 001410 BEQ 1$ ;BR IF NOT SET
8 007274 032760 040000 002140 BIT #BIT14,$RMCS1(R0) ;SEE IF 'TRE' SET
9 007302 001017 BNE ERPROC ;BR IF SET
10 007304 032760 040000 002152 BIT #BIT14,$RMDS(R0) ;SEE IF 'ERR' SET
11 007312 001013 BNE ERPROC ;BR IF SET
12
13          ;NO ERRORS DETECTED IN REGISTERS, DO SOME CHECKING ANYWAY
14
15 007314 004737 013062 1$: JSR PC,CKERR ;CHECK ERROR BITS
16 007320 004737 013154 JSR PC,CKBUS ;CHECK BUS ADDRESS & WORD COUNT
17 007324 032777 000002 171622 BIT #SW01,@SWR ;DO DATA COMPARE ?
18 007332 001002 BNE 2$ ;BR IF NO
19 007334 004737 013240 JSR PC,CMPAR ;COMPARE DATA W/O ERROR
20 007340 000207 RTS PC ;RETURN
21
22          ;COMMAND TERMINATED WITH AN ERROR - PROCESS THE ERROR
23
24 007342 032760 000200 000016 ERPROC: BIT #BIT07,$STATUS(R0) ;DONE BIT SET ?
25 007350 001402 BEQ ERPRC1 ;BR IF COMMAND DIDN'T COMPLETE NORMALLY
26 007352 000137 007550 JMP DONE ;PROCESS ERROR WITH 'DONE' BIT SET
27
28          ;PROCESS COMMAND COMPLETION WITH 'ERROR' & 'DONE NOT' BITS
29
30 ERPRC1:
39 007356 032760 001000 000016 BIT #BIT09,$STATUS(R0) ;TIMEOUT?
40 007364 001011 BNE SWTIM ;BR IF YES
41 007366 032760 040002 000016 BIT #BIT14!BIT01,$STATUS(R0) ;DRIVE WENT OFFLINE ?
42 007374 001024 BNE OFLIN ;BR IF IT DID
43 007376 032760 000004 000016 BIT #BIT2,$STATUS(R0) ;PORT REQUEST TIME OUT ?
44 007404 001042 BNE PRTIM ;BR IF IT DID
45 007406 000207 RTS PC ;ERROR. RETURN
46
47
48
49
50
51
52
53          ;SOFTWARE TIMEOUT OCCURRED
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 007410 004737 021134 SWTIM: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
86 007414 104414 070307 DISPLY ,EM13 ;PRINT THE TIME OUT MESSAGE
87 007420 004737 021214 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
88 007424 004737 021654 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
89 007430 004737 022324 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
90 007434 004737 024740 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
91 007440 004737 022746 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
92 007444 000207 RTS PC ;RETURN
93
94          ;DRIVE WENT OFFLINE
95
96 007446 104401 001203 OFLIN: TYPE ,$CRLF ;CR-LF
97 007452 004737 021134 JSR PC,LINE1 ;PRINT LINE 1 OF THE ERROR MESSAGE
98 007456 104414 070361 DISPLY ,EM14 ;PRINT OFFLINE MESSAGE
99 007462 004737 021214 JSR PC,LINE2 ;PRINT LINE 2 OF THE ERROR MESSAGE
100 007466 004737 021654 JSR PC,LINE3 ;PRINT LINE 3 OF THE ERROR MESSAGE

```

```

101 007472 004737 022324      JSR    PC,LINE4      ;PRINT LINE 4 OF THE ERROR MESSAGE
102 007476 004737 024740      JSR    PC,INCTOT    ;INCREMENT TOTAL ERROR COUNT
103 007502 004737 022746      JSR    PC,LINE7     ;PRINT LINE 7 OF THE ERROR MESSAGE
104 007506 000137 031204      JMP    DROP         ;DROP THE DRIVE
105
106                          ;PORT REQUEST TIMEOUT ERROR
107
108 007512 004737 021134      PRTIM: JSR    PC,LINE1 ;TYPE LINE 1 OF THE ERROR MESSAGE
109 007516 104414 070404      DISPLY ,EM15       ;PRINT PORT TIME OUT MESSAGE
112 007522 004737 021214      JSR    PC,LINE2     ;TYPE LINE 2 OF THE ERROR MESSAGE
      007526 004737 021654      JSR    PC,LINE3     ;TYPE LINE 3 OF THE ERROR MESSAGE
      007532 004737 022324      JSR    PC,LINE4     ;TYPE LINE 4 OF THE ERROR MESSAGE
113 007536 004737 024740      JSR    PC,INCTOT    ;INCREMENT TOTAL ERROR COUNT
114 007542 004737 022746      JSR    PC,LINE7     ;TYPE LINE 7 OF THE ERROR MESSAGE
115 007546 000207              RTS    PC           ;RETURN
116
117                          ;PROCESS COMMAND COMPLETION WITH 'ERROR' & 'DONE' BITS SET
118
124 007550 032760 000020 000016 DONE: BIT    #BIT04,$STATUS(R0) ;UNSAFE OCCURRED
125 007556 001402              BEQ    1$           ;BR IF NOT
126 007560 000137 012532      JMP    UNSAF        ;REPORT UNSAFE
127
128 007564 032760 040000 002154 1$: BIT    #BIT14,$RMER1(R0) ;UNSAFE OCCURRED
129 007572 001402              BEQ    2$           ;BR IF NOT
130 007574 000137 012532      JMP    UNSAF        ;REPORT UNSAFE
131
132 007600 032760 040000 002150 2$: BIT    #BIT14,$RMCS2(R0) ;IS 'WCE' SET ?
133 007606 001402              BEQ    3$           ;BRANCH IF NOT SET
134 007610 000137 010546      JMP    WCKER        ;WRITE CHECK ERROR
135
136 007614 032760 040000 002152 3$: BIT    #BIT14,$RMDS(R0) ;CHECK 'ERR'
137 007622 001002              BNE    4$           ;BR IF SET
138 007624 000137 012272      JMP    TRFER        ;PROCESS 'TRE'
139
140 007630 032760 000400 002154 4$: BIT    #BIT08,$RMER1(R0) ;'HCRC' SET?
141 007636 001402              BEQ    5$           ;BR IF NOT
142 007640 000137 011054      JMP    HRCER        ;PROCESS 'HCRC'
143
144 007644 032760 000020 002154 5$: BIT    #BIT04,$RMER1(R0) ;'FMT' SET?
145 007652 001402              BEQ    6$           ;BR IF NOT SET
146 007654 000137 011232      JMP    CKFMT        ;CHECK FORMAT ERROR
147
148 007660 032760 000200 002154 6$: BIT    #BIT07,$RMER1(R0) ;'HCC' SET?
149 007666 001402              BEQ    7$           ;BR IF NOT SET
150 007670 000137 011406      JMP    CKHCE        ;CHECK 'HCE' ERROR
151
152 007674 032760 020000 002154 7$: BIT    #BIT13,$RMER1(R0) ;'OPI' SET?
153 007702 001402              BEQ    8$           ;BR IF NOT SET
154 007704 000137 011666      JMP    OPIER        ;REPORT 'OPI'
155
156 007710 032760 000010 002154 8$: BIT    #BIT3,$RMER1(R0) ;'PAR' SET?
157 007716 001402              BEQ    9$           ;BR IF NOT SET
158 007720 000137 012014      JMP    PARER        ;REPORT 'PAR'
159
160 007724 032760 000040 002154 9$: BIT    #BIT5,$RMER1(R0) ;'WCF' SET?
161 007732 001402              BEQ    10$          ;BR IF NOT SET
162 007734 000137 012434      JMP    WCFER        ;REPORT 'WCF'

```

```

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

```

220 010204 004737 024740          JSR    PC,INCTOT      ;INCREMENT TOTAL ERROR COUNT
221 010210 012737 000003 001330  MOV    #3,RETRY      ;RETRY COUNT
222 010216 004737 015744          JSR    PC,$RETRY     ;RETRY THE COMMAND
223 010222 000403                   BR     4$            ;RETRY WAS NOT SUCCESSFUL
224 010224 004737 022706          JSR    PC,LINE6C     ;PRINT 'CORRECTED ON N RETRIES'
225 010230 000402                   BR     5$            ;FINISH THE ERROR REPORT
226
227 010232 004737 022714          4$:   JSR    PC,LINE6D     ;PRINT 'UNCORRECTABLE AFTER N RETRIES'
228 010236 004737 022746          5$:   JSR    PC,LINE7     ;TYPE LINE 7 OF ERROR MESSAGE
229 010242 000207                   RTS    PC            ;RETURN
230
231                               ;THE VALUES IN THE ECC REGISTERS ARE CORRECT, REPORT 'DCK' ERROR
232
233 010244 004737 021134          6$:   JSR    PC,LINE1     ;PRINT LINE 1 OF ERROR MESSAGE
234 010250 104414 070461          DISPLY ,EM21        ;DATA CHECK ERROR
235 010254
236
237 010254 004737 021214          DCKER1: JSR    PC,LINE2     ;PRINT LINE 2 OF ERROR MESSAGE
238 010260 004737 021654          JSR    PC,LINE3     ;PRINT LINE 3 OF ERROR MESSAGE
239 010264 004737 022324          JSR    PC,LINE4     ;PRINT LINE 4 OF ERROR MESSAGE
240 010270 004737 015236          JSR    PC,PRTBAD    ;SEE IF BAD SECTOR TO BE PRINTED
241 010274 012737 110100 001326  MOV    #BIT15!BIT12!BIT06,MASK ;LOAD ERROR MASK
242 010302 032760 010100 002154  BIT    #BIT12!BIT06,$RMER1(R0) ;CHECK 'DTE' & 'ECH'
243 010310 001003                   BNE    1$            ;BR IF SET
244 010312 004737 022666          JSR    PC,LINE6     ;PRINT 'SECTOR IS ECC CORRECTABLE'
245 010316 000460                   BR     9$            ;FINISH THE ERROR REPORT
246 010320 012737 000020 001330  1$:   MOV    #16.,RETRY   ;RETRY COUNT
250
251 010326 004737 016710          2$:   JSR    PC,GODRIV    ;RETRY
252 010332 005760 000016          3$:   TST    $STATUS(R0) ;TEST FOR DONE
253 010336 001775                   BEQ    3$            ;BR IF NOT DONE
254 010340 100057                   BPL    11$           ;BR IF NOT ERROR
255 010342 032760 000200 000016  BIT    #BIT7,$STATUS(R0) ;SEE IF COMMAND TERMINIATED NORMALLY
256 010350 001006                   BNE    4$            ;BR IF NOT
257 010352 004737 024740          JSR    PC,INCTOT    ;INCREMENT TOTAL ERROR COUNT
258 010356 104414 074343          DISPLY ,LINBM      ;'DIFFERENT ERROR DURING RETRY'
259 010362 000137 007356          JMP    ERPRC1       ;SEE WHICH ERROR
260
261 010366 033760 001326 002154  4$:   BIT    MASK,$RMER1(R0) ;LOOK AT CURRENT ERROR
262 010374 001412                   BEQ    6$            ;BR IF DIFFERENT ERROR
263 010376 032760 010100 002154  BIT    #BIT12!BIT6,$RMER1(R0) ;'ECH' OR 'DTE' STILL SET ?
264 010404 001421                   BEQ    8$            ;BR IF NEITHER SET
265 010406 105237 001331          INCB   RETRY+1      ;INCREMENT RETRY COUNT
266 010412 123737 001330 001331  CMPB   RETRY,RETRY+1 ;DONE ?
267 010420 001342                   BNE    2$            ;BR IF NOT
278 010422 004737 023170          6$:   JSR    PC,LINE8     ;PRINT LINE 8 OF ERROR MESSAGE
279 010426 004737 024644          7$:   JSR    PC,INCHRD   ;INCREMENT 'HARD' ERROR COUNT
280 010432 004737 024740          JSR    PC,INCTOT    ;INCREMENT TOTAL ERROR COUNT
281 010436 004737 022746          JSR    PC,LINE7     ;PRINT LINE 7 OF ERROR MESSAGE
282 010442 004737 015236          JSR    PC,PRTBAD    ;PRINT THE BAD SECTOR
283 010446 000436                   BR     14$           ;CLEAN UP AND RETURN
284
288 010450 004737 022666          8$:   JSR    PC,LINE6     ;PRINT 'SECTOR IS ECC CORRECTABLE'
290 010454 004737 022624          JSR    PC,LINE5B    ;PRINT LINE 5B OF THE ERROR MESSAGE
291 010460 004737 024620          9$:   JSR    PC,INCSOF   ;INCREMENT 'SOFT' ERROR COUNT
292 010464 004737 014472          JSR    PC,ECC       ;CORRECT THE ERROR USING ECC AND CHECK IT
293 010470 000407                   BR     12$           ;COMPARE THE BUFFER
    
```



```

294
295 010472 004737 022714      10$: JSR   PC,LINE6D      :PRINT 'UNCORRECTABLE AFTER N RETRIES'
296 010476 000753              BR     7$             :INCRFMENT ERROR COUNT
297
298 010500 004737 022700      11$: JSR   PC,LINE6A      :PRINT LINE 6A OF ERROR MESSAGE
299 010504 004737 024620      JSR   PC,INCSOF      :INCREMENT 'SOFT' ERROR COUNT
300 010510 012737 000001 001356 12$: MOV   #1,FRSTER     :SET PROCESSING 'DCKER' INDICATOR
301 010516 004737 013244      JSR   PC,COMPARD     :COMPARE THE BUFFER
302 010522 105737 001357      TSTB  FRSTER+1      :ERROR IN COMPARE ?
303 010526 100406              BMI   14$           :BRANCH IF ERROR
304 010530 004737 024740      JSR   PC,INCTOT     :INCREMENT TOTAL ERROR COUNT
305 010534 104414 074560      DISPLY .LIN9G      :'DATA COMPARE OK' MESSAGE
306 010540 004737 022746      13$: JSR   PC,LINE7      :PRINT LINE 7 OF ERROR MESSAGE
307 010544 000207      14$: RTS   PC         :RETURN
308
309 :WRITE CHECK ERROR PROCESSING
310
311 010546 032760 100000 002154 WCKER: BIT   #BIT15,$RMER1(R0) :SEE IF 'DCK' SET ALSO
312 010554 001034              BNE   2$           :BR IF IT IS
313 010556 004737 021134      JSR   PC,LINE1      :PRINT LINE 1 OF ERROR MESSAGE
314 010562 104414 070565      DISPLY .EM23       :PRINT WCE & DCK NOT
315 010566 005037 001326      CLR   MASK          :CLEAR ERROR MASK
318 010572 004737 021214      JSR   PC,LINE2      :PRINT LINE 2 OF ERROR MESSAGE
      010576 004737 021654      JSR   PC,LINE3      :PRINT LINE 3 OF ERROR MESSAGE
      010602 004737 022324      JSR   PC,LINE4      :PRINT LINE 4 OF ERROR MESSAGE
      010606 004737 022414      JSR   PC,LINE5      :PRINT LINE 5 OF ERROR MESSAGE
319 010612 004737 024740      JSR   PC,INCTOT     :INCREMENT TOTAL ERROR COUNT
320 010616 012737 000003 001330  MOV   #3,RETRY      :RETRY LIMIT
321 010624 004737 015744      JSR   PC,$RETRY     :RETRY THE OPERATION
322 010630 000403              BR     1$           :RETRY UNSUCCESSFUL
323 010632 004737 022706      JSR   PC,LINE6C     :PRINT 'CORRECTED ON N RETRIES'
324 010636 000501              BR     10$          :FINISH PROCESSING THE ERROR
325
326 010640 004737 022714      1$: JSR   PC,LINE6D      :PRINT 'UNCORRECTABLE AFTER N RETRIES'
327 010644 000476              BR     10$          :FINISH PROCESSING THE ERROR
328 010646 004737 020764      2$: JSR   PC,SPOTCK     :SEE IF ERROR AT BAD SECTOR ON THE DISK
329 010652 000477              BR     11$          :EXIT IF AT BAD SECTOR ON DISK
330 010654 004737 021134      JSR   PC,LINE1      :PRINT LINE 1 OF ERROR MESSAGE
331 010660 012737 070512 010706  MOV   #EM22,4$      :ASSUME THAT EM22 WILL BE PRINTED
332 010666 032760 040000 002150  BIT   #BIT14,$RMCS2(R0) :DID 'WCK' ALSO SET ?
333 010674 001003              BNE   3$           :BR IF IT DID
334 010676 012737 071413 010706  MOV   #EM37,4$      :MESSAGE FOR 'DCK' AND 'WCK' NOT DURING
335 :WRITE CHECK
336 010704 104414      3$: DISPLY :TYPE THE ERROR MESSAGE
337 010706 000000      4$: .WORD 0 :MESSAGE ADDRESS GOES HERE
340 010710 004737 021214      JSR   PC,LINE2      :PRINT LINE 2 OF ERROR MESSAGE
      010714 004737 021654      JSR   PC,LINE3      :PRINT LINE 3 OF ERROR MESSAGE
      010720 004737 022324      JSR   PC,LINE4      :PRINT LINE 4 OF ERROR MESSAGE
      010724 004737 022414      JSR   PC,LINE5      :PRINT LINE 5 OF ERROR MESSAGE
341 010730 032760 000100 002154  BIT   #BIT06,$RMER1(R0) :ECH SET ALSO ?
342 010736 001441              BEQ   10$           :FINISH PROCESSING THE ERROR
343 010740 012737 000020 001330  5$: MOV   #16,RETRY    :RETRY LIMIT - 16 (10)
344 010746 004737 016710      6$: JSR   PC,GODRIV     :RETRY THE COMMAND
345 010752 005760 000016      7$: TST   $STATUS(R0)  :COMMAND FINISHED ?
346 010756 001775              BEQ   7$           :BR IF NOT
347 010760 100405              BMI   8$           :BR IF ERROR ON COMMAND
348 010762 105237 001331      INCB  RETRY+1      :INCREMENT RETRY COUNT
    
```

```

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

```

411 011240 001402      BEQ      1$      ;BR IF NOT SET
412 011242 000137 0.054  JMP      HRCRER   ;REPORT HCRER ERROR
413
414 011246 004737 023254      1$:      JSR      PC,READDR ;GET CORRECTED TRACK & SECTOR ADDRSES
415 011252 004737 015652      JSR      PC,READHD ;READ HEADER
416 011256 032737 000400 067612  BIT      #BIT8,GENREG+RMER1 ;'HCRER' SET WHEN HEADER READ?
417 011264 001002      BNE      2$      ;BR IF 'HCRER' SET
418 011266 000137 012166      JMP      FMTER    ;NO, ERROR IS 'FMT' ONLY
419
420 011272 004737 020764      2$:      JSR      PC,SPOTCK ;SEE IF ERROR AT BAD SECTOR ON THE DISK
421 011276 000442      BR       5$      ;EXIT IF IT IS
422 011300 004737 021134      JSR      PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
423 011304 104414 070644      DISPLY  ,EM24    ;HEADER READ ERROR - FMT BIT DROPPED UP
424 011310 004737 021214      JSR      PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
425 011314 004737 021654      JSR      PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
426 011320 004737 022324      JSR      PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
432 011324 004737 022556      3$:      JSR      PC,LINESA ;DISPLAY HEADER
433 011330 004737 024620      JSR      PC,INCSOF ;INCREMENT SOFT ERROR COUNT
434 011334 004737 024740      JSR      PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
435 011340 012737 000020 001326  MOV      #BIT4,MASK ;SET ERROR MASK
436 011346 012737 000003 001330  MOV      #3,RETRY ;RETRY LIMIT
437 011354 004737 015744      JSR      PC,$RETRY ;RETRY THE COMMAND
438 011360 000405      BR       4$      ;RETRY NOT SUCESSFUL
439 011362 004737 022706      JSR      PC,LINE6C ;PRINT 'CORRECTED ON N RETRIES'
440 011366 004737 022746      JSR      PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
441 011372 000404      BR       5$      ;EXIT
442
443 011374 004737 022714      4$:      JSR      PC,LINE6D ;PRINT 'UNCORRECTABLE AFTER N RETRIES'
444 011400 004737 022746      JSR      PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
445 011404 000207      5$:      RTS       PC     ;RETURN
446
447      ;PROCESS HEADER COMPARE ('HCE') ERROR
448
449 011406 032760 000400 002154  CKHCE:  BIT      #BIT8,$RMER1(R0) ;HCRER SET ON ORIGINAL ERROR ?
450 011414 001402      BEQ      1$      ;BR IF NOT SET
451 011416 000137 011054      JMP      HRCRER   ;REPORT HEADER CRC ERROR
452 011422 004737 023254      1$:      JSR      PC,READDR ;GET CURRENT SECTOR & TRACK ADDR
453 011426 004737 015652      JSR      PC,READHD ;READ HEADER OF CURRENT SECTOR
454 011432 032737 000400 067612  BIT      #BIT8,GENREG+RMER1 ;'HCRER' SET ?
455 011440 001017      BNE      3$      ;BR IF SET
456 011442 013746 101174      MOV      CYLNR,-(SP) ;:PUSH CYLNR ON STACK
457 011446 042737 170000 101174  BIC      #170000,CYLNR ;CLEAR FORMAT,MFG,USER AND SSF BITS FROM HEADER
458 011454 026037 002174 101174  CMP      $RMDC(R0),CYLNR ;CORRECT CYLINDER ?
459 011462 001402      BEQ      2$      ;BR IF IT IS
460 011464 000137 011614      JMP      POSER    ;REPORT POSITIONING ERROR
461 011470
462 011470 012637 101174      2$:      MOV      (SP)+,CYLNR ;:POP STACK INTO CYLNR
463 011474 000137 012230      JMP      HCEER    ;REPORT 'HCE' ERROR
464
464 011500 004737 020764      3$:      JSR      PC,SPOTCK ;SEE IF ERROR AT BAD SECTOR
465 011504 000442      BR       6$      ;EXIT IF IT IS
466 011506 004737 021134      JSR      PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
467 011512 104414 070712      DISPLY  ,EM25    ;HEADER READ ERROR - 'HCE' SET
468 011516 004737 021214      JSR      PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
469 011522 004737 021654      JSR      PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
470 011526 004737 022324      JSR      PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
476 011532 004737 022556      4$:      JSR      PC,LINESA ;PRINT LINE 5 OF ERROR MESSAGE
    
```

```

477 011536 004737 024620      JSR      PC,INCSOF      ;INCREMENT SOFT ERROR COUNT
478 011542 004737 024740      JSR      PC,INCTOT      ;INCREMENT TOTAL ERROR COUNT
479 011546 012737 000200      MOV      #BIT7,MASK     ;SET ERROR MASK
480 011554 012737 000003      MOV      #3,RETRY       ;RETRY LIMIT
481 011562 004737 015744      JSR      PC,$RETRY      ;RETRY THE COMMAND
482 011566 000405              BR        5$            ;RETRY NOT SUCESSFUL
483 011570 004737 022706      JSR      PC,LINE6C      ;PRINT 'CORRECTED ON N RETRIES'
484 011574 004737 022746      JSR      PC,LINE7       ;PRINT LINE 7 OF ERROR MESSAGE
485 011600 000404              BR        6$            ;EXIT
486
487 011602 004737 022714      5$:      JSR      PC,LINE6D ;PRINT 'UNCORRECTABLE AFTER N RETRIES'
488 011606 004737 022746      JSR      PC,LINE7       ;PRINT LINE 7 OF ERROR MESSAGE
489 011612 000207              6$:      RTS        PC        ;RETURN
490
491                      ;REPORT POSSIBLE POSITIONING ERROR
492
493 011614 004737 021134      POSER:   JSR      PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
494 011620 104414 072335      DISPLY   ,EM51          ;PROGRAM DETECTED POSITIONING ERROR
495 011624 004737 021214      JSR      PC,LINE2       ;PRINT LINE 2 OF ERROR MESSAGE
496 011630 004737 021702      JSR      PC,LINE3C      ;PRINT LINE 3C OF ERROR MESSAGE
497 011634 012637 101174      MOV      (SP)+,CYLNDR   ;POP STACK INTO CYLNR
498 011640 004737 022556      JSR      PC,LINE5A      ;PRINT LINE 5A OF THE ERROR MESSAGE
499 011644 004737 024714      JSR      PC,INCMIS      ;INCREMENT MISPOSITIONING COUNT
500 011650 004737 024740      JSR      PC,INCTOT      ;INCREMENT TOTAL ERROR COUNT
501 011654 004737 023076      JSR      PC,LINE7A      ;PRINT LINE 7A OF ERROR MESSAGE
502 011660 004737 015534      JSR      PC,RECALT      ;RECALIBRATE
503 011664 000207              RTS        PC        ;EXIT
504
505                      ;REPORT 'OPI' ERROR
506
507 011666 004737 020764      OPIER:   JSR      PC,SPOTCK ;SEE IF ERROR AT BAD SECTOR
508 011672 000207              RTS        PC        ;RETURN IF IT IS
509 011674 004737 021134      JSR      PC,LINE1       ;PRINT LINE 1 OF ERROR MESSAGE
510 011700 104414 071107      DISPLY   ,EM31          ;'OPI' ERROR
511 011704 004737 021214      JSR      PC,LINE2       ;PRINT LINE 2 OF ERROR MESSAGE
512 011710 004737 021654      JSR      PC,LINE3       ;PRINT LINE 3 OF ERROR MESSAGE
513 011714 004737 022324      JSR      PC,LINE4       ;PRINT LINE 4 OF ERROR MESSAGE
514 011720 004737 024740      JSR      PC,INCTOT      ;INCREMENT TOTAL ERROR COUNT
515 011724 012737 020000      MOV      #BIT13,MASK    ;ERROR MASK
516 011732 012737 000003      OPIER1: MOV      #3,RETRY   ;RETRY LIMIT
517 011740 004737 015744      JSR      PC,$RETRY      ;RETRY THE COMMAND
518 011744 000405              BR        1$            ;RETRY UNSUCCESSFUL
519 011746 004737 022706      JSR      PC,LINE6C      ;PRINT 'CORRECTED ON N RETRIES'
520 011752 004737 022746      JSR      PC,LINE7       ;PRINT LINE 7 OF ERROR MESSAGE
521 011756 000207              RTS        PC        ;EXIT
522
523 011760 004737 022714      1$:      JSR      PC,LINE6D      ;PRINT 'UNCORRECTABLE AFTER N RETRIES'
524 011764 004737 022746      JSR      PC,LINE7       ;PRINT LINE 7 OF ERROR MESSAGE
525 011770 000207              RTS        PC        ;RETURN
526
527                      ;REPORT 'DTE' ERROR
528
529 011772 004737 020764      DTEER:   JSR      PC,SPOTCK ;SEE IF ERROR AT BAD SECTOR
530 011776 000207              RTS        PC        ;RETURN IF IT IS
531 012000 004737 021134      JSR      PC,LINE1       ;PRINT LINE 1 OF ERROR MESSAGE
532 012004 104414 071152      DISPLY   ,EM32          ;'DTE' ERROR
533 012010 000137 010254      JMP      DCKER1        ;FINISH PROCESSING THE 'DTE' ERROR
    
```

```

534
535      ;REPORT 'PAR' ERROR
536
537 012014 004737 021134      PARER: JSR    PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
538 012020 104414 071205      DISPLY  ,EM33      ;REPORT 'PAR'
539 012024 004737 021214      JSR    PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
540 012030 004737 021760      JSR    PC,LINE3E     ;PRINT LINE 3E OF ERROR MESSAGE
541 012034 004737 022324      JSR    PC,LINE4      ;PRINT LINE 4 OF ERROR MESSAGE
542 012040 004737 024740      JSR    PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
543 012044 012737 000010      MOV    #BIT03,MASK  ;ERROR MASK
544 012052 012737 000003      MOV    #3,RETRY     ;RETRY LIMIT
545 012060 004737 015744      JSR    PC,$RETRY    ;RETRY COMMAND
546 012064 000405                BR     2$           ;RETRY UNSUCCESSFUL
547 012066 004737 022706      JSR    PC,LINE6C    ;PRINT 'CORRECTED ON N RETRIES'
548 012072 004737 022746      1$:  JSR    PC,LINE7    ;PRINT LINE 7 OF ERROR MESSAGE
549 012076 000207                RTS    PC          ;EXIT
550
551 012100 004737 022714      2$:  JSR    PC,LINE6D  ;PRINT 'UNCORRECTABLE AFTER N RETRIES'
552 012104 000772                BR     1$          ;FINISH ERROR MESSAGE
553
554      ;REPORT 'IAE' ERROR
555
556 012106 004737 021134      IAEER: JSR    PC,LINE1  ;PRINT LINE 1 OF ERROR MESSAGE
557 012112 104414 071324      DISPLY  ,EM35      ;REPORT 'IAE'
558 012116 004737 021214      JSR    PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
559 012122 004737 022046      JSR    PC,LINE3F     ;PRINT LINE 3F OF ERROR MESSAGE
560 012126 004737 024740      JSR    PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
561 012132 004737 022746      JSR    PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
562 012136 000207                RTS    PC          ;RETURN
563
564      ;REPORT 'WLE' ERROR
565
566 012140 004737 021134      WLEER: JSR    PC,LINE1  ;PRINT LINE 1 OF ERROR MESSAGE
567 012144 104414 071362      DISPLY  ,EM36      ;REPORT 'WLE'
568 012150 004737 021214      JSR    PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
569 012154 004737 024740      JSR    PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
570 012160 004737 022746      JSR    PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
571 012164 000207                RTS    PC          ;RETURN
572
573      ;REPORT FORMAT ERROR
574
575 012166 004737 021134      FMTER: JSR    PC,LINE1  ;PRINT LINE 1 OF ERROR MESSAGE
576 012172 104414 070773      DISPLY  ,EM26      ;FORMAT ERROR
577 012176 004737 021214      JSR    PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
578 012202 004737 021654      JSR    PC,LINE3      ;PRINT LINE 3 OF ERROR MESSAGE
579 012206 004737 022324      JSR    PC,LINE4      ;PRINT LINE 4 OF ERROR MESSAGE
586 012212 004737 022556      JSR    PC,LINE5A     ;PRINT LINE 5A OF ERROR MESSAGE
587 012216 004737 024740      JSR    PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
588 012222 004737 022746      JSR    PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
589 012226 000207                RTS    PC
590
591      ;REPORT HEADER COMPARE ERROR
592
593 012230 004737 021134      HCEER: JSR    PC,LINE1  ;PRINT LINE 1 OF ERROR MESSAGE
594 012234 104414 071020      DISPLY  ,EM27      ;HEADER COMPARE ERROR
595 012240 004737 021214      JSR    PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
596 012244 004737 021654      JSR    PC,LINE3      ;PRINT LINE 3 OF ERROR MESSAGE
    
```

```

597 012250 004737 022324      JSR      PC,LINE4      ;PRINT LINE 4 OF ERROR MESSAGE
604 012254 004737 022556      JSR      PC,LINE5A     ;PRINT LINE 5A OF ERROR MESSAGE
605 012260 004737 024740      JSR      PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
606 012264 004737 022746      JSR      PC,LINE7     ;PRINT LINE 7 OF ERROR MESSAGE
607 012270 000207              RTS      PC            ;RETURN
608
609                          ;PROCESS CONTROL/INTERFACE TRANSFER ERROR
610
611 012272 004737 021134      TRFER:  JSR      PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
612 012276 104414 071475      DISPLY  ,EM40          ;RH CONTROLLER OR UNIBUS TRANSFER ERROR
613 012302 004737 021214      JSR      PC,LINE2     ;PRINT LINE 2 OF ERROR MESSAGE
614 012306 004737 021654      JSR      PC,LINE3     ;PRINT LINE 3 OF ERROR MESSAGE
615 012312 004737 022324      JSR      PC,LINE4     ;PRINT LINE 4 OF ERROR MESSAGE
616 012316 004737 024740      JSR      PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
617 012322 032760 121400 002150  BIT      #BIT15!BIT13!BIT9!BIT8,$RMCS2(R0) ;'DLT','UPE','MXF','MDPE' SET ?
618 012330 001415              BEQ      2$           ;BR IF NONE SET
619 012332 012737 000003 001330  MOV      #3,RETRY      ;RETRY LIMIT
620 012340 005037 001326      CLR      MASK         ;CLEAR ERROR MASK
621 012344 004737 015744      JSR      PC,$RETRY     ;RETRY THE OPERATION
622 012350 000403              BR       1$           ;RETURN HERE IF RETRY UNSUCCESSFUL
623 012352 004737 022706      JSR      PC,LINE6C    ;PRINT 'CORRECTED ON N RETRIES'
624 012356 000402              BR       2$           ;FINISH THE ERROR REPORT
625
626 012360 004737 022714      1$:     JSR      PC,LINE6D    ;PRINT 'UNCORRECTABLE AFTER N RETRIES'
627 012364 004737 022746      2$:     JSR      PC,LINE7     ;PRINT LINE 7 OF ERROR MESSAGE
628 012370 000207              RTS      PC
629
630                          ;PROCESS 'SKI' ERRORS
631
632 012372 004737 021134      SKIER:  JSR      PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
633 012376 104414 072277      DISPLY  ,EM50          ;'SKI' ERROR
634 012402 004737 021214      JSR      PC,LINE2     ;PRINT LINE 2 OF ERROR MESSAGE
635 012406 004737 021670      JSR      PC,LINE3B    ;PRINT LINE 3B OF ERROR MESSAGE
636 012412 004737 024740      JSR      PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
637 012416 004737 024670      JSR      PC,INCSKI    ;INCREMENT 'SKI' ERROR COUNT
638 012422 004737 023076      JSR      PC,LINE7A    ;PRINT LINE 7A OF ERROR MESSAGE
639 012426 004737 015534      JSR      PC,RECALT    ;RECALIBRATE
640 012432 000207              RTS      PC
641
642                          ;REPORT WRITE CLOCK FAILURE ('WCF')
643
644 012434 004737 021134      WCFER:  JSR      PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
645 012440 104414 071262      DISPLY  ,EM34          ;REPORT WRITE CLOCK FAILURE
646 012444 004737 021214      JSR      PC,LINE2     ;PRINT LINE 2 OF ERROR MESSAGE
647 012450 004737 021662      JSR      PC,LINE3A    ;PRINT LINE 3A OF ERROR MESSAGE
648 012454 004737 022324      JSR      PC,LINE4     ;PRINT LINE 4 OF ERROR MESSAGE
649 012460 004737 024740      JSR      PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
650 012464 004737 015236      JSR      PC,PRTBAD    ;SEE IF BAD SECTOR TO BE PRINTED
651 012470 012737 000003 001330  MOV      #3,RETRY      ;RETRY COUNT
652 012476 012737 000040 001326  MOV      #BIT05,MASK   ;ERROR MASK
653 012504 004737 015744      JSR      PC,$RETRY     ;RETRY THE COMMAND
654 012510 000405              BR       2$           ;RETURN HERE IF RETRY UNSUCCESSFUL
655 012512 004737 022706      JSR      PC,LINE6C    ;PRINT 'CORRECTED ON N RETRIES'
656 012516 004737 022746      1$:     JSR      PC,LINE7     ;PRINT LINE 7 OF ERROR MESSAGE
657 012522 000207              RTS      PC
658
659 012524 004737 022714      2$:     JSR      PC,LINE6D    ;PRINT 'UNCORRECTABLE AFTER N RETRIES'
    
```

```

660 012530 000772          BR      1$
661
662          ;PROCESS DRIVE UNSAFE ERROR
663
664 012532 004737 021134  UNSAF: JSR      PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
665 012536 104414 072444      DISPLY    ,EM60          ;REPORT DRIVE UNSAFE
666 012542 004737 021214      JSR      PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
667 012546 004737 021654      JSR      PC,LINE3      ;PRINT LINE 3 OF ERROR MESSAGE
668 012552 004737 024740      JSR      PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
669 012556 032760 040000 002202 BIT      #BIT14,$RMR2(R0) ;IS 'SKI' ALSO SET ?
670 012564 001016          BNE      2$            ;BR IF YES
671 012566 012737 040000 001326 MOV      #BIT14,MASK    ;LOAD THE ERROR MASK
672 012574 012737 000003 001330 MOV      #3,RETRY      ;RETRY COUNT
673 012602 004737 015744      JSR      PC,$RETRY     ;RETRY THE COMMAND
674 012606 000403          BR       1$            ;RETRY WAS UNSUCCESSFUL
675 012610 004737 022706      JSR      PC,LINE6C    ;PRINT 'CORRECTED ON N RETRIES'
676 012614 000402          BR       2$            ;CONTINUE WITH ERROR REPORT
677
678 012616 004737 022714      1$: JSR      PC,LINE6D    ;PRINT 'UNCORRECTABLE AFTER N RETRIES'
679 012622 004737 022746      2$: JSR      PC,LINE7    ;PRINT LINE 7 OF ERROR MESSAGE
680 012626 032760 040000 002202 BIT      #BIT14,$RMR2(R0) ;CHECK 'SKI' AGAIN
681 012634 001001          BNE      3$            ;BR IF SET
682 012636 000207          RTS      PC            ;RETURN
683 012640 004737 015534      3$: JSR      PC,RECALT   ;RECALIBRATE
684 012644 000207          RTS      PC            ;RETURN
685
686          ;REPORT AN 'UNKNOWN' DATA PATTERN
687
688 012646 105737 001356  NOMTCH: TSTB    FRSTER    ;FIRST ERROR IN THE SECTOR ?
689 012652 001013          BNE      1$            ;BR IF NOT OR IF PROCESSING 'DCKER'
690 012654 004737 021134      JSR      PC,LINE1    ;TYPE LINE 1 OF ERROR MESSAGE
691 012660 104414 071671      DISPLY    ,EM43      ;'CAN'T MATCH DATA WITH PATTERN'
692 012664 004737 021214      JSR      PC,LINE2    ;PRINT LINE 2 OF ERROR MESSAGE
693 012670 004737 021662      JSR      PC,LINE3A   ;PRINT LINE 3A OF ERROR MESSAGE
694 012674 004737 022324      JSR      PC,LINE4    ;PRINT LINE 4 OF ERROR MESSAGE
695 012700 000404          BR       2$            ;CONTINUE PROCESSING ERROR
696 012702 104414 071671      1$: DISPLY    ,EM43      ;'CAN'T MATCH DATA WITH PATTERN'
697 012706 104414 001203      DISPLY    ,$CRLF     ;CR-LF
698 012712 104414 074473      2$: DISPLY    ,LIN9I    ;HEADER FOR DATA PRINTOUT
706 012716 010146          MOV      R1,-(SP)     ;ADDRESS OF WORD 1
          JSR      PC,LINOC T ;TYPE WORD 1
          DISPLY    ,BLNKS2 ;TYPE 2 BLANKS
          MOV      (R1)+,-(SP) ;ADDRESS OF WORD 1
          JSR      PC,LINOC T ;TYPE WORD 1
          DISPLY    ,$CRLF ;CR-LF
          MOV      R1,-(SP) ;ADDRESS OF WORD 2
          JSR      PC,LINOC T ;TYPE WORD 2
          DISPLY    ,BLNKS2 ;TYPE 2 BLANKS
          MOV      (R1)+,-(SP) ;ADDRESS OF WORD 2
          JSR      PC,LINOC T ;TYPE WORD 2
          DISPLY    ,$CRLF ;CR-LF
          MOV      R1,-(SP) ;ADDRESS OF WORD 3
          JSR      PC,LINOC T ;TYPE WORD 3
          DISPLY    ,BLNKS2 ;TYPE 2 BLANKS
          MOV      (R1)+,-(SP) ;ADDRESS OF WORD 3
          JSR      PC,LINOC T ;TYPE WORD 3
          DISPLY    ,$CRLF ;CR-LF
    
```



```

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

```

1          ;COMPARE THE BUFFER
2
3 013240 005037 001356  CMPAR: CLR      FRSTER          ;CLEAR 'FIRST ERROR' INDICATOR
4
5 013244 132760 000004 000024  CMPARD: BITB   #4,$CODE(R0)  ;SEE IF READ COMMAND
6 013252 001001          BNE      1$          ;BR IF IT IS
7 013254 000207          RTS      PC          ;RETURN
8
9 013256 005037 001364  1$:   CLR      ERCTR          ;CLEAR THE ERROR COUNTER
10 013262 016001 000006  MOV     $BUF(R0),R1      ;BUFFER ADDRESS
11 013266 016037 000020 001370  MOV     $WRDL(R0),CMCNT ;WORD COUNT TO WORKING LOCATION
12 013274 066037 002142 001370  ADD     $RMWC(R0),CMCNT ;CALCULATE ACTUAL WORDS TRANSFERED
13 013302 001001          BNE      2$
14 013304 000207          RTS      PC          ;EXIT--NO WORDS XFERED
15
16 013306 016037 000012 001372  2$:   MOV     $CYL(R0),CMCYL ;CYLINDER ADDRESS WORKING LOCATION
17 013314 052737 170000 001372  BIS     #170000,CMCYL   ;SET MFG, USER, SSF AND FMT BITS
18 013322 016037 000010 001374  MOV     $SECT(R0),CMSEC ;SECTOR & TRACK ADDRESSES TO WORKING LOCNS
19 013330 013737 001460 001366  MOV     CMPLMT,LIMIT   ;DISPLAY LIMIT
20 013336 005237 001366  INC     LIMIT          ;CONVERT PARAMETER INTO LIMIT VALUE
21 013342 012737 177777 001354  CMSTR: MOV     #-1,ZROIND  ;CLEAR THE 'ZERO'S' INDICATOR
22 013350 005037 001360  CLR     SAVER1         ;CLEAR THE R1 SAVE WORD
23 013354 005037 001362  CLR     SAVER5         ;CLEAR THE R5 SAVE WORD
24 013360 023760 001370 000022  CMP     CMCNT,$SSEC(R0) ;IS BUFFER SIZE GREATER THAN ONE SECTOR ?
25 013366 101005          BHI     1$          ;BR IF IT IS
26 013370 013702 001370  MOV     CMCNT,R2       ;LESS THAN, USE REMAINING BUFFER
27 013374 005037 001370  CLR     CMCNT         ;SET COUNTER TO 0
28 013400 000405          BR      2$
29 013402 016002 000022  1$:   MOV     $SSEC(R0),R2  ;COMPARE SECTOR
30 013406 166037 000022 001370  SUB     $SSEC(R0),CMCNT ;DECREMENT WORD COUNT
31 013414 126027 000024 000005  2$:   CMPB   $CODE(R0),#5   ;READ HEADER & DATA?
32 013422 001026          BNE     CMDAT         ;BR IF NOT
33
34          ;COMPARE HEADER WORDS
35
36 013424 012705 001372  CMHED: MOV     #CMCYL,R5  ;ADDRESS OF COMPARING CYLINDER
37 013430 052711 170000  BIS     #170000,(R1)   ;SET BITS INCASE BAD SECTOR ENCOUNTER
38 013434 022521          CMP     (R5)+,(R1)+   ;CHECK CYLINDER
39 013436 001402          BEQ     1$          ;BR IF COMPARE OK
40 013440 004737 013466  JSR     PC,CMSTR2     ;REPORT ERROR
41 013444 022521  1$:   CMP     (R5)+,(R1)+   ;COMPARE SECTOR/TRACK
42 013446 001402          BEQ     2$          ;BR IF EQ
43 013450 004737 013466  JSR     PC,CMSTR2     ;REPORT ERROR
44 013454 162702 000002  2$:   SUB     #2,R2        ;SUBTRACT HEADER LENGTH FROM SIZE
45 013460 003007          BGT     CMDAT         ;BR IF NOT FINISHED
46 013462 000137 014052  JMP     CMPRX         ;COMPARE THE DATA PORTION
47
48 013466 005237 001364  CMSTR2: INC     ERCTR    ;INCREMENT THE ERROR COUNT
49 013472 004737 014060  JSR     PC,CMPRT     ;REPORT THE COMPARISON ERROR
50 013476 000207          RTS      PC          ;CHECK THE REST OF THE HEADER
51
52          ;COMPARE DATA FIELD
53
54 013500 004737 027462  CMDAT: JSR     PC,GETLMT ;GET ADDRESS LIMITS
55 013504 004737 014402  JSR     PC,MATCH     ;FIND THE PATTERN
56 013510 000403          BR      1$          ;FOUND A PATTERN
57 013512 004737 012646  JSR     PC,NOMTCH    ;RETURN HERE IF NO MATCH WITH PATTERN MADE
    
```

58	013516	000456			BR	7\$:BYPASS COMPARE ROUTINE
59							
60	013520	011405			1\$: MOV	(R4),R5	:ADDRESS OF PATTERN ADDRESS IN R4
61	013522	012703	000020		MOV	#16.,R3	:R3 IS PATTERN POS COUNTER
62	013526	022125			2\$: CMP	(R1)+,(R5)+	:COMPARE BUFFER WITH PATTERN
63	013530	001016			BNE	4\$:BR IF NOT EQUAL
64	013532	005737	001364		TSI	ERCTR	:ERRORS DETECTED ?
65	013536	001406			BEQ	3\$:BR IF NO ERRORS
66	013540	032777	000010	165406	BIT	#SW3,@SWR	:SWITCH 3 SET ?
67	013546	001402			BEQ	3\$:BR IF NOT SET
68	013550	004737	014060		JSR	PC,CMPRT	:DISPLAY THE WORD
69	013554	005302			3\$: DEC	R2	:DECREMENT SIZE COUNT
70	013556	003436			BLE	7\$:BR WHEN AT END
71	013560	005303			DEC	R3	:DECREMENT PATT POS COUNT
72	013562	001361			BNE	2\$:BR IF NOT AT END OF PATT
73	013564	000755			BR	1\$:RESTART THE PATTERN
74	013566	005761	177776		4\$: TST	-2(R1)	:IS MISCOMPARED CHARACTER=0
75	013572	001410			BEQ	5\$:BR IF YES
76	013574	012737	177777	001354	MOV	#-1,ZROIND	:SET NON-ZERO MISCOMPARED INDICATOR
77	013602	005237	001364		INC	ERCTR	:INCREMENT THE ERROR COUNTER
78	013606	004737	014060		JSR	PC,CMPRT	:REPORT ERROR
79	013612	000760			BR	3\$:CONTINUE COMPARE
80							
81	013614	105737	001356		5\$: TSTB	FRSTER	:FIRST ERROR?
82	013620	100407			BMI	6\$:BR IF NOT
83	013622	005037	001354		CLR	ZROIND	:SET THE ZERO INDICATOR
84	013626	010137	001360		MOV	R1,SAVER1	:SAVE CURRENT R1
85	013632	010537	001362		MOV	R5,SAVER5	:SAVE CURRENT R5
86	013636	000746			BR	3\$:CONTINUE COMPARE
87	013640	005737	001354		6\$: TST	ZROIND	:ANY MISCOMPARISONS NOT ZEROS ?
88	013644	001743			BEQ	3\$:BR IF NONE-ALL ERRORS=ZERO
89	013646	004737	014060		JSR	PC,CMPRT	:REPORT ERROR
90	013652	000740			BR	3\$:CONTINUE COMPARING
91							
92	013654	126027	000024	000005	7\$: CMPB	\$CODE(R0),#5	:READ HEAD AND DATA ?
93	013662	001414			BEQ	9\$:YES
94	013664	013702	001370		8\$: MOV	CMCNT,R2	:SET COUNTER = REMAIN BUFFER LENGTH
95	013670	020227	000004		CMP	R2,#4	:IS THERE AT LEAST 4 WORDS TO MATCH PATTERN ?
96	013674	002466			BLT	CMPRX	:BR IF NO
97	013676	162737	000400	001370	SUB	#256.,CMCNT	:GREATER THAN A SECTOR ?
98	013704	003675			BLE	CMDAT	:NO,RETURN TO COMPARE LOOP
99	013706	012702	000400		MOV	#256.,R2	:SET COUNTER =SECTOR SIZE
100	013712	000672			BR	CMDAT	:RETURN TO COMPARE LOOP
101							
102	013714	023727	001370	000002	9\$: CMP	CMCNT,#2	:IS THERE AT LEAST 2 WORDS TO COMPARE HEADER ?
103	013722	002453			BLT	CMPRX	:BR IF NO
104	013724	105237	001374		INCB	CMSEC	:INCREMENT COUNTER
105	013730	123737	001374	001424	CMPB	CMSEC,SECLMT	:MAX SECTOR # ?
106	013736	101424			BLOS	10\$:NO
107	013740	105037	001374		CLRB	CMSEC	:RESET SECTOR #
108	013744	105237	001375		INCB	CMTRK	:INCREMENT TRACK #
109	013750	123737	001375	001426	CMPB	CMTRK,TRKLMT	:MAX TRACK # ?
110	013756	101414			BLOS	10\$:NO
111	013760	105037	001375		CLRB	CMTRK	:RESET TRACK #
112	013764	005237	001372		INC	CMCYL	:INCREMENT CYLINDER NUMBER
113	013770	013746	001372		MOV	CMCYL,-(SP)	:GET COMPARING CYLINDER
114	013774	042716	170000		BIC	#170000,(SP)	:SAVE ONLY THE CYLINDER BITS

```

115 014000 022637 001422      CMP      (SP)+,CYLIMT      ;LAST CYLINDER ?
116 014004 101401              BLOS     10$              ;NO
117 014006 000421              BR       CMPRX            ;NORMAL RETURN,NOT WRAP AROUND
118
119 014010 012705 001372      10$:    MOV      #CMCYL,R5      ;ADDRESS OF COMPARING CYLINDER
120 014014 052711 170000      BIS      #170000,(R1)     ;SET BITS IN CASE BAD SECTOR ENCOUNTER
121 014020 022521              CMP      (R5)+,(R1)+     ;COMPARE 1ST 'EADER WORD
122 014022 001402              BEQ     11$              ;MATCH
123 014024 004737 013466      JSR     PC,CMSTR2        ;NOT MATCH
124 014030 022521      11$:    CMP      (R5)+,(R1)+     ;SECOND WORD OF HEADER
125 014032 001402              BEQ     12$              ;MATCH
126 014034 004737 013466      JSR     PC,CMSTR2        ;NOT MATCH
127 014040 162737 000002 001370 12$:    SUB      #2,CMCNT         ;ADJUST WORD COUNT
128 014046 003401              BLE     CMPRX            ;COMPARE IS DONE
129 014050 000705              BR       8$              ;RETURN TO COMPARE LOOP
130
131 014052 004737 014334      CMPRX:  JSR     PC,ENDCMP   ;PRINT LAST LINE IF ERRORS
132 014056 000207              RTS     PC
133
134      ;TYPE DATA COMPARE ERRORS
135
136 014060 005737 001360      CMPRT:  TST     SAVER1      ;PRINT SAVED VALUES ?
137 014064 001010              BNE     2$              ;BR IF YES
138 014066 105737 001356      TSTB   FRSTER           ;FIRST ERROR?
139 014072 100402              BMI     1$              ;BR IF NOT
140 014074 004737 014154      JSR     PC,4$           ;PRINT INITIAL MESSAGE INFO
141 014100 004737 014236      1$:    JSR     PC,8$           ;PRINT REMAINDER OF MESSAGE
142 014104 000422              BR       3$              ;EXIT
143 014106      2$:    MOV      R1,-(SP)        ;;PUSH R1 ON STACK
144 014110 010146      MOV      R5,-(SP)        ;;PUSH R5 ON STACK
145 014112 013701 001360      MOV      SAVER1,R1       ;DISPLAY SAVED R1
146 014116 013705 001362      MOV      SAVER5,R5       ;DISPLAY SAVED R5
147 014122 004737 014154      JSR     PC,4$           ;PRINT INITIAL MESSAGE INFO
148 014126 004737 014236      JSR     PC,8$           ;PRINT SAVED VALUES
149 014132 005037 001360      CLR     SAVER1          ;CLEAR SAVED REGISTER INDICATORS
150 014136 005037 001362      CLR     SAVER5          ;CLEAR THE OTHER ONE
151 014142 012605      MOV      (SP)+,R5        ;;POP STACK INTO R5
152 014144 012601      MOV      (SP)+,R1        ;;POP STACK INTO R1
153 014146 004737 014236      JSR     PC,8$           ;PRINT REMAINDER OF MESSAGE
154 014152 000207      3$:    RTS     PC              ;RETURN
155 014154 105737 001356      4$:    TSTB   FRSTER        ;FIRST ERROR ?
156 014160 100425      BMI     7$              ;BR IF NOT
157 014162 001013      BNE     5$              ;BR IF FIRST ERROR AND PROCESSING 'DCK' ERROR
158 014164 004737 021134      JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
159 014170 104414 071610      DISPLY  ,EM42           ;DATA COMPARE ERROR
160 014174 004737 021214      JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
161 014200 004737 021662      JSR     PC,LINE3A       ;PRINT LINE 3A OF ERROR MESSAGE
162 014204 004737 022324      JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
163 014210 000404      BR       6$              ;GO TO TYPE HEADER
164 014212 104414 074366      5$:    DISPLY  ,LIN9B        ;PRINT 'DATA COMPARISON ERRORS'
165 014216 104414 001203      DISPLY  ,$CRLF          ;CR-LF
166 014222 104414 074415      6$:    DISPLY  ,LIN9H        ;PRINT '
167 014226 012737 177777 001356      ;          LOC      EXPCTD  RECEVD
168 014234 000207      ;          DATA   DATA'
169      MOV      #-1,FRSTER   ;SET FIRST ERROR FLAG
170      RTS     PC              ;RETURN
    
```

```

169
170 014236 005737 001366      8$:  TST      LIMIT      ;TYPEOUT LIMIT REACHED ?
171 014242 001403              BEQ      9$           ;BR IF IT HAS
172 014244 005337 001366      DEC      LIMIT      ;DECREMENT LIMIT COUNTER
173 014250 001005              BNE      10$          ;BR IF NOT AT LIMIT
174 014252 032777 000200 164674 9$:  BIT      #SW07,@SWR   ;PRINT ALL DATA COMPARE ERRORS ?
175 014260 001001              BNE      10$          ;BR IF YES
176 014262 000207              RTS      PC           ;RETURN
177
178 014264 010146              10$:  MOV      R1,-(SP)   ;BUFFER ADDRESS
179 014266 162716 000002      SUB      #2,(SP)     ;ADJUST ADDRESS
180 014272 004737 023202      JSR      PC,LINOC   ;TYPE IT
181 014276 104414 075233      DISPLY  ,BLNKS2     ;TYPE 2 BLANKS
182 014302 016546 177776      MOV      -2(R5),-(SP);PUT GOOD DATA ON THE STACK
183 014306 004737 023202      JSR      PC,LINOC   ;TYPE IT
184 014312 104414 075233      DISPLY  ,BLNKS2     ;TYPE 2 BLANKS
185 014316 016146 177776      MOV      -2(R1),-(SP);BAD DATA
186 014322 004737 023202      JSR      PC,LINOC   ;TYPE IT
187 014326 104414 001203      DISPLY  ,$CRLF      ;CR-LF
188 014332 000207              RTS      PC           ;RETURN
189
190                          ;LAST LINE OF COMPARE ERROR REPORTING
191
192 014334 105737 001357      ENDCMP: TSTB     FRSTER+1 ;ANY COMPARE ERRORS FOUND ?
193 014340 001417              BEQ      2$           ;BR IF NOT
194 014342 005737 001364      TST     ERCTR       ;SEE HOW MANY ERRORS
195 014346 001410              BEQ      1$           ;BR IF ONLY CAN'T MATCH PATTERN
196 014350 104414 074531      DISPLY  ,LINEE     ;'NUMBER OF ERRORS='
197 014354 013746 001364      MOV     EPCTR,-(SP) ;NUMBER OF ERRORS
198 014360 004737 023234      JSR     PC,LINDEC  ;TYPE IT
199 014364 104414 001203      DISPLY  , $CRLF    ;CR-LF
200 014370 004737 024740      1$:  JSR     PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
201 014374 004737 022746      JSR     PC,LINE7   ;PRINT LINE 7 OF ERROR MESSAGE
202 014400 000207              2$:  RTS      PC           ;RETURN
203
204
205                          ;ROUTINE TO MATCH THE DATA WITH A PATTERN, ONLY WHEN LOCATION 'PATTERN'
206                          ;IS EQUAL TO 0 (RANDOM DATA PATTERN MODE). OTHERWISE, THIS ROUTINE WILL
207                          ;RETURN THE ADDRESS OF THE EXPECTED FIXED DATA PATTERN IN R4.
208                          ;CALL:
209                          ;
210                          ;
211                          ;
212                          ;
213                          ;
214                          ;
215                          ;
216                          ;
217                          ;
218                          ;
219                          ;
220 014402 010146              MATCH: MOV      R1,-(SP)   ;SAVE R1 ON THE STACK
221 014404 013704 001472      MOV     PATTERN,R4 ;WAS RANDOM PATTERN ENABLED ?
222 014410 001402              BEQ     1$           ;BR IF YES
223 014412 006304              ASL     R4           ;* 2
224 014414 000416              BR      4$           ;USE KNOWN PATTERN
225 014416 012704 000044      1$:  MOV     #44,R4    ;PATTERN TABLE INDEX
226 014422 011601              2$:  MOV     (SP),R1   ;RELOAD R1
227 014424 162704 000002      SJB    #2,R4        ;DECREMENT INDEX
228 014430 001413              BEQ     5$           ;BR IF PATTERN NOT MATCH
229 014432 016405 002324      MOV     SINDAT(R4),R5 ;ADDRESS OF PATTERN ADDRESS
230 014436 012703 000004      MOV     #4,R3       ;NUMBER OF LOCATIONS TO CHECK
231 014442 022125              3$:  CMP     (R1)+,(R5)+ ;COMPARE THE BUFFER AGAINST THE PATTERN

```

```

236 014444 001366          BNE      2$          ;BR IF NOT EQUAL, TRY NEXT PATTERN
237 014446 005303          DEC      R3          ;FINISHED CHECKING?
238 014450 001374          BNE      3$          ;BR IF NOT FINISHED
239 014452 062704 002324  4$:      ADD      #STNDAT,R4 ;MAKE PATTERN ADDRESS ABSOLUTE
240 014456 000403          BR       6$          ;EXIT
241 014460 062766 000002 000002 5$:      ADD      #2,2(SP) ;INCREMENT RETURN ADDRESS
242 014466 012601          6$:      MOV      (SP)+,R1 ;RESTORE R1
243 014470 000207          RTS      PC          ;RETURN
244
245                          ;USE ECC TO CORRECT THE DATA ERROR
246
247 014472 016037 002144 001400 ECC:    MOV      $RMBA(R0),ECSEC ;ADDRESS OF LAST LOCN XFERED
248 014500 016046 002142          MOV      $RMWC(R0),-(SP) ;ACT WORDS XFERED (2'S COMP)
249 014504 066016 000020          ADD      $WRDL(R0),(SP) ;ADD WORDS REQUESTED
250 014510 001002          BNE      1$
251 014512 005726          TST     (SP)+       ;RESTORE STACK
252 014514 000207          RTS      PC          ;EXIT--NO WORDS XFERRED
253 014516 005046          1$:      CLR      -(SP)      ;CLEAR NEXT STACK LOCN
254 014520 016046 000022          MOV      $SSEC(R0),-(SP) ;SECTOR SIZE
255 014524 004737 032040          JSR     PC,$DIV     ;DIVIDE WORDS XFERED BY SECTOR SIZE
256 014530 005716          TST     (SP)        ;PARTIAL SECTOR XFERED ?
257 014532 001413          BEQ     2$          ;BR IF NOT
258 014534 006316          ASL     (SP)        ;CONVERT INTO NUMBER OF BYTES
259 014536 161637 001400          SUB     (SP),ECSEC  ;SUBTRACT SECTOR RESIDUE
260 014542 122760 000005 000024 CMPB    #5,$CODE(R0) ;WAS OPERATION, READ HEAD & DATA
261 014550 001007          BNE      3$          ;BR IF NOT
262 014552 062737 000004 001400 ADD      #4,ECSEC   ;ADD HEADER SIZE (IN BYTES) BACK IN
263 014560 000403          BR       3$          ;GO ADJUST THE STACK POINTER
264 014562 162737 001000 001400 2$:      SUB     #256.*2,ECSEC ;SUBTRACT SECTOR DATA FIELD SIZE (IN BYTES)
265 014570 062706 000004 3$:      ADD      #4,SP      ;ADJUST THE STACK POINTER
266 014574 016037 002204 001376 MOV      $RMEC1(R0),ECBIT ;ECC POSITION COUNT
267 014602 005337 001376          DEC     ECBIT       ;ADJUST BIT POSITION
268 014606 013737 001376 001406 MOV      ECBIT,ECWRD ;LOAD THE WORD COUNT LOCATION
269 014614 042737 177760 001376 BIC     #^C17,ECBIT ;SAVE THE BIT OFFSET COUNT
270 014622 042737 000017 001406 BIC     #17,ECWRD   ;CLEAR THE BIT OFFSET
271 014630 006237 001406          ASR     ECWRD       ;CHANGE TO BYTE COUNT(DIVIDE BY 2)
272 014634 006237 001406          ASR     ECWRD       ;CHANGE TO BYTE COUNT(DIVIDE BY 4)
273 014640 006237 001406          ASR     ECWRD       ;CHANGE TO BYTE COUNT(DIVIDE BY 8.)
274 014644 104414 074606          DISPLY ,LIN10A     ;'ERROR BURST BEGINS AT '
275 014650 013746 001406          MOV      ECWRD,-(SP) ;PUT THE WORD COUNT ON THE STACK
276 014654 006216          ASR     (SP)        ;GET STARTING WORD FOR MESSAGE(DIVIDE BY 16.)
277 014656 004737 033230          JSR     PC,$SB2D   ;CONVERT THE WORD COUNT TO DECIMAL
278 014662 004737 032364          JSR     PC,$SUPRL  ;AND PRINT IT
279 014666 104414 074642          DISPLY ,LIN10B     ;' IN DATA FIELD OF ERROR SECTOR'
280 014672 063737 001400 001406 ADD      ECSEC,ECWRD ;FIND THE BEGINNING OF THE ERROR BURST
281 014700 026037 002144 001406 CMP      $RMBA(R0),ECWRD ;SEE IF BURST WAS IN DATA READ
282 014706 101002          BHI     4$          ;BR IF IN DATA READ
283 014710 000137 015224          JMP     ECC2        ;NOT IN DATA READ - REPORT IT
284
285 014714 016037 002206 001402 4$:      MOV      $RMEC2(R0),ECMSK0 ;GET THE ERROR BIT MASK
286 014722 005037 001404          CLR     ECMSK1     ;CLEAR THE UPPER MASK WORD
287 014726 005337 001376          5$:      DEC     ECBIT     ;DECREMENT THE BIT OFFSET COUNT
288 014732 002405          BLT     6$          ;BR IF DONE
289 014734 006337 001402          ASL     ECMSK0     ;SHIFT THE ERROR MASK
290 014740 006137 001404          ROL     ECMSK1     ;SHIFT THE LOWER INTO THE UPPER
291 014744 000770          BR      5$          ;CONTINUE THE SHIFT
292
    
```

```

293 014746 017737 164434 001412 6$:  MOV    @ECWRD,ECBADO  ;SAVE THE INCORRECT WORD
294 014754 013746 001402          MOV    ECMSK0,-(SP)  ;PUT LOWER MASK ON STACK
295 014760 047716 164422          BIC    @ECWRD,(SP)  ;CLEAR ERRONEOUS ONE BITS FROM MASK
296 014764 043777 001402 164414  BIC    ECMSK0,@ECWRD ;CLEAR ERRONEOUS ONE BITS FROM BAD WORD
297 014772 052677 164410          BIS    (SP)+,@ECWRD ;SET DROPPED BITS
298
299 014776 005737 001404          TST    ECMSK1       ;DOES ERROR GO INTO NEXT WORD ?
300 015002 001415          BEQ    7$          ;BR IF NO
301 015004 013737 001406 001414  MOV    ECWRD,ECWRD1 ;DUPLICATE ADDRESS
302 015012 062737 000002 001414  ADD    #2,ECWRD1   ;INCREMENT ERROR ADDRESS
303 015020 026037 002144 001414  CMP    $RMB(A(R0)),ECWRD1 ;IS NEXT WORD IN THE BUFFER ?
304 015026 101006          BHI    8$          ;BR IF YES, ELSE,
305 015030 005737 001402          TST    ECMSK0       ;WAS ERROR IN FIRST WORD ?
306 015034 001473          BEQ    ECC2        ;BR IF NO
307 015036 005037 001414          CLR    ECWRD1     ;CLEAR 2ND WORD ADDRESS
308 015042 000414          BR     ECC1        ;PRINT WORD CORRECTED
309
310 015044 017737 164344 001420 8$:  MOV    @ECWRD1,ECBAD1 ;SAVE THE SECOND BAD WORD
311 015052 013746 001404          MOV    ECMSK1,-(SP) ;PUT THE UPPER MASK ON THE STACK
312 015056 047716 164332          BIC    @ECWRD1,(SP) ;CLEAR ERRONEOUS ONE BITS FROM UPPER MASK
313 015062 043777 001404 164324  BIC    ECMSK1,@ECWRD1 ;CLEAR ERRONEOUS ONE BITS FROM DATA WORD
314 015070 052677 164320          BIS    (SP)+,@ECWRD1 ;SET DROPPED BITS
315
316 015074 104414 075006          ECC1:  DISPLY ,LIN10H      ;HEADER
321 015100 013746 001406          MOV    ECWRD,-(SP)  ;PUT ECWRD ON THE STACK
      015104 004737 023202          JSR    PC,LIN0CT    ;TYPE ECWRD
      015110 104414 075233          DISPLY ,BLNKS2     ;TYPE 2 BLANKS
      015114 013746 001412          MOV    ECBADO,-(SP) ;PUT ECBADO ON THE STACK
      015120 004737 023202          JSR    PC,LIN0CT    ;TYPE ECBADO
      015124 104414 075233          DISPLY ,BLNKS2     ;TYPE 2 BLANKS
      015130 017746 164252          MOV    @ECWRD,-(SP) ;PUT @ECWRD ON THE STACK
      015134 004737 023202          JSR    PC,LIN0CT    ;TYPE @ECWRD
      015140 104414 075233          DISPLY ,BLNKS2     ;TYPE 2 BLANKS
322
323 015144 005737 001414          TST    ECWRD1       ;PRINT THE NEXT WORD ?
324 015150 001427          BEQ    ECCX         ;BR IF NOT
325 015152 104414 001203          DISPLY ,$CRLF      ;CR-LF
330 015156 013746 001414          MOV    ECWRD1,-(SP) ;PUT ECWRD1 ON THE STACK
      015162 004737 023202          JSR    PC,LIN0CT    ;TYPE ECWRD1
      015166 104414 075233          DISPLY ,BLNKS2     ;TYPE 2 BLANKS
      015172 013746 001420          MOV    ECBAD1,-(SP) ;PUT ECBAD1 ON THE STACK
      015176 004737 023202          JSR    PC,LIN0CT    ;TYPE ECBAD1
      015202 104414 075233          DISPLY ,BLNKS2     ;TYPE 2 BLANKS
      015206 017746 164202          MOV    @ECWRD1,-(SP) ;PUT @ECWRD1 ON THE STACK
      015212 004737 023202          JSR    PC,LIN0CT    ;TYPE @ECWRD1
      015216 104414 075233          DISPLY ,BLNKS2     ;TYPE 2 BLANKS
331 015222 000402          BR     ECCX         ;EXIT
332
333 015224 104414 074702          ECC2:  DISPLY ,LIN10C    ;ERROR BURST WAS NOT TRANSFERED TO MEMORY
334 015230 104414 001203          ECCX:  DISPLY ,$CRLF  ;CR-LF
335 015234 000207          RTS    PC           ;RETURN
336
337          ;ROUTINE TO DISPLAY THE SECTOR WHICH GAVE THE HARD ERROR
338
339 015236 032777 000010 163710 PRTBAD: BIT    #SW3,@SWR    ;PRINT THE BAD SECTOR ?
340 015244 001520          BEQ    8$          ;BR IF NOT
341 015246 016001 002144          MOV    $RMB(A(R0)),R1 ;PUT THE END ADDRESS INTO R1
    
```



```

342 015252 016046 000020      MOV      $WRDL(R0),-(SP)  ;FIND THE BEGINNING OF THE SECTOR
343 015256 066016 002142      ADD      $RMWC(R0),(SP)  ;SUBTRACT THE WORDS NOT TRANSFERED
344 015262 001002                BNE      1$
345 015264 005726                TST      (SP)+           ;RESTORE STACK
346 015266 000207                RTS      PC              ;EXIT--NO WORDS XFERRED
347 015270 005046                CLR      -(SP)          ;MAKE THE UPPER DIVIDEND 0
348 015272 016046 000022      MOV      $SSEC(R0),-(SP) ;DIVIDE THE WORDS XFERED BY THE SECTOR SIZE
349 015276 004737 032040      JSR      PC,$DIV        ;DIVIDE
350 015302 005716                TST      (SP)          ;REMAINDER = 0 ?
351 015304 001403                BEQ      2$            ;BR IF IT IS - COMPLETE SECTOR TRANSFERED
352 015306 006316                ASL      (SP)          ;CONVERT THE RESIDUAL SECTOR INTO BYTE COUNT
353 015310 161601                SUB      (SP),R1       ;SUBTRACT IT FROM THE END ADDRESS
354 015312 000410                BR       3$            ;FINISH THE SIZING
355 015314 162701 001000      SUB      #256,*2,R1    ;SUBTRACT FULL SECTOR FROM END ADDR (IN BYTES)
356 015320 122760 000005 000024  CMPB     #5,$CODE(R0)  ;WAS OPERATION READ HEADER & DATA ?
357 015326 001002                BNE      3$            ;BR IF NOT
358 015330 162701 000004                SUB      #4,R1         ;SUBTRACT HEADER SIZE FROM ADDR
359 015334 062706 000004      ADD      #4,SP         ;RESTORE THE STACK POINTER
360 015340 104414 001203      DISPLY   ,$CRLF        ;CR-LF
361 015344 104414 075116      DISPLY   ,LIN11H       ;PRINT THE HEADER
362 015350 122760 000005 000024  CMPB     #5,$CODE(R0)  ;WAS OPERATION READ HEADER & DATA ?
363 015356 001021                BNE      4$            ;BR IF NOT
364 015360 104414 075171      DISPLY   ,LIN11        ;TYPE 'ADDR  HEADER'
365 015364 010146                MOV      R1,-(SP)      ;PUT THE ADDRESS ON THE STACK
366 015366 004737 023202      JSR      PC,LIN0CT     ;TYPE THE ADDRESS
367 015372 104414 075232      DISPLY   ,BLNKS3       ;TYPE 3 BLANKS
368 015376 012146                MOV      (R1)+,-(SP)   ;PUT WORD ON STACK
369 015400 004737 023202      JSR      PC,LIN0CT     ;TYPE THE 1ST HEADER WORD
370 015404 104414 075234      DISPLY   ,BLNKS1       ;TYPE 1 BLANK
371 015410 012146                MOV      (R1)+,-(SP)   ;PUT WORD ON STACK
372 015412 004737 023202      JSR      PC,LIN0CT     ;TYPE THE 2ND HEADER WORD
373 015416 104414 001203      DISPLY   ,$CRLF        ;CR-LF
374
375 015422 104414 075212      4$:     DISPLY   ,LIN11A  ;TYPE 'ADDR  DATA'
376 015426 012702 000010      5$:     MOV      #8.,R2      ;8. DATA WORDS PER LINE
377 015432 010146                MOV      R1,-(SP)      ;PUT THE ADDRESS ON THE STACK
378 015434 004737 023202      JSR      PC,LIN0CT     ;TYPE THE ADDRESS
379 015440 104414 075233      DISPLY   ,BLNKS2       ;TYPE 2 BLANKS
380 015444 020160 002144      6$:     CMP      R1,$RMB(A)   ;PRINTED ALL THE SECTOR ?
381 015450 001412                BEQ      7$            ;BR IF ALL PRINTED
382 015452 104414 075234      DISPLY   ,BLNKS1       ;TYPE 1 BLANK
383 015456 012146                MOV      (R1)+,-(SP)   ;PUT THE DATA ON THE STACK
384 015460 004737 023202      JSR      PC,LIN0CT     ;TYPE THE DATA
385 015464 005302                DEC      R2            ;DECREMENT THE HORIZONTAL COUNT
386 015466 001366                BNE      6$           ;BR IF NOT AT THE END OF THE LINE
387 015470 104414 001203      DISPLY   ,$CRLF        ;CR-LF
388 015474 000754                BR       5$            ;RESTORE THE WORDS/LINE COUNT
389 015476 104414 001203      7$:     DISPLY   ,$CRLF        ;CR-LF
390 015502 104414 001203      DISPLY   ,$CRLF        ;CR-LF
391 015506 000207      8$:     RTS      PC        ;RETURN
392
393      ;ROUTINE TO DO AN RTC - DRIVE SELECTED IN R0
394      ;CALL:
395      :     MOV      #DPB,R0      ;DPB ADDRESS
396      :     JSR      PC,R1NCTR
397      :     RETURN
398

```



```

399 015510 111037 067556          RTNCTR: MOV  (R0),GENDPB      ;MOVE THE DRIVE # TO THE GENERAL DPB
400 015514 112737 000117 067560  MOV  #RTC,GENDPB+$COMND  ;COMMAND CODE
401 015522 004037 041000          1$:   JSR  R0,RM80          ;DRIVER ENTRANCE
402 015526 067556                  GENDPB      ;DPB ADDRESS FOR COMMAND
403 015530 000774                  BR    1$      ;DRIVER DIDN'T ACCEPT COMMAND
404 015532 000207                  RTS    PC      ;RETURN
405
406          ;ROUTINE TO DO A RECALIBRATE USING ACTIVE DPB
407          ;CALL:
408          :       MOV  #DPB,R0          ;DPB ADDRESS
409          :       JSR  PC,RECALT
410          :       RETURN
411
412 015534 010037 015560          RECALT: MOV  R0,2$          ;LOAD THE DPB ADDRESS
413 015540 116060 002140 000027  MOV  $RMCS1(R0),$PREV0(R0) ;SAVE THE PREVIOUS COMMAND
414 015546 112760 000107 000002  MOV  #RECAL,$COMND(R0)    ;LOAD THE NEW COMMAND
415 015554 004037 041000          1$:   JSR  R0,RM80          ;START THE RECALIBRATE
416 015560 000000          2$:   .WORD 0          ;DPB ADDRESS
417 015562 000774          BR    1$      ;DRIVER DIDN'T ACCEPT THE COMMAND
418 015564 005760 000016          3$:   TST  $STATUS(R0)    ;SEE IF FINISHED
419 015570 001775          BEQ  3$      ;IF EQ NO
420 015572 004737 023254          JSR  PC,READDR          ;DECREMENT THE ADDRESSES
421 015576 012660 000034          MOV  (SP)+,$PREVA+2(R0)  ;MOVE THE CYLINDER ADDRESS
422 015602 112660 000033          MOV  (SP)+,$PREVA+1(R0)  ;MOVE THE TRACK ADDRESS
423 015606 112660 000032          MOV  (SP)+,$PREVA(R0)    ;MOVE THE SECTOR ADDRESS
424 015612 005060 000012          CLR  $CYL(R0)          ;CLEAR THE CURRENT CYLINDER ADDRESS
425 015616 005060 000010          CLR  $SEC(R0)          ;CLEAR THE CURRENT TRK/SEC ADDRESS
426 015622 000207          RTS    PC      ;RETURN
427
428          ;ROUTINE TO A RECAL WITH NO DPB ACTIVE
429          ;CALL:
430          :       MOV  #DRIVE,GENDPB    ;DRIVE ADDRESS
431          :       JSR  PC,RECALO
432          :       RETURN
433
434 015624 112737 000107 067560  RECALO: MOV  #RECAL,GENDPB+$COMND ;RELCALIBRATE COMMAND
435 015632 004037 041000          1$:   JSR  R0,RM80          ;DRIVER ENTRANCE
436 015636 067556                  GENDPB      ;DPB ADDRESS FOR COMMAND
437 015640 000774                  BR    1$      ;DRIVER DIDN'T ACCEPT THE COMMAND
438 015642 005737 067574          2$:   TST  GENDPB+$STATUS ;SEE IF FINISHED
439 015646 001775          BEQ  2$      ;BR IF NOT FINISHED
440 015650 000207          RTS    PC
441
442          ;UTILITY READ HEADER ROUTINE
443          ;CALL:
444          :       MOV  #DPB,R0          ;DPB ADDRESS
445          :       MOV  #SECTOR,-(SP)    ;SECTOR ADDRESS
446          :       MOV  #TRACK,-(SP)    ;TRACK ADDRESS
447          :       MOV  #CYLINDER,-(SP) ;CYLINDER ADDRESS
448          :       JSR  PC,READDR
449          :       RETURN
450
451 015652 116637 000004 067567  READHD: MOV  4(SP),GENDPB+$TRK    ;TRACK ADDRESS
452 015660 116637 000006 067566  MOV  6(SP),GENDPB+$SEC    ;SECTOR ADDRESS
453 015666 016637 000002 067570  MOV  2(SP),GENDPB+$CYL    ;CYLINDER ADDRESS
454 015674 111037 067556          MOV  (R0),GENDPB        ;DRIVE NUMBER
455 015700 112737 000173 067560  MOV  #RDHD,GENDPB+$COMND ;COMMAND
    
```

```

472 015706 012737 177776 067562      MOV      #-2,GENDPB+$WCNT      :WORD CTR = 2
473 015714 004037 041000      1$:     JSR      RO,RM80              :DRIVER ENTRANCE
474 015720 067556                      GENDPB                      :DPB ADDRESS FOR COMMAND
475 015722 000774                      BR      1$                   :DRIVER DIDN'T ACCEPT COMMAND
476 015724 005737 067574      2$:     TST      GENDPB+$STATUS   :FINISHED?
477 015730 001775                      BEQ     2$                   :BR IF NOT
478 015732 011666 000006      MOV     (SP),6(SP)           :ADJUST STACK FOR RETURN
479 015736 062706 000006      ADD     #6,SP                :ADJUST RFRTRN POINTER
480 015742 000207                      RTS      PC                  :RETURN
481
482      :RETRY THE PRESENT OPERATION
483      :CALL:
484      :
485      :
486      :
487      :
488      :
489      :
490      :
491 015744 004737 016710      $RETRY: JSR      PC,GODRIV        :RE-START COMMAND
492 015750 005760 000016      1$:     TST      $STATUS(RO)     :COMMAND FINISHED?
493 015754 001775                      BEQ     1$                   :BR IF NOT
494 015756 100405                      BMI     2$                   :BR IF ERROR
495 015760 105237 001331                      INCB   RETRY+1              :INCREMENT RETRY COUNT
496 015764 062716 000002                      ADD     #2,(SP)             :INCREMENT RETURN
497 015770 000425                      BR      5$                   :GO TO EXIT
498 015772 032760 000200 000016 2$:     BIT      #BIT7,$STATUS(RO)   :DID COMMAND TERMINATE NORMALLY ?
499 016000 001430                      BEQ     7$                   :BR IF NOT
500 016002 005737 001326                      TST     MASK                :IS ERROR MASK 0 ?
501 016006 001004                      BNE    3$                   :BR IF NOT
502 016010 005760 002154                      TST     $RMER1(RO)         :MAKE SURE THAT THE DRIVE ERROR REG IS CLEAR
503 016014 001014                      BNE    6$                   :BR IF NOT
504 016016 000404                      BR      4$                   :CONTINUE RETRY
505 016020 033760 001326 002154 3$:     BIT      MASK,$RMER1(RO)    :SAME ERROR?
506 016026 001407                      BEQ     6$                   :BR IF NOT
507 016030 105237 001331                      INCB   RETRY+1              :INCREMENT RETRY COUNT
508 016034 123737 001330 001331 4$:     CMPB   RETRY,RETRY+1        :DONE ?
509 016042 001340                      BNE    $RETRY               :BR IF NOT DONE
510 016044 000207                      5$:     RTS      PC                  :RETURN
511 016046 004737 023170      6$:     JSR      PC,LIN8            :REPORT DIFFERENT ERROR
512 016052 004737 022746                      JSR     PC,LIN7            :PRINT LINE 7
513 016056 005726                      TST     (SP)+               :ADJUST STACK POINTER FOR DIRECT RETURN
514 016060 000207                      RTS      PC                  :RETURN
515 016062 104414 074343      7$:     DISPLY  ,LIN8M           :'DIFFERENT ERROR DURING RETRY'
516 016066 000137 007356                      JMP     ERPRC1              :REPORT THE ERROR
    
```

```

1          ;ROUTINE TO UPDATE THE PERFORMANCE SUMMARY STATISTICS
2          ;CALL:
3          :
4          :       MOV      #DPB,RO          ;DPB ADDRESS
5          :       JSR      PC,STATIS
6          :       RETURN
7 016072 032760 000300 000016 STATIS: BIT    #BIT07!BIT06,$STATUS(RO) ;CHECK FOR DATA TERMINATION
8 016100 001456          BEQ      3$          ;BR IF NOT DATA TERMINATION
9 016102 016037 002144 016240      MOV      $RMBA(RO),FACTOR          ;STORE THE FINAL BUFFER ADDRESS
10 016110 166037 000006 016240     SUB      $BUF(RO),FACTOR ;SUBTRACT THE INITIAL ADDRESS
11 016116 001447          BEQ      3$          ;BR IF NO DATA TRANSFER
12 016120 006237 016240          ASR      FACTOR          ;CONVERT TO A WORD COUNT
13
14 016124 122760 000002 000024     CMPB    #2,$CODE(RO) ;SEE IF COMMAND WAS A WRITE
15 016132 001404          BEQ      1$          ;BRANCH IF YES
16 016134 122760 000000 000024     CMPB    #0,$CODE(RO) ;PRESENT OPERATION AN AUTO WRITE CHECK ?
17 016142 001012          BNE      2$          ;BR IF NO
18 016144 063760 016240 000060 1$:  ADD      FACTOR,$WRITN(RO) ;ADD WORDS WRITTEN DURING WRITE DATA
19 016152 005560 000062          ADC      $WRITN+2(RO) ;DID HIGH WORD OVFL0 AFTER ADDING CARRY ?
20 016156 102004          BVC      2$          ;BR IF NO
21 016160 005060 000062          CLR      $WRITN+2(RO) ;CLEAR HIGH WORD
22 016164 005260 000056          INC      $WTOFL(RO) ;AND COUNT WRITE OVERFLOW
23
24 016170 122760 000002 000024 2$:  CMPB    #2,$CODE(RO) ;SEE IF COMMAND WAS A WRITE
25 016176 001417          BEQ      3$          ;BRANCH IF YES
26 016200 063760 016240 000036     ADD      FACTOR,$ENDAT(RO) ;END OF PASS DATA WORD COUNT
27 016206 005560 000040          ADC      $ENDAT+2(RO) ;ADD ANY CARRY
28 016212 063760 016240 000066     ADD      FACTOR,$READ(RO) ;UPDATE THE READ WORD COUNT
29 016220 005560 000070          ADC      $READ+2(RO) ;DID HIGH WORD OVFL0 AFTER ADDING CARRY ?
30 016224 102004          BVC      3$          ;BR IF NO
31 016226 005060 000070          CLR      $READ+2(RO) ;CLEAR HIGH WORD
32 016232 005260 000064          INC      $RDOFL(RO) ;AND COUNT READ OVERFLOW
33 016236 000207          3$:  RTS      PC
34
35 016240 000000          FACTOR: .WORD 0 ;USED FOR WORDS TRANSFERED
36
37          ;ROUTINE TO GET A BUFFER
38          ;CALL:
39          :
40          :       MOV      #DPB,RO          ;DPB ADDRESS
41          :       CLR      -(SP)          ;CLEAR THE STACK
42          :       JSR      PC,GETBUF
43          :       RETURN          ;BUFFER ADDRESS WILL BE ON THE STACK
44          :                       ;STACK WILL BE ZERO IF NO BUFFER AVAILABLE
45 016242 010146          GETBUF: MOV      R1,-(SP) ;SAVE R1
46 016244 010246          MOV      R2,-(SP) ;SAVE R2
47 016246 010346          MOV      R3,-(SP) ;SAVE R3
48 016250 013702 001654          MOV      BUFTBL,R2 ;NUMBER OF SEPARATE BUFFERS
49 016254 001444          BEQ      5$          ;BR IF NONE AVAILABLE
50 016256 012701 001656          MOV      #BUFTBL+2,R1 ;FIRST ADDRESS OF ALLOCATION TABLE
51 016262 026061 000020 000002 1$:  CMP      $WRDL(RO),2(R1) ;SEE IF THERE IS A BLOCK LARGE ENOUGH
52 016270 101405          BLOS    2$          ;BRANCH IF IT IS
53 016272 005302          DEC      R2          ;DECREMENT TABLE COUNT
54 016274 001434          BEQ      5$          ;BR IF THROUGH TABLE
55 016276 062701 000004          ADD      #4,R1      ;INCREMENT TABLE POINTER
56 016302 000767          BR      1$          ;CONTINUE LOOKING
57 016304 011166 000010          2$:  MOV      (R1),10(SP) ;BUFFER ADDRESS TO STACK
    
```

```

58 016310 166061 000020 000002      SUB      $WRDL(R0),2(R1)  ;ADJUST BUFFER WRD CNT
59 016316 001407                    BEQ      3$              ;BR IF DIFFERENCE IS ZERO
60 016320 006360 000020              ASL      $WRDL(R0)       ;CONVERT # WORDS TO BYTES
61 016324 066011 000020              ADD      $WRDL(R0),(R1)  ;MAKE NEW STARTING ADDRESS
62 015330 006260 000020              ASR      $WRDL(R0)       ;RETURN # BYTES TO WORDS
63 016334 000414                    BR       5$              ;RETURN
64 016336 005337 001654              3$:     DEC      BUFTBL   ;DECREMENT ENTRIES COUNT
65 016342 001411                    BEQ      5$              ;BR IF ALLOCATION TABLE EMPTY
66 016344 005302                    DEC      R2              ;DECREMENT TABLE COUNT
67 016346 001407                    BEQ      5$              ;BR IF ITEM WERE LAST ENTRY
68 016350 010103                    MOV      R1,R3          ;MOVE TABLE POINTER
69 016352 062703 000004              ADD      #4,R3          ;POINT TO NEXT ENTRY
70 016356 012321                    4$:     MOV      (R3)+,(R1)+ ;MOVE ITEMS
71 016360 012321                    MOV      (R3)+,(R1)+
72 016362 005302                    DEC      R2              ;DECREMENT TABLE COUNT
73 016364 001374                    BNE      4$             ;CONTINUE IF NOT AT END OF TABLE
74 016366 012603                    5$:     MOV      (SP)+,R3   ;RESTORE R3
75 016370 012602                    MOV      (SP)+,R2       ;RESTORE R2
76 016372 012601                    MOV      (SP)+,R1       ;RESTORE R1
77 016374 000207                    RTS      PC              ;RETURN
78
79
80                                     ;ROUTINE TO PUT BUFFER BACK IN TABLE
81                                     ;CALL:
82                                     ;
83                                     ;
84                                     ;
85                                     ;
86 016376 010146      RELBUF:  MOV      R1,-(SP)   ;SAVE R1
87 016400 010246      MOV      R2,-(SP)   ;SAVE R2
88 016402 010346      MOV      R3,-(SP)   ;SAVE R3
89 016404 010446      MOV      R4,-(SP)   ;SAVE R4
90 016406 010546      MOV      R5,-(SP)   ;SAVE R5
91 016410 012701 001656      MOV      #BUFTBL+2,R1 ;BEGINNING OF TABLE
92 016414 013702 001654      MOV      BUFTBL,R2   ;ENTRY COUNT
93 016420 001424                    BEQ      2$           ;BR IF EMPTY TABLE
94 016422 016003 000110      MOV      $HLDWC(R0),R3 ;TRIAL ADDRESS
95 016426 006303                    ASL      R3           ;CHANGE TO BYTE COUNT
96 016430 066003 000006      ADD      $BUF(R0),R3  ;ADDRESS OF HIGHER ADJACENT BLOCK
97 016434 021103                    1$:     CMP      (R1),R3   ;UPPER ADJACENT BLOCK
98 016436 001424                    BEQ      3$           ;BR IF YES
99 016440 062701 000004      ADD      #4,R1        ;INCREMENT POINTER
100 016444 005302                    DEC      R2           ;DECREMENT ENTRY COUNT
101 016446 001372                    BNE      1$          ;CONTINUE SEARCHING
102 016450 016011 000006      MOV      $BUF(R0),(R1) ;PUT THE BUFFER BLOCK INTO THE TABLE
103 016454 016061 000110 000002      MOV      $HLDWC(R0),2(R1) ;BLOCK WRD CNT
104 016462 005237 001654      INC      BUFTBL      ;INCREMENT ENTRY COUNT
105 016466 005202                    INC      R2           ;INCREMENT R2 FOR USE LATER
106 016470 000414                    BR       4$           ;SEE IF A LOWER ADJACENT BLOCK IS IN THE TABLE
107 016472 016021 000006      2$:     MOV      $BUF(R0),(R1)+ ;BLOCK ADDRESS TO TABLE
108 016476 016021 000110      MOV      $HLDWC(R0),(R1)+ ;WRD CNT TO TABLE
109 016502 005237 001654      INC      BUFTBL      ;INCREMENT ENTRY COUNT
110 016506 000443                    BR       8$           ;EXIT
111 016510 016011 000006      3$:     MOV      $BUF(R0),(R1) ;RELEASED BUFFER IS LOWER ADJACENT
112 016514 066061 000110 000002      ADD      $HLDWC(R0),2(R1) ;INCREMENTED WRD CNT
113 016522 010246                    4$:     MOV      R2,-(SP)   ;SAVE R2
114 016524 013702 001654      MOV      BUFTBL,R2   ;ENTRY COUNT
    
```

MAIN PROGRAM

```

115 016530 012705 001656      MOV      #BUFTBL+2,R5      ;BEGINNING OF TABLE
116 016534 016504 000002      5$:     MOV      2(R5),R4      ;BLOCK SIZE (IN WORDS)
117 016540 006304              ASL      R4                ;CHANGE TO BYTE COUNT
118 016542 061504              ADD      (R5),R4          ;ADD BLOCK BEGINNING ADDRESS
119 016544 020411              CMP      R4,(R1)          ;R1 STILL POINTS TO INSERTED ENTRY
120 016546 001406              BEQ      6$                ;LOWER ADJACENT IN TABLE
121 016550 062705 000004      ADD      #4,R5            ;INCREMENT POINTER
122 016554 005302              DEC      R2                ;DECREMENT ENTRY COUNT
123 016556 001366              BNE      5$                ;CONTINUE LOOKING
124 016560 005726              TST      (SP)+            ;RESTORE STACK POINTER
125 016562 000415              BR       8$                ;END
126 016564 012602              6$:     MOV      (SP)+,R2        ;RESTORE R2
127 016566 066165 000002 000002  ADD      2(R1),2(R5)      ;INCREMENT LOWER BLOCK LENGTH
128 016574 005337 001654      DEC      BUFTBL           ;DECREMENT ENTRY COUNT
129 016600 010105              MOV      R1,R5            ;GET READY TO COMPRESS
130 016602 062705 000004      ADD      #4,R5            ;INCREMENT TO NEXT ENTRY
131 016606 012521              7$:     MOV      (R5)+,(R1)+    ;COMPRESS TABLE
132 016610 012521              MOV      (R5)+,(R1)+    ;MOVE SIZE FIELD DOWN
133 016612 005302              DEC      R2                ;DECREMENT ENTRY COUNT
134 016614 001374              BNE      7$                ;BR IF NOT FINISHED
135 016616 012605              8$:     MOV      (SP)+,R5        ;RESTORE R5
136 016620 012604              MOV      (SP)+,R4        ;RESTORE R4
137 016622 012603              MOV      (SP)+,R3        ;RESTORE R3
138 016624 012602              MOV      (SP)+,R2        ;RESTORE R2
139 016626 012601              MOV      (SP)+,R1        ;RESTORE R1
140 016630 000207              RTS      PC                ;RETURN
141
142      ;FILL THE ASSIGNED BUFFER (IF WRITE OR WRITE CHECK COMMAND)
143      ;CALL:
144      ;     MOV      #DPB,R0          ;DPB ADDRESS
145      ;     MOV      #BUFADR,$BUF(R0) ;LOAD BUFFER ADDRESS INTO THE DPB
146      ;     MOV      #PATTERN,$PATT(R0) ;PATTERN CODE
147      ;     JSR      PC,FILBUF
148      ;     RETURN
149
150 016632 104412      FILBUF: SAVREG           ;SAVE THE REGISTERS
151 016634 132760 000004 000024  BITB     #4,$CODE(R0)    ;SEE IF READ COMMAND
152 016642 001020              BNE      4$                ;BR IF READ
153 016644 016001 000006      1$:     MOV      $BUF(R0),R1    ;BUFFER ADDRESS
154 016650 016002 000020      MOV      $WRDL(R0),R2    ;POSITIVE WORD COUNT
155 016654 116004 000030      MOV      $PATT(R0),R4    ;RELATIVE PATTERN ADDRESS
156 016660 016405 002324      2$:     MOV      $STNDAT(R4),R5 ;PATTERN ADDRESS
157 016664 012703 000020      MOV      #16,R3          ;PATTERN COUNT
158 016670 012521      3$:     MOV      (R5)+,(R1)+    ;MOVE THE PATTERN INTO THE BUFFER
159 016672 005302              DEC      R2                ;DECREMENT THE WORD COUNT
160 016674 003403              BLE      4$                ;BR IF DONE (WORD COUNT = 0)
161 016676 005303              DEC      R3                ;DECREMENT THE PATTERN COUNT
162 016700 001373              BNE      3$                ;BR IF MORE PATTERN
163 016702 000766              BR       2$                ;CONTINUE DISTRIBUTING THE PATTERN
164 016704 104413      4$:     RESREG           ;RESTORE THE REGISTERS
165 016706 000207              RTS      PC                ;RETURN
166
167      ;START THE COMMAND FOR THE DPB IN R0
168      ;CALL:
169      ;     MOV      #DPB,R0          ;DPB ADDRESS
170      ;     JSR      PC,GODRIV
171      ;     RETURN

```

```
172  
173 016710 010046  
174 016712 010037 016722  
175 016716 004037 041000  
176 016722 000000  
177 016724 000000  
178 016726 012600  
179 016730 062760 000001 000046  
180 016736 005560 000050  
181 016742 026060 000034 000012  
182 016750 001412  
183 016752 062760 000001 000042  
184 016760 005560 000044  
185 016764 062760 000001 000052  
186 016772 005560 000054  
187 016776 000207  
  
GODRIV: MOV R0,-(SP) ;SAVE R0  
MOV R0,2$ ;CURRENT DPB ADDRESS  
1$: JSR R0,RM80 ;CALL THE DRIVE HANDLER  
2$: .WORD 0 ;DRIVE BLOCK ADDRESS GOES HERE  
HALT ;DRIVER REJECTED REQUEST  
MOV (SP)+,R0 ;RESTORE R0  
ADD #1,$OPERC(R0) ;INCREMENT THE OPERATION COUNT  
ADC $OPERC+2(R0)  
CMP $PREVA+2(R0),$CYL(R0) ;DID COMMAND REQUIRE A CYLINDER CHANGE ?  
BEQ 3$ ;BR IF NO  
ADD #1,$ENDSK(R0) ;INCREMENT END OF PASS SEEK COUNT  
ADC $ENDSK+2(R0) ;ADD ANY CARRY  
ADD #1,$POSIT(R0) ;INCREMENT SEEK COUNT  
ADC $POSIT+2(R0) ;ADD ANY CARRY  
3$: RTS PC
```

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14 017000 004737 027462
15 017004 005760 000050
16 017010 001011
17 017012 005760 000046
18 017016 001006
19 017020 012704 000010
20 017024 004737 020130
21 017030 000462
22 017032 000000
23
24 017034 116060 002140 000027 1$:
25 017042 016060 000010 000032
26 017050 016060 000012 000034
27 017056 016060 002146 000010
28 017064 016060 002174 000012
29
30 017072 032760 001000 002172
31 017100 001402
32 017102 005360 000010
33
34
35
36 017106 012704 000010 2$:
37 017112 004737 020130
38 017116 000427
39 017120 116060 000140 000010
40 017126 116060 000134 000011
41 017134 016060 000130 000012
42 017142 112760 000004 000024
43 017150 122760 177776 000026
44 017156 001473
45 017160 004737 031370
46 017164 032777 000020 161762
47 017172 001471
48 017174 000744
49
50 017176 012704 000010 3$:
51 017202 013705 001462
52 017206 004737 020330
53 017212 010560 000020
54 017216 042760 000377 000020
55 017224 001002
56 017226 105260 000021
57 017232 016060 000020 000004 4$:

```

```

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

```

```

WRTPK: JSR PC,GETLMT ;GET ADDRESS LIMITS
TST $OPERC+2(R0) ;IS THIS THE FIRST OPERATION ?
BNE 1$ ;BR IF NO
TST $OPERC(R0) ;IS THIS THE FIRST OPERATION ?
BNE 1$ ;BR IF NO
MOV #SSEC,R4 ;GET INDEX TO SECTOR ADDR STORAGE IN DPB
JSR PC,CKLMTS ;GO CHECK DISK ADDRESS LIMITS
BR 3$ ;BR IF NOT AT END OF SEQUENTIAL ADDRESSING
HALT ;SHOULD NOT GET HERE

1$: MOVB $RMCS1(R0),$PREVO(R0) ;SAVE CURRENT PARAMETERS
MOV $SSEC(R0),$PREVA(R0) ;SAVE PREVIOUS TRACK/SECTOR ADDRESS
MOV $CYL(R0),$PREVA+2(R0) ;SAVE PREVIOUS CYLINDER ADDRESS
MOV $RMDA(R0),$SSEC(R0) ;CURRENT SECTOR & TRACK ADDRESS
MOV $RMDC(R0),$CYL(R0) ;CURRENT CYLINDER ADDRESS

BIT #SSEI,$RMOF(R0) ;IS SSEI STILL SET ?
BEQ 2$ ;BR IF NOT
DEC $SSEC(R0) ;IF SO, THEN BACKUP ONE SECTOR TO REFLECT THE
;PROPER ADDRESS TO BE ACCESSED WHEN READING OR
;WRITING THE NEXT SEQUENTIAL SECTOR.

2$: MOV #SSEC,R4 ;GET INDEX TO SECTOR ADDR STORAGE IN DPB
JSR PC,CKLMTS ;GO CHECK DISK ADDRESS LIMITS
BR 3$ ;BR IF NOT AT END OF SEQUENTIAL ADDRESSING
MOVB MINSEC(R0),$SSEC(R0) ;RESET SECTOR ADDRESS
MOVB MINTRK(R0),$STRK(R0) ;RESET TRACK ADDRESS
MOV MINCYL(R0),$CYL(R0) ;RESET CYLINDER ADDRESS
MOVB #4,$CODE(R0) ;SET CODE TO READ DATA
CMPB #-2,$PACK(R0) ;WAS WRITE DATA PACK IN PROGRESS ?
BEQ 8$ ;BR IF YES (START TESTING)
JSR PC,EOP2 ;DROP THE DRIVE (NORMAL TERMINATION)
BIT #SW04,$SWR ;IS SWITCH 4 SET ?
BEQ 9$ ;BR IF NO
BR 2$ ;RE-CHECK FOR BSF & SSF TRACKS

3$: MOV #SSEC,R4 ;GET INDEX TO SECTOR STORAGE
MOV WRDCNT,R5 ;WORD COUNT IS MAXIMUM
JSR PC,CHKWC ;CHECK WORD COUNT FOR MAXCYL/MAXTRK
MOV R5,$WRDL(R0) ;GET WORD COUNT
BIC #377,$WRDL(R0) ;SECTOR BOUNDARY FOR WRITING
BNE 4$ ;NO
INCB $WRDL+1(R0) ;SET TO ONE SECTOR
MOV $WRDL(R0),$WCNT(R0) ;STORE FOR 2'S COMPLEMENT WORD

```

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


```

1          ;GENERATE PARAMETERS FOR THE OPERATION
2          ;CALL:
3          :
4          :      MOV      #DPB,R0      ;DPB ADDRESS
5          :      JSR      PC,GENPAR
6          :      RETURN
7 017370 004737 027462  GENPAR: JSR      PC,GETLMT      ;GET ADDRESS LIMITS
8 017374 004737 037024  JSR      PC,$RAND      ;CYCLE THE RANDOM NUMBER GENERATOR
9 017400 005737 001440  TST      RONLY        ;LOCKED IN READ ONLY MODE ?
10 017404 001016  BNE      1$           ;BR IF YES
11 017406 032777 000001 161540 BIT      #SW0,$SWR    ;SEE IF SW0 SET
12 017414 001012  BNE      1$           ;BR IF SET - READ ONLY
13 017416 012705 000010  MOV      #8,$R5       ;READ/WRITE SELECTION DIVISOR
14 017422 004737 032014  JSR      PC,GETREM    ;GET SELECTION VALUE
15 017426 020537 001476  CMP      R5,RATIO     ;DETERMINE IF READ OR WRITE
16 017432 103003  BHS     1$           ;BR IF READ
17 017434 012705 000002  MOV      #2,$R5       ;SELECT WRITE DATA COMMAND
18 017440 000407  BR       2$           ;SELECT ADDRESS
19
20 017442 013705 037124  1$:  MOV      $LONUM,$R5 ;SELECT READ OPERATION CODE
21 017446 000305  SWAB     R5           ;SWAP BYTES IN R5
22 017450 042705 177776  BIC      #^C1,$R5     ;MASK OUT ALL BUT BIT 0
23 017454 062705 000004  ADD      #4,$R5       ;TABLE OFFSET FOR READ CODE
24 017460 110560 000114  2$:  MOVB     R5,$NCODE(R0) ;COMMAND SELECTION CODE TO CONTROL BLOCK
25 017464 016060 002146 000116  MOV      $RMDA(R0),$NSEC(R0) ;SECTOR AND TRACK
26 017472 016060 002174 000120  MOV      $RMDC(R0),$NCTL(R0) ;CYLINDER NUMBER
27
28 017500 032760 001000 002172 BIT      #SSEI,$RMOF(R0) ;IS 'SSEI' STILL SET ?
29 017506 001402  BEQ     3$           ;BR IF NOT
30 017510 005360 000010  DEC      $SEC(R0)     ;IF SO, THEN BACKUP ONE SECTOR TO REFLECT THE
31                                     ;PROPER ADDRESS TO BE ACCESSED WHEN READING OR
32                                     ;WRITTING THE NEXT SEQUENTIAL SECTOR.
33 017514  3$:
34
35 017514 005737 001506  THEAD: TST      RANDOM ;ENABLE RANDOM ADDRESS SELECT ?
36 017520 001427  BEQ     RANCTL        ;YES
37 017522 005760 000050  TST      $OPERC+2(R0) ;IS THIS THE FIRST OPERATION ?
38 017526 001003  BNE     1$           ;BR IF NO
39 017530 005760 000046  TST      $OPERC(R0)  ;IS THIS THE FIRST OPERATION ?
40 017534 001405  BEQ     2$           ;BR IF YES
41
42 017536 012704 000116  1$:  MOV      #NSEC,$R4   ;GET INDEX TO SECTOR ADDR STORAGE IN DPB
43 017542 004737 020130  JSR      PC,CKLMTS   ;GO CHECK DISK ADDRESS LIMITS
44 017546 000412  BR      3$           ;BR IF NOT AT END OF SEQUENTIAL ADDRESSING
45 017550 116060 000140 000116  2$:  MOVB     MINSEC(R0),$NSEC(R0) ;RESET SECTOR ADDRESS
46 017556 116060 000134 000117  MOVB     MINTRK(R0),$NTRK(R0) ;RESET TRACK ADDRESS
47 017564 016060 000130 000120  MOV      MINCYL(R0),$NCTL(R0) ;RESET CYLINDER ADDRESS
48 017572 000761  BR      1$           ;RE-CHECK FOR BSF & SSF TRACKS
49 017574 000137 017746  3$:  JMP      RANSIZ     ;GO CHECK FOR RANDOM WORD SIZE
    
```

```

1          ;GENERATE A RANDOM CYLINDER ADDRESS BETWEEN VALUES 'MINCYL' & 'MAXCYL'
2
3 017600 016005 000126 RANCYL: MOV     MAXCYL(R0),R5 ;GET MAXIMUM CYLINDER ADDRESS
4 017604 026005 000130      CMP     MINCYL(R0),R5 ;'MINCYL' AND 'MAXCYL' THE SAME ?
5 017610 001407          BEQ     1$ ;BR IF THEY ARE
6 017612 166C05 000130      SUB     MINCYL(R0),R5 ;GET NUMBER OF ALLOWABLE CYLINDERS
7 017616 005205          INC     R5 ;INCREMENT DIFFERENCE TO USE AS DIVISOR
8 017620 004737 032014      JSR     PC,GETREM ;GET THE RANDOM AUGMENT
9 017624 066005 000130      ADD     MINCYL(R0),R5 ;NEW CYLINDER ADDRESS
10 017630 010560 000120     1$:  MOV     R5,$NCYL(R0) ;STORE CYLINDER ADDRESS IN DPB
11
12          ;GENERATE A RANDOM TRACK ADDRESS BETWEEN VALUES 'MINTRK' & 'MAXTRK'
13
14 017634 016005 000132 RANTRK: MOV     MAXTRK(R0),R5 ;GET MAXIMUM TRACK ADDRESS
15 017640 026005 000134      CMP     MINTRK(R0),R5 ;'MINTRK' AND 'MAXTRK' THE SAME ?
16 017644 001407          BEQ     1$ ;BR IF THEY ARE
17 017646 166005 000134      SUB     MINTRK(R0),R5 ;GET NUMBER OF ALLOWABLE TRACKS
18 017652 005205          INC     R5 ;INCREMENT DIFFERENCE TO USE AS DIVISOR
19 017654 004737 032014      JSR     PC,GETREM ;GET THE RANDOM AUGMENT
20 017660 066005 000134      ADD     MINTRK(R0),R5 ;NEW TRACK ADDRESS
21 017664 110560 000117     1$:  MOVB    R5,$NTRK(R0) ;STORE TRACK ADDRESS IN DPB
22
23          ;GENERATE A RANDOM SECTOR ADDRESS BETWEEN VALUES 'MINSEC' & 'MAXSEC'
24
25 017670 016005 000136 RANSEC: MOV     MAXSEC(R0),R5 ;GET MAXIMUM SECTOR ADDRESS
26 017674 026005 000140      CMP     MINSEC(R0),R5 ;'MINSEC' AND 'MAXSEC' THE SAME ?
27 017700 001407          BEQ     1$ ;BR IF THEY ARE
28 017702 166005 000140      SUB     MINSEC(R0),R5 ;GET NUMBER OF ALLOWABLE SECTORS
29 017706 005205          INC     R5 ;INCREMENT DIFFERENCE TO USE AS DIVISOR
30 017710 004737 032014      JSR     PC,GETREM ;GET THE RANDOM AUGMENT
31 017714 066005 000140      ADD     MINSEC(R0),R5 ;NEW SECTOR ADDRESS
32 017720 110560 000116     1$:  MOVB    R5,$NSEC(R0) ;STORE SECTOR ADDRESS IN DPB
33
34          ;MAKE SURE ADDRESS JUST GENERATED IS NOT 'BSF' OR 'SSF' TRACK
35
36 017724 012704 000116      MOV     #$NSEC,R4 ;GET INDEX TO SECTOR ADDR STORAGE IN DPB
37 017730 004737 020130      JSR     PC,CKLMTS ;GO CHECK DISK ADDRESS LIMITS
38 017734 000404          BR     2$ ;BR IF NOT AT END OF SEQUENTIAL ADDRESSING
39 017736 004737 037024      JSR     PC,$RAND ;CYCLE THE RANDOM NUMBER GENERATOR
40 017742 000137 017600      JMP     RANCYL ;GO GENERATE NEW ADDRESS
41 017746
42
43          ;GENERATE A RANDOM BUFFER LENGTH BETWEEN 6 & THE VALUE IN 'WRDCNT'
44
45 017746 013705 001462 RANSIZ: MOV     WRDCNT,R5 ;GET MAX WORD COUNT
46 017752 005737 001474      TST     RANDWC ;SELECT A RANDOM WORD COUNT ?
47 017756 001011          BNE    2$ ;BR IF NOT
48 017760 005205          INC     R5 ;INCREMENT THE MAXIMUM WRD CNT
49 017762 004737 032014      JSR     PC,GETREM ;DIVIDE BY MAX VALUE
50 017766 020527 000006      CMP     R5,#6 ;WORD COUNT LESS THAN 6 ?
51 017772 002003          BGE    2$ ;BR IF NO
52 017774 004737 037024     1$:  JSR     PC,$RAND ;CYCLE THE RANDOM NUMBER GENERATOR
53 020000 000762          BR     RANSIZ
54
55 020002 012704 000116     2$:  MOV     #$NSEC,R4 ;GET INDEX TO SECTOR STORAGE
56 020006 004737 020330      JSR     PC,CHKWC ;SEE IF WORD COUNT IS TOO LARGE TO FIT
57                                     ;IN REMAINDER OF TRACK. IF SO, THEN ADJUST
    
```

```

58                                     ;WORD COUNT TO FIT ON TRACK.
59 020012 122760 000002 000114 3$:  CMPB  #2,$NCODE(R0) ;WRITE OPERATION ?
60 020020 001005                                     ;BR IF NO
61 020022 042705 000377                                     ;WRITING PARTIAL SECTOR ?
62 020026 001002                                     ;BR IF NO, ELSE,
63 020030 012705 000400                                     ;WRITE AT LEAST ONE SECTOR
64 020034 010560 000020 4$:  MOV  R5,$WRDL(R0) ;WORD COUNT
65
66                                     ;GET A RANDOM PATTERN NUMBER
67
68 020040 122760 000002 000114 RANPAT: CMPB  #2,$NCODE(R0) ;WRITE OPERATION ?
69 020046 001004                                     ;BR IF NO
70 020050 004737 020070                                     ;GET PATTERN CODE
71 020054 110560 000115                                     ;MOVE PATTERN CODE TO CONTROL BLOCK
72 020060 012760 177777 000122 RANXIT: MOV  #-1,$NEXT(R0) ;SET PARAMETERS SELECTED INDICATOR
73 020066 000207                                     ;RETURN
74
75                                     ;ROUTINE TO SELECT A PATTERN
76
77 020070 012705 000020 GETPAT: MOV  #16.,R5 ;SELECT PATTERN
78 020074 005737 001472                                     ;ENABLE RANDOM PATTERN SELECTION ?
79 020100 001403                                     ;YES
80 020102 013705 001472                                     ;USE INDEXED PATTERN
81 020106 000406                                     ;NO
82 020110 004737 037024 1$:  JSR  PC,$RAND ;CYCLE THE RANDOM NUMBER GENERATOR
83 020114 004737 032014                                     ;GET CODE
84 020120 005705                                     ;WAS PATTERN ZERO SELECTED ?
85 020122 001762                                     ;BR IF YES
86 020124 006305 2$:  ASL  R5 ;MAKE CODE INTO TABLE INDEX
87 020126 000207                                     ;
    
```

```

1      ;THIS ROUTINE IS USED TO CHECK ADDRESS LIMITS BEFORE THE NEXT COMMAND
2      ;IS PERFORMED. THIS WILL PROTECT AGAINST WRITTING ON CUSTOMER DATA BY
3      ;CHECKING FOR MINIMUM ADDRESS VALUES. ALSO, IT WILL CHECK FOR MAXIMUM
4      ;ADDRESS LIMITS TO LOOK FOR AN END TO THE SEQUENTIAL ADDRESSING.
5      ;CALL:
6      ;
7      ;MOV      #DPB,R0      ;DPB ADDRESS
8      ;MOV      #POINTER,R4  ;POINTER TO SECTOR STORAGE ($SEC OR $NSEC) IN DPB
9      ;JSR      PC,CKLMTS    ;CALL ADDRESS LIMITS ROUTINE
10     ;BR       ???          ;RETURN HERE IF NOT END OF SEQUENTIAL ADDRESSING
11     ;-----             ;ELSE, RETURN HERE TO RESET DISK ADDRESS
12
13     ;R0 = DPB ADDRESS BEFORE CALLING THE ROUTINE
14     ;R4 = POINTER TO SECTOR STORAGE BEFORE CALLING THE ROUTINE
15     CKLMTS: ADD      R0,R4      ;POINT TO SECTOR STORAGE POINT IN DPB
16     CMP      2(R4),MINCYL(R0)  ;IS CYLINDER ADDRESS BELOW MIN. ?
17     BGE      1$              ;BR IF NO
18     MOV      MINCYL(R0),2(R4)  ;RESET CYLINDER TO MIN.
19     1$: CMPB    1(R4),MINTRK(R0) ;IS TRACK ADDRESS BELOW MIN. ?
20     BGE      2$              ;BR IF NO
21     MOVB    MINTRK(R0),1(R4)  ;RESET TRACK TO MIN.
22     2$: CMPB    (R4),MINSEC(R0) ;IS SECTOR ADDRESS BELOW MIN. ?
23     BGE      3$              ;BR IF NO
24     MOVB    MINSEC(R0),(R4)   ;RESET SECTOR TO MIN.
25
26     ;LOOK FOR MAXIMUM LIMITS AND END OF SEQUENTIAL ADDRESSING
27
28     3$: CMPB    (R4),MAXSEC(R0) ;IS SECTOR ADDRESS AT MAXIMUM ?
29     BLE      5$              ;BR IF NO
30     MOVB    MINSEC(R0),(R4)   ;RESET SECTOR ADDRESS
31     4$: INCB    1(R4)         ;INCREMENT TO NEXT TRACK ADDRESS
32     5$: CMPB    1(R4),MAXTRK(R0) ;IS TRACK ADDRESS OVER MAXIMUM ?
33     BLE      6$              ;BR IF NO
34     MOVB    MINSEC(R0),(R4)   ;RESET SECTOR ADDRESS
35     MOVB    MINTRK(R0),1(R4)  ;RESET TRACK ADDRESS
36     INC      2(R4)           ;INCREMENT TO NEXT CYLINDER ADDRESS
37     6$: CMP      2(R4),MAXCYL(R0) ;IS CYLINDER ADDRESS OVER MAXIMUM ?
38     BLE      7$              ;BR IF NO
39     ADD      #2,(SP)         ;ADJUST RETURN TO RESET DISK ADDRESS PARAMETERS
40     BR       9$
41
42     7$: MOV      FE1,-(SP)     ;CHECK NOT TO READ OR WRITE BAD SECTOR TRACK
43     DEC      (SP)           ;GET FIRST FE CYLINDER (LAST CYL+1)
44     CMP      2(R4),(SP)+     ;LOOK AT LAST USER CYLINDER
45     BNE      8$              ;ARE WE ON LAST USER CYLINDER ?
46     8$: CMPB    1(R4),TRKLMT  ;IS THIS THE BAD SECTOR TRACK ?
47     BEQ      4$              ;BR IF YES
48
49     8$: CMP      2(R4),FE1     ;CHECK NOT TO READ OR WRITE SKIP SECTOR FILE TRACKS
50     BNE      9$              ;ARE WE ON 1ST FE CYLINDER ?
51     9$: CMPB    1(R4),#1     ;ARE WE ON TRACK 0 OR 1 ?
52     BLE      4$              ;BR IF YES
53     9$: RTS      PC          ;RETURN
    
```

```

1      ;THIS ROUTINE IS USED TO CALCULATE AND CHECK THE WORD COUNT FOR THE
2      ;DRIVE THAT IS TO DO A DATA TRANSFER ON THE MAXIMUM TRACK OF THE MAXIMUM
3      ;CYLINDER. IF THE CALCULATED MAXIMUM WORD COUNT, EXCEEDS THE DESIRED WORD
4      ;COUNT (CONTENTS OF R5), THEN THE DESIRED WORD COUNT IS CHANGED, SO THAT
5      ;THE WORD COUNT WILL NOT CAUSE A TRACK OVERFLOW DURING THE TRANSFER.
6      ;CALL:
7      ;
8      ;       MOV     #DPB,R0           ;DPB ADDRESS
9      ;       MOV     #POINTER,R4      ;POINTER TO SECTOR STORAGE ($SEC OR $NSEC) IN DPB
10     ;       JSR     PC,CHKWC         ;CALL CHECK WORD COUNT ROUTINE
11     ;       RETURN                    ;RETURN WITH R5 CONTAINING THE DESIRED WORD COUNT
12
13     ;R0 = DPB ADDRESS BEFORE CALLING THE ROUTINE
14     ;R4 = POINTER TO SECTOR STORAGE BEFORE CALLING THE ROUTINE
15     ;R5 = DESIRED WORD COUNT BEFORE CALLING THE ROUTINE
16
17     CHKWC:  ADD     R0,R4             ;POINT TO SECTOR STORAGE POINT IN DPB
18             TSTB   MINSEC(R0)       ;ALLOW SPIRAL RD/WRT ?
19             BNE    2$                ;BR IF NO
20             CMPB   MAXSEC(R0),SECLMT ;ALLOW SPIRAL RD/WRT ?
21             BNE    2$                ;BR IF NO
22             TSTB   MINTRK(R0)       ;ALLOW SPIRAL RD/WRT ?
23             BNE    1$                ;BR IF NO
24             CMPB   MAXTRK(R0),TRKLMT ;ALLOW SPIRAL RD/WRT ?
25             BNE    1$                ;BR IF NO
26             ;WHEN SPIRAL RD/WRT IS ALLOWED, THEN CHECK
27             ;TO MAKE SURE YOU DO NOT SPIRAL OVER MAXIMUM
28             ;TRACK ON MAXIMUM CYLINDER
29             CMP    MAXCYL(R0),2(R4)  ;ON MAXIMUM CYLINDER ?
30             BNE    4$                ;BR IF NO
31             CMPB   MAXTRK(R0),1(R4)  ;ON MAXIMUM TRACK ?
32             BNE    4$                ;BR IF NO
33
34             2$:  MOVB   (R4),R4       ;GET STARTING SECTOR ADDRESS
35             MOV    MAXSEC(R0),-(SP)  ;GET MAXIMUM SECTOR
36             SUB    R4,(SP)          ;GET NUMBER SECTORS TO BE XFERD
37             CLR    R4               ;CLEAR R4
38             3$:  ADD    #256.,R4     ;ADD 1 SECTOR OF WORDS TO R4
39             DEC    (SP)             ;DONE ALL SECTORS YET ?
40             BGE    3$               ;BR IF NO
41             TST   (SP)+            ;RESTORE STACK
42             CMP   R5,R4            ;TOO MANY WORDS FOR TRACK ?
43             BLE   5$               ;BR IF NO
44             MOV   R4,R5            ;YES, CHANGE WORD COUNT
45             BR    5$
46
47             4$:  MOV    FE1,-(SP)    ;GET FIRST FE CYLINDER (LAST CYL+1)
48             DEC   (SP)             ;LOOK AT LAST USER CYLINDER
49             CMP   2(R4),(SP)+      ;ARE WE ON LAST USER CYLINDER ?
50             BNE   5$               ;BR IF NO
51             MOV   TRKLMT,-(SP)    ;GET LAST TRACK
52             DEC   (SP)             ;LOOK AT NEXT TO LAST TRACK
53             CLR   -(SP)            ;PUSH STACK
54             MOVB  1(R4),(SP)       ;GET CURRENT TRACK
55             CMP   (SP)+,(SP)+     ;IS IT TRACK BEFORE BAD SECTOR TRACK ?
56             BEQ   2$               ;BR IF YES (DON'T ALLOW SPIRAL TO BAD SEC TRK)
57             RTS    PC
    
```

```

1      ;ROUTINE TO GET THE PREVIOUSLY SELECTED PARAMETER VALUES
2      ;CALL:
3      :
4      :       MOV     #DPB,R0           ;DPB ADDRESS
5      :       JSR     PC,GENPAR        ;GENERATE THE PARAMETERS
6      :       JSR     PC,LODPAR       ;LOAD THE PARAMETERS JUST GENERATED
7      :       RETURN
8 020500 010546      LODPAR: MOV     R5,-(SP)           ;SAVE R5
9 020502 105760 000026  TSTB    $PACK(R0)           ;'R' OR 'W' COMMAND FOR THE DRIVE ?
10 020506 001106      BNE     4$                ;BR IF YES
11 020510 116060 002140 000027  MOVB   $RMCS1(R0),$PREVO(R0)       ;SAVE CURRENT PARAMETERS
12 020516 142760 177701 000027  BICB   #^C76,$PREVO(R0)           ;STRIP GO,AND IE BITS
13 020524 132760 000006 000114  BITB   #6,$NCODE(R0)              ;SEE IF NEXT OPERATION IS READ OR WRITE
14 020532 001007      BNE     1$                ;BR IF EITHER
15 020534 016060 000012 000034  MOV    $CYL(R0),$PREVA(R0)         ;SAVE STARTING CYLINDER
16 020542 016060 000010 000032  MOV    $SEC(R0),$PREVA(R0)         ;SAVE STARTING SECTOR AND TRACK
17 020550 000410      BR      2$
18 020552 004737 023254      1$: JSR    PC,READDR           ;GET THE DECREMENTED SECTOR AND TRACK ADDRESSES
19 020556 012660 000034      MOV    (SP)+,$PREVA+2(R0)          ;CYLINDER ADDRESS
20 020562 112660 000033      MOVB   (SP)+,$PREVA+1(R0)          ;TRACK ADDRESS
21 020566 112660 000032      MOVB   (SP)+,$PREVA(R0)           ;SECTOR ADDRESS
22
23 020572 032777 000100 160354  2$: BIT    #SW06,@SWR              ;SWITCH 6 SET ?
24 020600 001051      BNE     4$                ;BR IF SET
25 020602 116060 000114 000024  MOVB   $NCODE(R0),$CODE(R0)        ;LOGICAL CODE FOR OPERATION
26 020610 116005 000114      MOVB   $NCODE(R0),R5             ;LOAD R5 FOR USE AS TABLE INDEX
27 020614 116560 002076 000002  MOVB   COMTBL(R5),$COMND(R0)        ;COMMAND CODE
28 020622 122760 000151 000002  CMPB   #WCKD,$COMND(R0)           ;IS NEW COMMAND A WRITE CHECK DATA ?
29 020630 001012      BNE     3$                ;BR IF NO
30 020632 122760 000060 000027  CMPB   #60,$PREVO(R0)             ;WAS PREVIOUS COMMAND A WRITE DATA ?
31 020640 001431      BEQ    4$                ;BR IF YES
32 020642 112760 000171 000002  MOVB   #RDDAT,$COMND(R0)          ;CHANGE WRITE CHECK TO READ DATA COMMAND
33 020650 112760 000004 000024  MOVB   #4,$CODE(R0)              ;CODE NUMBER CHANGED TO READ DATA
34
35 020656 116060 000115 000030  3$: MOVB   $NPATC(R0),$PATT(C(R0)      ;PATTERN CODE
36 020664 016060 000116 000010  MOV    $NSEC(R0),$SEC(R0)          ;TRACK AND SECTOR ADDRESSES
37 020672 016060 000120 000012  MOV    $NCYL(R0),$CYL(R0)         ;CYLINDER ADDRESS
38 020700 012760 000400 000022  MOV    #256,$SSEC(R0)             ;INITIAL VALUE OF SECTOR SIZE
39 020706 132760 000001 000024  BITB   #1,$CODE(R0)              ;HEADER OPERATION ?
40 020714 001403      BEQ    4$                ;BR IF NOT
41 020716 062760 000002 000022  ADD    #2,$SSEC(R0)              ;ADD HEADER SIZE
42 020724 016060 000020 000004  4$: MOV    $WRDL(R0),$WCNT(R0)       ;GET WORD COUNT AND
43 020732 016060 000020 000110  MOV    $WRDL(R0),$HLDWC(R0)       ;HOLD WORD COUNT FOR 'RELBUF' ROUTINE
44 020740 005460 000004      NEG    $WCNT(R0)                ;MAKE IT 2'S COMPLEMENT
45 020744 012605      MOV    (SP)+,R5                 ;RESTORE R5
46 020746 000207      RTS     PC                     ;RETURN
    
```

```

1      ;ROUTINE TO COMPRESS A LIST
2      :CALL:
3      :      MOV      #ADDRS,R1      ;COMPRESS LIST STARTING AT THIS ADDRESS
4      :      JSR      PC,CMPRES
5      :      RETURN
6
7 020750 016111 000002      CMPRES: MOV      2(R1),(R1)      ;COMPRESS THE TABLE IN R1
8 020754 001402              BEQ      1$              ;BR WHEN ZERO FOUND
9 020756 005721              TST      (R1)+          ;INCREMENT R1
10 020760 000773             BR       CMPRES         ;CONTINUE COMPRESSING TABLE
11 020762 000207             1$:      RTS       PC          ;RETURN
12
13      ;ROUTINE TO DETERMINE OF ERROR IS AT A LOCATION ON THE DISK DEFINED
14      ;IN THE BAD TRACK/SECTOR TABLE FOR THE DRIVE.
15      :CALL:
16      :      JSR      PC,SPOTCK
17      :      RETURN1
18      :      RETURN2
19      :
20
21      SPOTCK:
22 020764 010146              MOV      R1,-(SP)        ;:PUSH R1 ON STACK
23 020766 012701 000146      MOV      #$BDSEC,R1    ;:INCREMENT FOR BAD SECTOR TABLE
24 020772 060001              ADD      R0,R1          ;:ADD THE BLOCK'S STARTING ADDRESS
25 020774 005037 001436      1$:      CLR      DEC2        ;:ASSUME DECREMENT SECTOR ONCE
26 021000 005760 000112      TST      $$SENB(R0)    ;:DID ERROR OCCUR DURING SKIP SECTORING ?
27 021004 001402              BEQ      2$              ;:BR IF NO
28 021006 005237 001436      INC      DEC2          ;:DECREMENT SECTOR TWICE
29 021012 004737 023254      2$:      JSR      PC,READDR    ;:DECREMENT THE SECTOR/TRACK ADDRESS
30 021016 021126              CMP      (R1),(SP)+    ;:ON THE SAME CYLINDER ?
31 021020 001023              BNE      6$              ;:BRANCH IF NOT
32 021022 122761 177777 000003  CMPB     #-1,3(R1)     ;:ALL BAD TRACKS ?
33 021030 001002              BNE      3$              ;:BR IF NO
34 021032 005726              TST      (SP)+          ;:ADJUST STACK AND
35 021034 000403              BR       4$              ;:GO CHECK SECTORS
36 021036 122661 000003      3$:      CMPB     (SP)+,3(R1) ;:COMPARE THE TRACK ADDRESS
37 021042 001013              BNE      7$              ;:BR IF IT IS NOT EQUAL
38 021044 122761 177777 000002  4$:      CMPB     #-1,2(R1)     ;:ALL BAD SECTORS ?
39 021052 001002              BNE      5$              ;:BR IF NO
40 021054 005726              TST      (SP)+          ;:ADJUST STACK AND
41 021056 000413              BR       9$              ;:CHECK 'MESSAGE'
42 021060 122661 000002      5$:      CMPB     (SP)+,2(R1)   ;:COMPARE THE SECTOR ADDRESS
43 021064 001003              BNE      8$              ;:BR IF NOT EQUAL
44 021066 000407              BR       9$              ;:CHECK 'MESSAGE'
45 021070 005726              TST      (SP)+          ;:CLEAR OFF THE STACK
46 021072 005726              TST      (SP)+          ;:INCREMENT THE STACK POINTER
47 021074 062701 000004      8$:      ADD      #4,R1          ;:GO TO THE NEXT LOCATION IN THE TABLE
48 021100 005711              TST      (R1)           ;:EMPTY ENTRY OR TERMINATOR ?
49 021102 100407              BMI     10$             ;:BR IF YES
50 021104 000733              BR       1$              ;:TRY NEXT SECTOR
51 021106 005737 001504      9$:      TST      MESSAGE      ;:PRINT THE ERROR ANYWAY ?
52 021112 001006              BNE     11$             ;:BR IF NOT
53 021114 012737 177777 001342  MOV      #-1,BADSEC    ;:SET THE INDICATOR FOR THE IDENTIFICATION LINE
54 021122 062766 000002 000002 10$:     ADD      #2,2(SP)       ;:INCREMENT THE RETURN
55 021130 012601              11$:     MOV      (SP)+,R1      ;:POP STACK INTO R1
56 021132 000207              RTS       PC            ;:RETURN
    
```

```

1          ::*****
2          ::SBTTL  ERROR MESSAGE GENERATION ROUTINES
3          ::*****
4
5          :PRINT LINE 1 OF ERROR MESSAGE:
6          :'HH:MM:SS'
7
8 021134 032777 002000 160012 LINE1: BIT      #SW10,@SWR      :SWITCH 10 SET ?
9 021142 001402                BEQ      1$          :BR IF NOT
10 021144 104401 001176        TYPE     ,SBELL        :RING THE BELL
11 021150 032777 020000 157776 1$: BIT      #SW13,@SWR      :INHIBIT TYPEOUT ?
12 021156 001405                BEQ      2$          :BR IF NOT
13 021160 104414 001203        DISPLY  ,SCRLF       :CR-LF
14 021164 104414 001203        DISPLY  ,SCRLF       :CR-LF
15 021170 000410                BR       3$          :EXIT
16 021172 104414 001203        2$: DISPLY ,SCRLF       :CR-LF
17 021176 104414 001203        DISPLY ,SCRLF       :CR-LF
18 021202 004737 024764        JSR     PC,$TIME    :TYPE THE TIME
19 021206 104414 075234        DISPLY ,BLNKS1     :TYPE 1 BLANK
20 021212 000207                3$: RTS      PC      :RETURN & TYPE DESCRIPTION
21
22          :PRINT LINE 2 OF ERROR MESSAGE
23          :'PRCNT COMMAND = XXXX  PREV COMMAND = XXXX'
24          :'* ERROR AT BAD TRACK/SECTOR'
25          :'DRV RMCS1  RMCS2  RMDS1  RMER1  RMMR2  RMER2  RMEC1  RMEC2'
26          :'RMWC  RMBA  RMDA  RMAS  RMLA  RMDB  RMMR1  RMDT'
27          :'RMSN  RMOF  RMDC  RMCC  STATUS'
28          :'RMBAE  RMCS3' (RH70 ONLY)
29          :'BUS ADDRESS OR WORD COUNT NOT CONSISTENT'
30          :'RMBA = XXXXXX  RMWC = XXXXXX'
31          :'BUFFER ADR = XXXXXX  WRD CNT = XXXX  ACTUAL NMBR WRDS XFRD = XXX'
32
33          LINE2:
34
35 021214 010346                MOV     R3,-(SP)    ;;PUSH R3 ON STACK
36 021216 010446                MOV     R4,-(SP)    ;;PUSH R4 ON STACK
37 021220 010546                MOV     R5,-(SP)    ;;PUSH R5 ON STACK
38 021222 101414 001203        DISPLY ,SCRLF       :CR-LF
39 021226 005037 021354        CLR     4$          :CLEAR MESSAGE ADDRESS STORAGE
40 021232 005004                CLR     R4          :WORKING REGISTER
41 021234 012737 073272 021354  MOV     #LIN2C,4$   :ADDRESS OF 'PRCNT COMMAND = ' MSG
42 021242 116004 002140        MOVB   $RMCS1(R0),R4 :GET THE OPCODE
43 021246 042704 177701        BIC    #^C76,R4    :SAVE ONLY SIGNIFICANT BITS
44 021252 004737 021310        JSR    PC,1$       :TYPE THE FIRST MNEMONIC
45 021256 005737 021360        TST    5$          :SEE IF MNEMONIC ENTRY FOUND
46 021262 001440                BEQ    LINE2A      :BR IF NOT
47 021264 012737 073312 021354  MOV     #LIN2P,4$   :ADDRESS OF 'PREVS COMMAND = ' MSG
48 021272 116004 000027        MOVB   $PREVO(R0),R4 :PREVIOUS OPERATION CODE
49 021276 042704 177701        BIC    #^C76,R4    :SAVE ONLY SIGNIFICANT BITS
50 021302 004737 021310        JSR    PC,1$       :TYPE THE PREVIOUS MNEMONIC
51 021306 000426                BR     LINE2A      :CONTINUE
52 021310 005005                1$: CLR     R5       :CLEAR THE TABLE INDEX
53 021312 126504 002104        2$: CMPB  OPTBL(R5),R4 :LOOK FOR THE OPCODE
54 021316 001405                BEQ    3$          :BR WHEN OPCODE COUNT EQUALS OPCODE
55 021320 105765 002104        TSTB  OPTBL(R5)    :LOOK FOR END OF TABLE
56 021324 100402                BMI    3$          :BR IF END
57 021326 005205                INC    R5          :INCREMENT THE POINTER
58 021330 000770                BR     2$          :CONTINUE - NOT END OF TABLE
    
```


57	021332	006305			3\$:	ASL	R5	:SHIFT INDEX
58	021334	006305				ASL	R5	:SHIFT THE INDEX
59	021336	006305				ASL	R5	:SHIFT THE INDEX
60	021340	012737	002124	021360		MOV	#MNTBL,5\$:ADDRESS OF ASCII TEXT TABLE
61	021346	060537	021360			ADD	R5,5\$:ADD THE INDEX
62	021352	104414				DISPLY		:TYPE IT
63	021354	000000			4\$:	.WORD	0	:ADDRESS OF 'PRESENT' OR 'PREVIOUS' MESSAGE
64	021356	104414				DISPLY		:TYPE THE OPERATION MNEMONIC
65	021360	000000			5\$:	.WORD	0	:ADDRESS OF MESSAGE
66	021362	000207				RTS	PC	:RETURN TO MAIN ROUTINE
67								
68	021364	005737	001342		LINE2A:	TST	BADSEC	:PRINT THE BAD SECTOR LINE ?
69	021370	001404				BEQ	LINE2B	:BR IF NOT
70	021372	104414	001203			DISPLY	,\$CRLF	:CR-LF
71	021376	104414	073333		LINE2B:	DISPLY	,\$LIN2S	:ERROR ADDRESS DEFINED AS BAD AREA
72	021402	104414	001203			DISPLY	,\$CRLF	:CR-LF
73	021406	104414	072644			DISPLY	,\$DH14	:STANDARD RM REGISTER HEADER
74	021412	104414	075234			DISPLY	,\$BLNKS1	:TYPE 1 BLANK
75	021416	013746	001324			MOV	DRVNO,-(SP)	:PUT THE DRIVE NUMBER ON THE STACK
76	021422	004737	023234			JSR	PC,LINDEC	:TYPE DRIVE NUMBER
77	021426	104414	075233			DISPLY	,\$BLNKS2	:TYPE 2 BLANKS
78	021432	012705	073204			MOV	#DT14,R5	:REGISTER INDEXES
79	021436	004737	021620			JSR	PC,3\$:PRINT THE REGISTERS
80	021442	032777	000040	157504		BIT	#SW05,\$SWR	:PRINT THE OPTIONAL REGISTERS ?
81	021450	001031				BNE	1\$:BR IF NOT
82	021452	104414	072746			DISPLY	,\$DH15	
83	021456	012705	073226			MOV	#DT15,R5	:SECOND DATA LINE
84	021462	004737	021620			JSR	PC,3\$:PRINT THEM
85	021466	104414	073044			DISPLY	,\$DH16	
86	021472	012705	073250			MOV	#DT16,R5	:THIRD DATA LINE
87	021476	004737	021620			JSR	PC,3\$:PRINT THE REGISTERS
89	021502	013746	001234			MOV	,\$CPUOP,-(SP)	:CHECK THE CPU (RM) TYPE
90	021506	042716	003777			BIC	#^C174000,(SP)	:LEAVE THE CPU BITS
91	021512	022726	030000			CMP	#30000,(SP)+	:SEE IF RH70
92	021516	001006				BNE	1\$:BR IF NO
93	021520	104414	073114			DISPLY	,\$DH17	
94	021524	012705	073264			MOV	#DT17,R5	:OPTIONAL FOURTH DATA LINE
95	021530	004737	021620			JSR	PC,3\$:PRINT THE REGISTERS
97	021534	032760	000100	000016	1\$:	BIT	#BIT6,\$STATUS(R0)	:DATA ERROR ?
98	021542	001422				BEQ	2\$:BR IF NOT
99	021544	016046	000020			MOV	,\$WRDL(R0),-(SP)	:TRANSFER WRD CNT
100	021550	066016	002142			ADD	,\$RMWC(R0),(SP)	:ADD REMAINING WORD COUNT
101	021554	006310				ASL	(SP)	:CONVERT TO AN BYTE INCREMENT
102	021556	066016	000006			ADD	,\$BUF(R0),(SP)	:BUFFER STARTING ADDRESS
103	021562	022660	002144			CMP	(SP)+,\$RMBA(R0)	:CORRECT BUFFER ADDRESS ?
104	021566	001410				BEQ	2\$:BR IF YES
105	021570	104414	072150			DISPLY	,\$EM46	: 'BUS ADDRESS AND WORD COUNT ARE NOT CONSISTENT'
106	021574	104414	001203			DISPLY	,\$CRLF	:CR-LF
107	021600	004737	021712			JSR	PC,LINE3D	:PRINT LINE 3D OF ERROR MESSAGE
108	021604	004737	022324			JSR	PC,LINE4	:PRINT LINE 4 OF ERROR MESSAGE
109	021610				2\$:			
	021610	012605				MOV	(SP)+,R5	::POP STACK INTO R5
	021612	012604				MOV	(SP)+,R4	::POP STACK INTO R4
	021614	012603				MOV	(SP)+,R3	::POP STACK INTO R3
110	021616	000207				RTS	PC	:RETURN TO ERROR PROCESSING ROUTINE
111	021620	012546			3\$:	MOV	(R5)+,-(SP)	:PUT THE REGISTER INDEX ON THE STACK
112	021622	060016				ADD	R0,(SP)	:ADD DRIVE'S TABLE ADDRESS

```

113 021624 017646 000000      MOV      @ (SP), -(SP)      ;VALUE
114 021630 004737 023202      JSR      PC,LIN0CT        ;TYPE IT
115 021634 005726              TST      (SP)+            ;CORRECT THE STACK POINTER
116 021636 104414 075233      DISPLY   ,BLNKS2         ;TYPE 2 BLANKS
117 021642 005715              TST      (R5)            ;AT END OF LINE ?
118 021644 001365              BNE      3$              ;BR IF NOT
119 021646 104414 001203      4$:     DISPLY   , $CRLF   ;CR-LF
120 021652 000207              RTS      PC              ;RETURN
121
122      ;PRINT LINE 3 OF ERROR MESSAGE
123      ;'ERROR AT CCC TT SS  PREV ADR = CCC TT SS'
124
125 021654 104414 073367      LINE3:  DISPLY   ,LINM3   ;LINE 3 ENTRANCE
126 021660 000517              BR       LIN3.1         ;FINISH PRINTOUT
127
128      ;PRINT LINE 3A OF ERROR MESSAGE
129      ;'START CYL = CCC  END CYL = CCC'
130
131 021662 104414 073405      LINE3A: DISPLY   ,LINM3   ;LINE 3A ENTRANCE
132 021666 000514              BR       LIN3.1         ;FINISH ERROR LINE
133
134      ;PRINT LINE 3B OF ERROR MESSAGE
135      ;'START CYL = CCC  END CYL = CCC  ACTUAL CYL = CCC'
136
137 021670 004737 022226      LINE3B: JSR      PC,LIN3.3 ;LINE 3B ENTRANCE
138 021674 104414 001203      DISPLY   , $CRLF   ;
139 021700 000207              RTS      PC
140
141      ;PRINT LINE 3C OF ERROR MESSAGE
142      ;'START CYL = CCC  END CYL = CCC  ACTUAL CYL = CCC  TRK = TT'
143
144 021702 004737 022226      LINE3C: JSR      PC,LIN3.3 ;LINE 3C ENTRANCE
145 021706 000137 022260      JMP      LIN3.4         ;FINISH MESSAGE
146
147      ;PRINT LINE 3D OF ERROR MESSAGE
148      ;'RMBA = XXXXXX  RMWC = XXXXXX'
149
150 021712 032777 000040 157234 LINE3D: BIT      #SW05,@SWR ;SWITCH 5 SET ?
151 021720 001416              BEQ      1$            ;BR IF IT IS
152 021722 104414 073544      DISPLY   ,LINB3        ;'RMBA = '
153 021726 016046 002144      MOV      $RMBA(RO),-(SP) ;BUFFER ADDR REG CONTENTS
154 021732 004737 023202      JSR      PC,LIN0CT        ;CONVERT TO OCTAL AND TYPE IT
155 021736 104414 073553      DISPLY   ,LINW3        ;' RMWC = '
156 021742 016046 002142      MOV      $RMWC(RO),-(SP) ;WORD COUNT REGISTER CONTENTS
157 021746 004737 023202      JSR      PC,LIN0CT        ;CONVERT TO OCTAL AND TYPE IT
158 021752 104414 001203      DISPLY   , $CRLF   ;
159 021756 000207              1$:     RTS      PC
160
161      ;PRINT LINE 3E OF ERROR MESSAGE
162      ;'START CYL = CCC  START TRK = TT  START SEC = SS'
163
164 021760 104414 073445      LINE3E: DISPLY   ,LINS3   ;'START CYL = '
165 021764 016046 000012      MOV      $CYL(RO),-(SP) ;MOVE CYL TO STACK
166 021770 004737 023234      JSR      PC,LINDEC      ;TYPE IT IN DECIMAL
167 021774 104414 075233      DISPLY   ,BLNKS2         ;TYPE 2 BLANKS
168 022000 104414 073564      DISPLY   ,LINST3        ;'START TRK = '
169 022004 005046              CLR      -(SP)         ;CLEAR STACK
    
```

```

170 022006 116016 000011      MOVB   $TRK(RO), (SP)      :TRACK TO STACK
171 022012 004737 023234      JSR    PC,LINDEC         :TYPE IT IN DECIMAL
172 022016 104414 075233      DISPLY ,BLNKS2           :TYPE 2 BLANKS
173 022022 104414 073600      DISPLY ,LINSS3          :'START SEC = '
174 022026 005046             CLR    -(SP)             :CLEAR STACK
175 022030 116016 000010      MOVB   $SEC(RO), (SP)    :SECTOR ADDR TO STACK
176 022034 004737 023234      JSR    PC,LINDEC         :TYPE IT IN DECIMAL
177 022040 104414 001203      DISPLY ,$CRLF           :
178 022044 000207             RTS    PC
179
180                          :PRINT LINE 3F OF ERROR MESSAGE
181                          :'RMDA = XXXXXX  RMCA = XXXXXX'
182
183 022046 032777 000040 157100 LINE3F: BIT   #SWS,@SWR    :SWITCH 5 SET ?
184 022054 001420             BEQ    1$                :BR IF NOT
185 022056 104414 073535      DISPLY ,LINDA3          :'RMDA = '
186 022062 016046 002146      MOV    $RMDA(RO),-(SP)  :PUT SECTOR/TRACK ADDRESS ON THE STACK
187 022066 004737 023202      JSR    PC,LINOC         :TYPE IT
188 022072 104414 075233      DISPLY ,BLNKS2         :TYPE 2 BLANKS
189 022076 104414 073524      DISPLY ,LINCA3         :' RMDC = '
190 022102 016046 002174      MOV    $RMDC(RO),-(SP) :PUT DESIRED CYLINDER ADDRESS ON THE STACK
191 022106 004737 023202      JSR    PC,LINOC         :TYPE IT
192 022112 104414 001203      DISPLY , $CRLF
193 022116 000207             RTS    PC
194
195                          :'CCC TT SS  PREV ADR = CCC TT SS'
196
200 022120 004737 023254      LIN3.1: JSR   PC,READDR   :DECREMENT TRACK AND SECTOR ADDRESS
201 022124 004737 023234      JSR    PC,LINDEC         :TYPE IT IN DECIMAL
202 022130 104414 073402      DISPLY ,T                :PRINT ' T'
206 022134 004737 023234      JSR    PC,LINDEC         :TYPE TRACK IN DECIMAL
207 022140 104414 073423      DISPLY ,S                :PRINT ' S'
208 022144 004737 023234      JSR    PC,LINDEC         :TYPE SECTOR ADDRESS
209 022150 104414 073426      DISPLY ,LINP3           :PRINT 'PREV ADDR'
210 022154 016046 000034      MOV    $PREVA+2(RO),-(SP) :PREVIOUS CYLINDER
211 022160 004737 023234      JSR    PC,LINDEC         :TYPE IT IN DECIMAL
212 022164 104414 073402      DISPLY ,T                :PRINT ' T'
213 022170 005046             CLR    -(SP)             :MAKE ROOM ON THE STACK
214 022172 116016 000033      MOVB   $PREVA+1(RO), (SP) :PREVIOUS TRACK ADDRESS
215 022176 004737 023234      JSR    PC,LINDEC         :TYPE IT IN DECIMAL
216 022202 104414 073423      DISPLY ,S                :PRINT ' S'
217 022206 005046             CLR    -(SP)             :MAKE ROOM ON THE STACK
218 022210 116016 000032      MOVB   $PREVA(RO), (SP)  :PREVIOUS SECTOR DDRESS
219 022214 004737 023234      JSR    PC,LINDEC         :TYPE IT IN DECIMAL
220 022220 104414 001203      DISPLY , $CRLF
221 022224 000207             RTS    PC
222
223                          :'START CYL = CCC  END CYL = CCC'
224
225 022226 104414 073445      LIN3.3: DISPLY ,LINS3     :LINE '3B & 3C' ENTRANCE
226 022232 016046 000034      MOV    $PREVA+2(RO),-(SP) :PREVIOUS CYLINDER
227 022236 004737 023234      JSR    PC,LINDEC         :TYPE IT IN DECIMAL
228 022242 104414 073461      DISPLY ,LINEN3         :PRINT 'END CYL'
229 022246 016046 002174      MOV    $RMDC(RO),-(SP)  :PRESENT CYLINDER
230 022252 004737 023234      JSR    PC,LINDEC         :TYPE IT IN DECIMAL
231 022256 000207             RTS    PC
232
    
```

```

233          ;'ACTUAL CYL = CCC   TRK = TT'
234
235 022260 104414 073475          LIN3.4: DISPLY  ,LINA3          ;PRINT 'ACTUAL'
236 022264 013746 101174          MOV          CYLNDR,-(SP)      ;ACTUAL CYLINDER
237 022270 042716 010000          BIC          #BIT12,(SP)     ;CLEAR THE FORMAT BIT
238 022274 004737 023234          JSR          PC,LINDEC       ;TYPE IT IN DECIMAL
239 022300 104414 073514          DISPLY      ,LINT3          ;PRINT TRACK
240 022304 005046                    CLR          -(SP)           ;CLEAR STACK WORD
241 022306 116016 002147          MOVB        $RMDA+1(R0),(SP) ;PUT TRACK ON STACK
242 022312 004737 023234          JSR          PC,LINDEC       ;TYPE IT IN DECIMAL
243 022316 104414 001203          DISPLY      ,%CRLF
244 022322 000207          RTS          PC
245
246          ;PRINT LINE 4 OF ERROR MESSAGE
247          ;'BUFFER ADR = XXXXXX   WRD CNT = XXXX   ACTUAL NMBR WRDS XFRD = XXX'
248
249 022324 032760 000100 000016 LINE4:  BIT          #BIT06,$STATUS(R0) ;DATA ERROR ?
250 022332 001427                    BEQ          1$              ;BR IF NOT
251 022334 104414 073614          DISPLY      ,LINM4          ;'PRINT BUFFER'
252 022340 016046 000006          MOV          $BUF(R0),-(SP) ;BUFFER ADDR ON STACK
253 022344 004737 023202          JSR          PC,LINOC7       ;CONVERT TO OCTAL & PRINT
254 022350 104414 073632          DISPLY      ,LINS4          ;PRINT 'WRD CNT'
255 022354 016046 000020          MOV          $WRDL(R0),-(SP) ;WORD LENGTH SIZE(WORD COUNT)
256 022360 004737 023234          JSR          PC,LINDEC       ;TYPE IT IN DECIMAL
257 022364 104414 073646          DISPLY      ,LINX4          ;'ACTUAL NMBR WRDS XFRD = '
258 022370 016046 002144          MOV          $RMB A(R0),-(SP) ;VALUE IN BUFFER ADDR REGISTER
259 022374 166016 000006          SUB          $BUF(R0),(SP)   ;SUBTRACT STARTING ADDRESS
260 022400 006216                    ASR          (SP)           ;CONVERT INTO A WORD COUNT
261 022402 004737 023234          JSR          PC,LINDEC       ;TYPE IT IN DECIMAL
262 022406 104414 001203          DISPLY      ,%CRLF
263 022412 000207          1$:          RTS          PC          ;RETURN
264
265          ;PRINT LINE 5 OF ERROR MESSAGE
266          ;'EXPCTD DATA = XXXXXX   RECEVD DATA = XXXXXX   WORD POS = XXX'
267
268 022414 104414 073700          LINES:  DISPLY      ,LIND5          ;PRINT 'EXPCTD DATA'
269 022420 162760 000002 002144          SUB          #2,$RMB A(R0)    ;BACK THE ADDRESS UP
270 022426 013746 001234          MOV          $CPUOP,-(SP)     ;CHECK THE CPU (RH) TYPE
271 022432 042716 003777          BIC          #*C17400,(SP)   ;LEAVE THE CPU BITS
272 022436 022726 030000          CMP          #30000,(SP)+    ;SEE IF RH70
273 022442 001012                    BNE          1$              ;BR IF NO
274 022444 162760 000004 002144          SUB          #4,$RMB A(R0)    ;BACKUP THE BUFFER POINTER
275 022452 032760 004000 002212          BIT          #BIT11,$RMC S3(R0) ;SEE WHICH WORD HALF DIDN'T COMPARE
276 022460 001403                    BEQ          1$              ;IF EQ, EVEN HALF DIDN'T COMPARE
277 022462 162760 000002 002144          SUB          #2,$RMB A(R0)    ;BACKUP THE BUFFER POINTER AGAIN
278 022470 017046 002144          1$:          MOV          @ $RMB A(R0),-(SP) ;'EXPCTD' DATA - AT THE BUFFER LOCATION
279 022474 004737 023202          JSR          PC,LINOC7       ;TYPE IT
280 022500 104414 073716          DISPLY      ,LINB5          ;PRINT 'RECEVD DATA'
281 022504 016046 002162          MOV          $RMDB(R0),-(SP) ;RECEVD DATA FROM BUFFER
282 022510 004737 023202          JSR          PC,LINOC7       ;TYPE IT
283 022514 016046 002142          MOV          $RMC W(R0),-(SP) ;WORD LENGTH ON STACK
284 022520 066016 000020          ADD          $WRDL(R0),(SP)  ;MAKE INTO A POSITIVE NUMBER
285 022524 005046                    CLR          -(SP)           ;UPPER DIVIDEND TO ZERO
286 022526 016046 000022          MOV          $SSEC(R0),-(SP) ;SECTOR SIZE ON THE STACK
287 022532 004737 032040          JSR          PC,$DIV         ;DIVIDE WORDS XFERED BY SECTOR SIZE
288 022536 012616                    MOV          (SP)+,(SP)      ;MOVE REMAINDER UP THE STACK
289 022540 104414 073736          DISPLY      ,LINP5          ;PRINT 'WORD POS'
    
```

```
294 022544 004737 023234      JSR    PC,LINDEC      ;TYPE THE POSITION
295 022550 104414 001203      DISPLY $CRLF
296 022554 000207      RTS    PC
297
298      ;PRINT LINE 5A OF THE ERROR MESSAGE
299      ;'HEADER FROM ERROR SECTOR XXXXXX XXXXXX XXXXXX XXXXXX'
300
301 022556
302 022556 104414 073753      LINE5A:
303 022562 013746 101174      2$:  DISPLY ,LINS5      ;'HEADER CONTENTS OF ERROR SECTOR'
      MOV    (CYLNDR,-(SP)) ;HEADER POSITION
      JSR    PC,LINOC      ;TYPE IT
      DISPLY ,BLNKS2      ;TYPE 2 BLANKS
      MOV    (CYLNDR+2,-(SP)) ;HEADER POSITION +2
      JSR    PC,LINOC      ;TYPE IT
      DISPLY ,BLNKS2      ;TYPE 2 BLANKS
308 022612 104414 075236      DISPLY ,LINX5      ;APPENDING INFO 1/23/77
309 022616 104414 001203      3$:  DISPLY $CRLF
310 022622 000207      RTS    PC
311
312      ;PRINT LINE 5B OF ERROR MESSAGE
313      ;'RMEC1 = XXXXXX RMEC2 = XXXXXX'
314
315 022624 104414 074006      LINE5B: DISPLY ,LINEP5      ;'RMEC1 = '
316 022630 016046 002204      MOV    $RMEC1(RO),-(SP) ;PUT REGISTER CONTENTS ON THE STACK
317 022634 004737 023202      JSR    PC,LINOC      ;TYPE IT
318 022640 104414 075233      DISPLY ,BLNKS2      ;TYPE 2 BLANKS
319 022644 104414 074016      DISPLY ,LINEO5      ;' RMEC2 = '
320 022650 016046 002206      MOV    $RMEC2(RO),-(SP) ;PUT REGISTER CONTENTS ON THE STACK
321 022654 004737 023202      JSR    PC,LINOC      ;TYPE IT
322 022660 104414 001203      DISPLY $CRLF
323 022664 000207      RTS    PC      ;RETURN
324
325      ;PRINT LINE 6 OF ERROR MESSAGE
326      ;'SECTOR IS ECC CORRECTABLE'
327
328 022666 104414 074030      LINE6: DISPLY ,LINB6      ;ECC CORRECTABLE
329 022672 104414 001203      DISPLY $CRLF
330 022676 000207      RTS    PC
331
332
333
334
335
336
337
338
339
340      ;PRINT LINE 6A OF THE ERROR MESSAGE
341      ;'SECTOR READ CORRECTLY'
342
343 022700 104414 074063      LINE6A: DISPLY ,LINC6      ;PRINT 'SECTOR READ CORRECTLY ON N RETRIES'
344 022704 000406      BR    LIN6.2      ;TYPE THE REST OF THE LINE
345
346
347
348
349
350
351
352
353
354      ;PRINT LINE 6C OF THE ERROR MESSAGE
355      ;'CORRECTED ON NTH RETRY'
356
357 022706 104414 074112      LINE6C: DISPLY ,LING6      ;'CORRECTED ON N RETRIES'
358 022712 000403      BR    LIN6.2      ;TYPE THE REST OF THE LINE
359
360
361      ;PRINT LINE 6D OF THE ERROR MESSAGE
362      ;'UNCORRECTABLE AFTER N RETRIES'
363 022714 104414 074142      LINE6D: DISPLY ,LINUO6      ;'UNCORRECTABLE AFTER N RETRIES'
364 022720 000400      BR    LIN6.2      ;FINISH
365
```

377
 378
 379 022722 005046
 380 022724 113716 001331
 381 022730 004737 023234
 382 022734 104414 074130
 383 022740 104414 001203
 384 022744 000207
 385
 386
 387
 388
 396 022746 104414 074254
 397 022752 016046 000072
 398 022756 004737 023234
 399 022762 104414 074272
 400 022766 016046 000056
 401 022772 004737 023234
 402 022776 104414 074302
 403 023002 012746 000060
 404 023006 060016
 405 023010 004737 037222
 406 023014 004737 032364
 407 023020 104414 074317
 408 023024 016046 000064
 409 023030 004737 023234
 410 023034 104414 074327
 411 023040 012746 000066
 412 023044 060016
 413 023046 004737 037222
 414 023052 004737 032364
 415 023056 104414 001203
 416 023062 032777 100000 156064
 417 023070 001401
 418 023072 000000
 419 023074 000207
 420
 421
 422
 423
 431 023076 104414 074214
 432 023102 012746 000052
 433 023106 060016
 434 023110 004737 037222
 435 023114 004737 032364
 436 023120 104414 074167
 437 023124 016046 000102
 438 023130 004737 023234
 439 023134 104414 074232
 440 023140 016046 000100
 441 023144 004737 023234
 442 023150 104414 001203
 443 023154 032777 100000 155772
 444 023162 001401
 445 023164 000000
 446 023166 000207
 447

```

;RETRY COUNT TYPEOUT
LINE6.2: CLR      -(SP)          ;CLEAR STACK
          MOV     RETRY+1,(SP)  ;RETRY COUNT
          JSR     PC,LINDEC     ;TYPE IT IN DECIMAL
          DISPLY ,LINR6        ;'RETRY'
          DISPLY ,%CRLF
          RTS     PC

;PRINT LINE 7 OF THE ERROR MESSAGE
;'TOTAL ERRORS:XXX  WOFL:X WRDS WRITN:XXXXXXX  ROFL:0 WRDS READ:XXXXXXX'
LINE7:  DISPLY ,LIN7T          ;TOTAL ERRORS
          MOV     $TOTAL(RO),-(SP) ;TO STACK
          JSR     PC,LINDEC     ;TYPE IT IN DECIMAL
          DISPLY ,LIN7OX       ;PRINT 'WTOFL'
          MOV     $WTOFL(RO),-(SP) ;PUSH $WTOFL(RO) ON STACK
          JSR     PC,LINDEC     ;TYPE IT IN DECIMAL
          DISPLY ,LIN7X        ;PRINT 'WRDS WRITN'
          MOV     #SWRITN,-(SP) ;ADDRESS OF LOW WORD ON STACK
          ADD     RO,(SP)
          JSR     PC,$DB2D      ;CONVERT
          JSR     PC,$SUPRL     ;PRINT
          DISPLY ,LIN7OR       ;PRINT 'ROFL'
          MOV     $RDOFL(RO),-(SP) ;PUSH $RDOFL(RO) ON STACK
          JSR     PC,LINDEC     ;TYPE IT IN DECIMAL
          DISPLY ,LIN7R        ;'WRDS READ'
          MOV     #SREAD,-(SP) ;LOW WORD ADDRESS
          ADD     RO,(SP)
          JSR     PC,$DB2D      ;CONVERT
          JSR     PC,$SUPRL     ;PRINT IT
          DISPLY ,%CRLF        ;CR-LF
          BIT     #SW15,@SWR    ;SEE IF 'HALT ON ERROR' - SWITCH 15
          BEQ     1$           ;BR IF NOT
          HALT
1$:      RTS     PC

;PRINT LINE 7A OF ERROR MESSAGE
;'TOTAL SEEKS=XXXXX  TOTAL MISPOS ERR = XXX  TOTAL SKI= XXX'
LINE7A: DISPLY ,LIN7P          ;'TOTAL SEEKS = '
          MOV     #SPOSIT,-(SP) ;TOTAL SEEKS
          ADD     RO,(SP)       ;DEVICE TABLE ADDRESS
          JSR     PC,$DB2D      ;CONVERT THE SEEK COUNT
          JSR     PC,$SUPRL     ;PRINT IT
          DISPLY ,LIN7M        ;' TOTAL MISPOS ERR = '
          MOV     $MISPO(RO),-(SP) ;TOTAL ERRORS
          JSR     PC,LINDEC     ;TYPE IT IN DECIMAL
          DISPLY ,LIN7S        ;' TOTAL SKI ERR = '
          MOV     $SKI(RO),-(SP) ;CONVERT & PRINT IT
          JSR     PC,LINDEC     ;TYPE IT IN DECIMAL
          DISPLY ,%CRLF        ;CR-LF
          BIT     #SW15,@SWR    ;SEE IF HALT ON ERROR - SWITCH 15 SET
          BEQ     1$           ;BR IF NOT
          HALT
1$:      RTS     PC
    
```

```

448                                     :PRINT LINE 8 OF THE ERROR MESSAGE
449                                     : 'DIFFERENT ERROR DURING RETRY'
450
451 023170 104414 074343                LINE8: DISPLY ,LIN8M
452 023174 004737 021214                JSR      PC,LIN2      :PRINT LINE 2 OF ERROR MESSAGE
453 023200 000207                       RTS      PC
454
455                                     :OCTAL TYPEOUT ROUTINE
456                                     :CALL:
457                                     :
458                                     :   MOV      NUM,-(SP)      :PUT THE NUMBER ON THE STACK
459                                     :   JSR      PC,LINOCT
460                                     :   RETURN
461 023202 016646 000002                LINOCT: MOV      2(SP),-(SP)  :PUT NUMBER IN PROPER LOCATION ON STACK
462 023206 004737 033260                JSR      PC,$SB20      :CONVERT THE NUMBER TO OCTAL
463 023212 012637 023226                MOV      (SP)+,1$     :GET THE ADDRESS OF THE ASCII STRING
464 023216 062737 000005 023226        ADD      #5.,1$       :ADDRESS THE LAST 6 ASCII DIGITS
465 023224 104414                       DISPLY   :TYPE IT
466 023226 000000                1$: .WORD 0          :ADDRESS
467 023230 012616                MOV      (SP)+,(SP)   :CORRECT THE STACK
468 023232 000207                RTS      PC          :RETURN
469
470                                     :ROUTINE TO CONVERT THE INPUT NUMBER TO DECIMAL AND TYPE IT WITH
471                                     :LEADING ZERO SUPRESSION
472                                     :CALL:
473                                     :
474                                     :   MOV      NUM,-(SP)      :PUT THE NUMBER ON THE STACK
475                                     :   JSR      PC,LINDEC
476                                     :   RETURN
477 023234 016646 000002                LINDEC: MOV      2(SP),-(SP)  :SET UP STACK FOR CONVERT
478 023240 004737 033230                JSR      PC,$SB20      :CONVERT IT TO DECIMAL
479 023244 004737 032364                JSR      PC,$SUPRL     :TYPE IT (WITH LEADING ZEROS SUPRESSED)
480 023250 012616                MOV      (SP)+,(SP)   :RESTORE STACK POINTER
481 023252 000207                RTS      PC

```

```

1      .SBTTL  GENERAL SUPPORT SUBROUTINES
2
3      :DECREMENT THE SECTOR-TRACK ADDRESS
4      :CALL:
5          MOV     #DPB,RO           ;DPB ADDRESS
6          JSR     PC,READDR
7          RETURN
8
9      :ON RETURN THE STACK CONTAINS THE FOLLOWING:
10         4(SP) = SECTOR ADDRESS
11         2(SP) = TRACK ADDRESS
12         (SP) = CYLINDER ADDRESS
13
14 023254 004737 027462 READDR: JSR     PC,GETLMT       ;GET ADDRESS LIMITS
15 023260 162706 000006 SUB     #6,SP           ;DECREMENT THE STACK POINTER
16 023264 016616 000006 MOV     6(SP), (SP)    ;MOVE THE RETURN ADDR DOWN THE STACK
17 023270 005066 000006 CLR     6(SP)         ;CLEAR STACK FOR SECTOR
18 023274 005066 000004 CLR     4(SP)         ;CLEAR STACK FOR TRACK
19 023300 116066 002146 000006 MOVB   $RMDA(RO),6(SP)  ;SECTOR ON STACK
20 023306 116066 002147 000004 MOVB   $RMDA+1(RO),4(SP) ;TRACK ADDRESS
21 023314 016066 002174 000002 MOV     $RMDC(RO),2(SP) ;CYLINDER ADDRESS
22 023322 005766 000006 1$: TST     6(SP)         ;SECTOR 0 ?
23 023326 001403 BEQ     2$           ;BRANCH IF SO
24 023330 105366 000006 DECB   6(SP)         ;DECREMENT ONE SECTOR
25 023334 000424 BR      4$           ;BRANCH TO EXIT
26 023336 005766 000004 2$: TST     4(SP)         ;ALSO ON TRACK 0 ?
27 023342 001406 BEQ     3$           ;BRANCH IF SO
28 023344 113766 001424 000006 MOVB   SECLMT,6(SP)    ;LAST SECTOR
29 023352 105366 000004 DECB   4(SP)         ;DECREMENT ONE TRACK
30 023356 000413 BR      4$           ;EXIT
31 023360 005766 000002 3$: TST     2(SP)         ;ALSO ON CYLINDER 0 ?
32 023364 001410 BEQ     4$           ;BR IF YES
33 023366 113766 001424 000006 MOVB   SECLMT,6(SP)    ;LAST SECTOR
34 023374 113766 001426 000004 MOVB   TRKLM,4(SP)    ;GET LAST TRACK
35 023402 005366 000002 DEC     2(SP)         ;DECREMENT ONE CYLINDER COUNT
36 023406 005337 001436 4$: DEC     DEC2          ;DECREMENT TWICE YET ?
37 023412 002343 BGE    1$           ;BR IF NO
38 023414 005037 001436 CLR     DEC2          ;DECREMENT TRK/SEC ONLY ONCE NEXT TIME
39 023420 000207 RTS      PC           ;RETURN
40
41      :ROUTINE TO CHECK FOR KW11-L OR KW11-P CLOCKS
42
43 023422 012737 177777 001312 CKCLK: MOV     #-1,CLKFLG  ;CLEAR CLOCK AVAILABILITY FLAG
44 023430 012737 177777 001310 MOV     #-1,PCLOCK     ;CLEAR KW11-P CLOCK AVAILABILITY FLAG
45 023436 013746 000004 MOV     ERRVEC,-(SP)   ;PUSH ERRVEC ON STACK
46 023442 012737 025516 000004 MOV     #CKCLK1,ERRVEC ;SET UP VECTOR FOR CLOCK CHECK
47 023450 005777 155622 TST     @CLKCSR       ;CHECK FOR KW11-P
48 023454 005037 001312 CLR     CLKFLG        ;SET CLOCK AVAILABILITY FLAG
49 023460 005037 001310 CLR     PCLOCK        ;SET KW11-P CLOCK FLAG
50 023464 013701 001302 MOV     $LPVEC,R1     ;KW11-P VECTOR ADDRESS
51 023470 012721 025110 MOV     #CLOCK,(R1)+  ;SET UP KW11-P VECTOR
52 023474 012711 000300 MOV     #300,(R1)     ;PSW - PRI 6
53 023500 012777 174575 155572 MOV     #-1667,@CLKCSB ;LOAD COUNTER BUFFER WITH 16.67
54 023506 012777 000131 155562 MOV     #131,@CLKCSR  ;SET CLOCK - CNT UP, 10US, CONT INT
55 023514 000441 BR      CKCLK3
56
57 023516 012716 023524 CKCLK1: MOV    #1$,(SP) ;SETUP RETURN ADDRESS
    
```



```

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

```

1
2
3
4
5
6
7 023626 010046
8 023630 010446
9 023632 005737 001542
21 023640 104401 001203
22 023641 004737 023742
23 023650 000004
24 023652 006304
25 023654 016400 002056
26 023660 006204
27 023662 136437 040300 001542
28 023670 001402
29 023672 004737 023752
30 023676 005204
31 023700 020427 000010
32 023704 001362
33 023706
    023706 012604
    023710 012600
    023712 000207
34
35
36
37
38
39
40
41
42
43 023714 010046
44 023716 010446
45 023720 004737 023742
46 023724 005004
47 023726 111004
48 023730 004737 023752
49 023734 012604
50 023736 012600
51 023740 000207
52
53
54
55
56
57
58 023742 004537 032574
59 023746 075626
60 023750 000207
61
62
63
64

```

```

:ROUTINE TO DISPLAY STATISTICS FOR ASSIGNED DRIVES
:CALL:
:   JSR   PC,STATPR
:   RETURN
:
STATPR:
MOV     R0,-(SP)           ;;PUSH R0 ON STACK
MOV     R4,-(SP)           ;;PUSH R4 ON STACK
TST     ASNLST             ;;ANY DRIVES ASSIGNED ?
BEQ     5$                 ;;BR IF NOT
2$:     TYPE   ,SCLF        ;;CR-LF
        JSR   PC,SHDTYP    ;;TYPE THE HEADING
        CLR   R4           ;;CLEAR THE DRIVE INDEX
        ASL   R4           ;;CHANGE TO INDEX WORDS
        MOV   BLKADR(R4),R0 ;;GET THE DRIVE'S BLOCK ADDRESS
        ASR   R4           ;;RESTORE R4
        BITB ATABIT(R4),ASNLST ;;IS THIS DRIVE ASSIGNED ?
        BEQ   4$           ;;BR IF NOT
        JSR   PC,SDETAL    ;;TYPE THE PERFORMANCE SUMMARY
        INC   R4           ;;INCREMENT THE INDEX
        CMP   R4,#8        ;;FINISHED ?
        BNE   3$           ;;BR IF NO
5$:     MOV   (SP)+,R4      ;;POP STACK INTO R4
        MOV   (SP)+,R0      ;;POP STACK INTO R0
        RTS   PC           ;;RETURN

:ROUTINE TO TYPE THE PERFORMANCE SUMMARY (STATISTICS) FOR AN INDIVIDUAL
:DRIVE.
:CALL:
:   MOV   #DPB,R0           ;;DPB ADDRESS
:   JSR   PC,SUMARY
:   RETURN
:
SUMARY: MOV   R0,-(SP)       ;;SAVE R0
        MOV   R4,-(SP)       ;;SAVE R4
        JSR   PC,SHDTYP     ;;TYPE THE HEADING
        CLR   R4           ;;CLEAR R4 FOR DRIVE NUMBER
        MOVB (R0),R4        ;;DRIVE NUMBER
        JSR   PC,SDETAL    ;;TYPE THE STATISTICS
        MOV   (SP)+,R4      ;;RESTORE R4
        MOV   (SP)+,R0      ;;RESTORE R0
        RTS   PC           ;;RETURN

:TYPE THE HEADER FOR THE DRIVE PERFORMANCE SUMMARY TYPEOUT
:CALL:
:   JSR   PC,SHDTYP
:   RETURN
:
SHDTYP: JSR   R5,TYPRI4    ;;TYPE SUMMARY HEADER
        SUMHD
        RTS   PC           ;;RETURN

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

```

```

65      :      MOV      #DPB,RO      :DPB ADDRESS
66      :      RETURN
67
68 023752 SDETAL:
    023752 104401 023760      TYPE      ,65$      ::TYPE ASCIZ STRING
    023756 000404      BR      ,64$      ::GET OVER THE ASCIZ
    ::65$: .ASCIZ <CRLF>/TIME /
    64$:
69 023770      JSR      PC,$TIME      :TYPE ELAPSED TIME
    023770 004737 024764
70
71      :TYPE LINE 2 OF SUMMARY
72 023774 104401 001203      TYPE      ,$CRLF      :CR-LF
73 024000 104401 075304      TYPE      ,UNTMSG      :TYPE 'DRIVE'
74 024004 010446      MOV      R4,-(SP)      ::SAVE R4 FOR TYPEOUT
    024006 104403      TYPOS      ::GO TYPE--OCTAL ASCII
    024010      .BYTE      2      ::TYPE 2 DIGIT(S)
    024011      .BYTE      0      ::SUPPRESS LEADING ZEROS
75 024012 104401 075234      TYPE      ,BLNKS1      :TYPE 1 BLANK
76 024016 104401 075300      TYPE      ,DASH      :TYPE '-'
77 024022 104401 075234      TYPE      ,BLNKS1      :TYPE 1 BLANK
78 024026 104401 075520      TYPE      ,RM80      :TYPE DRIVE TYPE
79 024032 104401 075276      TYPE      ,COMMA      :TYPE ','
80 024036 104401 024044      TYPE      ,67$      ::TYPE ASCIZ STRING
    024042 000404      BR      ,66$      ::GET OVER THE ASCIZ
    ::67$: .ASCIZ / PASS /
    66$:
81 024054 016046 000104      MOV      $PASSC(RO),-(SP) :PUT THE PASS COUNT ON THE STACK
82 024060 004737 033230      JSR      PC,$SB2D      :CONVERT IT
83 024064 004537 032470      JSR      R5,REPLZ      :TYPE IT
84 024070 000003      .WORD      3      :TYPE 3 DIGITS
85 024072 104401 076141      TYPE      ,PERIOD      :TYPE '.'
86 024076 104401 075234      TYPE      ,BLNKS1      :TYPE 1 BLANK
87 024102 004737 032622      JSR      PC,TYDRV      :TYPE DRV SERIAL NUMBER
88 024106 104401 075234      TYPE      ,BLNKS1      :TYPE 1 BLANK
89 024112 004737 032652      JSR      PC,TYHDA      :TYPE HDA SERIAL NUMBER
90
91      :TYPE LINE 3 OF SUMMARY
92 024116 104401 024124      TYPE      ,69$      ::TYPE ASCIZ STRING
    024122 000407      BR      ,68$      ::GET OVER THE ASCIZ
    ::69$: .ASCIZ <CRLF><LF>/WT OFLOW /
    68$:
93 024142 016046 000056      MOV      $WTOFL(RO),-(SP) ::SAVE $WTOFL(RO) FOR TYPEOUT
    024146 104405      TYPDS      ::CO TYPE--DECIMAL ASCII WITH SIGN
94 024150 104401 076141      TYPE      ,PERIOD      :TYPE '.'
95 024154 104401 024162      TYPE      ,71$      ::TYPE ASCIZ STRING
    024160 000407      BR      ,70$      ::GET OVER THE ASCIZ
    ::71$: .ASCIZ / WRDS WRITN /
    70$:
96 024200 010046      MOV      RO,-(SP)      :GET ADDRESS OF DPB
97 024202 062716 000060      ADD      #$WRITN,(SP) :POINT TO LOW NUMBER OF WRDS WRITTEN
98 024206 004737 037222      JSR      PC,$DB2D      :CONVERT DECIMAL NUMBER
99 024212 004737 032300      JSR      PC,SUPRS      :SUPPRESS LEADING ZEROS AND TYPE
100 024216 104401 076141      TYPE      ,PERIOD      :TYPE '.'
101
102      :TYPE LINE 4 OF SUMMARY
103 024222 104401 024230      TYPE      ,73$      ::TYPE ASCIZ STRING
    024226 000406      BR      ,72$      ::GET OVER THE ASCIZ
    
```

```

    024244      016046 000064      ::73$: .ASCIZ <CRLF>/RD OFLOW /
104 024244      016046 000064      72$: MOV      $RDOFL(R0),-(SP)      ;;SAVE $RDOFL(R0) FOR TYPEOUT
    024250      104405      TYPDS      ;;GO TYPE--DECIMAL ASCII WITH SIGN
105 024252      104401 076141      TYPE      ,PERIOD      ;;TYPE ' '
106 024256      104401 024264      TYPE      75$      ;;TYPE ASCIZ STRING
    024262      000407      BR      74$      ;;GET OVER THE ASCIZ

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

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

1
15
16

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

17

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

18

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

19

```
:ROUTINE TO INCREMENT $MISPO
:NOTE: $MISPO WILL NOT BE INCREMENTED BEYOND 77777 (32767.)
024714 005737 001342 INCMIS: TST BADSEC ;SEE IF BAD TRK/SEC INDICATOR SET
024720 001006 BNE 1$ ;BR IF IT'S SET, DON'T INCREMENT COUNT
024722 026027 000102 0.7777 CMP $MISPO(RO),#77777 ;IS $MISPO ALREADY AT MAXIMUM?
024730 103002 BHIS 1$ ;BR IF IT IS
024732 005260 000102 INC $MISPO(RO) ;INCREMENT $MISPO
024736 000207 1$: RTS PC ;RETURN
```

20

```
:ROUTINE TO INCREMENT $TOTAL
:NOTE: $TOTAL WILL NOT BE INCREMENTED BEYOND 77777 (32767.)
024740 005737 001342 INCTOT: TST BADSEC ;SEE IF BAD TRK/SEC INDICATOR SET
024744 001006 BNE 1$ ;BR IF IT'S SET, DON'T INCREMENT COUNT
024746 026027 000072 077777 CMP $TOTAL(RO),#77777 ;IS $TOTAL ALREADY AT MAXIMUM?
024754 103002 BHIS 1$ ;BR IF IT IS
024756 005260 000072 INC $TOTAL(RO) ;INCREMENT $TOTAL
024762 000207 1$: RTS PC ;RETURN
```

```

1
2
3          ;ROUTINE TO TYPE THE TIME
4 024764 005737 001312 $TIME: TST CLKFLG          ;CLOCK ON THE SYSTEM ?
5 024770 001046          BNE 3$          ;BR IF NOT
6 024772 012737 000002 025040 MOV #2,2$          ;ASSUME 2 DIGITS TO TYPE
7 025000 013746 001344 MOV HOUR,-(SP)          ;PUT 'HOURS' ON THE STACK
8 025004 021627 000144 CMP (SP),#100.          ;100. HOURS OR MORE ?
9 025010 002407          BLT 1$          ;BR IF NO
10 025012 005237 025040 INC 2$          ;TYPE 3 DIGITS
11 025016 021627 001750 CMP (SP),#1000.          ;1000. HOURS OR MORE ?
12 025022 002402          BLT 1$          ;BR IF NO
13 025024 005237 025040 INC 2$          ;TYPE 4 DIGITS
14 025030 004737 033230 1$: JSR PC,$SB2D          ;CONVERT TO DECIMAL
15 025034 004537 032464 JSR R5,FILLZ          ;TYPE IT
16 025040 000000          .WORD 0          ;NUMBER OF HOUR DIGITS TO TYPE
17 025042 104401 076332 TYPE ,COLON          ;:
18 025046 013746 001346 MOV MINUTE,-(SP)          ;PUT 'MINUTES' ON THE STACK
19 025052 004737 033230 JSR PC,$SB2D          ;CONVERT TO DECIMAL
20 025056 004537 032464 JSR R5,FILLZ          ;TYPE IT
21 025062 000002          .WORD 2          ;TYPE 2 DIGITS
22 025064 104401 076332 TYPE ,COLON          ;:
23 025070 013746 001350 MOV SECOND,-(SP)          ;PUT SECONDS ON THE STACK
24 025074 004737 033230 JSR PC,$SB2D          ;CONVERT TO DECIMAL
25 025100 004537 032464 JSR R5,FILLZ          ;TYPE IT
26 025104 000002          .WORD 2          ;TYPE 2 DIGITS
27 025106 000207          RTS PC
28
29          ;CLOCK HANDLER ROUTINE
30
31 025110 005337 001352 CLOCK: DEC ONESEC          ;INCREMENT THE ONE SECOND COUNTER
32 025114 001027          BNE 1$          ;BR IF A SECOND NOT COUNTED
33 025116 013737 001314 001352 MOV HZ,ONESEC          ;RESTORE THE VALUE
34 025124 005237 001350 INC SECOND          ;COUNT THE SECOND
35 025130 022737 000074 001350 CMP #60.,SECOND          ;AT MAXIMUM ?
36 025136 001016          BNE 1$          ;BR IF NOT
37 025140 005037 001350 CLR SECOND          ;CLEAR THE SECOND'S COUNTER
38 025144 005237 001466 INC INTRVL+2          ;COUNT THE PERFORMANCE SUMMARY INTERVAL
39 025150 005237 001346 INC MINUTE          ;COUNT THE MINUTE
40 025154 022737 000074 001346 CMP #60.,MINUTE          ;AT MAXIMUM ?
41 025162 001004          BNE 1$          ;BR IF NOT
42 025164 005037 001346 CLR MINUTE          ;CLEAR THE MINUTE'S COUNTER
43 025170 005237 001344 INC HOUR          ;COUNT THE HOURS
44 025174 022737 000062 001314 1$: CMP #50.,HZ          ;CPU RUNNING @ 50HZ ?
45 025202 001403          BEQ 2$          ;BR IF YES
46 025204 012746 000020 MOV #16.,-(SP)          ;16MS ON THE STACK @ 60HZ
47 025210 000402          BR 3$
48 025212 012746 000024 2$: MOV #20.,-(SP)          ;20MS ON THE STACK @ 50HZ
49 025216 004737 044152 3$: JSR PC,RMTMR          ;DRIVER TIMER ROUTINE
50 025222 005737 001464 TST INTRVL          ;DISPLAY THE PERFORMANCE SUMMARY ?
51 025226 001411          BEQ 4$          ;BR IF NOT
52 025230 023737 001464 001466 CMP INTRVL,INTRVL+2          ;DISPLAY INTERVAL FINISHED ?
53 025236 001005          BNE 4$          ;BR IF NOT
54 025240 012737 177777 001316 MOV #-1,STATIN          ;SET PERFORMANCE SUMMARY DISPLAY FLAG
55 025246 005037 001466 CLR INTRVL+2          ;CLEAR THE PERFORMANCE INTERVAL COUNTER
56 025252 000002          RTS
    
```

```

1
2
3      :COMMAND DECODE ROUTINE
4      :CALL:
5
6      :
7      :
8      :
9
10     025254 005737 040276      KSR:  TST      DTUW      :ANY DATA TRANSFERS UNDER WAY ?
11     025260 100375              BPL      KSR          :BR IF YES
12     025262 104412              JSR      PC,KSR      :ROUTINE IN INTERRUPT MODE
13     025264 012737 000200 177776      :
14     025272 013704 040310      :
15     025276 012764 000040 000010      :
16     025304 005037 001340      :
17     025310 104401 001203      :
18     025314 004737 024764      :
19     025320 004737 033326      :
20     025324 104401 076231      :
21
22     025330 104411              RDLIN
23     025332 012605              MOV      (SP)+,R5    :READ THE KEYBOARD
24     025334 005737 001340      MOV      CFLAG      :GET ADDRESS OF INPUT STRING
25     025340 001405              TST      CFLAG      :CHECK THE CONTROL C FLAG
26     025342 005737 001542      BEQ      2$          :BR IF NO 'CONTROL C' ENTERED
27     025346 001136              TST      ASNLST     :ANY DRIVES ASSIGNED ?
28     025350 000137 003532      BNE      13$        :BR IF YES
29
30     025354 005205              JMP      START      :JUMP TO START
31
32     025356 122715 000124      2$:  INC      R5          :POINT TO SECOND CHARACTER
33     025362 001465              CMPB    #'T,(R5)    :EQ TO A 'T' ?
34     025364 122715 000101      BEQ      9$          :YES
35     025370 001410              CMPB    #'A,(R5)    :EQ TO AN 'A'
36     025372 121527 000067      BEQ      3$          :BR IF IT IS
37     025376 101117              CMPB    (R5),#'7    :DRIVE NUMBER GREATER THAN AN ASCII 7 ?
38     025400 121527 000060      BHI     12$         :BR IF IT IS
39     025404 103514              CMPB    (R5),#'0    :DRIVE NUMBER LESS THAN AN ASCII 0 ?
40     025406 142715 177770      BLO     12$         :BR IF IT IS
41     025412 122765 000124 177777 3$:  BICB    #'^C7,(R5)  :LEAVE ONLY LOWER 3 BITS IF CHAR NOT 'A'
42     025420 001003              CMPB    #'T,-1(R5) :EQ TO 'T'
43     025422 004737 026556      BNE     4$          :BR IF NOT EQ
44     025426 000506              JSR     PC,NEWASN   :ASSIGN DRIVE FOR TEST
45     025430 122765 000104 177777 4$:  BR      13$        :EXIT
46     025436 001003              CMPB    #'D,-1(R5) :EQ TO 'D' ?
47     025440 004737 026356      BNE     5$          :BR IF NOT EQ
48     025444 000477              JSR     PC,DEASGN  :DEASSIGN DRIVE
49     025446 122765 000123 177777 5$:  BR      13$        :EXIT
50     025454 001003              CMPB    #'S,-1(R5) :EQ TO 'S'
51     025456 004737 026464      BNE     6$          :BR IF NOT EQ
52     025462 000470              JSR     PC,SCMND   :TYPE STATISTICS
53     025464 122765 000127 177777 6$:  BR      13$        :EXIT
54     025472 001012              CMPB    #'W,-1(R5) :EQ TO 'W'
55     025474 005737 001440      BNE     8$          :BR IF NOT EQ
56     025500 001053              TST     RDNLY      :LOCKED IN 'READ ONLY' MODE ?
57     025502 032777 000001 153444  BNE     11$         :BR IF YES
58     025510 001047              BIT     #SW0,@SWR  :IS SWITCH 0 SET ?
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
    
```



```

58 025512 004737 026600      7$:   JSR     PC, DATAPK      ;WRITE A DATA PACK
59 025516 000452              BR      13$             ;EXIT
60 025520 122765 000122 177777 8$:   CMPB   #'R,-1(R5)      ;EQ TO 'R' ?
61 025526 001043              BNE    12$             ;BR IF NOT EQ
62 025530 004737 026566      JSR     PC, REDAPK     ;READ A DATA PACK
63 025534 000443              BR      13$             ;EXIT
64 025536 122765 000127 177777 9$:   CMPB   #'W,-1(R5)      ;WT COMMAND ?
65 025544 001034              BNE    12$             ;NO
66 025546 005737 001440      TST    RONLY          ;LOCKED IN 'READ ONLY' MODE ?
67 025552 001026              BNE    11$             ;BR IF YES
68 025554 032777 000001 153372  BIT    #SW0,@SWR      ;IS SWITCH 0 SET ?
69 025562 001022              BNE    11$             ;BR IF SET, CAN'T DO 'W' COMMAND
70 025564 122765 000101 000001  CMPB   #'A,1(R5)      ;ALL DRIVES ?
71 025572 001413              BEQ    10$             ;YES
72 025574 126527 000001 000067  CMPB   1(R5),#'7      ;GREAT THAN 7
73 025602 101015              BHI    12$             ;YES
74 025604 126527 000001 000060  CMPB   1(R5),#'0      ;LESS THAN 0
75 025612 103411              BLO    12$             ;YES
76 025614 142765 177770 000001  BICB   #'C7,1(R5)    ;CHOP OFF THE HIGHER BITS
77 025622 004737 026612      10$:  JSR     PC, WATPAK     ;ASSIGN DRIVES WITH WT COMMAND
78 025626 000406              BR      13$             ;
79 025630 104401 076143      11$:  TYPE   ,MSWRO        ;TYPE 'CAN'T WRITE IN READ ONLY MODE'
80 025634 000616              BR      1$             ;TRY AGAIN
81 025636 104401 076206      12$:  TYPE   ,INVLD        ;TYPE 'INVALID COMMAND' MESSAGE
82 025642 000613              BR      1$             ;TRY AGAIN
83 025644 104413      13$:  RESREG ;RESTORE R0 - R5
84 025646 005777 153310      TST    @STKB          ;CLEAR THE TTY BUFFER
85 025652 052777 000100 153300  BIS    #BIT06,@STKS   ;SET TTY INTERRUPT ENABLE
86 025660 005037 177776      CLR    PS             ;SET PRIORITY BACK TO ZERO
87 025664 000207              RTS     PC             ;RETURN
88
89
90
91 025666 111504      ASSIGN: MOVB   (R5),R4      ;PUT DRIVE # IN R4
92 025670 005037 001340 1$:   CLR    CFLAG         ;CLEAR CONTROL C FLAG
93 025674 005037 001442      CLR    DRVPAR        ;ASSUME CHANGING DRIVE PARAMETERS
94 025700 104401 077344      TYPE   ,MSPRM        ;TYPE 'CHANGE DRIVE PARAMETERS ?'
95 025704 104411      RDLIN ;READ THE ENTRY
96 025706 012600      MOV    (SP)+,R0      ;SAVE ADDRESS OF RESPONSE
97 025710 005737 001340      TST    CFLAG         ;WAS IT CONTROL C ?
98 025714 001365              BNE    1$             ;BR IF YES
99 025716 105710      TSTB   (R0)          ;WAS RESPONSE A CARRIAGE RETURN (DEFAULT 'N')?
100 025720 001414      BEQ    3$             ;BR IF YES
101 025722 105760 000001      TSTB   1(R0)         ;WAS IT TERMINATED WITH CARRIAGE RETURN ?
102 025726 001006              BNE    2$             ;BR IF NO
103 025730 122710 000131      CMPB   #'Y,(R0)      ;WAS IT A 'Y' RESPONSE ?
104 025734 001410              BEQ    4$             ;BR IF YES
105 025736 122710 000116      CMPB   #'N,(R0)      ;WAS IT A 'N' RESPONSE ?
106 025742 001403              BEQ    3$             ;BR IF YES
107 025744 104401 076341      2$:   TYPE   ,BADENT      ;TYPE BAD ENTRY MESSAGE
108 025750 000747              BR      1$             ;TRY AGAIN
109 025752 005237 001442      3$:   INC    DRVPAR        ;DO NOT CHANGE DRIVE PARAMETERS
110 025756 122704 000101      4$:   CMPB   #'A,R4      ;ASSIGN ALL DRIVES ?
111 025762 001426              BEQ    ASGN2          ;BR IF YES
112
113 025764 012737 075355 031200 ASGN1: MOV    #UNTASN,ASNMSG ;'DRIVE ASSIGNED' MESSAGE ADDRESS
114 025772 005737 001444      TST    XXDP          ;LOADED FROM THIS DEVICE ?
    
```

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

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

```

1
2
3
4 026356 005004
5 026360 012703 000010
6 026364 122715 000101
7 026370 001403
8 026372 111504
9 026374 012703 000001
10 026400 136437 040300 001542 1$:
11 026406 001417
12 026410 146437 040300 001542
13 026416 146437 040300 032012
14 026424 006304
15 026426 016464 002056 001544
16 026434 006204
17 026436 005303 2$:
18 026440 001410
19 026442 005204
20 026444 000755
21 026446 012737 075333 031200 3$:
22 026454 004737 031154
23 026460 000766
24 026462 000207 4$:
25
26
27
28 026464
29 026464 013746 001542
30 026470 122715 000101
31 026474 001416
32 026476 111504
33 026500 136416 040300
34 026504 001404
35 026506 116437 040300 001542
36 026514 000411
37 026516 012737 075333 031200 1$:
38 026524 004737 031154
39 026530 000407
40 026532 105737 001542 2$:
41 026536 001404
42 026540 004737 023626 3$:
43 026544 104401 075617
44 026550 4$:
45 026550 012637 001542
46 026554 000207
47
48
49 026556 005037 001320
50 026562 000137 025666
51
52
53
54 026566 012737 000001 001320 REDAPK:
55 026574 000137 025666
    
```

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

```

DEASGN: CLR R4 ;START WITH DRIVE 0
MOV #8,R3 ;COUNTER
CMPB #'A,(R5) ;DEASSIGN ALL DRIVES ?
BEQ 1$ ;BR IF YES
MOVB (R5),R4 ;GET DRIVE NUMBER
MOV #1,R3 ;SET R3 FOR ONE DRIVE
BITB ATABIT(R4),ASNLST ;DRIVE ASSIGNED ?
BEQ 3$ ;BR IF NOT
BICB ATABIT(R4),ASNLST ;DELETE THE DRIVE FROM THE ASSIGNED LIST
BICB ATABIT(R4),AUTLST ;DELETE DRIVE FROM AUTO ASSIGN LIST
ASL R4 ;MAKE ADDR INTO A WORD INDEX
MOV BLKADR(R4),DDRVS(R4) ;PUT ADDRESS IN DEASSIGN LIST
ASR R4
2$: DEC R3 ;ANY MORE DRIVES ?
BEQ 4$ ;BR IF NOT
INC R4
BR 1$
3$: MOV #UNTNOT,ASNMSG ;ADDR OF 'NOT ASSIGNED' MESSAGE
JSR PC,ASNERR ;REPORT IT
BR 2$
4$: RTS PC
    
```

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

```

SCMND: MOV ASNLST -(SP) ;;PUSH ASNLST ON STACK
CMPB #'A,(R5) ;ALL STATISTICS ?
BEQ 2$ ;BR IF YES
MOVB (R5),R4 ;GET DRIVE NUMBER
BITB ATABIT(R4),(SP) ;IS THIS DRIVE ASSIGNED ?
BEQ 1$ ;BR IF NO
MOVB ATABIT(R4),ASNLST ;GET DRIVE ASSIGN BIT
BR 3$
1$: MOV #UNTNOT,ASNMSG ;ADDR OF 'NOT ASSIGNED' MSG
JSR PC,ASNERR ;TYPE ERROR MESSAGE
BR 4$ ;EXIT
2$: TSTB ASNLST ;ANY DRIVE ASSIGNED ?
BEQ 4$ ;BR IF NO
3$: JSR PC,STATPR ;TYPE ALL STATISTICS
TYPE ,STAR5 ;TYPE '****...ETC'
4$: MOV (SP)+,ASNLST ;;POP STACK INTO ASNLST
RTS PC
    
```

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

```

NEWASN: CLR PACK ;SET 'T' COMMAND INDICATOR
JMP ASSIGN ;GO TO THE ASSIGN ROUTINE
    
```

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

```

REDAPK: MOV #1,PACK ;SET THE 'READ' INDICATOR
JMP ASSIGN ;ASSIGN THE REQUESTED DRIVE
    
```

```
56  
57  
58  
59 026600 012737 177777 001320 DATAPK: MOV #-1,PACK ;SET THE 'W' COMMAND INDICATOR  
60 026606 000137 025666 JMP ASSIGN ;ASSIGN REQUESTED DRIVE  
61  
62  
63  
64 026612 116515 000001 WATPAK: MOVB 1(R5),(R5) ;ADJUST DRIVE NUMBER ADDRESS  
65 026616 012737 177776 001320 MOV #-2,PACK ;PACK WRITE COMMAND  
66 026624 000137 025666 JMP ASSIGN ;JUMP TO ASSIGN ROUTINE
```

```

1
2
3
4
5
6
7
8
9
10 026630
    026630 010146
    026632 010346
    026634 010446
    026636 010546
11 026640 005737 037716
12 026644 001076
13 026646 010004
14 026650 062704 000002
15 026654 012703 000012
16 026660 005024
17 026662 162703 000002
18 026666 001374
19 026670 062704 000002
20 026674 012703 000106
21 026700 005024
22 026702 162703 000002
23 026706 001374
24 026710 062704 002004
25
27 026714 012703 000064
31 026720 012764 177777 177776
32 026726 005024
33 026730 162703 000002
34 026734 001374
35 026736 113760 001514 000024
36 026744 013701 001514
37 026750 116160 002076 000002
38 026756 113760 001512 000030
39 026764 106360 000030
40 026770 013760 001516 000020
41 026776 013760 001516 000004
42 027004 005460 000004
43 027010 012760 000400 000022
44 027016 012760 000001 000104
45 027024 132760 000001 000024
46 027032 001403
47 027034 062760 000002 000022
48 027042
    027042 012605
    027044 012604
    027046 012603
    027050 012601
49 027052 000207
50
51
52
53

```

```

:ROUTINE TO CLEAR THE DPB FOR THE ASSIGNED DRIVE
:CALL:
:      MOV      #DPB,R0          ;DPB ADDRESS
:      JSR      PC,CLRDPB
:      RETURN
:RO = DPB ADDRESS BEFORE CALLING THE ROUTINE

CLRDPB:
MOV      R1,-(SP)              ;;PUSH R1 ON STACK
MOV      R3,-(SP)              ;;PUSH R3 ON STACK
MOV      R4,-(SP)              ;;PUSH R4 ON STACK
MOV      R5,-(SP)              ;;PUSH R5 ON STACK
TST      PWRFLG                ;RETURNING FROM POWER FAIL ?
BNE      4$                    ;BRANCH IF YES
MOV      R0,R4                  ;GET THE DPB ADDRESS
ADD      #2,R4                  ;ADDRESS OF FIRST LOCN TO BE CLEARED
MOV      #<$CYL-$COMND>+2,R3    ;NUMBER OF LOCNS TO BE CLEARED
1$:      CLR      (R4)+          ;CLEAR THE LOCATION
SUB      #2,R3                  ;DONE CLEARING YET ?
BNE      1$                    ;BR IF NO
ADD      #2,R4                  ;SKIP OVER THE '$REG' LOCATION
MOV      #<$NEXT-$STATUS>+2,R3  ;NUMBER OF LOCNS TO BE CLEARED
2$:      CLR      (R4)+          ;CLEAR THE LOCATION
SUB      #2,R3                  ;DONE CLEARING YET ?
BNE      2$                    ;BR IF NO
ADD      #<$DRVSN-$FIRST>,R4    ;SKIP OVER FIRST FLAG, MIN/MAX ADRS
                                ;LIMITS AND BAD SECTOR TABLE
MOV      #<$RMCS3-$DRVSN>+2,P3  ;NUMBER OF LOCNS TO BE CLEARED
3$:      MOV      #-1,-2(R4)     ;INITIALIZE TERMINATOR FOR BAD SECTOR TABLE
CLR      (R4)+                  ;CLEAR A LOCATION
SUB      #2,R3                  ;DONE CLEARING YET ?
BNE      3$                    ;BR IF NO
MOVB     BEGCD,$CODE(R0)        ;INITIAL COMMAND CODE
MOV      BEGCD,R1               ;GET THE ACTUAL OP CODE
MOVB     COMTBL(R1),$COMND(R0)  ;OPERATION CODE
MOVB     BEGPAT,$PATT(R0)      ;PATTERN CODE
ASLB     $PATT(R0)              ;CONVERT CODE TO A TABLE INDEX
MOV      BEGWC,$WRDL(R0)       ;BEGINNING WORD COUNT
MOV      BEGWC,$WCNT(R0)       ;VALUE FOR DATA TRANSFER
NEG      $WCNT(R0)              ;MAKE IT INTO 2'S COMPLEMENT
MOV      #256,$SSEC(R0)        ;INITIAL VALUE OF SECTOR SIZE
MOV      #1,$PASSC(R0)         ;PRESET PASS COUNT TO 1
BITB     #1,$CODE(R0)          ;HEADER COMMAND ?
BEQ      4$                    ;BR IF NOT
ADD      #2,$SSEC(R0)          ;ADD HEADER SIZE TO SECTOR SIZE
4$:      MOV      (SP)+,R5       ;;POP STACK INTO R5
MOV      (SP)+,R4              ;;POP STACK INTO R4
MOV      (SP)+,R3              ;;POP STACK INTO R3
MOV      (SP)+,R1              ;;POP STACK INTO R1
RTS      PC                    ;RETURN

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

```

```

54      :      JSR      PC,DRVPRM      ;CALL ROUTINE
55      :
56      :RO = DPB ADDRESS BEFORE CALLING THE ROUTINE
57
58 027054 010346      DRVPRM: MOV      R3,-(SP)      ;SAVE R3
59 027056 010446      MOV      R4,-(SP)      ;SAVE R4
60 027060 105737 001150  TSTB     $AUTOB      ;RUNNING IN AUTO MODE ?
61 027064 001010      BNE     1$           ;BR IF YES
62 027066 005737 001336  TST     CHGADR      ;PROGRAM STARTED AT 200 ?
63 027072 003005      BGT     1$           ;BR IF YES
64 027074 005737 001442  TST     DRVPAR      ;CHANGE DRIVE PARAMETERS ?
65 027100 001002      BNE     1$           ;BR IF NO
66 027102 104401 076252  TYPE     ,ENTLMT     ;'ENTER ADDRESS LIMITS'
67
68 027106 004737 027462 1$:      JSR     PC,GETLMT     ;GET ADDRESS LIMITS
69 027112 062760 177777 000124  ADD     #-1,$FIRST(R0) ;SEE IF FIRST TIME STARTED
70 027120 103426      BCS     4$           ;BR IF NOT
71 027122 013760 001422 000126  MOV     CYLIMT,MAXCYL(R0) ;LOAD MAXIMUM CYLINDER
72 027130 013760 001426 000132  MOV     TRKLMT,MAXTRK(R0) ;LOAD MAXIMUM TRACK
73 027136 013760 001424 000136  MOV     SECLMT,MAXSEC(R0) ;LOAD MAXIMUM SECTOR
74 027144 005737 001434      TST     FEFLAG      ;USING FE CYLINDERS ONLY ?
75 027150 001004      BNE     2$           ;BR IF NO
76 027152 013760 001430 000130  MOV     FE1,MINCYL(R0) ;RESET MINIMUM CYLINDER ADDRESS
77 027160 000402      BR      3$
78 027162 005060 000130 2$:      CLR     MINCYL(R0)   ;CLEAR MINIMUM CYLINDER
79 027166 005060 000134 3$:      CLR     MINTRK(R0)  ;CLEAR MINIMUM TRACK
80 027172 005060 000140      CLR     MINSEC(R0)  ;CLEAR MINIMUM SECTOR
81
82 027176 105737 001150 4$:      TSTB     $AUTOB      ;RUNNING IN AUTO MODE ?
83 027202 001113      BNE     9$           ;BR IF YES
84 027204 005737 001336  TST     CHGADR      ;PROGRAM STARTED AT 200 ?
85 027210 003110      BGT     9$           ;BR IF YES
86 027212 005737 001442  TST     DRVPAR      ;CHANGE DRIVE PARAMETERS ?
87 027216 001105      BNE     9$           ;BR IF NO
88 027220 016403 100042  MOV     TABLE(R4),R3 ;PARAMETER TABLE ADDRESS
89 027224 013763 001422 000002  MOV     CYLIMT,2(R3)   ;LOAD CYLINDER LIMIT FOR MINCYL
90 027232 013763 001422 000010  MOV     CYLIMT,10(R3)  ;LOAD CYLINDER LIMIT FOR MAXCYL
91 027240 013763 001426 000016  MOV     TRKLMT,16(R3) ;LOAD TRACK LIMIT FOR MINTRK
92 027246 013763 001426 000024  MOV     TRKLMT,24(R3) ;LOAD TRACK LIMIT FOR MAXTRK
93 027254 013763 001424 000032  MOV     SECLMT,32(R3) ;LOAD SECTOR LIMIT FOR MINSEC
94 027262 013763 001424 000040  MOV     SECLMT,40(R3) ;LOAD SECTOR LIMIT FOR MAXSEC
95 027270 004737 031030      JSR     PC,PARENT    ;GET THE DRIVE'S PARAMETERS
96
97 027274 016003 000130      MOV     MINCYL(R0),R3 ;STORE MINCYL VALUE
98 027300 016004 000126      MOV     MAXCYL(R0),R4 ;STORE MAXCYL VALUE
99 027304 020304      CMP     R3,R4        ;IS MIN. LESS THAN OR EQUAL TO MAX. ?
100 027306 003404      BLE     5$          ;BR IF YES
101 027310 010360 000126      MOV     R3,MAXCYL(R0) ;SWAP MIN. TO MAX.
102 027314 010460 000130      MOV     R4,MINCYL(R0) ;SWAP MAX. TO MIN.
103 027320 016003 000134 5$:      MOV     MINTRK(R0),R3 ;STORE MINTRK VALUE
104 027324 016004 000132      MOV     MAXTRK(R0),R4 ;STORE MAXTRK VALUE
105 027330 020304      CMP     R3,R4        ;IS MIN. LESS THAN OR EQUAL TO MAX. ?
106 027332 003404      BLE     6$          ;BR IF YES
107 027334 010360 000132      MOV     R3,MAXTRK(R0) ;SWAP MIN. TO MAX.
108 027340 010460 000134      MOV     R4,MINTRK(R0) ;SWAP MAX. TO MIN.
109 027344 016003 000140 6$:      MOV     MINSEC(R0),R3 ;STORE MINSEC VALUE
110 027350 016004 000136      MOV     MAXSEC(R0),R4 ;STORE MAXSEC VALUE
    
```

```

111 027354 020304      CMP      R3,R4      ;IS MIN. LESS THAN OR EQUAL TO MAX. ?
112 027356 003404      BLE      7$        ;BR IF YES
113 027360 010360 000136    MOV      R3,MAXSEC(R0) ;SWAP MIN. TO MAX.
114 027364 010460 000140    MOV      R4,MINSEC(R0) ;SWAP MAX. TO MIN.
115
116 027370 005737 001434      7$:     TST      FEFLAG      ;USING FE CYLINDERS ONLY ?
117 027374 001016      BNE      9$        ;BR IF NO
118 027376 026037 000130 001430    CMP      MINCYL(R0),FE1 ;IS MIN. CYLINDER < 1ST FE CYLINDER ?
119 027404 103003      BNE      8$        ;BR IF NO
120 027406 013760 001430 000130    MOV      FE1,MINCYL(R0) ;YES, RESET MIN. CYLINDER
121 027414 026037 000126 001430    8$:     CMP      MAXCYL(R0),FE1 ;IS MAX. CYLINDER < 1ST FE CYLINDER ?
122 027422 103003      BNE      9$        ;BR IF NO
123 027424 013760 001432 000126    MOV      FE2,MAXCYL(R0) ;YES, RESET MAX. CYLINDER
124
125 027432 016060 000130 000012    9$:     MOV      MINCYL(R0),$CYL(R0) ;INITIAL CYLINDER VALUE
126 027440 116060 000134 000011    MOV      MINTRK(R0),$TRK(R0) ;INITIAL TRACK VALUE
127 027446 116060 000140 000010    MOV      MINSEC(R0),$SEC(R0) ;INITIAL SECTOR VALUE
128 027454 012604      MOV      (SP)+,R4    ;POP STACK INTO R4
128 027456 012603      MOV      (SP)+,R3    ;POP STACK INTO R3
129 027460 000207      RTS      PC          ;RETURN
130
131      ;ROUTINE TO GET THE ADDRESS LIMITS FOR THE CURRENT DRIVE TYPE
132      ;CALL:
133      ;      JSR      PC,GETLMT      ;CALL ROUTINE
134      ;
135      ;RO = DPB ADDRESS BEFORE CALLING THE ROUTINE
136
137 027462 005737 001434      GETLMT: TST      FEFLAG      ;USING FE CLYINDERS ONLY ?
138 027466 001004      BNE      1$        ;BR IF NO
139 027470 013737 001432 001422    MOV      FE2,CYLIMT    ;GET 2ND FE CYLINDER
140 027476 000403      BR       2$
141
142 027500 012737 001056 001422    1$:     MOV      #558.,CYLIMT ;GET LAST CYLINDER
143 027506 012737 000015 001426    2$:     MOV      #13.,TRKLMT ;GET LAST TRACK FOR AN RM80
144 027514 012737 000036 001424      MOV      #30.,SECLMT ;GET LAST SECTOR
145 027522 032760 001000 002172    BIT      #SSEI,$RMOF(R0) ;IS SKIP SECTOR INHIBIT SET ?
146 027530 001003      BNE      3$        ;BR IF YES
147 027532 005760 000112      TST      $$SEN8(R0) ;WAS SKIP SECTORING ENABLED DURING XFER ?
148 027536 001403      BEQ      4$        ;BR IF NO
149 027540 012737 000037 001424    3$:     MOV      #31.,SECLMT ;GET LAST SECTOR
150 027546 000207      4$:     RTS      PC          ;RETURN
    
```



```

1      ;ROUTINE TO GET THE DRIVE (MBA) SERIAL NUMBER FROM RMSN REGISTER
2      ;THIS NUMBERS CONTAINED IN THE REGISTER ARE ONLY THE 4 LSD'S OF THE
3      ;SERIAL NUMBER.
4      ;CALL:
5      ;
6      ;       MOV     #DPB,R0           ;DP3 ADDRESS
7      ;       JSR     PC,GETID        ;CALL ROUTINE
8      ;
9      ;RO = DPB ADDRESS BEFORE CALLING THE ROUTINE
10     GETID:
11     MOV     R0,-(SP)                ;;PUSH R0 ON STACK
12     MOV     R1,-(SP)                ;;PUSH R1 ON STACK
13     MOV     R2,-(SP)                ;;PUSH R2 ON STACK
14     MOV     R5,-(SP)                ;;PUSH R5 ON STACK
15     MOV     R0,R2                  ;GET INDEX TO DPB
16     JSR     PC,SVRH70              ;SAVE ALL REGISTERS
17     MOV     #4,R2                  ;FOUR DIGITS TO STORE
18     MOV     $RMSN(R0),R1           ;SERIAL NUMBER
19     1$:   CLR     R5                 ;ZERO
20     ROL     R1                     ;PUT THE NEXT DIGIT
21     ROL     R5                     ;INTO R5
22     ROL     R1
23     ROL     R5
24     ROL     R1
25     ROL     R5
26     ROL     R1
27     ROL     R5
28     ADD     #'0,R5                 ;MAKE IT ASCII
29     MOVB    R5,$DRVSN(R0)          ;SAVE DRIVE (MBA) SERIAL NUMBER DIGIT
30     INC     R0                     ;GET NEXT INDEX FOR DRIVE (MBA) SERIAL NUMBER
31     DEC     R2                     ;ALL DIGITS TYPED?
32     BGT     1$                     ;NO -- BRANCH
33     MOV     (SP)+,R5                ;;POP STACK INTO R5
34     MOV     (SP)+,R2                ;;POP STACK INTO R2
35     MOV     (SP)+,R1                ;;POP STACK INTO R1
36     MOV     (SP)+,R0                ;;POP STACK INTO R0
37     RTS     PC                     ;RETURN
    
```

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

.SBTTL READ DEC144 FILE

: THIS ROUTINE IS USED TO READ THE DEC144 BAD SECTOR FILE FROM CYLINDER
 : 558. TRACK 13. AND TO STORE THE FILE IN IT'S RESPECTIVE DPB TABLE.
 : THE DPB TABLE HAS ENOUGH ROOM TO SAVE THE ENTIRE MFG AND USR PORTIONS
 : OF THE DEC144 FILE. (MFG=126. ENTRIES AND USR=126. ENTRIES) EVERY TIME
 : THE DRIVE IS ASSIGNED THE DEC144 FILE IS READ TO DETERMINE THE STATUS
 : OF THE CURRENT HDA SERIAL NUMBER. BUT, IN ORDER TO INITIALIZE THE
 : BAD SECTOR ENTRY TABLE, AT LEAST ONE OF FOLLOWING STATEMENTS MUST BE VALID.

- : 1) FIRST TIME PROGRAM WAS STARTED
- : OR
- : 2) LOCATION 'BADBLK' IS EQUAL TO 1
- : OR
- : 3) LOCATION 'BADBLK' IS EQUAL TO 0 AND THE HDA
 SERIAL NUMBER CHANGED SINCE THE LAST TIME IT WAS
 READ. (DEFAULT)

: NOTE: IF THE SERIAL NUMBER HAS CHANGED, THIS MOST LIKELY MEANS THAT THE
 HDA OR DRIVE HAD BEEN REPLACED WHILE THE DRIVE WAS DEASSIGNED.

: THIS ROUTINE CHECKS THAT THE TWO SERIAL NUMBER WORDS ARE NOT ZERO
 : AND THE ENTIRE SERIAL NUMBER IS POSITIVE. ALSO, WORDS 3 AND 4 ARE
 : CHECKED TO BE ALL ZERO WORDS. IF THE DEC144 FILE DOES NOT COMPLY
 : WITH THIS STRUCTURE, AN ERROR MESSAGE IS TYPED AND THE ROUTINE IS EXITED.

: CALL
 : MOV #DPB,R0 :DPB ADDRESS
 : JSR R5,GETADR :READ DEC144 BAD SECTOR FILES
 : RO = DPB ADDRESS BEFORE CALLING THE ROUTINE

GETADR:

```

MOV R1,-(SP)      ;;PUSH R1 ON STACK
MOV R2,-(SP)      ;;PUSH R2 ON STACK
MOV R3,-(SP)      ;;PUSH R3 ON STACK
JSR PC,GETLMT     :GET ADDRESS LIMITS
MOV R0,R1         :DPB ADDRESS
AND #5BDSEC,R1    :ADDRESS OF BAD SECTOR TABLE
MOV R1,-(SP)      ;;PUSH R1 ON STACK
MOVB (R0),GENDPB  :DRIVE NUMBER
MOV #558,GENDPB+$CYL :LAST CYLINDER
MOVB TRKLMT,GENDPB+$TRK :GET LAST TRACK
MOVB #0,GENDPB+$SEC  :GET STARTING SECTOR OF 16 BIT MFG FILE
MOV #-256,GENDPB+$WCNT :ONE SECTOR WORD COUNT
MOVB #RDDAT,GENDPB+$COMND :READ DATA COMMAND
MOV #8,$CDW2       :GET LAST SECTOR OF 16 BIT MFG FILE
1$: MOV #CYLNDR,R3  :GET READ BUFFER ADDRESS
JSR R0,RM80       :READ CURRENT SECTOR
GENDPB
BR 1$            :WAIT FOR QUE
2$: TST GENDPB+$STATUS :READ DONE YET ?
BEQ 2$          :BR IF NO
BPL 3$          :BR IF NO ERROR, ELSE
ADD #2,GENDPB+$SEC :INCREMENT NEXT SECTOR TO READ
CMPB $CDW2,GENDPB+$SEC :WERE ALL SECTORS TRIED ?
BHIS 1$         :BR IF NO
  
```

```

027650
027650 010146
027652 010246
027654 010346
34 027656 004737 027462
35 027662 010001
36 027664 062701 000146
37 027670 010146
38 027672 111037 067556
39 027676 012737 001056 067570
40 027704 113737 001426 067567
41 027712 112737 000000 067566
42 027720 012737 177400 067562
43 027726 112737 000171 067560
44 027734 012737 000010 001270
45 027742 012703 101174
46 027746 004037 041000
47 027752 067556
48 027754 000772
49 027756 005737 067574
50 027762 001775
51 027764 100010
52 027766 062737 000002 067566
53 027774 123737 001270 067566
54 030002 103357
  
```

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

030246 012602
030250 012601
108 030252 000205

MOV (SP)+,R2
MOV (SP)+,R1
RTS R5

::POP STACK INTO R2
::POP STACK INTO R1
;EXIT

```
1      .SBTTL  ENTER BAD SECTOR ROUTINE
2
3      :ROUTINE TO ENTER BAD SECTOR INFORMATION MANUALLY
4      :CALL:
5          MOV      #DPB,RO      ;DPB ADDRESS
6          JSR      PC,MANTER    ;CALL ROUTINE
7
8      :RO = DPB ADDRESS BEFORE CALLING THE ROUTINE
9
10     MANTER:
11     030254      010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
12     030254      010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
13     030260      010346      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
14     030262      010446      MOV      R4,-(SP)      ;;PUSH R4 ON STACK
15     030264      105737      TSTB     $AUTOB      ;RUNNING IN AUTO MODE ?
16     030270      001162      BNE     19$          ;BRANCH IF SO
17     030272      005737      TST     DRVPAR      ;CHANGE DRIVE PARAMETERS ?
18     030276      001157      BNE     19$          ;BR IF NO
19     030300      005037      1$:    CLR     CFLAG      ;CLEAR THE CONTROL-C FLAG
20     030304      104401      TYPE    ,ENTADR     ;MESSAGE TO ENTER...
21     030310      012704      MOV     #SBDSEC,R4  ;INDEX VALUE OF TABLE ADDRESS
22     030314      060004      ADD     R0,R4       ;TABLE STARTING ADDRESS
23     030316      012701      MOV     #<126.*2>,R1 ;256. TOTAL BAD SECTORS ALLOWED
24     030322      022714      2$:    CMP     #-1,(R4)  ;ENTRY IN THE TABLE ?
25     030326      001407      BEQ     3$          ;BRANCH IF SO
26     030330      062704      ADD     #4,R4       ;ADJUST THE TABLE ENTRY POINTER
27     030334      005301      DEC     R1          ;DECREMENT THE BAD SECTOR COUNT
28     030336      001371      BNE     2$          ;BR IF TO NEXT ENTRIES POSITION
29     030340      104401      TYPE    ,MSFULL    ;TYPE 'BAD SECTOR TABLE IS FULL'
30     030344      000534      BR     19$         ;EXIT..
31     030346      010146      3$:    MOV     R1,-(SP)  ;SAVE THE COUNTER AND FIRST
32     030350      010446      MOV     R4,-(SP)  ;ENTRY POINTER PAIR
33     030352      012714      4$:    MOV     #-1,(R4) ;RESET CYLINDER TO -1
34     030356      012764      MOV     #-1,2(R4) ;RESET TRACK/SECTOR FIELD TO -1
35     030364      104401      TYPE    ,MSGCTS   ;TYPE 'CYL,TRK,SEC = '
36     030370      104411      RDLIN   ;READ THE ADDRESS
37     030372      012601      MOV     (SP)+,R1  ;READ IN TEXT ADDRESS
38     030374      005737      TST     CFLAG     ;CONTROL-C ENTERED ?
39     030400      001011      BNE     5$          ;BRANCH IF YES
40     030402      105761      TSTB     1(R1)    ;WAS IT TERMINATED WITH CARRIAGE RETURN ?
41     030406      001021      BNE     7$          ;BR IF NO
42     030410      122711      CMPB    #'L',(R1) ;WAS CHARACTER AN 'L' ?
43     030414      001016      BNE     7$          ;BR IF NO
44     030416      004737      JSR     PC,TYLIST ;TYPE BAD SECTOR LIST FOR USER
45     030422      000753      BR     4$
46     030424      012604      5$:    MOV     (SP)+,R4  ;RETRIEVE THE ENTRY POINTER
47     030426      012601      MOV     (SP)+,R1  ;RETRIEVE THE COUNT
48     030430      012724      6$:    MOV     #-1,(R4)+ ;RESET THE TABLE
49     030434      012724      MOV     #-1,(R4)+ ;TO -1
50     030440      005301      DEC     R1        ;ALL DONE ?
51     030442      001372      BNE     6$        ;BRANCH IF NOT
52     030444      104401      TYPE    ,ALOST    ;TYPE ' * ALL CURRENT ENTRIES LOST * '
53     030450      000713      BR     1$        ;ENTER AGAIN FROM THE FIRST POINTER
54     030452      000002      7$:    MOV     000002,R1 ;
55
```

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

.SBTTL TYPE BAD SECTOR LIST

```

85 :ROUTINE TO LIST BAD SECTORS ON THE TERMINAL IN DECIMAL NUMBERS
86 :FORMAT IS: CYL,TRK,SEC
87 :CALL :
88 :       MOV      #DPB,RO      :DPB ADDRESS
89 :       JSR      PC,TYLIST    :CALL ROUTINE
90 :
91 :RO = DPB ADDRESS BEFORE CALLING THE ROUTINE
92
93 TYLIST:
94 030650 010146      MOV      R1,-(SP)      ::PUSH R1 ON STACK
95 030652 104401 077216 TYPE      ,LSTHDR      :TYPE 'DEC144 AND MANUAL BAD SECTOR LIST'
96 030656 012701 000146 MOV      #SBDSEC,R1    :INDEX VALUE OF TABLE ADDRESS
97 030662 060001      ADD      RO,R1         :TABLE STARTING ADDRESS
98 030664 010146      MOV      R1,-(SP)     :SAVE ADDRESS FOR LATER
99 030672 001444      BEQ      #-1,(R1)     :TERMINATOR OR NO ENTRY IN THE TABLE ?
100 030674 011146     1$:      MOV      (R1),-(SP)    :BRANCH IF YES
101 030676 004737 033230 JSR      PC,$SB2D      :GET CYLINDER NUMBER
102 030702 004737 032264 JSR      PC,SUPRSL     :CONVERT NUMBER
103 030706 005046      CLR      -(SP)        :LEFT JUSTIFY AND TYPE
104 030710 116116 000003 MOV      3(R1),(SP)    :CLEAR HI BYTE AND PUSH STACK
105 030714 100407      BMI      2$          :GET TRACK NUMBER
106 030716 104401 C'5276      TYPE      ,COMMA      :BR IF ALL BAD
107 030722 004737 033230 JSR      PC,$SB2D      :TYPE ','
108 030726 004737 032264 JSR      PC,SUPRSL     :CONVERT NUMBER
109 030732 000401      BR      3$          :LEFT JUSTIFY AND TYPE
110 030734 005726     2$:      TST      (SP)+        :RESTORE STACK
111 030736 005046     3$:      CLR      -(SP)        :CLEAR HI BYTE AND PUSH STACK
112 030740 116116 000002 MOV      2(R1),(SP)    :GET SECTOR NUMBER
113 030744 100407      BMI      4$          :BR IF ALL BAD
114 030746 104401 075276 TYPE      ,COMMA      :TYPE ','
115 030752 004737 033230 JSR      PC,$SB2D      :CONVERT NUMBER
116 030756 004737 032264 JSR      PC,SUPRSL     :LEFT JUSTIFY AND TYPE
117 030762 000401      BR      5$          :RESTORE STACK
118 030764 005726     4$:      TST      (SP)+        :CR-LF
119 030766 104401 001203     5$:      TYPE      ,SCLF      :INCREMENT POINTER
120 030772 062701 000004 ADD      #4,R1         :CONTROL-C ENTERED ?
121 030776 005737 001340 TST      CFLAG        :BRANCH IF NO
122 031002 001731     6$:      BEQ      1$          :ANY ENTRIES ?
123 031004 022601      CMP      (SP)+,R1    :BR IF YES
124 031006 001002      BNE      7$          :TYPE 'NO ENTRIES'
125 031010 104401 077176     7$:      TYPE      ,NOENTY   :CR-LF
126 031014 104401 001203      TYPE      ,SCLF      :CLEAR CONTROL FLAG
127 031020 005037 001340 CLR      CFLAG        :POP STACK INTO R1
128 031024 012601      MOV      (SP)+,R1    :RETURN
129 031026 000207      RTS      PC
  
```

1				.SBTTL	PARAMETER ENTRY ROUTINE	
2						
3				:	PARAMETER ENTRY ROUTINE	
4				:	CALL:	
5				:	MOV #ADR,R3	:PARAMETER TABLE ADDRESS
6				:	JSR PC,PARENT	:GET THE PARAMETERS
7						
8	031030	010346		PARENT:	MOV R3,-(SP)	:SAVE THE PARAMETER TABLE ADDRESS
9	031032	005037	001340		CLR CFLAG	:CLEAR THE 'CONTROL C' FLAG
10	031036	012337	031046	1\$:	MOV (R3)+,3\$:ADDRESS OF PARAMETER NAME
11	031042	001442			BEQ 9\$:BR IF AT END OF TABLE
12	031044	104401			TYPE	:TYPE THE PARAMETER NAME
13	031046	000000		3\$:	.WORD 0	:ADDRESS OF PARAMETER NAME TEXT
14	031050	012302			MOV (R3)+,R2	:MAXIMUM PARAMETER VALUE
15	031052	012305			MOV (R3)+,R5	:ADDRESS OF PARAMETER
16	031054	011546			MOV (R5),-(SP)	:CURRENT VALUE OF PARAMETER
17	031056	104405			TYPDS	:TYPE THE CURRENT VALUE OF THE PARAMETER
18	031060	104401	077572		TYPE .SLASH	: ' / '
19	031064	104411			RDLIN	:READ THE KEYBOARD
20	031066	012601			MOV (SP)+,R1	:INPUT ASCII STRING ADDRESS
21	031070	005737	001340		TST CFLAG	: 'CONTROL C' ENTERED ?
22	031074	001021			BNE 8\$:BR IF IT WAS
23	031076	004537	033072		JSR R5,CK.DIG	:CHECK THE DIGIT(S)
	031102	031036			1\$:CARRIAGE RETURN ONLY ENTERED
	031104	031150			9\$:PERIOD ONLY ENTERED
	031106	031122			6\$:ILLEGAL INPUT
	031110	031116			5\$:TERMINATED WITH A CARRIAGE RETURN
	031112	031122			6\$:TERMINATED WITH A '..'
	031114	031134			7\$:TERMINATED WITH A '...'
24	031116	010215		5\$:	MOV R2,(R5)	:MOVE NEW VALUE TO PARAMETER LOCATION
25	031120	000746			BR 1\$:GET MORE PARAMETERS
26	031122	104401	076341	6\$:	TYPE .BADENT	: 'BAD ENTRY'
27	031126	162703	000006		SUB #6,R3	:DECREMENT THE TABLE POINTER
28	031132	000741			BR 1\$:TRY AGAIN
29	031134	010215		7\$:	MOV R2,(R5)	:NEW VALUE
30	031136	000404			BR 9\$:EXIT
31	031140	005037	001340	8\$:	CLR CFLAG	:CLEAR THE 'CONTROL C' FLAG
32	031144	011603			MOV (SP),R3	:RELOAD THE PARAMETER TABLE ADDRESS
33	031146	000733			BR 1\$:TRY AGAIN
34	031150	005726		9\$:	TST (SP)+	:CORRECT THE STACK POINTER
35	031152	000207			RTS PC	:RETURN


```
1 :TYPEOUT ASSIGN/DEASSIGN ERROR MESSAGE
2 :CALL:
3 :     MOV     #MESADR,ASNMSG ;ERROR MESSAGE ADDRESS
4 :     JSR     PC,ASNERR
5 :     RETURN
6
7 031154 104401 001203   ASNERR: TYPE     ,SCLF           ;CR-LF
8 031150 104401 075272   TYPE     ,QUES           ; '?'
9 031164 104401 075304   TYPE     ,UNMSG          ;TYPE 'DRIVE'
10 031170 010446         MOV     R4,-(SP)         ;;SAVE R4 FOR TYPEOUT
                            ;;TYPE DRIVE NUMBER
                            ;;GO TYPE--OCTAL ASCII
031172 104403         TYPOS
031174     002         .BYTE     2           ;;TYPE 2 DIGIT(S)
031175     000         .BYTE     0           ;;SUPPRESS LEADING ZEROS
11 031176 104401         TYPE
12 031200 000000   ASNMSG: .WORD     0           ;TYPE SPECIFIC MESSAGE
13 031202 000207   RTS     PC             ;MESSAGE ADDRESS
14
15 :DEASSIGN DRIVE IF A FATAL ERROR OCCURS
16 :CALL:
17 :     JSR     PC,DROP
18 :     RETURN
19
20 031204 005004   DROP:  CLR     R4           ;CLEAR R4 FOR DRIVE NUMBER
21 031206 111004   MOVB    (R0),R4         ;MOVE DRIVE NUMBER TO R4
22 031210 146437 040300 001542   BICB    ATABIT(R4),ASNLST ;REMOVE DRIVE FROM ASSIGNED LIST
23 031216 146437 040300 032012   BICB    ATABIT(R4),AUTLST ;DELETE DRIVE FROM AUTO ASSIGN LIST
24 031224 006304   ASL     R4           ;MAKE DRIVE NUMBER INTO A TABLE INDEX
25 031226 010064 001544   MOV     R0,DDRVS(R4)   ;PUT DRIVE IN DROP LIST
26 031232 104401 001203   TYPE     ,SCLF           ;TYPE 'FATAL OR EXCESSIVE ERRORS'
27 031236 104401 075700   TYPE     ,DROPNNG        ;TYPE 'ON'
28 031242 104401 075753   TYPE     ,MSGON          ;TYPE 'ON'
29 031246 104401 075304   TYPE     ,UNMSG          ;TYPE 'DRIVE'
30 031252 006204   ASR     R4           ;DRIVE NUMBER
31 031254 010446   MOV     R4,-(SP)         ;;SAVE R4 FOR TYPEOUT
                            ;;TYPE DRIVE NUMBER
                            ;;GO TYPE--OCTAL ASCII
031256 104403         TYPOS
031260     002         .BYTE     2           ;;TYPE 2 DIGIT(S)
031261     000         .BYTE     0           ;;SUPPRESS LEADING ZEROS
32 031262 000207   1$:    RTS     PC
33
34 :ROUTINE TO DEASSIGN DRIVE IF ERRORS BECOMES EXCESSIVE
35
36 031264 032777 000020 147662   ABNRML: BIT     #SW04,@SWR ;SEE IF SWITCH 4 SET
37 031272 001006         BNE     1$           ;B) IF IT'S SET
38 031274 023760 001456 000072   CMP     MAXER,$TOTAL(R0) ;CHECK TOTAL ERROR VALUE
39 031302 101002         BHI     1$           ;BR IF ERRORS DO NOT EXCEED MAX
40 031304 000137 031204         JMP     DROP          ;DEASSIGN THE DRIVE
41 031310 000207   1$:    RTS     PC             ;RETURN
42
43 :ROUTINE TO CHECK FOR END OF PASS AND END OF TEST
44
```

1

.SBTTL END OF PASS ROUTINE

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

```

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

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

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

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

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

031542	002			.BYTE	2	::TYPE 2 DIGIT(S)
031543	000			.BYTE	0	::SUPPRESS LEADING ZEROS
031544	146437	040300	001542	BICB	ATABIT(R4),ASNLST	::DELETE DRIVE FROM ASSIGNED LIST
031552	006304			ASL	R4	::MAKE DRIVE NUMBER INTO TABLE INDEX
031554	010064	001544		MOV	R0,DDRV5(R4)	::PUT BLOCK ADDRESS INTO DROP LIST
031560	105737	001542		TSTB	ASNLST	::ALL DRIVES ARE DEASSIGNED ?
031564	001041			BNE	7\$::BR IF NO
031566	005237	001216		INC	\$DEVCT	::INCREMENT DEVICE COUNT
031572	005237	001214		INC	\$PASS	::INCREMENT THE PASS COUNT
031576	042737	100000	001214	BIC	#100000,\$PASS	::AVOID NEGATIVE NUMBER
031604	000431			BR	7\$	
031606	032777	000400	147340	3\$: BIT	#SW08,@SWR	::INHIBIT END OF PASS TYPEOUT ?
031614	001002			BNE	4\$::BR IF YES
031616	004737	023714		JSR	PC,SUMARY	::TYPE THE DRIVE'S STATISTICS SUMMARY
031622	010346			4\$: MOV	R3,-(SP)	::SAVE R3
031624	010004			MOV	R0,R4	::DRIVE'S BLOCK ADDRESS
031626	062704	000036		ADD	#SENDAT,R4	::ADD THE STARTING ADDR OF SECTIONS TO CLEAR
031632	012703	000006		MOV	#6,R3	::NUMBER OF LOCNS TO BE CLEARED
						::(CLEAR SENDAT, SENDSK AND \$OPERC COUNTERS)
031636	005024			5\$: CLR	(R4)+	::CLEAR THE LOCN
031640	005303			DEC	R3	::DECREMENT THE LOCATION COUNTER
031642	001375			BNE	5\$::BR IF MORE TO GO
031644	012603			MOV	(SP)+,R3	::RESTORE R3
031646	005260	000104		INC	\$PASSC(R0)	::INCREMENT THE PASS COUNT
031652	105737	001150		6\$: TSTB	\$AUTOB	::RUNNING IN AUTO MODE ?
031656	001404			BEQ	7\$::BR IF NO
031660	023737	0015'2	032012	CMP	ASNLST,AUTLST	::HAVE ALL DRIVES COMPLETED PASS IN AUTO MODE ?
031666	001402			BEQ	8\$::BR IF YES
031670	012604			7\$: MOV	(SP)+,R4	::RESTORE R4
031672	000207			RTS	PC	::RETURN
031674	005237	032012		8\$: INC	AUTLST	::CLEAR AUTO ASSIGN LIST FOR NEXT PASS AND
031700	001375			BNE	8\$::WAIT FOR TTY
031702	012737	000000	177776	MOV	#PRO,PS	::ALLOW INTERRUPTS
031710	005237	001216		INC	\$DEVCT	::INCREMENT DEVICE COUNT
031714	005237	001214		INC	\$PASS	::INCREMENT THE PASS NUMBER
031720	042737	100000	001214	BIC	#100000,\$PASS	::DON'T ALLOW A NEG. NUMBER
031726	005327			DEC	(PC)+	::LOOP?
031730	000001			\$EOPCT: .WORD	1	
031732	003013			BGT	\$DOAGN	::YES
031734	012737			MOV	(PC)+,@(PC)+	::RESTORE COUNTER
031736	000001			\$ENDCT: .WORD	1	
031740	031730			\$EOPCT		
031742	013700	000042		\$GET42: MOV	@#42,R0	::GET MONITOR ADDRESS
031746	001405			BEQ	\$DOAGN	::BRANCH IF NO MONITOR
031750	000005			RESET		::CLEAR THE WORLD
031752	004710			\$ENDAD: JSR	PC,(R0)	::GO TO MONITOR
031754	000240			NOP		::SAVE ROOM
031756	000240			NOP		::FOR
031760	000240			NOP		::ACT11
031762				\$DOAGN:		
031762	000137			JMP	@(PC)+	::RETURN
031764	031766			\$RTNAD: .WORD	RTURN	
2						
3	031766	012706	001100	RTURN: MOV	#STACK,SP	::RESTORE STACK

4 031772 005237 001212
5 031776 004737 033326
6 032002 004737 023422
7 032006 000137 006240
8
9 032012 000000

INC \$TESTN
JSR PC,\$TKINT
JSR PC,CKCLK
JMP MAIN

;INCREMENT THE TEST NUMBER IN THE MAIL BOX
;MAKE SURE KEYBOARD INTERRUPT AND
;SYSTEM CLOCK ARE STILL ON.
;RETURN TO LOOP

AUTLST: .WORD 0

;AUTO ASSIGN LIST (USED IN AUTO RUN MODE)

```

1
2
3
4
5
6
7
8 032014 013746 037124
9 032020 013746 037122
10 032024 010546
11 032026 004737 032040
12 032032 012605
13 032034 005726
14 032036 000207
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 032040 104412
40 032042 016605 000026
41 032046 005004
42 032050 016602 000030
43 032054 016603 000032
44 032060 005000
45 032062 005001
46 032064 004737 032106
47 032070 010166 000030
48 032074 010366 000032
49 032100 104413
50 032102 012616
51 032104 000207
52
53
54
55
56
57

;ROUTINE TO GET THE REMAINDER OF THE RANDOM NUMBER
;CALL:
;:
;:      MOV      NUMBER,R5      ;DIVISOR INTO R5
;:      JSR      PC,GETREM
;:      RETURN
;:      ;REMAINDER IS IN R5

GETREM: MOV      $LONUM,-(SP)    ;STORE RANDOM NUMBER ON THE STACK FOR DIVIDE
;:      MOV      $HINUM,-(SP)    ;UPPER PART
;:      MOV      R5,-(SP)        ;PUT THE DIVISOR ONTO THE STACK
;:      JSR      PC,$DIV         ;DIVIDE THE RANDOM NUMBERS
;:      MOV      (SP)+,R5        ;PUT THE REMAINDER INTO R5
;:      TST      (SP)+          ;ADJUST THE STACK POINTER
;:      RTS      PC

.SBTTL  INTEGER DIVIDE ROUTINE

;*****
;*THIS ROUTINE WILL DIVIDE A 32-BIT TWO'S COMPLEMENT INTEGER
;*DIVIDEND BY A 16-BIT TWO'S COMPLEMENT INTEGER DIVISOR GIVING
;*A 16-BIT TWO'S COMPLEMENT INTEGER QUOTIENT AND A 16-BIT REMAINDER.
;*DIVISION WILL BE PERFORMED SO THAT THE REMAINDER IS OF THE
;*SAME SIGN AS THE DIVIDEND.
;*CALL:
;*      MOV      LOW DIVIDEND,-(SP)    ;;THE HIGH DIVIDEND MUST BE < 1/2
;*      MOV      HIGH DIVIDEND,-(SP)  ;;AS LARGE AS THE DIVISOR
;*      MOV      DIVISOR,-(SP)
;*      JSR      PC,$DIV
;*      RETURN                          ;;QUOTIENT & REMAINDER ARE ON THE STACK
;*
;*      STACK  NO ERROR      OVERFLOW      DIVIDE BY ZERO
;*      -----
;*      TOP    REMAINDER     ALL ZEROS     ALL ONES
;*      +2    QUOTIENT       ALL ZEROS     ALL ONES
;*
;*NOTE: THIS ROUTINE WILL LINK TO THE DIVISION SUBROUTINE ('M.DPID').

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

.SBTTL  DOUBLE PRECISION DIVISION SUBROUTINE

;CALL:
;:      JSR      PC,M.DPID
;:

```

```

58      :      DIVIDEND = R0-R1-R2-R3 (R0=MSD)
59      :      DIVISOR = R4-R5 (R4=MSD)
60      :
61      :RETURN
62      :
63      :      REMAINDER AFTER DIVISION = R0-R1 (R0=MSD)
64      :      QUOTIENT AFTER DIVISION = R2-R3 (R2=MSD)
65
66 032106 012746 000040      M.DPID: MOV      #40,-(SP)      :COUNTER FOR DIVISION CYCLES
67 032112 010446              MOV      R4,-(SP)      :HIGH ORDER
68 032114 010546              MOV      R5,-(SP)      :LOW ORDER DIVISOR TO THE STACK
69 032116 005466 000002      NEG      2(SP)      :FORM NEGATIVE
70 032122 005416              NEG      @SP          :VERSION OF THE DIVISOR
71 032124 005666 000002      SBC      2(SP)
72 032130 061601              ADD      @SP,R1
73 032132 005500              ADC      R0          :PERFORM THE INITIAL SUBTRACTION
74 032134 066600 000002      ADD      2(SP),R0
75 032140 103445              BCS      M.DP50      :IF CARRY THEN OVERFLOW HAS OCCURRED
76 032142 005046              CLR      -(SP)      :THIS IS A LONGER LASTING CARRY BIT
77 032144 006103      M.DP40: ROL      R3
78 032146 006102              ROL      R2
79 032150 006101              ROL      R1
80 032152 006100              ROL      R0
81 032154 005716              TST      @SP          :TEST "CARRY" INDICATOR
82 032156 001410              BEQ      M.DP41      :IF NO "CARRY" THEN ADD ELSE SUBTRACT
83 032160 005016              CLR      @SP          :CLEAR UP FOR NEXT TIME
84 032162 066601 000002      ADD      2(SP),R1
85 032166 005500              ADC      R0          :ADD -(DIVISOR)
86 032170 005516              ADC      @SP          :SET "CARRY"
87 032172 066600 000004      ADD      4(SP),R0; <- I
88 032176 000404              BR      M.DP42
89 032200 060501      M.DP41: ADD      R5,R1
90 032202 005500              ADC      R0          :ADD +(DIVISOR)
91 032204 005516              ADC      @SP          :SET "CARRY"
92 032206 060400              ADD      R4,R0      :<- I
93 032210 005516      M.DP42: ADC      @SP          :SET "CARRY"
94 032212 005716              TST      @SP          :TEST THE UPDATE INDICATOR
95 032214 001401              BEQ      .+4          :IF ZERO FORGET IT
96 032216 005203              INC      R3          :NO CARRY POSSIBLE HERE
97 032220 005366 000006      DEC      6(SP)      :DECREMENT COUNTER
98 032224 003347              BGT      M.DP40      :BRANCH IF MORE TO DO
99 032226 006003              ROR      R3
100 032230 103404              BCS      M.DP44
101 032232 060501              ADD      R5,R1
102 032234 005500              ADC      R0
103 032236 060400              ADD      R4,R0
104 032240 000241              CLC
105 032242 006103      M.DP44: ROL      R3
106 032244 062706 000010      ADD      #10,SP      :ADJUST STACK BY 4 WORDS
107 032250 000242              CLV
108 032252 000207              RTS      PC
109 032254 062706 000006      M.DP50: ADD      #6,SP
110 032260 000262              SEV
111 032262 000207              RTS      PC
    
```

```

1      .SBTTL SUPRS - TYPE ASCIZ, REPLACE LEADING 0'S WITH BLANKS
2      .SBTTL SUPRSL -TYPE ASCIZ, LEFT JUSTIFY
3
4      :*****
5      :CALL:
6      :      MOV      #NUMADR, -(SP)      ;FIRST ADDRESS OF ASCIZ STRING
7      :      JSR      PC, SUPRS
8      :      OR
9      :      MOV      #NUMADR, -(SP)      ;FIRST ADDRESS OF ASCIZ STRING
10     :      JSR      PC, SUPRSL
11
12     032264 010046 SUPRSL: MOV      R0, -(SP)      ;SAVE R0
13     032266 016600 000004 MOV      4(SP), R0      ;GET POINTER TO MESSAGE
14     032272 005037 032354 CLR      SUPR2
15     032276 000405 BR       SUPR1
16
17     032300 010046 SUPRS:  MOV      R0, -(SP)      ;SAVE R0
18     032302 016600 000004 MOV      4(SP), R0      ;GET POINTER TO MESSAGE
19     032306 010037 032354 MOV      R0, SUPR2      ;GET POINTER FOR TYPING
20     032312
21     032312 105710 SUPR1:  1$:   TSTB     (R0)      ;TEST FOR TERMINATOR
22     032314 001406 BEQ      2$           ;YES
23     032316 122710 000060 CMPB     #'0', (R0)    ;IS THIS A '0' ?
24     032322 001006 BNE      3$           ;NO
25     032324 112720 000040 MOVB    #40, (R0)+    ;REPLACE IT WITH A 'BLANK'
26     032330 000770 BR       1$           ;NEXT CHAR.
27     032332 005300 2$:   DEC      R0          ;BACKUP 1
28     032334 112710 000060 MOVB    #'0', (R0)    ;MAKE IT '0'
29     032340 005737 032354 3$:   TST      SUPR2      ;LEFT JUSTIFY ?
30     032344 001002 BNE      4$           ;NO
31     032346 010037 032354 MOV      R0, SUPR2    ;YES
32     032352 104401 4$:   TYPE     .WORD     0
33     032354 000000 SUPR2:  MOV      (SP)+, R0      ;RESTORE R0
34     032356 012600 MOV      (SP)+, (SP)   ;RESTORE STACK
35     032360 012616 RTS
36     032362 000207 PC
  
```

CZRMAAO RM80 PERF EXER MACRO V04.00 14-JAN-82 15:16:58 PAGE 38
\$SUPRS - TYPE ASCIZ, REPLACE LEADING 0'S WITH BLANKS

1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16	032364	010046	
17	032366	016600	000004
18	032372	005037	032454
19	032376	000405	
20			
21	032400	010046	
22	032402	016600	000004
23	032406	010037	032454
24	032412		
25	032412	105710	
26	032414	001406	
27	032416	122710	000060
28	032422	001006	
29	032424	112720	000040
30	032430	000770	
31	032432	005300	
32	032434	112710	000060
33	032440	005737	032454
34	032444	001002	
35	032446	010037	032454
36	032452	104414	
37	032454	000000	
38	032456	012600	
39	032460	012616	
40	032462	000207	

```

.SBTTL $SUPRS - TYPE ASCIZ, REPLACE LEADING 0'S WITH BLANKS
.SBTTL $SUPRL - TYPE ASCIZ, LEFT JUSTIFY
:*****
:THIS ROUTINE IS SAME AS 'SUPRSL' AND 'SUPRS', EXCEPT THAT IT
:WILL SUPPRESS THE ERROR TYPEOUT IF SW13=1. THIS ACCOMPLISHED BY
:USED TRAP CALL 'DISPLY', INSTEAD OF 'TYPE'.
:CALL:
:      MOV    #NUMADR, -(SP)    ;FIRST ADDRESS OF ASCIZ STRING
:      JSR    PC, $SUPRS
:
:      OR
:      MOV    #NUMADR, -(SP)    ;FIRST ADDRESS OF ASCIZ STRING
:      JSR    PC, $SUPRL
:
:$SUPRL: MOV    R0, -(SP)        ;SAVE R0
:        MOV    4(SP), R0       ;GET POINTER TO MESSAGE
:        CLR    $SUPR2
:        BR     $SUPR1
:
:$SUPRS: MOV    R0, -(SP)        ;SAVE R0
:        MOV    4(SP), R0       ;GET POINTER TO MESSAGE
:        MOV    R0, $SUPR2      ;GET POINTER FOR TYPING
:$SUPR1:
1$:     TSTB   (R0)              ;TEST FOR TERMINATOR
:        BEQ   2$               ;YES
:        CMPB  #'0', (R0)        ;IS THIS A '0' ?
:        BNE   3$               ;NO
:        MOVB #40, (R0)+         ;REPLACE IT WITH A 'BLANK'
:        BR    1$              ;NEXT CHAR.
2$:     DEC   R0                 ;BACKUP 1
:        MOVB #'0', (R0)        ;MAKE IT '0'
3$:     TST   $SUPR2             ;LEFT JUSTIFY ?
:        BNE   4$               ;NO
:        MOV   R0, $SUPR2        ;YES
4$:     DISPLY ;TYPE, UNLESS SW13=1
:$SUPR2: .WORD 0
:        MOV   (SP)+, R0         ;RESTORE R0
:        MOV   (SP)+, (SP)      ;RESTORE STACK
:        RTS   PC

```



```

1      ;ROUTINE TO REPLACE LEADING ZEROS IN A NUMERIC STRING WITH SPACES
2      :CALL:
3      :
4      :
5      :
6      :
7      :
8      :
9      :
10     :
11     032464 005237 032572      FILLZ: INC      FILL0      ;LEAVE ZERO'S
12     :
13     032470 010046      REPLZ: MOV      R0,-(SP)      ;SAVE R0
14     032472 016600 000004      MOV      4(SP),R0      ;ADDRESS OF NUMBER TO R0
15     032476 005737 032572      TST      FILL0      ;LEAVE PRECEDING ZEROS ?
16     032502 001014      BNE      3$      ;BR IF YES
17     032504 122710 000060      1$: CMPB     #'0,(R0)      ;BYTE EQUAL TO ASCII '0' ?
18     032510 001004      BNE      2$      ;BR IF NOT
19     032512 112710 000040      MOVB     #40,(R0)      ;REPLACE THE ZERO WITH A SPACE
20     032516 005200      INC      R0      ;INCREMENT THE BYTE ADDRESS
21     032520 000771      BR       1$      ;GO BACK AND LOOK FOR MORE LEADING ZEROS
22     032522 105710      2$: TSTB     (R0)      ;SEE IF ZERO BYTE TERMINATOR
23     032524 001003      BNE      3$      ;BR IF NOT
24     032526 005300      DEC      R0      ;BACKUP STRING POINTER
25     032530 112710 000060      MOVB     #'0,(R0)      ;PUT A ZERO BACK IN
26     032534 016600 000004      3$: MOV      4(SP),R0      ;PUT ADDRESS OF FIRST CHARACTER ON STACK
27     032540 105720      4$: TSTB     (R0)+      ;SEE IF ZERO BYTE TERMINATOR
28     032542 001376      BNE      4$      ;BR IF NOT
29     032544 005300      DEC      R0      ;BACKUP STRING POINTER
30     032546 162500      SUB      (R5)+,R0      ;ADJUST ADDRESS
31     032550 010037 032556      MOV      R0,5$      ;GET ADDRESS FOR TYPEOUT
32     032554 104401      TYPE     ;TYPE THE NUMBER
33     032556 000000      5$: .WORD    0      ;ADDRESS OF NUMBER
34     032560 012600      MOV      (SP)+,R0      ;POP STACK INTO R0
35     032562 012616      MOV      (SP)+,(SP)      ;RESTORE STACK
36     032564 005037 032572      CLR      FILL0      ;RESET FILL FLAG
37     032570 000205      RTS      R5      ;RETURN
38     :
39     032572 000000      FILL0: .WORD    0      ;IF SET, LEAVE PRECEDING ZEROS FOR TYPE
40     :
41     ;ROUTINE TO TYPE AT PRIORITY 4
42     :
43     032574 013746 177776      TYPRI4: MOV     @MPS,-(SP)      ;SAVE THE PRESENT STATUS
44     032600 012737 000200 177776      MOV     #200,@MPS      ;CHANGE THE PRIORITY TO 4
45     032606 012537 032616      MOV     (R5)+,1$      ;MESSAGE ADDRESS
46     032612 004737 035550      JSR     PC,$TYPE      ;TYPE THE MESSAGE
47     032616 000000      1$: .WORD    0      ;MESSAGE ADDRESS GOES HERE
48     032620 000205      RTS     R5      ;RETURN
49     :
50     ;ROUTINE TO TYPE THE DRIVE (MBA) SERIAL NUMBER IN DECIMAL
51     :CALL:
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     :
93     :
94     :
95     :
96     :
97     :
98     :
99     :
100    :
101    :
102    :
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    :
160    :
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    :
217    :
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    :
274    :
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    :
330    :
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    :
388    :
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    :
445    :
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    :
503    :
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    :
561    :
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    :
619    :
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    :
677    :
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    :
735    :
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    :
793    :
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    :
819    :
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    :
872    :
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    :
930    :
931    :
932    :
933    :
934    :
935    :
936    :
937    :
938    :
939    :
940    :
941    :
942    :
943    :
944    :
945    :
946    :
947    :
948    :
949    :
950    :
951    :
952    :
953    :
954    :
955    :
956    :
957    :
958    :
959    :
960    :
961    :
962    :
963    :
964    :
965    :
966    :
967    :
968    :
969    :
970    :
971    :
972    :
973    :
974    :
975    :
976    :
977    :
978    :
979    :
980    :
981    :
982    :
983    :
984    :
985    :
986    :
987    :
988    :
989    :
990    :
991    :
992    :
993    :
994    :
995    :
996    :
997    :
998    :
999    :
1000   :

```

```

58      ;RO = DPB ADDRESS BEFORE CALLING THE ROUTINE
59
60 032622 104401 077332 TYDRV: TYPE ,DRVSN ;TYPE 'DRV S/N:'
61 032626 010037 032642 TYPDRV: MOV R0,1$ ;ADDRESS OF DPB
62 032632 062737 002130 032642 ADD #SDRVSN,1$ ;INDEX TO DRIVE (MBA) SERIAL NUMBER
63 032640 104401 TYPE ;TYPE THE DRIVE (MBA) SERIAL NUMBER
64 032642 000000 1$: .WORD 0 ;ADDRESS OF DRIVE (MBA) SERIAL NUMBER FIELD
65 032644 104401 076141 TYPE ,PERIOD ;TYPE '.'
66 032650 000207 RTS PC ;RETURN
67
68 ;ROUTINE TO TYPE THE HDA SERIAL NUMBER IN DECIMAL
69 :CALL:
70 : MOV #DPB,R0 ;ADDRESS OF DRIVE PARAMETER BLOCK
71 : JSR PC,TYHDA ;CALL ROUTINE
72 :
73 : OR
74 : MOV #DPB,R0 ;ADDRESS OF DRIVE PARAMETER BLOCK
75 : JSR PC,TYPHDA ;CALL ROUTINE(WITH NO HEADER MESSAGE)
76
77 ;RO = DPB ADDRESS BEFORE CALLING THE ROUTINE
78 032652 104401 077320 TYHDA: TYPE ,HDASN ;TYPE 'HDA S/N:'
79 032656 005760 000144 TYPHDA: TST $HSNM(R0) ;IS SERIAL NUMBER VALID ?
80 032662 100003 BPL 1$ ;YES TYPE IT, ELSE
81 032664 104401 076334 TYPE ,NONE
82 032670 000207 RTS PC ;RETURN
83 032672 010046 1$: MOV R0,-(SP) ;DPB ADDRESS
84 C 2674 062716 000142 ADD #HSNLS,(SP) ;ADDRESS OF LOW NUMBER
85 032700 004737 037222 JSR PC,$DB2D ;CONVERT TO DOUBLE DECIMAL NUMBER
86 032704 004737 032264 JSR PC,SUPRSL ;AND TYPE IT LEFT JUSTIFIED
87 032710 104401 076141 TYPE ,PERIOD ;TYPE '.'
88 032714 000207 RTS PC ;RETURN
89
90 ;ROUTINE TO TYPE ERRORS
91 :CALL:
92 : DISPLY ;MUST DEFINED IN 'TRAP' TABLE
93 : MESADR ;ADDRESS OF MESSAGE
94 : RETURN
95
96 032716 032777 020000 146230 $DSPLY: BIT #BIT13,@SWR ;INHIBIT ERROR TYPEOUT ?
97 032724 001004 BNE 1$ ;BR IF YES
98 032726 005037 177776 CLR @MPS ;SET PRIORITY TO ZERO
99 032732 000137 035550 JMP $TYPE ;TYPE THE MESSAGE
100 032736 062716 000002 1$: ADD #2,(SP) ;INCREMENT THE RETURN
101 032742 000002 RTI ;RETURN
102
103 ;THIS ROUTINE IS USED TO CHECK IF AN
104 ;ASCII CHARACTER IS A DIGIT BETWEEN 0 AND 7.
105 :CALL:
106 : MOV #ADR,R1 ;ADDRESS OF ASCII CHARACTER
107 : JSR R5,CK.OCT ;CHECK THE CHARACTER
108 : RETURN1 ;CHARACTER IS NOT BETWEEN 0-7
109 : RETURN2 ;CHARACTER IS IN R2 AS A
110 : ;OCTAL DIGIT
111
112 032744 121127 000060 CK.OCT: CMPB (R1),#0 ;LESS THAN ZERO?
113 032750 103407 BLO 1$ ;YES -- BRANCH
114 032752 121127 000067 CMPB (R1),#7 ;GREATER THAN SEVEN?
  
```

```

115 032756 101004          BHI      1$          :YES -- BRANCH
116 032760 111102          MOVB    (R1),R2      :GET THE CHARACTER
117 032762 042702 177770  BIC     #'C7,R2     :STRIP AWAY THE ASCII
118 032766 005725          TST     (R5)+       :ADJUST FOR RETURN
119 032770 000205          1$:     RTS      R5          :RETURN
120
121          :THIS ROUTINE IS USED TO CHECK AN ASCII CHARACTER
122          :AND DETERMINE IF IT IS A DIGIT BETWEEN 0 AND 9.
123          :CALL:
124          :     MOV     #ADR,R1          :ADDRESS OF ASCII CHARACTER
125          :     JSR     R5,CK.DEC       :CHECK THE CHARACTER
126          :     RETURN1          :NOT BETWEEN 0 AND 9
127          :     RETURN2          :BETWEEN 0 AND 9
128          :     :                :R2 = DIGIT
129
130 032772 121127 000060  CK.DEC: CMPB    (R1),#'0      :LESS THAN ZERO?
131 032776 103407          BLO     1$          :YES -- BRANCH
132 033000 121127 000071  CMPB    (R1),#'9      :GREATER THAN NINE?
133 033004 101004          BHI     1$          :YES -- BRANCH
134 033006 111102          MOVB    (R1),R2      :GET THE CHARACTER
135 033010 042702 000060  BIC     #'0,R2       :STRIP AWAY THE ASCII
136 033014 005725          TST     (R5)+       :ADJUST FOR RETURN
137 033016 000205          1$:     RTS      R5          :RETURN
138
139          :THIS ROUTINE WILL CHECK AN ASCII CHARACTER TO
140          :DETERMINE WHAT IT IS.
141          :CALL:
142          :     MOV     #ADR,R1          :ADDRESS OF ASCII CHARACTER
143          :     JSR     R5,CK.CHR       :CHECK CHARACTER
144          :     RETURN ADR1          :UNKNOWN CHARACTER
145          :     RETURN ADR2          :CARRIAGE RETURN * (R1)=ADR+1
146          :     RETURN ADR3          :COMMA * (R1)=ADR+1
147          :     RETURN ADR4          :PERIOD * (R1)=ADR+1
148          :     RETURN ADR5          :DIGIT BETWEEN 0 AND 7.
149          :     RETURN ADR6          :DIGIT BETWEEN 8 AND 9.
150          :     :                :R2 = DIGIT * (R1)=ADR+1
151
152 033020 105711          CK.CHR: TSTB   (R1)          :'"CARRIAGE RETURN'?
153 033022 001417          BEQ     3$          :YES -- BRANCH
154 033024 121127 000054  CMPB    (R1),#',     :'"COMMA'?
155 033030 001413          BEQ     2$          :YES -- BRANCH
156 033032 121127 000056  CMPB    (R1),#'.     :'"PERIOD'?
157 033036 001407          BEQ     1$          :YES -- BRANCH
158 033040 004537 032772  JSR     R5,CK.DEC    :'"DIGIT'?
159 033044 000410          BR      4$          :NO -- BRANCH
160 033046 004537 032744  JSR     R5,CK.OCT    :OCTAL ?
161 033052 005725          TST     (R5)+       :DIGIT BETWEEN 8-9
162 033054 005725          TST     (R5)+       :DIGIT BETWEEN 0-7
163 033056 005725          1$:     TST     (R5)+       :PERIOD
164 033060 005725          2$:     TST     (R5)+       :COMMA
165 033062 005725          3$:     TST     (R5)+       :CARRIAGE RETURN
166 033064 005201          INC     R1          :MOVE POINTER TO NEXT CHARACTER
167 033066 011505          4$:     MOV     (R5),R5    :UNKNOWN CHARACTER
168 033070 000205          RTS      R5          :RETURN
169
170          :THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL
171          :CHARACTERS AND FORMS A DECIMAL VALUE BINARY NUMBER IN R2.

```

```

172                                     :CALL:
173                                     :      MOV      #ADR,R1      :ADDRESS OF ASCIZ STRING
174                                     :      MOV      #NUM,R2      :MAX. MAGNITUDE OF INPUT NUMBER
175                                     :      JSR      R5,CK.DIG     :CHECK DIGITS
176                                     :      RETURN   ADR1         :'CR' ONLY ENTERED -- R2=0
177                                     :      RETURN   ADR2         :'PERIOD' ONLY ENTERED -- R2=0
178                                     :      RETURN   ADR3         :ILLEGAL CHARACTER OR INPUT TOO LARGE -- R2=?
179                                     :      RETURN   ADR4         :'CR' -- R2 = NUMBER
180                                     :      RETURN   ADR5         :'COMMA' -- R2 = NUMBER
181                                     :      RETURN   ADR6         :'PERIOD' -- R2 = NUMBER
182
183 033072 010446      CK.DIG: MOV      R4,-(SP)      :SAVE R4
184 033074 010346      MOV      R3,-(SP)      :SAVE R3
185 033076 010246      MOV      R2,-(SP)      :SAVE THE MAX. SIZE ON THE STACK
186 033100 005002      CLR      R2              :START WITH 0
187 033102 005003      CLR      R3
188 033104 005004      CLR      R4
189 033106 004537 033020 JSR      R5,CK.CHR       :CHECK ONE CHARACTER
190 033112 033206      6$          :ILLEGAL CHARACTER
191 033114 033214      9$          :CARRIAGE RETURN
192 033116 033206      6$          :
193 033120 033210      7$          :
194 033122 033126      1$          :DIGIT 0-7
195 033124 033126      1$          :DIGIT 8-9
196 033126 062705 000004 1$: ADD      #4,R5          :STEP RETURN POINTER PAST 'CR' & 'PERIOD' RETURNS
197 033132 006303      2$: ASL      R3              :INPUT NUMBER *2
198 033134 010346      MOV      R3,-(SP)      :SAVE *2
199 033136 006303      ASL      R3              :*4
200 033140 006303      ASL      R3              :*8
201 033142 062603      ADD      (SP)+,R3          :(*2)+(*8) = *10
202 033144 060203      ADD      R2,R3          :UPDATE THE INPUT NUMBER
203 033146 004537 033020 JSR      R5,CK.CHR       :CHECK ONE CHARACTER
204 033152 033212      8$          :ILLEGAL CHARACTER
205 033154 033176      5$          :CARRIAGE RETURN
206 033156 033174      4$          :
207 033160 033166      3$          :
208 033162 033132      2$          :DIGIT 0-7
209 033164 033132      2$          :DIGIT 8-9
210 033166 105711      3$: TSTB     (R1)          :DOES A 'CR' FOLLOW THE 'PERIOD'
211 033170 001010      BNE     8$          :BR IF NOT
212 033172 005724      TST     (R4)+        :INCREMENT THE RETURN
213 033174 005724      4$: TST     (R4)+        :INCREMENT THE RETURN
214 033176 005724      5$: TST     (R4)+        :INCREMENT THE RETURN
215 033200 020316      CMP     R3,(SP)        :CHECK THE MAGNITUDE OF THE NUMBER
216 033202 101004      BHI     9$          :BR IF ENTERED NUMBER TOO LARGE
217 033204 000402      BR     8$          :BYPASS INCREMENT
218 033206 005725      6$: TST     (R5)+        :INCREMENT RETURN PAST INVALID RETURN
219 033210 005725      7$: TST     (R5)+        :INCREMENT RETURN
220 033212 060405      8$: ADD     R4,R5        :SETUP RETURN POINTER
221 033214 010302      9$: MOV     R3,R2        :ENTERED VALUE
222 033216 005726      TST     (SP)+        :CLEAN MAX. SIZE OFF OF STACK
223 033220 012603      MOV     (SP)+,R3      :RESTORE R3
224 033222 012604      MOV     (SP)+,R4      :RESTORE R4
225 033224 011505      MOV     (R5),R5      :GET RETURN ADDRESS
226 033226 000205      RTS     R5          :RETURN

```

;THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN

```
217      :UNSIGNED DECIMAL ASCIZ NUMBER.  
218      :CALL:  
219      :      MOV      NUMBER,-(SP)      :PUT THE NUMBER ON THE STACK  
220      :      JSR      PC,$SB2D          :CALL:  
221      :      RETURN                     :ADDRESS OF THE 1ST ASCIZ CHAR IS ON THE STACK  
222      :  
223      :NOTE: THE PROGRAM REQUIRES THIS FORM OF '$SB2D', NOT THE VERSION ON  
224      :      THE SYSMAC LIBRARY, REV C AND LATER  
225      :  
226 033230 016637 000002 033254 $SB2D: MOV      2(SP),1$      :SAVE THE BINARY NUMBER  
227 033236 012746 033254          MOV      #1$,-(SP)    :SET THE POINTER  
228 033242 004737 037222          JSR      PC,$DB2D     :CALL THE DOUBLE LENGTH CONVERT  
229 033246 012666 000002          MOV      (SP)+,2(SP)  :PICKUP THE POINTER  
230 033252 000207          RTS      PC           :RETURN  
231 033254 000000 000000          1$:      .WORD      0,0  
232      :  
233      :THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN  
234      :UNSIGNED OCTAL ASCIZ NUMBER.  
235      :CALL:  
236      :      MOV      NUMBER,-(SP)      :PUT THE NUMBER ON THE STACK  
237      :      JSR      PC,$SB20          :CALL:  
238      :      RETURN                     :ADDRESS OF THE 1ST ASCIZ CHAR IS ON THE STACK  
239      :  
240      :NOTE: THE PROGRAM REQUIRES THIS FORM OF '$SB20', NOT THE VERSION ON  
241      :      THE SYSMAC LIBRARY, REV C AND LATER  
242      :  
243 033260 016637 000002 033304 $SB20: MOV      2(SP),1$      :SAVE THE BINARY NUMBER  
244 033266 012746 033304          MOV      #1$,-(SP)    :SET THE POINTER  
245 033272 004737 037416          JSR      PC,$DB20     :CALL THE DOUBLE LENGTH CONVERT  
246 033276 012666 000002          MOV      (SP)+,2(SP)  :PICKUP THE POINTER  
247 033302 000207          RTS      PC           :RETURN  
248 033304 000000 000000          1$:      .WORD      0,0
```

1

.SBTTL TY INPUT ROUTINE

033310 000000
033312 000000
033314 000000
033316 033325

```
.ENABL LSB
$TKCNT: .WORD 0      ;;NUMBER OF ITEMS IN QUEUE
$TKQIN: .WORD 0      ;;INPUT POINTER
$TKQOUT: .WORD 0     ;;OUTPUT POINTER
$TKQSRT: .BLKB 7     ;;TTY KEYBOARD QUEUE
$TKQEND=.
.EVEN
```

```
;*TK INITIALIZE ROUTINE
;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
```

```
;*CALL:
;*      JSR      PC,$TKINT
;*      RETURN
```

033326 005037 033310
033332 012737 033316 033312
033340 013737 033312 033314
033346 012737 033376 000060
033354 012737 000200 000062
033362 005777 145574
033366 012777 000100 145564
033374 000207

```
$TKINT: CLR      $TKCNT      ;;CLEAR COUNT OF ITEMS IN QUEUE
        MOV      #$TKQSRT,$TKQIN ;;MOVE THE STARTING ADDRESS OF THE
        MOV      $TKQIN,$TKQOUT ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
        MOV      #$TKSRV,@$TKVEC ;;INITIALIZE THE KEYBOARD VECTOR
        MOV      #200,@$TKVEC+2 ;;'BR' LEVEL 4
        TST     @$TKB        ;;CLEAR DONE FLAG
        MOV      #100,@$TKS   ;;ENABLE TTY KEYBOARD INTERRUPT
        RTS      PC         ;;RETURN TO CALLER
```

```
;*TK SERVICE ROUTINE
;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
;*IT IN THE QUEUE.
;*IF THE CHARACTER IS A 'CONTROL-C' (^C) $TKINT IS CALLED AND
;*UPON RETURN EXIT IS MADE TO THE 'CONTROL-C' RESTART ADDRESS (CTRAP)
```

033376 117746 145560
033402 042716 177600
033406 021627 000021
033412 001002
033414 005726
033416 000002
033420
033420 021627 000003
033424 001007
033426 104401 034533
033432 004737 033326
033436 005726
033440 000137 034574
033444 021627 000007
033450 001004
033452 022737 000176 001154
033460 001500

```
$TKSRV: MOVB    @$TKB, -(SP)      ;;PICKUP THE CHARACTER
        BIC     #^C177, (SP)     ;;STRIP THE JUNK
        CMP     (SP),#$XON      ;;IS IT A RANDOM XON?
        BNE    30$              ;;BRANCH IF NO
        TST    (SP)+           ;;CLEAN RANDOM XON OFF STACK
        RTI                      ;;RETURN
```

```
30$:    CMP     (SP),#3          ;;IS IT A CONTROL C?
        BNE    1$              ;;BRANCH IF NO
        TYPE   , $CNTLC        ;;TYPE A CONTROL-C (^C)
        JSR    PC,$TKINT      ;;INIT THE KEYBOARD
        TST    (SP)+         ;;CLEAN UP STACK
        JMP    CTRAP         ;;CONTROL C RESTART
1$:    CMP     (SP),#7          ;;IS IT A CONTROL G?
        BNE    2$              ;;BRANCH IF NO
        CMP    #SWREG,SWR     ;;IS SOFT-SWR SELECTED?
        BEQ    6$              ;;GO TO SWR CHANGE
```

033462
033462 022737 000007 033310
033470 001004
033472 104401 001176

```
2$:    CMP     #7,$TKCNT      ;;IS THE QUEUE FULL?
        BNE    3$              ;;BRANCH IF NO
        TYPE   , $BELL        ;;RING THE TTY BELL
```

```

033476 005726          TST      (SP)+          ;;CLEAN CHARACTER OFF OF STACK
033500 000451          BR       5$              ;;EXIT
033502 021627 000023  3$:  CMP      (SP),#23          ;;IS IT A CONTROL-S?
033506 001021          BNE      32$              ;;BRANCH IF NO
033510 005077 145444  CLR      @STKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
033514 005726          TST      (SP)+          ;;CLEAN CHAR OFF STACK
033516 105777 145436  31$:  TSTB     @STKS           ;;WAIT FOR A CHAR
033522 100375          BPL      31$              ;;LOOP UNTIL ITS THERE
033524 117746 145432  MOVB     @STKB,-(SP)      ;;GET THE CHARACTER
033530 042716 177600  BIC      #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
033534 022627 000021  CMP      (SP)+,#21       ;;IS IT A CONTROL-Q?
033540 001366          BNE      31$              ;;BRANCH IF NO
033542 012777 000100 145410 MOV      #100,@STKS      ;;REENABLE TTY KEYBOARD INTERRUPTS
033550 000002          RTI                      ;;RETURN
033552 005237 033310  32$:  INC      $TKCNT         ;;COUNT THIS CHARACTER
033556 021627 000140  CMP      (SP),#140       ;;IS IT UPPER CASE?
033562 002405          BLT      4$              ;;BRANCH IF YES
033564 021627 000175  CMP      (SP),#175       ;;IS IT A SPECIAL CHAR?
033570 003002          BGT      4$              ;;BRANCH IF YES
033572 042716 000040  BIC      #40,(SP)        ;;MAKE IT UPPER CASE
033576 112677 177510  4$:  MOVB     (SP)+,@STKQIN  ;;AND PUT IT IN QUEUE
033602 005237 033312  INC      $TKQIN          ;;UPDATE THE POINTER
033606 023727 033312 033325  CMP      $TKQIN,$STKQEND ;;GO OFF THE END?
033614 001003          BNE      5$              ;;BRANCH IF NO
033616 012737 033316 033312  MOV      #$STKQRT,$TKQIN ;;RESET THE POINTER
033624 000002          RTI                      ;;RETURN

```

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

```

033626 022737 000176 001154  $CKSWR: CMP      #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED
033634 001124          BNE      15$              ;;EXIT IF NOT
033636 105777 145316  TSTB     @STKS           ;;IS A CHAR WAITING?
033642 100121          BPL      15$              ;;IF NOT, EXIT
033644 117746 145312  MOVB     @STKB,-(SP)      ;;YES
033650 042716 177600  BIC      #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
033654 021627 000007  CMP      (SP),#7         ;;IS IT A CONTROL-G?
033660 001300          BNE      2$              ;;IF NOT, PUT IT IN THE TTY QUEUE
                                ;;AND EXIT

```

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

```

033662 123727 001150 000001  6$:  CMPB     $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
033670 001674          BEQ      2$              ;;BRANCH IF YES
033672 005726          TST      (SP)+          ;;CLEAR CONTROL-G OFF STACK
033674 004737 033326  JSR      PC,$TKINT       ;;FLUSH THE TTY INPUT QUEUE
033700 005077 145254  CLR      @STKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
033704 112737 000001 001151  MOVB     #1,$INTAG       ;;SET INTERRUPT MODE INDICATOR

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

```

```

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

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

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

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

034036 021627 000015      10$:   CMP     (SP), #15   ;;IS IT A <CR>?
034042 001022          BNE     16$         ;;BRANCH IF NO
034044 005766 000004      TST     4(SP)       ;;YES, IS IT THE FIRST CHAR?
034050 001403          BEQ     11$         ;;BRANCH IF YES
034052 016677 000002 145074      MOV     2(SP), @SWR  ;;SAVE NEW SWR
034060 062706 000006      11$:   ADD     #6, SP      ;;CLEAR UP STACK
034064 104401 001203      14$:   TYPE    ,SCRLF     ;;ECHO <CR> AND <LF>
034070 123727 001151 000001    CMPB    $INTAG, #1   ;;RE-ENABLE TTY KBD INTERRUPTS?
034076 001003          BNE     15$         ;;BRANCH IF NOT
034100 012777 000100 145052      MOV     #100, @STKS ;;RE-ENABLE TTY KBD INTERRUPTS
034106 000002          RTI                    ;;RETURN
034110 004737 035762      15$:   JSR     PC, $TYPEC   ;;ECHO CHAR
034114 021627 000060      16$:   CMP     (SP), #60   ;;CHAR < 0?
034120 002420          BLT     18$         ;;BRANCH IF YES
034122 021627 000067      CMP     (SP), #67   ;;CHAR > 7?
034126 003015          BGT     18$         ;;BRANCH IF YES
034130 042726 000060      BIC     #60, (SP)+  ;;STRIP-OFF ASCII
034134 005766 000002      TST     2(SP)       ;;IS THIS THE FIRST CHAR
034140 001403          BEQ     17$         ;;BRANCH IF YES
034142 006316          ASL     (SP)        ;;NO, SHIFT PRESENT
034144 006316          ASL     (SP)        ;;CHAR OVER TO MAKE
034146 006316          ASL     (SP)        ;;ROOM FOR NEW ONE.
034150 005266 000002      17$:   INC     2(SP)       ;;KEEP COUNT OF CHAR
034154 056616 177776      BIS     -2(SP), (SP) ;;SET IN NEW CHAR
034160 000667          BR      7$         ;;GET THE NEXT ONE
034162 104401 001202      18$:   TYPE    ,SQUES     ;;TYPE ?<CR><LF>
034166 000720          BR      20$         ;;SIMULATE CONTROL-U
.DSABL  LSB

```

```

: *THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
: *CALL:
: *      RDCHR          ::GET A CHARACTER FROM THE QUEUE
: *      RETURN HERE   ::CHARACTER IS ON THE STACK
: *                   ::WITH PARITY BIT STRIPPED OFF
:
034170 011646          $RDCHR: MOV      (SP),-(SP)      ::PUSH DOWN THE PC AND
034172 016666 000004 000002  MOV      4(SP),2(SP)    ::THE PS
034200 005066 000004          CLR      4(SP)          ::GET READY FOR A CHARACTER
034204 005046          CLR      -(SP)          ::PUT NEW PS ON STACK
034206 012746 034214          MOV      #64$,-(SP)      ::PUT NEW PC ON STACK
034212 000002          RTI                          ::POP NEW PC AND PS
034214
034214 005737 033310 64$:   TST      $TKCNT          ::WAIT ON A CHARACTER
034220 001775 1$:       BEQ      1$
034222 005337 033310          DEC      $TKCNT          ::DECREMENT THE COUNTER
034226 117766 177062 000004  MOVVB   @$TKQOUT,4(SP)  ::GET ONE CHARACTER
034234 005237 033314          INC      $TKQOUT          ::UPDATE THE POINTER
034240 023727 033314 033325  CMP      $TKQOUT,#$TKQEND ::DID IT GO OFF OF THE END?
034246 001003          BNE      2$          ::BRANCH IF NO
034250 012737 033316 033314  MOV      #$TKQRT,$TKQOUT ::RESET THE POINTER
034256 000002          RTI                          ::RETURN
: *****
: *THIS ROUTINE WILL INPUT A STRING FROM THE TTY
: *CALL:
: *      RDLIN          ::INPUT A STRING FROM THE TTY
: *      RETURN HERE   ::ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
: *                   ::TERMINATOR WILL BE A BYTE OF ALL 0'S
:
034260 010346          $RDLIN: MOV      R3, -(SP)      ::SAVE R3
034262 005046          CLR      -(SP)          ::CLEAR THE RUBOUT KEY
034264 012703 034514 1$:   MOV      #$TTYIN,R3      ::GET ADDRESS
034270 022703 034533 2$:   CMP      #$TTYIN+15.,R3  ::BUFFER FULL?
034274 011456          BLOS   4$          ::BR IF YES
034276 011456          RDCHR          ::GO READ ONE CHARACTER FROM THE TTY
034300 112613          MOVVB   (SP)+,(R3)      ::GET CHARACTER
034302 122713 000177 10$:  CMPB   #177,(R3)      ::IS IT A RUBOUT
034306 001022          BNE   5$          ::BR IF NO
034310 005716          TST   (SP)          ::IS THIS THE FIRST RUBOUT?
034312 001007          BNE   6$          ::BR IF NO
034314 112737 000134 034512  MOVVB   #'\,9$        ::TYPE A BACK SLASH
034322 104401 034512          TYPE   ,9$
034326 012716 177777          MOV   #-1,(SP)      ::SET THE RUBOUT KEY
034332 005303 6$:     DEC   R3          ::BACKUP BY ONE
034334 020327 034514          CMP   R3,$$TTYIN    ::STACK EMPTY?
034340 103434          BLO   4$          ::BR IF YES
034342 111337 034512          MOVVB (R3),9$      ::SETUP TO TYPEOUT THE DELETED CHAR.
034346 104401 034512          TYPE   ,9$
034352 000746          BR    2$          ::GO TYPE
034354 005716 5$:     TST   (SP)          ::GO READ ANOTHER CHAR.
034356 001406          BEQ   7$          ::RUBOUT KEY SET?
034360 112737 000134 034512  MOVVB   #'\,9$        ::TYPE A BACK SLASH
034366 104401 034512          TYPE   ,9$
034372 005016          CLR   (SP)          ::CLEAR THE RUBOUT KEY
034374 122713 000025 7$:   CMPB   #25,(R3)      ::IS CHARACTER A CTRL U?
034400 001003          BNE   8$          ::BR IF NO
  
```

```

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

2
3
4
5 034574 012737 000001 001340  CTRAP: MOV      #1,CFLAG          ;;SET THE 'CONTROL C' FLAG
6 034602 005237 033310          INC      $TKCNT          ;;COUNT THIS CHARACTER
7 034606 112777 000015 176476  MOVB     #15,@$TKQIN      ;;PUT 'RETURN' CHARACTER IN QUEUE
8 034614 005237 033312          INC      $TKQIN          ;;UPDATE THE POINTER
9 034620 023727 033312 033325  CMP      $TKQIN,$$TKQEND ;;GO OFF THE END ?
10 034626 001003          BNE      1$          ;;BR IF YES
11 034630 012737 033316 033312  MOV      #$TKQSRRT,$TKQIN ;;RESET THE POINTER
12 034636 000002      1$:  RTI          ;;RETURN

```

;THIS ROUTINE WILL PROCESS THE (^C) CHARACTER

CZF
SAV

1

.SBTTL ERROR HANDLER ROUTINE

```

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

```

034640 105037 035226 $ERROR: CLRB      IBSAVE      ;;CLEAR THE ITEM BYTE SAVE LOCATION
034644 104407          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
034646 010337 001322      MOV      R3,ATTN      ;;SAVE THE ATTENTION REGISTER CONTENTS
034652 010137 001220      MOV      R1,DRIVE     ;;DRIVE NUMBER
034656 032777 020000 144270 BIT      #SW13,@SWR    ;;INHIBIT PRINTOUTS ?
034664 001004          BNE      .+12         ;;BR IF YES
034666 104401 001203      TYPE     , $CRLF      ;;CR-LF
034672 104401 001203      TYPE     , $CRLF      ;;CR-LF
034676 004737 024764      JSR      PC,$TIME     ;;TYPE THE TIME
034702 105237 001117 7$:      INCB      $ERFLG      ;;SET THE ERROR FLAG
034706 001775          BEQ      7$          ;;DON'T LET THE FLAG GO TO ZERO
034710 013777 001116 144240 MOV      $STNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
034716 032777 002000 144230 BIT      #BIT10,@SWR   ;;BELL ON ERROR?
034724 001402          BEQ      1$          ;;NO - SKIP
034726 104401 001176      TYPE     , $BELL      ;;RING BELL
034732 005237 001126 1$:      INC      $ERTTL      ;;COUNT THE NUMBER OF ERRORS
034736 011637 001132      MOV      (SP), $ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
034742 162737 000002 001132 SUB      #2, $ERRPC
034750 117737 144156 001130 MOV      @ $ERRPC, $ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
034756 032777 001000 14417J BIT      #BIT09,@SWR   ;;SEE IF LOOP ON ERROR IS SET
034764 001060          BNE      1004$       ;;BRANCH AROUND ROUTINE IF SO
034766 122737 000177 001130 CMP      #177, $ITEMB  ;;SEE IF THIS IS THE POWER FAIL CALL
034774 001454          BEQ      1004$       ;;BRANCH AROUND ROUTINE IF IT IS
034776 105737 035226      TSTB     IBSAVE      ;;SEE IF THIS IS THE 2ND ERROR CALL IN THIS ROUTINE
035002 001047          BNE      1003$       ;;BRANCH IF SO
035004 022737 177777 035224 CMP      #-1, CPSAVE   ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
035012 001445          BEQ      1004$       ;;BRANCH IF SO
035014 013746 000004      MOV      ERRVEC, -(SP) ;;SAVE CONTENTS OF ERROR VECTOR
035020 012737 035036 000004 MOV      #1000$, ERRVEC ;;SETUP 'TRAP' RETURN ADDRESS
035026 013737 177766 035224 MOV      177766, CPSAVE ;;MOVE CPU ERROR REGISTER TO CPSAVE FOR TEST
035034 000406          BR      1001$
035036 012737 177777 035224 1000$: MOV      #-1, CPSAVE   ;;SET CPU ERROR REGISTER TIMEOUT INDICATOR
035044 012716 035052      MOV      #1001$, (SP) ;;SETUP RETURN ADDRESS
035050 000002          RTI
035052 012637 000004 1001$: MOV      (SP)+, ERRVEC ;;RESTORE CONTENTS OF ERROR VECTOR

035056 022737 177777 035224 1002$: CMP      #-1, CPSAVE   ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
035064 001420          BEQ      1004$       ;;BRANCH IF SO
035066 032737 000001 035224 BIT      #BIT00, CPSAVE ;;SEE IF POWER MONITOR BIT IS SET IN CPU ERR REG
035074 001414          BEQ      1004$       ;;BRANCH IF OK
035076 042737 000001 177766 BIC      #BIT00, 177766 ;;CLEAR THE BIT FOUND SET
035104 113737 001130 035226 MOV      $ITEMB, IBSAVE ;;MAKE IBSAVE NON-ZERO FOR DUAL ERROR CALL
035112 112737 000177 001130 MOV      #177, $ITEMB  ;;SET $ITEMB TO SPECIAL POWER FAIL POINTER
035120 000402          BR      1004$       ;;BRANCH OVER IBSAVE CLEARING
    
```

```

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

.SBTTL ERROR MESSAGE TYPEOUT ROUTINE

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

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

035542 000 000 000 PFECH4: .BYTE 0,0,0,0
035546 000000 PFTSTN: .WORD 0

::CONTAINS TEST NUMBER FOR PF BIT ERROR

.SBTTL TYPE ROUTINE

```

*****
*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
  
```

```

*CALL:
*1) USING A TRAP INSTRUCTION
*   TYPE ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
*OR
*   TYPE
*   MESADR
  
```

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

;HORIZONTAL TAB PROCESSOR

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

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

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

036250	052703	000060		BIS	#'0,R3	::MAKE THIS DIGIT ASCII
036254	052703	000040	5\$:	BIS	#',R3	::MAKE ASCII IF NOT ALREADY
036260	110337	036324		MOVB	R3,8\$::SAVE FOR TYPING
036264	104401	036324		TYPE	,8\$::GO TYPE THIS DIGIT
036270	105337	036326	7\$:	DECB	\$OCNT	::COUNT BY 1
036274	003347			BGT	2\$::BR IF MORE TO DO
036276	002402			BLT	6\$::BR IF DONE
036300	005204			INC	R4	::INSURE LAST DIGIT ISN'T A BLANK
036302	000744			BR	2\$::GO DO THE LAST DIGIT
036304	012605		6\$:	MOV	(SP)+,R5	::RESTORE R5
036306	012604			MOV	(SP)+,R4	::RESTORE R4
036310	012603			MOV	(SP)+,R3	::RESTORE R3
036312	016666	000002 000004		MOV	2(SP),4(SP)	::SET THE STACK FOR RETURNING
036320	012616			MOV	(SP)+,(SP)	
036322	000002			RTI		::RETURN
036324	000		8\$:	.BYTE	0	::STORAGE FOR ASCII DIGIT
036325	000			.BYTE	0	::TERMINATOR FOR TYPE ROUTINE
036326	000		\$OCNT:	.BYTE	0	::OCTAL DIGIT COUNTER
036327	000		\$OFILL:	.BYTE	0	::ZERO FILL SWITCH
036330	000000		\$OMODE:	.WORD	0	::NUMBER OF DIGITS TO TYPE

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

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

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

\$TYPDS:

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

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

.SBTTL APT COMMUNICATIONS ROUTINE

```

*****
036556 112737 006001 037022 $ATY1:  MOV  #1,$FFLG      ;;TO REPORT FATAL ERROR
036564 112737 000001 037020 $ATY3:  MOV  #1,$MFLG      ;;TO TYPE A MESSAGE
036572 000403                BR      $ATYC
036574 112737 000001 037022 $ATY4:  MOV  #1,$FFLG      ;;TO ONLY REPORT FATAL ERROR
036602                $ATYC:
036602 010046                MOV   R0,-(SP)      ;;PUSH R0 ON STACK
036604 010146                MOV   R1,-(SP)      ;;PUSH R1 ON STACK
036606 105737 037020                TSTB  $MFLG      ;;SHOULD TYPE A MESSAGE?
036612 001450                BEQ   5$          ;;IF NOT: BR
036614 122737 000001 001226        CMPB  #APTENV,$ENV  ;;OPERATING UNDER APT?
036622 001031                BNE   3$          ;;IF NOT: BR
036624 132737 000100 001227        BITB  #APTPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
036632 001425                BEQ   3$          ;;IF NOT: BR
036634 017600 000004                MOV   @4(SP),R0    ;;GET MESSAGE ADDR.
036640 062766 000002 000004        ADD   #2,4(SP)    ;;BUMP RETURN ADDR.
036646 005737 001206        1$:  TST  $MSGTYPE    ;;SEE IF DONE W/ LAST XMISSION?
036652 001375                BNE   1$          ;;IF NOT: WAIT
036654 010037 001222        MOV   R0,$MSGAD   ;;PUT ADDR IN MAILBOX
036660 105720        2$:  TSTB (R0)+      ;;FIND END OF MESSAGE
036662 001376                BNE   2$
036664 163700 001222        SUB   $MSGAD,R0   ;;SUB START OF MESSAGE
036670 006200                ASR   R0          ;;GET MESSAGE LNGTH IN WORDS
036672 010037 001224        MOV   R0,$MSGGLT  ;;PUT LENGTH IN MAILBOX
036676 012737 000004 001206        MOV   #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
036704 000413                BR    5$
036706 017637 000004 036732        3$:  MOV   @4(SP),4$    ;;PUT MSG ADDR IN JSR LINKAGE
036714 062766 000002 000004        ADD   #2,4(SP)    ;;BUMP RETURN ADDRESS
036722 013746 177776                MOV   177776,-(SP) ;;PUSH 177776 ON STACK
036726 004737 035550                JSR   PC,$TYPE    ;;CALL TYPE MACRO
036732 000000        4$:  .WORD  0
036734        5$:
036734 105737 037022        10$: TSTB  $FFLG      ;;SHOULD REPORT FATAL ERROR?
036740 001416                BEQ   12$        ;;IF NOT: BR
036742 005737 001226                TST  $ENV        ;;RUNNING UNDER APT?
036746 001413                BEQ   12$        ;;IF NOT: BR
036750 005737 001206        11$: TST  $MSGTYPE    ;;FINISHED LAST MESSAGE?
036754 001375                BNE   11$        ;;IF NOT: WAIT
036756 017637 000004 001210        MOV   @4(SP),$FATAL ;;GET ERROR #
036764 062766 000002 000004        ADD   #2,4(SP)    ;;BUMP RETURN ADDR.
036772 005237 001206                INC  $MSGTYPE    ;;TELL APT TO TAKE ERROR
036776 105037 037022        12$: CLRB  $FFLG      ;;CLEAR FATAL FLAG
037002 105037 037021                CLRB $LFLG      ;;CLEAR LOG FLAG
037006 105037 037020                CLRB $MFLG      ;;CLEAR MESSAGE FLAG
037012 012601                MOV  (SP)+,R1    ;;POP STACK INTO R1
037014 012600                MOV  (SP)+,R0    ;;POP STACK INTO R0
037016 000207                RTS   PC        ;;RETURN
037020 000                $MFLG: .BYTE  0  ;;MESSG. FLAG
037021 000                $LFLG: .BYTE  0  ;;LOG FLAG
037022 000                $FFLG: .BYTE  0  ;;FATAL FLAG
                .EVEN
                APTSIZE = 200
                APTENV  = 001
                APTPOOL = 100
                APTCSUP = 040
  
```

.SBTTL RANDOM NUMBER GENERATOR ROUTINE

::*****
 ::*THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR
 ::*WITH A RANGE OF 0 TO 2(+33)-1.
 ::*CALL:

```

    *      JSR      PC,$RAND      ::CALL THE ROUTINE
    *      RETURN                                ::RETURN HERE THE RANDOM
    *                                          ::NUMBER WILL BE IN
    *                                          ::$HINUM,$LONUM
    
```

```

037024
037024 010046
037026 010146
037030 010246
037032 013700 037124
037036 013701 037122
037042 012702 177771
037046 006300
037050 006101
037052 005202
037054 001374
037056 063700 037124
037062 005501
037064 063701 037122
037070 062700 001057
037074 005501
037076 062701 047401
037102 010037 037124
037106 010137 037122
037112 012602
037114 012601
037116 012600
037120 000207
037122 176543
037124 123456
    
```

```

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

.SBTTL SAVE AND RESTORE R0-R5 ROUTINES

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

```
037126  
037126 010046  
037130 010146  
037132 010246  
037134 010346  
037136 010446  
037140 010546  
037142 016646 000022  
037146 016646 000022  
037152 016646 000022  
037156 016646 000022  
037162 000002
```

```
$SAVREG:  
MOV R0,-(SP) ::PUSH R0 ON STACK  
MOV R1,-(SP) ::PUSH R1 ON STACK  
MOV R2,-(SP) ::PUSH R2 ON STACK  
MOV R3,-(SP) ::PUSH R3 ON STACK  
MOV R4,-(SP) ::PUSH R4 ON STACK  
MOV R5,-(SP) ::PUSH R5 ON STACK  
MOV 22(SP),-(SP) ::SAVE PS OF MAIN FLOW  
MOV 22(SP),-(SP) ::SAVE PC OF MAIN FLOW  
MOV 22(SP),-(SP) ::SAVE PS OF CALL  
MOV 22(SP),-(SP) ::SAVE PC OF CALL  
RTI
```

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

```
037164  
037164 012666 000022  
037170 012666 000022  
037174 012666 000022  
037200 012666 000022  
037204 012605  
037206 012604  
037210 012603  
037212 012602  
037214 012601  
037216 012600  
037220 000002
```

```
MOV (SP)+,22(SP) ::RESTORE PC OF CALL  
MOV (SP)+,22(SP) ::RESTORE PS OF CALL  
MOV (SP)+,22(SP) ::RESTORE PC OF MAIN FLOW  
MOV (SP)+,22(SP) ::RESTORE PS OF MAIN FLOW  
MOV (SP)+,R5 ::POP STACK INTO R5  
MOV (SP)+,R4 ::POP STACK INTO R4  
MOV (SP)+,R3 ::POP STACK INTO R3  
MOV (SP)+,R2 ::POP STACK INTO R2  
MOV (SP)+,R1 ::POP STACK INTO R1  
MOV (SP)+,R0 ::POP STACK INTO R0  
RTI
```

.SBTTL DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

 *THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
 *DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
 *POSITIVE.
 *CALL

```
*      MOV      #PNTR,-(SP)      ;; POINTER TO LOW WORD OF BINARY NUMBER
*      JSR      PC,@#$DB2D      ;; THE FIRST ADDRESS OF ASCII
*      RETURN                                ;; IS ON THE STACK
```

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


037372 000012
037374 000000
037376 000001
037400 000000
037402

12
0
1
0
SDECVL: .BLKB 12.

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

.SBTTL DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE

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

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

037416	104412		\$DB20:	SAVREG		::SAVE ALL REGISTERS
037420	016601	000002		MOV	2(SP),R1	::PICKUP THE POINTER TO LOW WORD
037424	012705	037535		MOV	#\$OCTVL+13.,R5	:: POINTER TO DATA TABLE
037430	012704	000014		MOV	#12.,R4	::DO ELEVEN CHARACTERS
037434	012703	177770		MOV	#^C7,R3	::MASK
037440	012100			MOV	(R1)+,R0	::LOWER WORD
037442	012101			MOV	(R1)+,R1	::HIGH WORD
037444	005002			CLR	R2	::TERMINATOR
037446	110245		1\$:	MOVB	R2,-(R5)	::PUT CHARACTER IN DATA TABLE
037450	010002			MOV	R0,R2	::GET THIS DIGIT
037452	005304			DEC	R4	::COUNT THIS CHARACTER
037454	0C3007			BGT	3\$::BR IF NOT THE LAST DIGIT
037456	001405			BEQ	2\$::BR IF IT IS THE LAST DIGIT
037460	005205			INC	R5	::ALL DIGITS DONE-ADJUST POINTER FOR FIRST
037462	010566	000002		MOV	R5,2(SP)	::ASCII CHAR. & PUT IT ON THE STACK
037466	104413			RESREG		::RESTORE ALL REGISTERS
037470	000207			RTS	PC	::RETURN TO USER
037472	006203		2\$:	ASR	R3	::POSITION THE MASK FOR THE LAST DIGIT
037474	006001		3\$:	ROR	R1	::POSITION THE BINARY NUMBER FOR
037476	006000			ROR	R0	:: THE NEXT OCTAL DIGIT
037500	006001			ROR	R1	
037502	006000			ROR	R0	
037504	006001			ROR	R1	
037506	006000			ROR	R0	
037510	040302			BIC	R3,R2	::MASK OUT ALL JUNK
037512	062702	000060		ADD	#'0,R2	::MAKE THIS CHAR. ASCII
037516	000753			BR	1\$::GO PUT IT IN THE DATA TABLE
037520			\$OCTVL:	.BLKB	14.	::RESERVE DATA TABLE

.SBTTL POWER DOWN AND UP ROUTINES

:POWER DOWN ROUTINE

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

:POWER UP ROUTINE

037610	012737	037710	000024	\$PWRUP: MOV	#\$ILLUP,@#PWRVEC	::SET FOR FAST DOWN
037616	013706	037714		MOV	\$\$SAVR6,SP	::GET SP
037622	005037	037714		CLR	\$\$SAVR6	::WAIT LOOP FOR THE TTY
037626	005237	037714		1\$: INC	\$\$SAVR6	::WAIT FOR THE INC
037632	001375			BNE	1\$::OF WORD
037634	005337	037716		2\$: DEC	PWRFLG	::WAIT AND SET POWER FAIL FLAG
037640	003775			BLE	2\$::WAIT FOR FLAG= 1
037642	012677	141306		MOV	(SP)+,@SWR	::POP STACK INTO @SWR
037646	012605			MOV	(SP)+,R5	::POP STACK INTO R5
037650	012604			MOV	(SP)+,R4	::POP STACK INTO R4
037652	012603			MOV	(SP)+,R3	::POP STACK INTO R3
037654	012602			MOV	(SP)+,R2	::POP STACK INTO R2
037656	012601			MOV	(SP)+,R1	::POP STACK INTO R1
037660	012600			MOV	(SP)+,R0	::POP STACK INTO R0
037662	012737	037536	000024	MOV	#\$PWRDN,@#PWRVEC	::SET UP THE POWER DOWN VECTOR
037670	012737	000340	000026	MOV	#340,@#PWRVEC+2	::PRIO:7
037676	104401			TYPE		::REPORT THE POWER FAILURE
037700	037720			\$PWRMG: .WORD	\$POWER	::POWER FAIL MESSAGE POINTER
037702	012716			MOV	(PC)+,(SP)	::RESTART AT SATPOW
037704	037736			\$PWRAD: .WORD	SATPOW	::RESTART ADDRESS
037706	000002			RTI		
037710	000000			\$ILLUP: HALT		::THE POWER UP SEQUENCE WAS STARTED
037712	000776			BR	.-2	::BEFORE THE POWER DOWN WAS COMPLETE
037714	000000			\$\$SAVR6: 0		::PUT THE SP HERE
2 037716	000000			PWRFLG: .WORD	0	::INDICATES POWER FAIL OCCURRED WHEN SET
3 037720	200	042	120	\$POWER: .ASCIZ	<CRLF>/'POWER UP'/<CRLF>	
4				.EVEN		
5						
6						
7						
8						
9 037736	005227	000000		SATPOW: INC	#0	::TTY LOOP, WAIT FOR INCREMENT
10 037742	001375			BNE	.-4	::OF WORD
11 037744	000005			RESET		::CLEAR THE WORLD
12 037746	005037	001350		CLR	SECOND	::RESET SECOND COUNT
13 037752	005037	001466		CLR	INTRVL+2	::RESET THE INTERVAL COUNT
14 037756	005037	177776		CLR	PS	::CLEAR PSW

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

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

1

.SBTTL TRAP DECODER

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

040060	010046		\$TRAP: MOV	R0,-(SP)	::SAVE R0
040062	016600	000002	MOV	2(SP),R0	::GET TRAP ADDRESS
040066	005740		TST	-(R0)	::BACKUP BY 2
040070	111000		MOVB	(R0),R0	::GET RIGHT BYTE OF TRAP
040072	6300		ASL	R0	::POSITION FOR INDEXING
040074	016000	040114	MOV	\$TRPAD(R0),RC	::INDEX TO TABLE
040100	000200		RTS	R0	::GO TO ROUTINE

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

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

.SBTTL TRAP TABLE

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

			:	ROUTINE	
			:	-----	
040114	040102		\$TRPAD: .WORD	\$TRAP2	
040116	035550		\$TYPE	::CALL=TYPE	TRAP+1(104401) TTY TYPEOUT ROUTINE
040120	036130		\$TYPOC	::CALL=TYPOC	TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
040122	036104		\$TYPOS	::CALL=TYPOS	TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
040124	036144		\$TYPON	::CALL=TYPON	TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
040126	036332		\$TYPDS	::CALL=TYPDS	TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
040130	033716		\$GTSWR	::CALL=GTSWR	TRAP+6(104406) GET SOFT-SWR SETTING
040132	033626		\$CKSWR	::CALL=CKSWR	TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
040134	034170		\$RDCHR	::CALL=RDCHR	TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
040136	034260		\$RDLIN	::CALL=RDLIN	TRAP+11(104411) TTY TYPEIN STRING ROUTINE
040140	037126		\$SAVREG	::CALL=SAVREG	TRAP+12(104412) SAVE R0-R5 ROUTINE
040142	037164		\$RESREG	::CALL=RESREG	TRAP+13(104413) RESTORE R0-R5 ROUTINE
2 040144	032716		\$DSPLY	::CALL=DISPLY	TRAP+14(104414) ROUTINE TO TYPE ERROR MESSAGES
3	000032		\$TERM=.	-\$TRPAD	

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

040200 000
040201 000
040202 000
040203 000

.BYTE 0 :DRIVE 4
.BYTE 0 :DRIVE 5
.BYTE 0 :DRIVE 6
.BYTE 0 :DRIVE 7

74
75
76
77
78

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

79 040204 000
82 040205 000
040206 000
040207 000
040210 000
040211 000
040212 000
040213 000

DPINT: .BYTE 0 :DRIVE 0
.BYTE 0 :DRIVE 1
.BYTE 0 :DRIVE 2
.BYTE 0 :DRIVE 3
.BYTE 0 :DRIVE 4
.BYTE 0 :DRIVE 5
.BYTE 0 :DRIVE 6
.BYTE 0 :DRIVE 7

83
84
85
86
87

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

88 040214 000
91 040215 000
040216 000
040217 000
040220 000
040221 000
040222 000
040223 000

DPRQS: .BYTE 0 :DRIVE 0
.BYTE 0 :DRIVE 1
.BYTE 0 :DRIVE 2
.BYTE 0 :DRIVE 3
.BYTE 0 :DRIVE 4
.BYTE 0 :DRIVE 5
.BYTE 0 :DRIVE 6
.BYTE 0 :DRIVE 7

92
93
94
95
98

;DRIVE REQUEST QUE WORDS

040224 000000
040226 000000
040230 000000
040232 000000
040234 000000
040236 000000
040240 000000
040242 000000

QDRV: .WORD 0 :DRIVE 0
.WORD 0 :DRIVE 1
.WORD 0 :DRIVE 2
.WORD 0 :DRIVE 3
.WORD 0 :DRIVE 4
.WORD 0 :DRIVE 5
.WORD 0 :DRIVE 6
.WORD 0 :DRIVE 7

99
100
101
102
103

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

104 040244 000000

TRNSWT: .WORD 0

105
106
107
108
109

;SEARCH WAIT KEYS (SRCHWT=1 WORD)
;THIS IS A ONE WORD QUEUE THAT WILL CONTAIN A KEY FOR EACH OF
;THE DRIVES THAT ARE PERFORMING A SEARCH COMMAND FOR THE I/O
;REQUEST THAT IS AT THE TOP OF THEIR REQUEST QUEUE.
;EACH DRIVE IS ASSIGNED ONE BIT, STARTING AT BIT0 FOR DRIVE 0.

110
111
112 040246 000000

SRCHWT: .WORD 0

113
114

;RM DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)

```
115           ;ACTDRV=0 IF DRIVER IS INACTIVE
116           ;ACTDRV>0 IF DRIVER IS ACTIVE
117
118 040250     000      ACTDRV: .BYTE  0
119
120           ;SOFTWARE TIMER ROUTINE ACTIVE FLAG (ACTSTR=1 BYTE)
121           ;ACTSTR=0 IF SOFTWARE TIMER ROUTINE IS INACTIVE
122           ;ACTSTR>0 IF SOFTWARE TIMER ROUTINE IS ACTIVE
123
124 040251     000      ACTSTR: .BYTE  0
125
126           ;SAVE REGISTERS FLAG (SAVEFG =1 WORD)
127           ;SAVEFG <0 IF SAVE THE RH/RM REGISTERS WHEN THE
128           ;OPERATION IS COMPLETED AS PER (DPB+14).
129           ;SAVEFG=0 IF SAVE THE RH/RM REGISTERS, AS PER
130           ;(DPB+14), AFTER AN ERROR.
131
132 040252     000000    SAVEFG: .WORD  0
133
134           ;SEEK FLAG (SEEKFG=1 WORD)
135           ;SEEKFG=0 IF WHEN THE DISK ADDRESS ISN'T IN THE WINDOW
136           ;FOR A DATA TRANSFER START A SEARCH COMMAND
137           ;SEEKFG<0 IF DATA TRANSFER WILL DO IMPLIED SEEKS,
138           ;DISREGARD THE WINDOW
139
140 040254     177777    SEEKFG: .WORD  -1
141
142           ;TIMEOUT TABLE (TIMER=8 WORDS)
143           ;THIS TABLE CONTAINS THE TIME ALLOWED FOR AN OPERATION
144
145 040256     177777    TIMER:  .WORD  -1           ;DRIVE 0
148 040260     177777    .WORD  -1           ;DRIVE 1
149 040262     177777    .WORD  -1           ;DRIVE 2
150 040264     177777    .WORD  -1           ;DRIVE 3
151 040266     177777    .WORD  -1           ;DRIVE 4
152 040270     177777    .WORD  -1           ;DRIVE 5
153 040272     177777    .WORD  -1           ;DRIVE 6
154 040274     177777    .WORD  -1           ;DRIVE 7
155
156           ;DATA TRANSFER UNDERWAY INDICATOR (DTUW=1 WORD)
157           ;DTUW<0 IF NO DATA TRANSFER UNDERWAY
158           ;DTUW=+N (WHERE N=0 TO 7) IMPLIES DATA TRANSFER UNDERWAY ON DRIVE N
159
160 040276     177777    DTUW:  .WORD  -1
161
162           ;ATTENTION BITS TABLE (ATABIT=8 BYTES)
163           ;THIS TABLE CONTAINS THE CORRESPONDING BIT TO EACH DRIVES
164           ;ATTENTION BIT
165
166 040300     001      002      004    ATABIT: .BYTE  1,2,4,10,20,40,100,200
167
168           ;STORAGE FOR RMADR (THE FIRST ADDRESS (776700) OF THE RH70/RM80),
169           ;RMVEC (THE VECTOR ADDRESS (254)), AND RMVEC+2 (THE BR LEVEL (5)).
170
171 040310     176700    RMADR:  .WORD  176700
172 040312     000254    RMVEC:  .WORD  254,5*32.
173 040316     000050    RHEXT:  .WORD  50           ;OFFSET TO RMBAE
```


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
198
199
201

040320 000005

:SEARCH DIFFERENCE IS 5 FIVE SECTORS

MXWWDW: .WORD 5

:DEFINITIONS OF THE RH70/RM80 ADDRESS INDEXES

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

```

1      ;RH70/RM80 DRIVER INITIALIZATION CODE
2      ;THIS ROUTINE WILL DETERMINE WHICH RM80 DRIVES ARE
3      ;AVAILABLE FOR TESTING AND SET THE DRVSTA INDICATOR
4      ;TO THE PROPER STATE FOR EACH DRIVE.
5      ;NOTE: THIS ROUTINE CALLS DRVINT
6
7      ;CALL:
8
9      JSR    PC,RMINIT
10     RETURN
11
12     ;NOTE: THE 'P' OR 'L' CLOCK MUST BE STARTED
13
14     RMINIT: SAVREG                ;SAVE R0 - R5
15     MOV    @#PS,-(SP)            ;SAVE THE PRESENT PROCESSOR STATUS
16     MOV    #<5*32.>,@#PS        ;CHANGE THE PRIORITY TO 5
17     JSR    PC,CLRQUE            ;CLEAR ALL REQUEST QUEUES
18     MOV    #RMERRS,R1          ;FIRST ADDRESS TO BE CLEARED
19     MOV    #SEEKFG,R2          ;LAST ADDRESS TO BE CLEARED
20     CLR    (R1)+                ;CLEAR
21     CMP    R1,R2                ;ARE WE DONE?
22     BLOS   1$                  ;BRANCH IF NO
23     MOV    #DTUW,R2            ;LAST ADDRESS
24     MOV    #-1,(R1)+           ;INITIALIZE
25     CMP    R1,R2                ;DONE?
26     BLOS   2$                  ;LOOP IF NO
27     CLR    DRVSTA              ;SET ALL DRIVES TO OFFLINE
28     CLR    DRVSTA+2
29     CLR    DRVSTA+4
30     CLR    DRVSTA+6
31     MOV    RMVEC,R3            ;SETUP THE RH70/RM80 VECTOR
32     MOV    #ISR,(R3)+
33     MOV    RMVEC+2,(R3)
34     MOV    RMADR,R4            ;FIRST ADDRESS OF RH70/RM80
35     MOV    #40,RMCS2(R4)      ;MASSBUS INIT
36     CLR    R1                  ;START WITH DRIVE 0
37     JSR    R0,DRVINT           ;INIT THE DRIVE
38     BR     4$                  ;'DVA' NOT SET
39     BR     5$                  ;NORMAL RETURN
40     CLR    DRVSTA(R1)          ;SET DRIVE STATUS TO OFFLINE
41     INC    R1                  ;GO TO NEXT DRIVE
42     BIC    #^C7,R1            ;MASK OUT UNUSED BITS
43     BNE    3$                  ;BR IF MORE DRIVES TO GO
44     MOV    #7,R1              ;START WITH DRIVE 7
45     CLR    @#PS                ;CLEAR THE PROCESSOR STATUS
46     TSTB   DPINT(R1)          ;WAITING FOR DRIVE TO SWITCH PORTS ?
47     BEQ    8$                  ;BR NOT WAITING
48     JSR    PC,SET.IE          ;SET INTERRUPT
49     TSTB   DPINT(R1)          ;DRIVE SWITCHED PORTS ?
50     BNE    7$                  ;BR IF NOT
51     DEC    R1                  ;GO TO THE NEXT DRIVE
52     BPL    6$                  ;CHECK NEXT DRIVE
53     MOV    (SP)+,@#PS         ;RESTORE THE PROCESSOR STATUS
54     RESREG
55     RTS    PC                  ;RESTORE R0 - R5
56     ;BYE-BYE
57
;DRIVE INITIALIZATION ROUTINE

```

```

58                                     :THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
59                                     :AN RM80. IF IT IS, A 'READ-IN PRESET' IS ISSUED AND FMT16
60                                     :IS SET TO A '1'. THEN MOL, DPR, DRY, AND V1 ARE CHECKED TO
61                                     :INSURE THEY ARE ALL ON A '1'. AND DEPENDING ON THEIR STATE,
62                                     :DRVSTA IS SET TO THE PROPER CONDITION.
63                                     :CALL
64                                     :
65                                     MOV     #DRVNUM,R1      :DRIVE NUMBER TO R1
66                                     MOV     RMADR,R4       :UNIBUS ADDRESS OF RH70/RM80 (RMCS1)
67                                     JSR     R0,DRVINT      :CALLED BY A JSR
68                                     RETURN1 :ERROR OCCURRED ('NED')
69                                     RETURN2 :NORMAL RETURN
70                                     :
71 040534 010546 DRVINT: MOV     R5,-(SP)      :SAVE R5
72 040536 105061 040164 CLRB   DRVSTA(R1)  :START DRIVE STATUS AS OFFLINE
73 040542 105061 040174 CLRB   DRVSTYP(R1) :CLEAR THE DRIVE TYPE INDICATOR
74 040546 010164 000010 MOV     R1,RMCS2(R4) :SELECT A DRIVE
75 040552 112764 000111 000000 MOV    #111,RMCS1(R4) :DO A DRIVE CLEAR COMMAND (& .EIZE DRIVE)
76 040560 032764 010000 000010 BIT    #BIT12,RMCS2(R4) :NONEXISTENT DRIVE?
77 040566 001403 BEQ    1$          :NO---BRANCH
78 040570 004737 045226 JSR    PC,SET.IE   :GO SET 'IE' WITHOUT A 'TRE'
79 040574 000476 BR     4$          :LEAVE THIS ROUTINE
80 040576 105061 040164 1$: CLRB   DRVSTA(R1)  :SET DRIVE STATUS TO OFFLINE
81 040602 032764 004000 000000 BIT    #BIT11,RMCS1(R4) :SEE IF DRIVE AVAILABLE
82 040610 001470 BEQ    4$          :BR IF DRIVE NOT AVAILABLE
83 040612 004037 044670 JSR    R0,RD.RM   :CALL THE READ ROUTINE
84 040616 000026 RMDT   5$          :REGISTER OFFSET
85 040620 040774 5$          :'NED' RETURN
86 040622 012605 MOV     (SP)+,R5    :PUT DRIVE TYPE IN R5
87 040624 112761 000001 040174 MOV    #1,DRVSTYP(R1) :SET RM80 INDICATOR
88 040632 022705 020026 CMP    #20026,R5     :IS IT A SINGLE PORT RM80?
89 040636 001407 BEQ    2$          :BRANCH IF YES
90 040640 022705 024026 CMP    #24026,R5     :IS IT A DUAL PORT RM80?
91 040644 001404 BEQ    2$          :BR IF YES
92 040646 112761 177777 040174 MOV    #-1,DRVSTYP(R1) :SET INDICATOR TO 'OTHER'
93 040654 000446 BR     4$          :EXIT
94 040656 012746 000121 2$: MOV    #121,-(SP)   :DO A 'READ-IN PRESET'
95 040658 004037 044750 JSR    R0,WRT.RM   :CALL THE WRITE ROUTINE
96 040666 000000 RMCS1  5$          :REGISTER OFFSET
97 040670 040774 5$          :'NED' RETURN
98 040672 012746 010000 MOV    #BIT12,-(SP) :SET FMT16=1
99 040676 004037 044750 JSR    R0,WRT.RM   :CALL THE WRITE ROUTINE
100 040702 000032 RMOF   5$          :REGISTER OFFSET
101 040704 040774 5$          :'NED' RETURN
102 040706 004037 044670 JSR    R0,RD.RM   :CALL THE READ ROUTINE
103 040712 000012 RMDS   5$          :REGISTER OFFSET
104 040714 040774 5$          :'NED' RETURN
105 040716 012605 MOV     (SP)+,R5    :AND SAVE IT IN R5
106 040720 100015 BPL    3$          :BRANCH IF ATA=0
107 040722 116164 040300 000016 MOV    ATABIT(R1),RMAS(R4) :CLEAR ATTENTION BIT
108 040730 004037 044670 JSR    R0,RD.RM   :CALL THE READ ROUTINE
109 040734 000014 RMER1  5$          :REGISTER OFFSET
110 040736 040774 5$          :'NED' RETURN
111 040740 006126 ROL    (SP)+      :IS IT UNSAFE?
112 040742 100004 BPL    3$          :BR IF NOT
113 040744 112761 177777 040164 MOV    #-1,DRVSTA(R1) :SET UNSAFE INDICATOR
114 040752 000407 BR     4$          :EXIT
    
```

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

```

1          ;REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
2
3          ;CALL:
4
5          ;
6          ;
7          ;
8          ;
9
10
11 041000 013746 177776          RM80:  MOV    @#PS, -(SP)      ;SAVE THE CALLING STATUS
12 041004 013737 040314 177776  MOV    RMVEC+2, @#PS    ;DON'T ALLOW ANY RM80 INTERRUPTS
13 041012 112737 000001 040250  MOVB   #1, ACTDRV      ;SET 'ACTIVE DRIVER' FLAG
14 041020 104412                SAVREG                ;SAVE R0 - R5
15 041022 011002                MOV    (R0), R2        ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
16 041024 005062 000016        CLR    16(R2)         ;CLEAR THE STATUS/ERROR INDICATOR
17 041030 111201                MOVB   (R2), R1        ;PICKUP THE DRIVE NUMBER
18 041032 013704 040310        MOV    RMADR, R4       ;UNIBUS ADDRESS OF RMCS1
19 041036 105761 040164        TSTB  DRVSTA(R1)      ;CHECK DRIVES STATUS
20 041042 003011                BGT    1$             ;BRANCH IF ONLINE
21 041044 105761 040204        TSTB  DPINT(R1)       ;TRYING TO INIT THE DRIVE
22 041050 001027                BNE    4$             ;BR IF YES
23 041052 004037 040534        JSR    R0, DRVINT      ;GO INIT. THE DRIVE
24 041056 000421                BR     3$             ;ERROR RETURN
25 041060 105761 040164        TSTB  DRVSTA(R1)      ;IS DRIVE STATUS ONLINE?
26 041064 003432                BLE    5$             ;BR IF NOT
27 041066 105761 040214        1$:   TSTB  DPRQS(R1)     ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
28 041072 001016                BNE    4$             ;BR IF YES
29 041074 010164 000010        MOV    R1, RMCS2(R4)  ;SELECT THE DRIVE
30 041100 004037 045344        JSR    R0, DRVQIF     ;PUT THIS REQUEST IN QUEUE
31 041104 000445                BR     8$             ;QUEUE IS FULL
32 041106 105761 040154        2$:   TSTB  DRVACT(R1)  ;IS THIS DRIVE ACTIVE?
33 041112 001037                BNE    7$             ;BR IF YES
34 041114 004737 041236        JSR    PC, OPT        ;CALL THE OPTIMIZER
35 041120 000434                BR     7$
36 041122 004737 042334        3$:   JSR    PC, CI8     ;GO HANDLE THE 'NED'
37 041126 000431                BR     7$
38 041130 004037 045344        4$:   JSR    R0, DRVQUE  ;PUT REQUEST IN QUEUE
39 041134 000431                BR     8$             ;QUEUE IS FULL
40 041136 032714 000100        BIT    #BIT06, (R4)   ;IS 'IE' SET ALREADY ?
41 041142 001023                BNE    7$             ;BR IF IT IS
42 041144 004737 045226        JSR    PC, SET.IE    ;SET INTERRUPT
43 041150 000420                BR     7$             ;RETURN, REQUEST IN QUEUE
44 041152 105761 040164        5$:   TSTB  DRVSTA(R1)  ;SEE IF DRIVE OFFLINE OR UNSAFE
45 041156 002412                BLT    6$             ;BR IF UNSAFE
46 041160 012762 140000 000016  MOV    #BIT15!BIT14,16(R2) ;SET OFFLINE ERROR INDICATOR
47 041166 105761 040174        TSTB  DRVSTYP(R1)    ;SEE IF OFFLINE OR NONEXISTENT
48 041172 001007                BNE    7$             ;BR IF OFFLINE
49 041174 012762 100002 000016  MOV    #BIT15!BIT01,16(R2) ;REPORT DRIVE NONEXISTENT
50 041202 000403                BR     7$             ;GO TO EXIT
51 041204 012762 110000 000016  6$:   MOV    #BIT15!BIT12,16(R2) ;DRIVE IS UNSAFE
52 041212 104413                7$:   RESREG                ;RESTORE R0 - R5
53 041214 005720                TST   (R0)+           ;SETUP FOR NORMAL RETURN
54 041216 000401                BR     9$             ;FINISH UP, THEN
55 041220 104413                8$:   RESREG                ;RESTORE R0 - R5
56 041222 005720                9$:   TST   (R0)+           ;CORRECT THE RETURN ADDRESS
57 041224 105037 040250        CLRB  ACTDRV         ;CLEAR 'ACTIVE DRIVER' FLAG
    
```

```

58 041230 012637 177776      MOV      (SP)+,@#PS      ;RETURN 'PS' TO USER LEVEL
59 041234 000200      RTS      R0              ;RETURN TO CALLER
60
61      ;OPTIMIZER-CALLED FOR A PARTICULAR DRIVE
62
63      ;CALL:
64      ;
65      ;
66      ;
67 041236 104412      OPT:    SAVREG           ;SAVE R0 - R5
68 041240 013746      MOV      @#PS,-(SP)      ;SAVE PROC. STATUS
69 041244 004737      JSR      PC,GETREQ      ;GET 'DPB' POINTER OF REQUEST
70 041250 005702      TST      R2              ;IS THERE A REQUEST IN QUEUE?
71 041252 001510      BEQ      9$              ;NO--BRANCH TO EXIT
72 041254 032764 004000 000000  BIT      #BIT11,RMCS1(R4) ;IS DVA SET?
73 041262 001407      BEQ      2$              ;BRANCH IF NOT
74 041264 032764 000100 000012  BIT      #BIT6,RMDS(R4)  ;IS VV SET ?
75 041272 001003      BNE      2$              ;BR IF IT IS
76 041274 004037 040534      1$:    JSR      R0,DRVINT     ;SEE IF DRIVE STILL ONLINE ?
77 041300 000473      BR       8$              ;'DVA' NOT SET
78 041302 105761 040164      2$:    TSTB     DRVSTA(R1) ;IS DRIVE ONLINE?
79 041306 003014      BGT      3$              ;YES--BRANCH
80 041310 004737 045400      JSR      PC,POPQUE      ;NO--REMOVE REQUEST FROM QUEUE
81 041314 012762 140000 000016  MOV      #BIT15!BIT14,16(R2) ;SET OFFLINE STATUS/ERROR INDICATOR
82 041322 105761 040164      TSTB     DRVSTA(R1)     ;IS DRIVE UNSAFE ?
83 041326 100067      BPL      10$             ;BR TO EXIT IF NOT
84 041330 012762 110000 000016  MOV      #BIT15!BIT12,16(R2) ;SET UNSAFE STATUS/ERROR INDICATOR
85 041336 000463      BR       10$             ;BRANCH TO EXIT
86 041340 012746 000111 3$:    MOV      #111,-(SP)     ;LOAD COMMAND ONTO THE STACK
87 041344 004037 044750      JSR      R0,WRT.RM      ;CALL THE WRITE ROUTINE
      041350 000000      RMCS1           ;REGISTER OFFSET
      041352 041470      8$          ;'NED' RETURN
88 041354 032714 004000      BIT      #BIT11,(R4)    ;DRIVE AVAILABLE ?
89 041360 001432      BEQ      7$              ;BR IF NOT
90 041362 122762 000150 000002  CMPB     #150,2(R2)     ;IS THE REQUEST FOR I/O?
91 041370 002403      BLT      4$              ;YES--BRANCH
92 041372 004737 041756      JSR      PC,C14         ;CALL THE COMMAND INITIATOR
93 041376 000443      BR       10$             ;BRANCH TO EXIT
94 041400 005737 040276      4$:    TST      DTUW         ;DATA TRANSFER UNDERWAY?
95 041404 002015      BGE      6$              ;YES--GO START A SEARCH
96 041406 136137 040300 040246  BITB     ATABIT(R1),SRCHWT ;FINISHED A SEARCH ?
97 041414 001003      BNE      5$              ;IF NE, YES
98 041416 005737 040254      TST      SEEKFG        ;DO IMPLIED SEEKS?
99 041422 100006      BPL      6$              ;IF PL DO A SEARCH
100 041424 146137 040300 040246 5$:    BICB     ATABIT(R1),SRCHWT ;CLEAR 'SEARCH WAIT' KEY
101 041432 004737 041516      JSR      PC,C11         ;START A DATA TRANSFER
102 041436 000423      BR       10$             ;
103 041440 004737 041650      6$:    JSR      PC,C13         ;START A SEARCH
104 041444 000420      BR       10$             ;GO TO THE EXIT
105 041446 112761 177777 040214 7$:    MOVB     #-1,DPRQS(R1)  ;SET PORT REQUEST INDICATOR
106 041454 010103      MOV      R1,R3          ;SET UP TO ADDRESS WORDS
107 041456 006303      ASL      R2              ;CONVERT TO WORD INDEX
108 041460 012763 035230 040256  MOV      #15000.,TIMER(R3) ;START 15 SECOND TIMER
109 041466 000402      BR       9$              ;EXIT
110 041470 004737 042334      8$:    JSR      PC,C18         ;PROCESS THE 'NED'
111 041474 032714 000100      9$:    BIT      #BIT06,(R4) ;SEE IF 'IE' ALREADY SET
112 041500 001002      BNE      10$            ;BR IF SET
    
```

```
113 041502 004737 045226      JSR    PC,SET.IE      ;SET 'IE' WITHOUT A 'TRE'
114 041506 012637 177776      10$:  MOV    (SP)+,@#PS  ;RESTORE PROC. STATUS
115 041512 104413                RESREG                ;RESTORE R0 - R5
116 041514 000207                RTS     PC
117
118      ;COMMAND INITIATOR
119
120      ;CALL:
121      MOV    #DRVNUM,R1      ;DRIVE NUMBER
122      MOV    #DPB,R2         ;ADDRESS OF DPB
123      JSR    PC,CI?         ;CI?= CI1,CI3, OR CI4
124      ;WHERE:
125      ;CI1=DATA TRANSFER
126      ;CI2=SEARCH REQUESTED BY DATA XFER
127      ;CI4=NOT DATA TRANSFER
128
129      ;START A DATA TRANSFER
130
131 041516 004737 045400      CI1:  JSR    PC,POPQUE    ;REMOVE REQUEST FROM 'DRIVES WAIT' QUEUE
132 041522 010237 040244      MOV    R2,TRNSWT        ;PUT REQ. IN TRANSFER WAIT QUEUE
133 041526 010203                MOV    R2,R3            ;DPB ADDRESS TO R3
134 041530 013704 040310      MOV    RMADR,R4         ;RMCS1 ADDRESS
135 041534 010164 000010      MOV    R1,RMCS2(R4)    ;SELECT DRIVE
136 041540 062703 000004      ADD    #4,R3            ;DESIRED WORD COUNT
137 041544 062704 000002      ADD    #2,R4            ;RMWC ADDRESS
138 041550 012324                MOV    (R3)+,(R4)+      ;LOAD WORD COUNT
139 041552 012324                MOV    (R3)+,(R4)+      ;LOAD BUFFER ADDRESS
140 041554 012346                MOV    (R3)+,-(SP)      ;LOAD SECTOR AND TRACK
141 041556 004037 044750      JSR    R0,WRT.RM        ;CALL THE WRITE ROUTINE
142      041562 000006                RMDA                   ;REGISTER OFFSET
143      041564 042334                CI8                    ;'NED' RETURN
144 041566 012346                MOV    (R3)+,-(SP)      ;LOAD CYLINDER ADDRESS
145 041570 004037 044750      JSR    R0,WRT.RM        ;CALL THE WRITE ROUTINE
146      041574 000034                RMDC                   ;REGISTER OFFSET
147      041576 042334                CI8                    ;'NED' RETURN
148 041600 004037 044670      JSR    R0,RD.RM         ;CALL THE READ ROUTINE
149      041604 000032                RMOF                   ;REGISTER OFFSET
150      041606 042334                CI8                    ;'NED' RETURN
151 041610 042716 001000      BIC    #BIT09,(SP)      ;CLEAR 'SKIP SECTOR ERROR INHIBIT'
152 041614 004037 044750      JSR    R0,WRT.RM        ;CALL THE WRITE ROUTINE
153      041620 000032                RMOF                   ;REGISTER OFFSET
154      041622 042334                CI8                    ;'NED' RETURN
155 041624 016246 000002      MOV    2(R2),-(SP)      ;LOAD 'COMMAND+GO', 'A17&A16', AND 'PSEL'
156 041630 004037 044750      JSR    R0,WRT.RM        ;CALL THE WRITE ROUTINE
157      041634 000000                RMCS1                  ;REGISTER OFFSET
158      041636 042334                CI8                    ;'NED' RETURN
159 041640 010137 040276      MOV    R1,DTUW          ;SET 'DATA TRANSFER UNDERWAY'
160 041644 000137 042312      JMP    CI5
161
162      ;START A SEARCH
163
164 041650 013704 040310      CI3:  MOV    RMADR,R4         ;RMCS1 ADDRESS
165 041654 010164 000010      MOV    R1,RMCS2(R4)    ;SELECT DRIVE
166 041660 016246 000012      MOV    12(R2),-(SP)    ;DESIRED CYLINDER ADDRESS
167 041664 004037 044750      JSR    R0,WRT.RM        ;CALL THE WRITE ROUTINE
168      041670 000034                RMDC                   ;REGISTER OFFSET
169      041672 042334                CI8                    ;'NED' RETURN
```

158	041674	116203	000010		MOVB	10(R2),R3	:PICKUP SECTOR ADDRESS
159	041700	163703	040320		SUB	MXWNDW,R3	:BACKUP BY MAX. SEARCH FOR I/O WINDOW
160	041704	002002			BGE	1\$	
161	041706	062703	000037		ADD	#31,R3	
162	041712	010346		1\$:	MOV	R3,-(SP)	:COMBINE THE ADJUSTED SECTOR WITH
163	041714	116266	000011	0000C1	MOVB	11(R2),1(SP)	:THE DESIRED TRACK
164	041722	004037	044750		JSR	RO,WRT.RM	:CALL THE WRITE ROUTINE
	041726	000006			RMDA		:REGISTER OFFSET
	041730	042334			CIB		: 'NED' RETURN
165	041732	012746	000131		MOV	#131,-(SP)	:START A SEARCH
166	041736	004037	044750		JSR	RO,WRT.RM	:CALL THE WRITE ROUTINE
	041742	000000			RMCS1		:REGISTER OFFSET
	041744	042334			CIB		: 'NED' RETURN
167	041746	156137	040300	040246	BISB	ATABIT(R1),SRCHWT	:SET 'SEARCH WAIT' KEY
168	041754	000556			BR	CIS	
169							
170							:INITIATE A NON-I/O OPERATION
171							
172	041756	013704	040310		MOV	RMADR,R4	:RMCS1 ADDRESS
173	041762	010164	000010		MOV	R1,RMCS2(R4)	:SELECT DRIVE
174	041766	116203	000002		MOVB	2(R2),R3	:PICKUP THE REQUESTED COMMAND
175	041772	122703	000131		CMPB	#131,R3	:IS IT A SEARCH COMMAND?
176	041776	001007			BNE	1\$:BRANCH IF NO
177	042000	016246	000010		MOV	10(R2),-(SP)	:LOAD DESIRED TRACK & SECTOR
178	042004	004037	044750		JSR	RO,WRT.RM	:CALL THE WRITE ROUTINE
	042010	000006			RMDA		:REGISTER OFFSET
	042012	042334			CIB		: 'NED' RETURN
179	042014	000403			BR	2\$:GO LOAD CYLINDER
180	042016	122703	000105		CMPB	#105,R3	:IS IT A SEEK COMMAND
181	042022	001007			BNE	3\$:BRANCH IF NO
182	042024	016246	000012		MOV	12(R2),-(SP)	:LOAD DESIRED CYLINDER
183	042030	004037	044750		JSR	RO,WRT.RM	:CALL THE WRITE ROUTINE
	042034	000034			RMDC		:REGISTER OFFSET
	042036	042334			CIB		: 'NED' RETURN
184	042040	000517			BR	14\$	
185	042042	122703	000115		CMPB	#115,R3	:IS IT AN 'OFFSET' COMMAND?
186	042046	001013			BNE	4\$:BR IF NO
187	042050	004037	044670		JSR	RO,RD.RM	:CALL THE READ ROUTINE
	042054	000032			RMOF		:REGISTER OFFSET
	042056	042334			CIB		: 'NED' RETURN
188	042060	116216	000001		MOVB	1(R2),(SP)	:BYTE WHEN LOADING THE REGISTER
189	042064	004037	044750		JSR	RO,WRT.RM	:CALL THE WRITE ROUTINE
	042070	000032			RMOF		:REGISTER OFFSET
	042072	042334			CIB		: 'NED' RETURN
190	042074	000501			BR	14\$	
191	042076	122703	000107		CMPB	#107,R3	:IS IT A 'RECALIBRATE' COMMAND?
192	042102	001006			BNE	5\$:IF NE, NO
193	042104	005046			CLR	-(SP)	:CYLINDER ZERO
194	042106	004037	044750		JSR	RO,WRT.RM	:CALL THE WRITE ROUTINE
	042112	000034			RMDC		:REGISTER OFFSET
	042114	042334			CIB		: 'NED' RETURN
195	042116	000470			BR	14\$:CONTINUE
196	042120	122703	000143		CMPB	#143,R3	:IS IT A 'SET FORMAT' COMMAND?
197	042124	001014			BNE	6\$:BRANCH IF NO
198	042126	004037	044670		JSR	RO,RD.RM	:CALL THE READ ROUTINE
	042132	000032			RMOF		:REGISTER OFFSET
	042134	042334			CIB		: 'NED' RETURN


```

199 042136 116266 000001 000001      MOVB    1(R2),1(SP)      ;COMBINE 'FMT16','ECI','HCI', & 'SSEI'
200 042144 004037 044750              JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
      042150 000032              RMOF                   ;REGISTER OFFSET
      042152 042334              CIB                      ;'NED' RETURN
201 042154 C00436                      BR      12$
202 042156 122703 000141      6$:    CMPB    #141,R3      ;IS IT A 'GET REGISTER' COMMAND?
203 042162 001023              BNE                      ;BRANCH IF NO
204 042164 016203 000006      7$:    MOV     6(R2),R3     ;POINTS TO 1ST ADDRESS OF WHERE
      205                          ;TO PUT THE REGISTER(S)
206 042170 116237 000010 042206      MOVB    10(R2),9$
207 042176 116205 000011              MOVB    11(R2),R5
208 042202 004037 044670      8$:    JSR     R0,RD.RM     ;INIT. THE INDEX FOR THE FIRST REG.
209 042206 000000 9$:          RMCS1    ;INDEX OF LAST REG. TO MOVE
      210                          ;READ RM80 REGISTER
      211                          ;INDEX OF REG. TO READ
210 042210 042334              CIB
211 042212 012623              MOV     (SP)+,(R3)+
212 042214 023705 042206      CMP     9$,R5           ;GET THE CONTENTS OF RH70/RM80 REG.
213 042220 001414              BEQ     12$             ;LAST REG. BEEN READ?
214 042222 062737 000002 042206      ADD     #2,9$          ;GET OUT IF YES
215 042230 000764              BR      8$              ;INCREASE THE INDEX BY 2
216 042232 122703 C00145      10$:   CMPB    #145,R3     ;LOOP--MORE TO READ
      217                          ;IS IT A 'SELECT DRIVE' COMMAND?
      218                          ;BRANCH IF YES
      219                          ;LOAD THE COMMAND
218 042240 010346 11$:    MOV     R3,-(SP)       ;CALL THE WRITE ROUTINE
219 042242 004037 044750      JSR     R0,WRT.RM
      220                          ;REGISTER OFFSET
      221                          ;'NED' RETURN
      222                          ;REMOVE REQ. FROM QUEUE
220 042252 004737 045400 12$:   JSR     PC,POPQUE
221 042256 052762 000200 000016      BIS    #BIT07,16(R2)   ;SET THE 'DONE' BIT
222 042264 005737 040252      TST    SAVEFG          ;SAVE THE RH70/RM80 REGISTERS?
223 042270 100002              BPL     13$            ;BRANCH IF NO
224 042272 004737 045062      JSR     PC,SVRH70      ;YES--GO SAVE THE REGISTERS
225 042276 000207 13$:   RTS     PC              ;RETURN TO USER
226
227 ;START A NON-DATA TRANSFER OPERATION
228
229 042300 010346 14$:   MOV     R3,-(SP)       ;LOAD THE COMMAND
230 042302 004037 044750      JSR     R0,WRT.RM
      231                          ;CALL THE WRITE ROUTINE
      232                          ;REGISTER OFFSET
      233                          ;'NED' RETURN
      234                          ;START THE COMMAND TIMER
231 042312 006301 C15:   ASL     R1              ;SET A 10 SECOND TIMER
232 042314 012761 023420 040256      MOV    #10000.,TIMER(R1)
233 042322 006201      ASR     R1
234 042324 112761 000001 040154      MOVB   #1,DRVACT(R1)   ;SET THE DRIVE ACTIVE
235 042332 000207      RTS     PC              ;RETURN TO THE USER
236
237 ;PROCESS A NON-EXISTENT DRIVE
238
239
240
241 C18:   SAVREG
242 042334 104412      TSTB   DRVACT(R1)     ;SAVE R0 - R5
243 042336 105761 040154      BEQ    3$              ;DRIVE ACTIVE?
244 042342 001431      MOV    TRNSWT,R2      ;BRANCH IF NO
245 042344 013702 040244      CMP    R1,DTUW        ;GET THE 'TRANSFER WAIT' QUEUE
246 042350 020137 040276      BEQ    1$              ;DID THIS DRIVE HAVE AN I/O IN PROGRESS?
247 042354 001402      JSR    PC,GETREQ      ;BRANCH IF YES
248 042356 004737 045366      TST    R2              ;GET THE DPB POINTER
249 042362 005702      ;QUEUE ENTRY FOR DRIVE ?
    
```

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

```

1          ;INTERRUPT SERVICE ROUTINE
2
3 042452 112737 000001 040250 ISR:  MOVB  #1,ACTDRV  ;SET 'ACTIVE DRIVER' FLAG
4 042460 104412          SAVREG  ;SAVE R0 - R5
5 042462 013704 040310          MOV   RMADR,R4  ;ADDRESS OF RMCS1
6 042466 013701 040276          MOV   DTUW,R1  ;GET 'DATA TRANSFER UNDERWAY' INDICATOR
7 042472 002402          BLT   1$      ;BRANCH IF NO DATA TRANSFER UNDERWAY
8 042474 004737 042514          JSR   PC,TD    ;CALL TRANSFER DONE
9 042500 004737 043334          1$: JSR   PC,SC   ;CALL SPECIAL CONDITIONS
10 042504 104413          RESREG ;RESTORE R0 - R5
11 042506 105037 040250          CLRB  ACTDRV  ;CLEAR 'ACTIVE DRIVER' FLAG
12 042512 000002          RTI   ;RETURN
13
14          ;TRANSFER DONE ROUTINE
15
16 042514 105061 040154          TD:  CLRB  DRVACT(R1) ;SET DRIVE ACTIVE INDICATOR TO IDLE
17 042520 012737 177777 040276          MOV   #-1,DTUW ;NO DATA TRANSFERS UNDERWAY
18 042526 006301          ASL   R1
19 042530 012761 177777 040256          MOV   #-1,TIMER(R1) ;CANCEL TIMEOUT
20 042536 006201          ASR   R1
21 042540 013702 040244          MOV   TRNSW,R2 ;GET 'DPB' ADDRESS FROM THE
22 042544 005037 040244          CLR   TRNSW  ;TRANSFER WAIT QUEUE--CLEAR QUEUE
23 042550 052762 000200 000016          BIS   #BIT07,16(R2) ;SET DONE
24 042556 010164 000010          MOV   R1,RMCS2(R4) ;SELECT THE DRIVE
25 042562 004037 044670          JSR   R0,RD.RM ;CALL THE READ ROUTINE
    042566 000000          RMCS1 ;REGISTER OFFSET
    042570 042334          CIB   ;'NED' RETURN
26 042572 006126          ROL   (SP)+  ;IS TRE=1 ?
27 042574 100421          BMI   3$    ;BR IF YES
28 042576 005737 040252          TST   SAVEFG ;SAVE THE RH70/RM80 REGISTERS?
29 042602 100002          BPL   1$    ;BRANCH IF NO
30 042604 004737 045062          JSR   PC,SVRH70 ;YES--SAVE THE REGISTERS
32 042610 004737 043246          1$: JSR   PC,WC.HK ;SEE IF WRITE CHECK TO BE PUT IN QUEUE
33 042614 004737 043366          JSR   PC,GETREQ ;GET DPB POINTER
34 042620 005702          TST   R2    ;ENTRY FOR DRIVE ?
35 042622 001403          BEQ   2$    ;BR IF NOT
36 042624 004737 041236          JSR   PC,OPT ;CALL OPTIMIZER
40 042630 000207          RTS   PC    ;RETURN
41 042632 012714 000113          2$: MOV   #113,(R4) ;RELEASE THE DRIVE
42 042636 000207          RTS   PC    ;RETURN
43
44          3$:
    042640          JSR   R0,RD.RM ;CALL THE READ ROUTINE
    042640 004037 044670          RMER1 ;REGISTER OFFSET
    042644 000014          CIB   ;'NED' RETURN
45 042650 032726 000600          BIT   #BIT8!BIT7,(SP)+ ;SEE IF HCRC OR HCE ERRORS
46 042654 001016          BNE   4$    ;IF NE, YES
47 042656 004037 044670          JSR   R0,RD.RM ;CALL THE READ ROUTINE
    042662 000042          RMER2 ;REGISTER OFFSET
    042664 042334          CIB   ;'NED' RETURN
48 042666 032726 000040          BIT   #SSE,(SP)+ ;SEE IF SKIP SECTOR ERROR
49 042672 001407          BEQ   4$    ;IF EQ, NO
50 042674 004037 044670          JSR   R0,RD.RM ;CALL THE READ ROUTINE
    042700 000032          RMOF ;REGISTER OFFSET
    042702 042334          CIB   ;'NED' RETURN
51 042704 032726 001000          BIT   #SSEI,(SP)+ ;IS THE INHIBIT BIT ALREADY SET ?
52 042710 001416          BEQ   ;IF EQ, NO
    
```

```

53 042712 052762 100100 000016 4$: BIS #BIT15!BIT06,16(R2) ;SET DATA ERROR FLAG
54 042720 004737 045332 JSR PC,EMPTYQ ;EMPTY THE 'DRIVE'S WAIT' QUEUE
55 042724 004737 045052 JSR PC,SVRH70 ;SAVE THE RH70/RM80 REGISTERS
56 042730 012714 040111 MOV #40111,(R4) ;ISSUE A 'DRIVE CLEAR'
60 042740 012714 000113 MOV #113,(R4) ;ISSUE A RELEASE TO THE DRIVE
61 042744 000207 RTS PL ;RETURN
62
63 ;SKIP SECTOR HANDLING ROUTINE
64
65 042746 010137 040276 SKIP: MOV R1,DTUW ;LOAD ACTIVE DRIVE NUMBER
66 042752 010237 040244 MOV R2,TRNSWT ;RESTORE TRANSFER FLAG
67 042756 005062 000016 CLR STATUS(R2) ;CLEAR THE DRIVER STATUS
68 042762 132762 000002 000002 BITB #2,$COMND(R2) ;SEE IF HEADER ORDER
70 042770 001447 BEQ 4$ ;IF EQ, NO
74 042772 004037 044670 JSR R0,RD.RM ;CALL THE READ ROUTINE
    042776 000034 RMDC ;REGISTER OFFSET
    043000 042334 C18 ;'NED' RETURN
75 043002 012662 000012 MOV (SP)+,$CYL(R2) ;SAVE NEW STARTING CYLINDER ADDRESS
76 043006 004037 044670 JSR R0,RD.RM ;CALL THE READ ROUTINE
    043012 000006 RMDA ;REGISTER OFFSET
    043014 042334 C18 ;'NED' RETURN
77 043016 012662 000010 MOV (SP)+,$SEC(R2) ;SAVE NEW STARTING TRK/SEC ADDRESS
78 043022 004037 044670 JSR R0,RD.RM ;CALL THE READ ROUTINE
    043026 000002 RMWC ;REGISTER OFFSET
    043030 042334 C18 ;'NED' RETURN
79 043032 012600 MOV (SP)+,R0 ;GET REMAINING WORD COUNT
80 043034 001005 BNE 2$ ;BR IF PARTIAL SECTOR LEFT
82 043036 016200 000004 MOV $WCNT(R2),R0 ;STARTING WORD COUNT
86 ;IF RMWC WAS AT ZERO AFTER TRANSFER,
87 043042 162700 177376 1$: SUB #-258.,R0 ;FIND THE # NUMBER OF WORDS FOR LAST SECTOR
88 043046 002775 BLT 1$ ;BR IF NOT DONE YET
89 043050 062700 177376 2$: ADD #-258.,R0 ;ADD 1 SECTOR BACK
90 043054 020027 177772 CMP R0,#-6 ;IS WORD COUNT AT LEAST 6 ?
91 043060 003402 BLE 3$ ;BR IF YES
92 043062 012700 177772 MOV #-6,R0 ;SET WORD COUNT TO 6
93 043066 010062 000004 3$: MOV R0,$WCNT(R2) ;STORE NEW WORD COUNT IN DPB
94 043072 010062 000020 MOV R0,$WRDL(R2) ;STORE NEW WORD COUNT LENGTH IN DPB AND
95 043076 005462 000020 NEG $WRDL(R2) ;MAKE IT POSITIVE
96 043102 016203 000006 MOV $BUF(R2),R3 ;STARTING BUFFER ADDRESS
97 043106 000417 BR 5$
98
99 043110 4$: JSR R0,RD.RM ;CALL THE READ ROUTINE
    043110 004037 044670 RMWC ;REGISTER OFFSET
    043114 000002 C18 ;'NED' RETURN
105 043120 166216 000004 SUB $WCNT(R2),(SP) ;CALCULATE THE NUMBER OF WORD TRANSFERED
106 043124 042716 000377 BIC #377,(SP) ;LEAVE ONLY SECTOR MULTIPLES
107 043130 011603 MOV (SP),R3 ;COPY THE DIFFERENCE
108 043132 066216 000004 ADD $WCNT(R2),(SP) ;NEW WORD COUNT
109 043136 012600 MOV (SP)+,R0 ;COPY THE WORD COUNT
110 043140 000303 ASL R3 ;CONVERT WORD DIFFERENCE TO A BYTE DIFFERENCE
111 043142 066203 000006 ADD $BUF(R2),R3 ;NEW BUFFER ADDRESS
113 043146 010346 5$: MOV R3,-(SP) ;BUFFER ADDRESS
117 043150 004037 044750 JSR R0,WRT.RM ;CALL THE WRITE ROUTINE
    043154 000004 RMBA ;REGISTER OFFSET
    043156 042334 C18 ;'NED' RETURN
118 043160 010046 MOV R0,-(SP) ;WORD COUNT
    
```

```

119 043162 004037 044750      JSR      R0,WRT.RM      ;CALL THE WRITE ROUTINE
      043166 000002      RMWC      ;REGISTER OFFSET
      043170 042334      CI8      ;'NED' RETURN
120 043172 012714 040111      MOV      #40111,(R4)    ;CLEAR THE DRIVE
121 043176 004037 044670      JSR      R0,RD.RM      ;CALL THE READ ROUTINE
      043202 000032      RMOF     ;REGISTER OFFSET
      043204 042334      CI8      ;'NED' RETURN
122 043206 052716 001000      BIS      #SSEI,(SP)    ;SET THE INHIBIT BIT
123 043212 004037 044750      JSR      R0,WRT.RM      ;CALL THE WRITE ROUTINE
      043216 000032      RMOF     ;REGISTER OFFSET
      043220 042334      CI8      ;'NED' RETURN
124 043222 012762 000001 000112  MOV      #1,$SSENB(R2)  ;INDICATE THAT SKIP SECTORING WAS ENABLED
125 043230 016246 000002      MOV      $COMND(R2),-(SP) ;GET THE ORIGINAL COMMAND
126 043234 004037 044750      JSR      R0,WRT.RM      ;CALL THE WRITE ROUTINE
      043240 000000      RMCS1   ;REGISTER OFFSET
      043242 042334      CI8      ;'NED' RETURN
127 043244 000207      RTS      PC      ;RETURN
128
129
130      ;FORCED WRITE CHECK ROUTINE
131
133 043246 005737 001502      WC.HK:  TST      WRTCHK      ;DO WRITE CHECK ?
134 043252 001427      BEQ      2$      ;BR IF NOT
135 043254 122762 000161 000002  CMPB     #WRTDAT,$COMND(R2) ;LAST OPERATION A WRITE COMMAND ?
139 043262 001023      BNE      2$      ;BR IF NOT
140 043264 004037 045344      1$:     JSR      R0,DRVQUE      ;PUT THE OPERATION IN THE QUEUE
141 043270 000420      BR      2$      ;QUEUE IS FULL
142 043272 005062 000016      CLR      STATUS(R2)    ;CLEAR 'DONE' BIT IN DPB
143 043276 116262 002140 000027  MOVB     $RMCS1(R2),$PREV0(R2) ;SAVE WRITE OPERATION CODE
144 043304 016262 000012 000034  MOV      $CYL(R2),$PREVA+2(R2) ;SAVE CYLINDER
145 043312 016262 000010 000032  MOV      $SEC(R2),$PREVA(R2)  ;SAVE SECTOR AND TRACK ADDRESSES
146 043320 105062 000024      CLRB     $CODE(R2)     ;CHANGE WRITE DATA TO WRITE CHECK
147 043324 112762 000151 000002  MOVB     #WCKD,$COMND(R2) ;CHANGE FUNCTION CODE TO WRITE CHECK
148 043332 000207      2$:     RTS      PC      ;EXIT
149
150
151      ;SPECIAL CONDITION ROUTINE
152
153 043334 116403 000016      SC:     MOVB     RMAS(R4),R3    ;READ 'RMAS'
154 043340 001012      BNE      2$      ;BRANCH IF ANY 'ATA' BITS SET
155 043342 004037 044670      JSR      R0,RD.RM      ;CALL THE READ ROUTINE
      043346 000000      RMCS1   ;REGISTER OFFSET
      043350 042334      CI8      ;'NED' RETURN
156 043352 106126      ROLB     (SP)+        ;IS 'IE'=1?
157 043354 100403      BMI      1$      ;YES, NO DRIVES TO CHECK
158 043356 104001      EMT      1          ;REPORT AN ILLEGAL INTERRUPT
159 043360 004737 045226      JSR      PC,SET.IE    ;SET INTERRUPT ENABLE
160 043364 000207      1$:     RTS      PC      ;RETURN
161 043366 005046      2$:     CLR      -(SP)     ;PROCESS ALL DRIVES THAT HAVE
162 043370 110316      MOVB     R3,(SP)     ;AN 'ATA'=1
163 043372 012703 000001      MOV      #1,R3
164 043376 005001      CLR      R1
165
166      ;PROCESS ALL DRIVES WITH 'ATA' SET
167
168 043400 030316      SC3:    BIT      R3,(SP)    ;ATA=1?
169 043402 001005      BNE      SC5        ;YES--BRANCH
170 043404 005201      SC4:    INC      R1          ;MOVE TO THE NEXT DRIVE
171 043406 106303      ASLB     R3
    
```

```

172 043410 001373          BNE    SC3          ;BRANCH IF MORE TO CHECK?
173 043412 005726          TST    (SP)+        ;CLEAN OFF THE STACK
174 043414 000207          RTS     PC           ;RETURN TO USER
175
176          ;DETERMINE IF THE DRIVE WITH 'ATA' SET IS ACTIVE WITH A COMMAND
177
178 043416 105761 040204    SC5:   TSTB   DPINT(R1) ;INITIALIZING THE DRIVE ?
179 043422 001402          BEQ    1$           ;BR IF NOT
180 043424 000137 044012    JMP    SC13         ;PROCESS THE DRIVE
181 043430 105761 040214    1$:   TSTB   DPRQS(R1) ;PORT REQUEST OUTSTANDING ?
182 043434 001402          BEQ    2$           ;BR IF NOT
183 043436 000137 044012    JMP    SC13         ;START THE OUTSTANDING COMMAND
184 043442 105761 040154    2$:   TSTB   DRVACT(R1) ;DRIVE ACTIVE ?
185 043446 001016          BNE    SC6           ;BR IF ACTIVE
186
187          ;THE DRIVE WAS NOT ACTIVE, FIND OUT WHY IT INTERRUPTED
188
189 043450 004737 043750    JSR    PC,SC12      ;SAVE RMD5, RMER1 AND RMER2
190          ;ALSO DO A DRIVE INIT (DRVINT)
191 043454 105761 040204    TSTB   DPINT(R1)   ;TRYING TO INIT THE DRIVE ?
192 043460 001351          BNE    SC4           ;BR IF YES, CHECK ON MORE DRIVES
193 043462 032737 020000 040154  BIT    #BIT13,RMERRS+6 ;ADDRESS PLUG CHANGED ?
194 043470 001003          BNE    4$           ;BR IF YES
195 043472 011605    3$:   MOV    (SP),R5    ;PICKUP (RMAS) BEFORE THE ERROR CALL:
196 043474 104002          EMT    2            ;REPORT THE UNEXPECTED ATTENTION
197 043476 000742          BR     SC4           ;GO CHECK FOR MORE ATA'S
198 043500          4$:   EMT    5            ;REPORT THE ADDRESS PLUG CHANGE
199 043502 104005          BR     SC4           ;CHECK FOR MORE DRIVES
200
201          ;THE DRIVE COMPLETED A NON-I/O COMMAND
202
203 043504 006301          SC6:   ASL    R1            ;SETUP TO ADDRESS WORDS
204 043506 012761 177777 040256  MOV    #-1,TIMER(R1) ;STOP THE TIMER
205 043514 006201          ASR    R1            ;RESTORE THE DRIVE ADDRESS
206 043516 004737 045366    JSR    PC,GETREQ    ;GET THE DPB POINTER FROM THE QUEUE
207 043522 010164 000010    MOV    R1,RMCS2(R4) ;SELECT DRIVE
208 043526 004037 044670    JSR    R0,RD.RM     ;CALL THE READ ROUTINE
209          RMD5          ;REGISTER OFFSET
210          SC8          ;'NED' RETURN
211 043536 011605    MOV    (SP),R5      ;AND PUT IT IN R5
212 043540 006126    ROL    (SP)+        ;WAS THERE AN ERROR?
213 043542 100053    BPL    SC11         ;BR IF NO ERROR
214 043544 004037 044670    JSR    R0,RD.RM     ;CALL THE READ ROUTINE
215          RMER1      ;REGISTER OFFSET
216          SC8          ;'NED' RETURN
217 043554 012605    MOV    (SP)+,R5     ;AND SAVE IT IN R5
218 043556 004737 045062    JSR    PC,SVRH70    ;SAVE RH70/RM80 REGISTERS
219 043562 012746 000111    MOV    #11,-(SP)   ;ISSUE A DRIVE CLEAR
220 043566 004037 044750    JSR    R0,WRT.RM    ;CALL THE WRITE ROUTINE
221          RMCS1      ;REGISTER OFFSET
222          SC8          ;'NED' RETURN
223 043576 006105    ROL    R5           ;WAS 'UNSAFE' CONDITION =1?
224 043600 100404    BMI    1$           ;BRANCH IF YES
225 043602 052762 100240 000016  BIS    #BIT15!BIT07!BIT05,16(R2) ;INFORM USER OF ERROR
226 043610 000403          BR     2$           ;
227 043612 052762 100220 000016  1$:   BIS    #BIT15!BIT07!BIT04,16(R2) ;INFORM USER OF UNSAFE ERROR
    
```

```

222 043620 105061 040154      2$:  CLRB   DRVACT(R1)      ;SET DRIVE TO IDLE
223 043624 004737 045332      JSR    PC,EMPTYQ      ;DUMP THE QUEUE
224 043630 146137 040300 040246  BICB   ATABIT(R1),SRCHWT ;CLEAR THE SEARCH WAIT FLAG
225 043636 012746 000113      MOV    #113,-(SP)     ;RELEASE COMMAND
226 043642 004037 044750      JSR    RO,WRT.RM      ;CALL THE WRITE ROUTINE
      043646 000000      RMCS1                ;REGISTER OFFSET
      043650 043656      SC8                  ;'NED' RETURN
227 043652 000137 043404      JMP    SC4            ;CHECK FOR MORE DRIVES
228
229
230
231 043656 004737 045332      SC8:   JSR    PC,EMPTYQ ;CLEAR THE DRIVE'S QUEUE
232 043662 004737 042334      JSR    PC,C18         ;PROCESS THE 'NED'
233 043666 000137 043404      JMP    SC4            ;CHECK MORE DRIVES
234
235
236
237 043672 105061 040154      SC11:  CLRB   DRVACT(R1) ;SET DRIVE IDLE
238 043676 136137 040300 040246  BITB   ATABIT(R1),SRCHWT ;DOING A SEARCH OPERATION FOR
      ;AN I/O COMMAND?
239
240 043704 001012      BNE    1$            ;BRANCH IF YES
241 043706 004737 045400      JSR    PC,POPQUE     ;REMOVE REQUEST FROM QUEUE
242 043712 052762 000200 000016  BIS    #BIT07,16(R2) ;SET 'DONE' BIT
243 043720 005737 040252      TST   SAVEFG        ;SAVE THE REGISTERS?
244 043724 100002      BPL    1$            ;BRANCH IF NO
245 043726 004737 045062      JSR    PC,SVRH70     ;YES--SAVE ALL OF THE RH70/RM80 REG'S
246 043732 116164 040300 000016  1$:   MOVB   ATABIT(R1),RMAS(R4) ;CLEAR ATTENTION BIT
247 043740 004737 041236      JSR    PC,OPT        ;START A REQUEST
248 043744 000137 043404      JMP    SC4            ;CHECK FOR MORE DRIVES
249
250
251
252 043750 010164 000010      SC12:  MOV    R1,RMCS2(R4) ;SELECT DRIVE
253 043754 016437 000012 040146  MOV    RMDS(R4),RMERRS ;SAVE THE FOUR REGISTERS THAT
254 043762 016437 000014 040150  MOV    RMER1(R4),RMERRS+2 ;WILL TELL US SOMETHING
255 043770 016437 000042 040152  MOV    RMER2(R4),RMERRS+4
256 043776 004037 040534      JSR    RO,DRVINT     ;INIT. THE STATE OF THE DRIVE
257 044002 000401      BR    1$            ;TAKE ERROR EXIT
258 044004 000207      RTS   PC            ;RETURN
259 044006 005726      1$:   TST   (SP)+      ;POP PC OFF OF THE STACK
260 044010 000722      BR    SC8           ;PROCESS THE PARITY ERROR
261
262
263
264 044012 006301      SC13:  ASL    R1            ;SETUP TO ADDRESS WORDS
265 044014 012761 177777 040256  MOV    #-1,TIMER(R1) ;STOP THE TIMER
266 044022 006201      ASR   R1            ;
267 044024 010164 000010      MOV    R1,RMCS2(R4) ;SELECT THE DRIVE
268 044030 116164 040300 000016  MOVB   ATABIT(R1),RMAS(R4) ;CLEAR THE ATTENTION BIT
269 044036 032714 004000      BIT   #BIT11,(R4)   ;DRIVE AVAILABLE ?
270 044042 001006      BNE   1$            ;BR IF AVAILABLE
271 044044 006301      ASL   R1            ;
272 044046 012761 035230 040256  MOV    #15000.,TIMER(R1) ;START 15 SECOND TIMER AGAIN
273 044054 006201      ASR   R1            ;
274 044056 000433      BR    3$            ;EXIT
275 044060 105761 040204      1$:   TSTB   DPINT(R1)   ;INITIALIZING THE DRIVE ?
276 044064 001424      BEQ   2$            ;BR IF NOT
    
```



```

1      ;RM80 TIMER ROUTINE
2      :CALL:
3      :
4      :      MOV      #TIME,-(SP)      ;ELAPSED TIME IN MILLISECONDS ON THE STACK
5      :      JSR      PC,RMTMR        ;CALL RM80 TIME ROUTINE
6 044152 005737 040250 RMTMR: TST      ACTDRV      ;CHECK 'ACTDRV & ACTSTR'
7 044156 001030      BNE      4$          ;IF NON ZERO EXIT
8 044160 112737 000001 040251 MOVB     #1,ACTSTR    ;SET 'ACTSTR'
9 044166 104412      SAVREG     ;SAVE R0 - R5
10 044170 005001      CLR      R1          ;START WITH DRIVE 0
11 044172 005003      CLR      R3
12 044174 005763 040256 1$: TST      TIMER(R3)    ;IS THE TIMER RUNNING?
13 044200 002407      BLT      2$          ;BRANCH IF NO
14 044202 166663 000002 040256 SUB      2(SP),TIMER(R3) ;COUNT THE INTERVAL
15 044210 003003      BGT      2$          ;BR IF NO SOFTWARE TIMEOUT
16 044212 004737 044244      JSR      PC,STO      ;CALL SOFTWARE TIMEOUT ROUTINE
17 044216 000405      BR      3$          ;GO TO THE EXIT
18 044220 005201 2$: INC      R1          ;MOVE TO NEXT DRIVE
19 044222 005723      TST      (R3)+
20 044224 022701 000010      CMP      #8.,R1     ;OUT OF DRIVES?
21 044230 003361      BGT      1$          ;BRANCH IF NO
22 044232 104413 3$: RESREG     ;RESTORE R0 - R5
23 044234 105037 040251 CLRB     ACTSTR      ;ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
24 044240 012616 4$: MOV      (SP)+,(SP) ;ADJUST THE STACK
25 044242 000207      RTS      PC        ;RETURN
26
27      ;SOFTWARE TIMEOUT ROUTINE
28
29      :NOTE: THIS ROUTINE MUST BE ENTERED AT PRIORITY 6
30      :OR GREATER
31
32      :CALL:
33      :      STO      #DRVNUM,R1      ;DRIVE NUMBER
34      :      JSR      PC,STO          ;CALL:
35      :      RETURN
36
37 044244 010146 STO: MOV      R1,-(SP)    ;SAVE R1
38 044246 010346      MOV      R3,-(SP)    ;SAVE R3
39 044250 013704 040310      MOV      RMADR,R4      ;GET ADDRESS OF 'RMCS1'
40 044254 010164 000010      MOV      R1,RMCS2(R4)  ;SELECT THE DRIVE
41 044260 004037 044670      JSR      R0,RD.RM      ;CALL THE READ ROUTINE
42 044264 000012      RMDS      ;REGISTER OFFSET
43 044266 044556      STOS     ;'NED' RETURN
44 044270 105726      TSTB     (SP)+        ;IS 'DRY'=1?
45 044272 100473      BMI      ST02        ;BR IF YES
46 044274 105761 040204 ST01: TSTB     DPINT(R1)   ;TRYING TO INTIALIZE THE DRIVE ?
47 044300 001070      BNE      ST02        ;BR IF YES
48 044302 105761 040214      TSTB     DPRQS(R1)    ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
49 044306 001065      BNE      ST02        ;BR IF YES
50 044310 013702 040244      MOV      TRNSWT,R2    ;PICKUP TRANSFER WAIT QUEUE
51 044314 020137 040276      CMP      R1,DTUW      ;TRANSFER UNDERWAY ON THIS DRIVE?
52 044320 001402      BEQ      1$          ;BRANCH IF YES
53 044322 004737 045366      JSR      PC,GETREQ    ;GET DPB ADDRESS
54 044326 052762 101000 000016 1$: BIS      #BIT15!BIT09,16(R2) ;SET THE ERROR FLAGS
55 044334 004737 045062      JSR      PC,SVRH70    ;SAVE RH70/RM80 REGISTERS
56 044340 012764 000040 000010      MOV      #BIT05,RMCS2(R4) ;'INIT' THE MASS BUS
57 044346 105061 040154      CLRB     DRVACT(R1)  ;DRIVE IS IDLE
    
```

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~

```

56 044352 005001 CLR R1 ;START WITH DRIVE 0
57 044354 005003 CLR R3
58 044356 004037 040534 2$: JSR R0,DRVINT ;INIT. THIS DRIVE
59 044362 00047 BR ST05 ;PARITY ERROR RETURN
60 044364 105761 040154 TSTB DRVACT(R1) ;DRIVE IDLE BEFORE THE INIT.?
61 044370 001414 BEQ 4$ ;YES--BRANCH
62 044372 013702 040244 MOV TRNSWT,R2 ;GET TRANSFER WAIT QUEUE
63 044376 023701 040276 CMP DTUW,R1 ;WAS THERE I/O ON THIS DRIVE?
64 044402 001402 BEQ 3$ ;YES--BRANCH
65 044404 004737 045366 JSR PC,GETREQ ;GET THE DPB POINTER FROM QUEUE
66 044410 052762 100400 000016 3$: BIS #BIT15!BIT10,16(R2) ;INFORM USER OF INIT.
67 044416 105061 040154 CLRB DRVACT(R1) ;SET DRIVE ACTIVE TO IDLE
68 044422 012763 177777 040256 4$: MOV #-1,TIMER(R3) ;STOP THE TIMER
69 044430 005723 TST (R3)+ ;UPDATE THE INDEX
70 044432 005201 INC R1 ;INCREMENT THE DRIVE NUMBER
71 044434 022701 000010 CMP #8.,R1 ;LAST DRIVE BEEN CHECKED?
72 044440 003346 BGT 2$ ;NO--LOOP
73 044442 012737 177777 040276 MOV #-1,DTUW ;NO DATA TRANSFERS UNDERWAY
74 044450 005037 040244 CLR TRNSWT ;CLEAR TRANSFER WAIT QUEUE
75 044454 004737 045300 JSR PC,CLRQUE ;CLEAR ALL REQUEST QUEUES
76 044460 000500 BR ST09 ;EXIT
77 044462 116405 000016 ST02: MOVB RMA5(R4),R5 ;READ ATTENTION REG
78 044466 136105 040300 BITB ATABIT(R1),R5 ;IS ATTENTION FOR THIS DRIVE UP ?
79 044472 001017 BNE ST03 ;YES--BRANCH
80 044474 105761 040204 TSTB DPINT(R1) ;TRYING TO INITIALIZE THE DRIVE ?
81 044500 001031 BNE ST06 ;BR IF YES - DRIVE NOT ONLINE
82 044502 105761 040214 TSTB DPRQS(R1) ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
83 044506 001045 BNE ST07 ;BR IF YES - NO RESPONSE TO REQUEST
84 044510 020137 040276 CMP R1,DTUW ;DATA TRANSFER UNDERWAY FOR THIS DRIVE
85 044514 001267 BNE ST01 ;BR IF NO
86 044516 004037 044670 JSR R0,RD.RM ;CALL THE READ ROUTINE
    044522 000000 'CS1 ;REGISTER OFFSET
    044524 044556 ST05 ;'NED' RETURN
87 044526 105726 TSTB (SP)+
88 044530 100261 BPL ST01 ;BR IF 'RDY'=0
89 044532 105761 040204 ST03: TSTB DPINT(R1) ;INITIALIZING THE DRIVE ?
90 044536 001003 BNE 1$ ;BR IF INIT PENDING
91 044540 105761 040214 TSTB DPRQS(.1) ;PORT REQUEST PENDING ?
92 044544 001446 BEQ ST09 ;BR IF NOT
93 044546 012763 177777 040256 1$: MOV #-1,TIMER(R3) ;STOP THE TIMER
94 044554 000442 BR ST09 ;EXIT
95 044556 004737 042334 ST05: JSR PC,C18 ;GO HANDLE THE 'NED'
96 044562 000437 BR ST09
97 044564 105061 040204 ST06: CLRB DPINT(R1) ;CLEAR THE INITIALIZE INDICATOR
98 044570 105061 040164 CLRB DRVSTA(R1) ;SET DRIVE OFFLINE
99 044574 012763 177777 040256 MOV #-1,TIMER(R3) ;STOP THE TIMER
100 044602 004737 045366 JSR PC,GETREQ ;GET THE DPB ADDRESS
101 044606 005702 TST R2 ;REQUEST IN QUEUE ?
102 044610 001424 BEQ ST09 ;BR IF NOT
103 044612 052762 140000 000016 BIS #BIT15!BIT14,16(R2) ;INFORM THE USER DRIVE NOT AVAILABLE
104 044620 000414 BR ST08 ;FINISH
105 044622 012763 177777 040256 ST07: MOV #-1,TIMER(R3) ;STOP THE TIMER
106 044630 105061 040214 CLRB DPRQS(R1) ;CLEAR PORT REQUEST INDICATOR
107 044634 004737 045366 JSR PC,GETREQ ;GET DPB ADDRESS
108 044640 005702 TST R2 ;QUEUE ENTRY FOR DRIVE ?
109 044642 001407 BEQ ST09 ;BR IF NONE
110 044644 012762 100004 000016 MOV #BIT15!BIT2,16(R2) ;INFORM USER OF PORT REQUEST ERROR
    
```

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

```

1      :ROUTINE TO READ A RH70/RM80 REGISTER
2
3      :CALL:
4      JSR      RO,RD.RM      ;GO READ A REGISTER
5      INDEX   ;REG. INDEX FROM BASE
6      ERRADR  ;ERROR ADDRESS--PROCESS ERROR STARTING
7      ;AT THIS ADDRESS
8      RETURN  ;CONTENTS OF REG. IS ON THE STACK
9
10     044670  011646      RD.RM:  MOV      (SP),-(SP)      ;SAVE R0 FOR RETURN
11     044672  013737  040310  044706  MOV      RMADR,RD.ADR  ;FORM THE DESIRED ADDRESS
12     044700  062037  044706      ADD      (R0)+,RD.ADR  ;USING THE BASE AND THE INDEX
13     044704  013727      RD.RM1: MOV      @ (PC)+,(PC)+  ;READ THE DESIRED REGISTER OF THE RM80
14     044706  000000      RD.ADR: .WORD   0      ;ADDRESS IS FORMED HERE
15     044710  000000      RD.WRD: .WORD   0      ;REG. CONTENTS PUT HERE
16     044712  013766  044710  000002  MOV      RD.WRD,2(SP)  ;RETURN IT TO THE USER
17     044720  013746  040310      MOV      RMADR,-(SP)  ;PUT THE ADDRESS ON THE STACK
18     044724  062716  000010      ADD      #RMCS2,(SP)  ;FORM THE ADDRESS OF RMCS2
19     044730  032736  010000      BIT      #BIT12,@(SP)+ ;CHECK THE 'NED' BIT
20     044734  001002      BNE     RD.RM3      ;IF NE, DRIVE NOT PRESENT
21     044736  005720      TST     (R0)+      ;ERROR FREE RETURN
22     044740  000402      BR      RD.RM4      ;EXIT
23     044742  011000      RD.RM3: MOV      (R0),R0  ;ERROR EXIT
24     044744  012616      MOV     (SP)+,(SP)
25     044746  000200      RD.RM4: RTS      R0
26
27      :ROUTINE TO WRITE A REGISTER
28
29      :CALL:
30      MOV      DATA,-(SP)  ;DATA TO BE LOADED ON THE STACK
31      JSR      RO,WRT.RM    ;CALL THE ROUTINE TO LOAD (WRITE) THE REG.
32      INDEX   ;INDEX OF THE REGISTER TO BE LOADED
33      'NED' RETURN ;ADDRESS TO RETURN TO IF 'NED' ERROR
34      RETURN  ;ERROR FREE RETURN
35
36     044750  016637  000002  045030  WRT.RM: MOV      2(SP),WRT.WD  ;SAVE THE WORD TO WRITE
37     044756  012616      MOV     (SP)+,(SP)    ;ADJUST THE STACK
38     044760  012037  045032      MOV     (R0)+,WRT.AD  ;GET INDEX OF REGISTER TO BE WRITTEN
39     044764  001015      BNE     1$           ;BRANCH IF NOT RMCS1
40     044766  122737  000150  045030  CMPB    #150,WRT.WD  ;IS THE COMMAND FOR DATA TRANSFERS?
41     044774  002411      BLT    1$           ;YES--DON'T GET THE OLD A16 & A17, & PSEL
42     044776  004037  044670      JSR     RO,RD.RM    ;CALL THE READ ROUTINE
43     045002  000000      RMCS1  ;REGISTER OFFSET
44     045004  045052      WRT.R3  ;'NED' RETURN
45     ;THE COMMAND BEFORE SENDING TO THE
46     ;THE RH70/RM80
47
48     045006  000316      SWAB   (SP)
49     045010  042716  177770      BIC    #^C7,(SP)
50     045014  112637  045031      MOVB  (SP)+,WRT.WD+1
51     045020  063737  040310  045032  1$: ADD      RMADR,WRT.AD  ;FORM THE ADDRESS OF THE DISK REG.
52     045026  012737      WRT.R1: MOV     (PC)+,@(PC)+ ;LOAD THE DESIRED REG.
53     045030  000000      WRT.WD: .WORD   0      ;WORD TO WRITE GOES HERE
54     045032  000000      WRT.AD: .WORD   0      ;ADDRESS IS FORMED HERE
55     045034  013746  040310      MOV     RMADR,-(SP)  ;PUT THE ADDRESS ON THE STACK
56     045040  062716  000010      ADD     #RMCS2,(SP)  ;FORM THE ADDRESS OF RMCS2
57     045044  032736  010000      BIT     #BIT12,@(SP)+ ;CHECK THE 'NED' BIT
58     045050  001402      BEQ    WRT.R4      ;IF EQ, DRIVE IS PRESENT
    
```

```

56 045052 011000      WRT.R3: MOV      (R0),R0      :TAKE THE 'NED' EXIT
57 045054 000401      WRT.R3: BR       WRT.R5      :EXIT
58 045056 005720      WRT.R4: TST      (R0)+      :ADJUST FOR ERROR FREE EXIT
59 045060 000200      WRT.R5: RTS      R0
60
61                    ;ROUTINE TO SAVE THE RH70/RM80 REGISTERS
62
63                    ;CALL:
64                    ;
65                    ;   MOV      #DPBADR,R2      :DPB POINTER TO R2
66                    ;   JSR      PC,SVRH70      :SAVE THE DRIVES REG'S
67 045062 104412      SVRH70: SAVREG      :SAVE R0 - R5
68 045064 005702      TST      R2              :QUEUE ENTRY FOR THE DRIVE ?
69 045066 001442      BEQ      6$             :BR IF NONE
70 045070 013704 040310  MOV      RMADR,R4
71 045074 111264 000010  MOVVB   (R2),RMCS2(R4)  :SELECT DRIVE
72 045100 016203 000014  MOV      14(R2),R3      :GET THE ERROR TABLE POINTER
73 045104 001433      BEQ      6$             :EXIT IF NO ADDRESS
74 045106 005037 045142  CLR      3$             :COUNTER & POINTER
75 045112 023727 045142 000022 1$: CMP      3$,#RMDB      :REACHED THE BUFFER REGISTER ?
76 045120 001006      BNE      2$             :BR IF NOT
77 045122 032764 000200 000010  BIT      #BIT07,RMCS2(R4) :'OR' SET ?
78 045130 001002      BNE      2$             :BR IF SET
79 045132 005023      CLR      (R3)+          :STORE RMDB AS ZEROES
80 045134 000405      BR       4$             :CONTINUE
81 045136 004037 044670  2$: JSR      R0,RD.RM      :READ THE SELECTED REGISTER
82 045142 000000      3$: .WORD 0             :REGISTER INDEX
83 045144 045170      5$: MOV      (SP)+,(R3)+      :ERROR RETURN ADDRESS
84 045146 012623      MOV      (SP)+,(R3)+      :STORE THE REGISTER CONTENTS
85 045150 023727 045142 000046 4$: CMP      3$,#RMEC2      :REACHED THE END ?
86 045156 001406      BEQ      6$             :BR IF YES
87 045160 062737 000002 045142  ADD      #2,3$          :INCREMENT THE REGISTER INDEX
88 045166 000751      BR       1$             :CONTINUE READING THE REGISTERS
89 045170 004737 042334 5$: JSR      PC,C18        :PROCESS THE 'NED'
90 045174 013746 001234 6$: MOV      $CPUOP,-(SP)    :CHECK THE CPU (RH) TYPE
91 045200 042716 003777  BIC      #^C174000,(SP)  :LEAVE THE CPU TYPE BITS
92 045204 022726 030000  CMP      #30000,(SP)+    :SEE IF RH70
93 045210 001004      BNE      7$             :IF NE, NO
94 045212 063704 040316  ADD      RHEXT,R4        :POINT TO RMBAE
95 045216 012423      MOV      (R4)+,(R3)+      :STORE THE CONTENTS
96 045220 011413      MOV      (R4),(R3)        :GET RMCS3
97 045222 104413      7$: RESREG      :RESTORE R0 - R5
98 045224 000207      RTS      PC            :RETURN
99
100                   ;ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A 'TRE''
101                   ;CALL:
102                   ;
103                   ;   MOV      #DRVNUM,R1      :DRIVE NUMBER TO R1
104                   ;   JSR      PC,SET.IE      :SET 'IE''
105                   ;
106                   ;   RETURN
107
108
109
110 045226 010446      SET.IE: MOV      R4,-(SP)    :SAVE R4
111 045230 013704 040310  MOV      RMADR,R4        :PICKUP ADDRESS OF RMCS1
112 045234 010164 000010  MOV      R1,RMCS2(R4)    :SELECT DRIVE
113 045240 011446      MOV      (R4),-(SP)      :READ RMCS1
114 045242 052716 040000  B'S     #BIT14,(SP)      :SET THE 'TRE'' BIT OF THE WORD READ
115 045246 000316      SIAB   (SP)            :ADJUST FOR DATO
116 045250 112714 000100  MOVVB   #BIT06,(R4)      :SET 'IE''
    
```

```

117 045254 032764 010000 000010      BIT      #BIT12,RMCS2(R4)      :IS 'NED'=1?
118 045262 001002                    BNE      1$                  :YES--CLEAR 'TRE'
119 045264 005725                    TST      (SP)+                :CLEAN OFF THE STACK
120 045266 000402                    BR       2$
121 045270 112664 000001      1$:    MOVB      (SP)+,1(R4)      :CLEAR 'TRE'
122 045274 012604      2$:    MOV       (SP)+,R4          :RESTORE R4
123 045276 000207                    RTS      PC                  :RETURN TO CALLER
124
125      :ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
126      :
127      :CALL:
128      :      JSR      PC,CLRQUE
129
130 045300 104412                    CLRQUE: SAVREG                :SAVE R0 - R5
131 045302 012702 040224      MOV       #QDRV,R2          :QUEUE BASE ADDRESS
132 045306 005022                    CLR      (R2)+                :CLEAR ENTRY
133 045310 005022                    CLR      (R2)+                :CLEAR ENTRY
134 045312 005022                    CLR      (R2)+                :CLEAR ENTRY
135 045314 005022                    CLR      (R2)+                :CLEAR ENTRY
136 045316 005022                    CLR      (R2)+                :CLEAR ENTRY
137 045320 005022                    CLR      (R2)+                :CLEAR ENTRY
138 045322 005022                    CLR      (R2)+                :CLEAR ENTRY
139 045324 005022                    CLR      (R2)+                :CLEAR ENTRY
140 045326 104413                    RESREG
141 045330 000207                    RTS      PC                  :RESTORE R0 - R5
142
143      :EMPTY THE QUEUE SPECIFIED BY R1
144      :
145      :CALL:
146      :      MOV       DRVNUM,R1      :DRIVE NUMBER TO R1
147      :      JSR      PC,EMPTYQ
148 045332 006301                    EMPTYQ: ASL      R1
149 045334 005061 040224      CLR      QDRV(R1)          :CLEAR DRIVE QUEUE
150 045340 006201                    ASR      R1                  :RESTORE R1
151 045342 000207                    RTS      PC
152
153      :ROUTINE TO PUT A REQUEST IN QUEUE
154      :
155      :CALL:
156      :      MOV       #DRVNUM,R1      :DRIVE NUMBER
157      :      MOV       #DPB,R2         :ADDRESS OF PARAMETER BLOCK
158      :      JSR      R0,DRVQUE        :GO PUT REQUEST IN QUEUE
159      :      RETURN1                    :RETURN HERE IF QUEUE IS FULL
160      :      RETURN2                    :RETURN HERE IF REQUEST IS IN QUEUE
161
162 045344 006301                    DRVQUE: ASL      R1
163 045346 005761 040224      TST      QDRV(R1)          :TEST THE QUEUE ENTRY
164 045352 001003                    BNE      1$                  :IF NE, QUEUE ENTRY ALREADY THERE
165 045354 010261 040224      MOV       R2,QDRV(R1)      :ADD THE QUEUE ENTRY
166 045360 005720                    TST      (R0)+                :TAKE RETURN 2
167 045362 006201      1$:    ASR      R1
168 045364 000200      2$:    RTS      R0          :RETURN TO USER
169
170      :ROUTINE TO GET THE 'DPB' ADDRESS OF NEXT REQUEST IN QUEUE
171      :
172      :CALL:
    
```

```
173      :      MOV      #DRVNUM,R1      ;DRIVE NUMBER TO R1
174      :      JSR      PC,GETREQ      ;GO GET THE REQUEST
175      :      RETURN     ;R2='DPB' ADDRESS OF THE REQUEST
176      :      ;R2=0 IF NO REQUEST IN QUEUE
177
178 045366 006301      GETREQ: ASL      R1      ;
179 045370 016102 040224  MOV      QDRV(R1),R2 ;GET THE REQUEST
180 045374 006201      ASR      R1      ;
181 045376 000207      RTS      PC      ;RETURN
182
183      ;ROUTINE TO 'POP' THE REQUEST FROM QUEUE
184
185      ;CALL:
186      :      MOV      #DRVNUM,R1      ;DRIVE NUMBER TO R1
187      :      JSR      PC,POPQUE      ;CALL TO REMOVE REQUEST
188      :      RETURN     ;R2=ADDRESS OF DPB REMOVED
189
190 045400 006301      POPQUE: ASL      R1
191 045402 016102 040224  MOV      QDRV(R1),R2 ;GET THE QUEUE ENTRY
192 045406 005061 040224  CLR      QDRV(R1)   ;CLEAR THE QUEUE
193 045412 006201      ASR      R1
194 045414 000207      RTS      PC      ;RETURN TO USER
195
```

```

1      .SBTTL DATA, CONTROL, & STATUS BLOCKS
2
3      ;BLOCK LOCATION EQUATE STATEMENTS
4
5      000001      $FMT      = 1      ;FMT, HCI, ECI OR OFFSET CODE
6      000002      $COMND    = $FMT+1 ;OPERATION CODE
7      000003      $PSEL     = $FMT+2 ;PORT SELECT & BITS A16, A17
8      000004      $WCNT     = $FMT+3 ;WORD COUNT (2'S COMP)
9      000006      $BUF      = $FMT+5 ;BUFFER ADDR OR REGISTER TABLE POINTER
10     000010      $SEC      = $FMT+7 ;SECTOR ADDRESS OR 1ST REG ADDR
11     000011      $TRK      = $FMT+10 ;TRACK ADDRESS OF LAST REG ADDR
12     000012      $CYL     = $FMT+11 ;CYLINDER ADDR
13     000014      $REG      = $FMT+13 ;REGISTER STORAGE (IF ERROR)
14     000016      $STATUS   = $FMT+15 ;STATUS WORD (SET BY DRIVER)
15
16     ;DRIVE'S HISTORY AND CURRENT INDICATOR STORAGE EQUATES
17
18     000020      $WRDL     = $FMT+17 ;WORD COUNT LENGTH (POSITIVE)
19     000022      $$SEC     = $WRDL+2 ;SECTOR SIZE FOR CURRENT OPERATION (256. OR 258.)
20     000024      $CODE     = $WRDL+4 ;PRESENT COMMAND SELECTION CODE
21     000026      $PACK     = $WRDL+6 ;WRITE DATA PACK INDICATOR
22     000027      $PREVO    = $WRDL+7 ;PREVIOUS COMMAND SELECTION CODE
23     000030      $PATT     = $WRDL+10 ;PATTERN CODE
24     000032      $PREVA    = $WRDL+12 ;PREVIOUS ADDRESS- TRK, SEC, CYL (DOUBLE WORD)
25     000036      $ENDAT    = $WRDL+16 ;END OF PASS DATA COUNT (DOUBLE WORD)
26     000042      $ENDSK    = $WRDL+22 ;END OF PASS SEEK COUNT (DOUBLE WORD)
27     000046      $OPERC    = $WRDL+26 ;OPERATION COUNT (DOUBLE WORD) PER PASS
28     000052      $POSIT    = $WRDL+32 ;SEEK COUNT (DOUBLE WORD)
29     000056      $WTOFL    = $WRDL+36 ;TOTAL WORDS WRITTEN OVERFLOW COUNT
30     000060      $WRITN    = $WRDL+40 ;TOTAL WORDS WRITTEN COUNT (DOUBLE WORD)
31     000064      $RDOFL    = $WRDL+44 ;TOTAL WORDS READ OVERFLOW COUNT
32     000066      $READ     = $WRDL+46 ;TOTAL WORDS READ COUNT (DOUBLE WORD)
33     000072      $TOTAL    = $WRDL+52 ;TOTAL ERRORS (ALL TYPES) COUNT
34     000074      $$SOFT    = $WRDL+54 ;'SOFT' ERROR COUNT
35     000076      $HARD     = $WRDL+56 ;'HARD' ERROR COUNT
36     000100      $SKI      = $WRDL+60 ;'SKI' ERROR COUNT
37     000102      $MISPO    = $WRDL+62 ;PROG DETECTED MIS-POSITIONING ERROR COUNT
38     000104      $PASSC    = $WRDL+64 ;PASS COUNTER
39     000106      $FAIR     = $WRDL+66 ;OPERATION QUEUE 'FAIRNESS' COUNT
40     000110      $HLDWC    = $WRDL+70 ;HOLD WORD COUNT FOR 'RELBUF' ROUTINE
41     000112      $SENSE    = $WRDL+72 ;SKIP SECTOR INDICATOR
42
43     ;INDEX EQUATES TO THE NEXT OPERATION PARAMETERS
44
45     000114      $NCODE    = $WRDL+74 ;NEXT OPERATION CODE
46     000115      $NPATC    = $NCODE+1 ;NEXT PATTERN
47     000116      $NSEC     = $NCODE+2 ;NEXT SECTOR
48     000117      $NTRK     = $NCODE+3 ;NEXT TRACK
49     000120      $NCYL     = $NCODE+4 ;NEXT CYLINDER
50     000122      $NEXT     = $NCODE+6 ;PARAMETER SELECTION INDICATOR
51     000124      $FIRST    = $NCODE+10 ;FIRST OPERATION INDICATOR
52
53     ;INDEX EQUATES FOR MAXIMUM/MINIMUM ADDRESSES
54
55     000126      $MAXCYL   = $NCODE+12 ;MAXIMUM CYLINDER ADDRESS
56     000130      $MINCYL   = $MAXCYL+2 ;MINIMUM CYLINDER ADDRESS
57     000132      $MAXTRK   = $MAXCYL+4 ;MAXIMUM TRACK ADDRESS
    
```



```

58          000134          MINTRK = MAXCYL+6          ;MINIMUM TRACK ADDRESS
59          000136          MAXSEC = MAXCYL+10         ;MAXIMUM SECTOR ADDRESS
60          000140          MINSEC = MAXCYL+12         ;MINIMUM SECTOR ADDRESS
61
62          ;HDA SERIAL NUMBER CONTAINED IN DEC144 FILE
63
64          000142          $HSNL = MAXCYL+14          ;LSB'S OF SERIAL NUMBER (DECIMAL)
65          000144          $HSNM = $HSNL+2            ;MSB'S OF SERIAL NUMBER (DECIMAL)
66
67          ;DEC144 BAD SECTOR ADDRESS STORAGE AREA INDEX EQUATE
68
69          000146          $BDSEC = $HSNL+4            ;BAD SECTOR STORAGE TABLE PLUS TERMINATOR
70
71          ;DRIVE (MBA) SERIAL NUMBER AREA INDEX EQUATE
72
73          002130          $DRVSN = $BDSEC+<126.*8.>+2 ;DRIVE (MBA) SERIAL NUMBER
74
75          ;RH/RM REGISTER EQUATES
76
77          002140          $RMCS1 = $DRVSN+10          ;RM REGISTER STORAGE
78          002142          $RMWC = $RMCS1+2
79          002144          $RMBA = $RMCS1+4
80          002146          $RMDA = $RMCS1+6
81          002150          $RMCS2 = $RMCS1+10
82          002152          $RMDS = $RMCS1+12
83          002154          $RMER1 = $RMCS1+14
84          002156          $RMAS = $RMCS1+16
85          002160          $RMLA = $RMCS1+20
86          002162          $RMDB = $RMCS1+22
87          002164          $RMMR1 = $RMCS1+24
88          002166          $RMDT = $RMCS1+26
89          002170          $RMSN = $RMCS1+30
90          002172          $RMOF = $RMCS1+32
91          002174          $RMDC = $RMCS1+34
92          002176          $RMHR = $RMCS1+36
93          002200          $RMMR2 = $RMCS1+40
94          002202          $RMER2 = $RMCS1+42
95          002204          $RMEC1 = $RMCS1+44
96          002206          $RMEC2 = $RMCS1+46
97          002210          $RMBAE = $RMCS1+50
98          002212          $RMCS3 = $RMCS1+52
99
101         ;BLOCK FOR DRIVE 0
113         045416          000          000          DRIVE0: .BYTE 0,0          ;DRIVE NUMBER 0
          045420          .BLKW 5
          045432          047556          .WORD .+$RMCS1-$REG
          045434          .BLKB $RMCS3-$REG

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

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

```

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

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

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

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

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

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

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

		.SBTTL		ERROR MESSAGES	
1					
2					
3	067646	122	110	040	EM1: .ASCIZ /RH CONTROLLER INTERRUPT OCCURRED (RMAS=0)/
4	067720	125	116	105	EM2: .ASCIZ /UNEXPECTED ATTENTION OCCURRED/
5	067756	115	101	123	EM3: .ASCIZ /MASSBUS PARITY ERROR (MCPE=1)/
6	070014	115	101	123	EM4: .ASCIZ /MASSBUS PARITY ERROR (PAR=1)/
7	070051	101	104	104	EM5: .ASCIZ /ADDRESS PLUG CHANGE BIT SET/
8	070105	122	110	040	EM6: .ASCIZ /RH CONTROLLER DIDN'T RESPOND TO ADDRESSING/
9	070160	125	116	103	EM10: .ASCIZ /UNCORRECTABLE MASSBUS PARITY ERROR/
10	070223	106	101	124	EM11: .ASCIZ /FATAL MASSBUS PARITY ERROR/
11	070256	120	105	122	EM12: .ASCIZ /PERSISTENT DEVICE UNSAFE/
12	070307	117	120	105	EM13: .ASCIZ /OPERATION NOT COMPLETED WITHIN TIME LIMIT/
13	070361	104	122	111	EM14: .ASCIZ /DRIVE WENT OFFLINE/
14	070404	116	117	040	EM15: .ASCIZ /NO RESPONSE TO PORT REQUEST/
15	070440	110	105	101	EM20: .ASCIZ /HEADER CRC ERROR/
16	070461	104	101	124	EM21: .ASCIZ /DATA CHECK ('DCK') ERROR/
17	070512	127	122	111	EM22: .ASCIZ /WRITE CHECK ERROR - DATA CHECK ('DCK') SET/
18	070565	127	122	111	EM23: .ASCIZ /WRITE CHECK ERROR - DATA CHECK ('DCK') NOT SET/
19	070644	110	105	101	EM24: .ASCIZ /HEADER READ ERROR - 'FMT' BIT DROPPED/
20	070712	110	105	101	EM25: .ASCIZ /HEADER READ ERROR - HEADER COMPARE ('HCE') ERROR/
21	070773	106	117	122	EM26: .ASCIZ /FORMAT ERROR ('FER')/
22	071020	110	105	101	EM27: .ASCIZ /HEADER COMPARE ('HCE') ERROR/
23	071055	115	111	123	EM30: .ASCIZ /MISCELLANEOUS DRIVE ERROR/
24	071107	117	120	105	EM31: .ASCIZ /OPERATION INCOMPLETE ('OPI') ERROR/
25	071152	104	122	111	EM32: .ASCIZ /DRIVE TIMING ('DTE') ERROR/
26	071205	120	101	122	EM33: .ASCIZ /PARITY ('PAR') ERROR AFTER OPERATION STARTED/
27	071262	127	122	111	EM34: .ASCIZ /WRITE CLOCK FAILURE ('WCF') ERROR/
28	071324	111	116	126	EM35: .ASCIZ /INVALID ADDRESS ('IAE') ERROR/
29	071362	127	122	111	EM36: .ASCIZ /WRITE LOCK ('WLE') ERROR/
30	071413	104	101	124	EM37: .ASCIZ /DATA CHECK ('DCK') SET DURING WRITE CHECK COMMAND/
31	071475	122	110	040	EM40: .ASCIZ /RH CONTROLLER OR UNIBUS TRANSFER ERROR/
32	071544	102	125	123	EM41: .ASCIZ /BUS ADDRESS OR WORD COUNT INCORRECT/
33	071610	104	101	124	EM42: .ASCIZ /DATA COMPARE ERRORS - NO OTHER ERROR(S) DETECTED/
34	071671	103	101	116	EM43: .ASCIZ /CAN'T MATCH DATA READ WITH A PATTERN - UNKNOWN DATA PATTERN/
35	071765	105	122	122	EM44: .ASCIZ /ERROR BIT(S) SET, BUT NO ERROR SIGNALLED BY THE RH CONTROLLER/
36	072062	105	103	103	EM45: .ASCIZ /ECC LOGIC FAILURE - POSITION REGISTER VALUE NOT VALID/
37	072150	102	125	123	EM46: .ASCIZ /BUS ADDRESS AND WORD COUNT NOT CONSISTENT/
38	072222	105	103	103	EM47: .ASCIZ /ECC LOGIC FAILURE - PATTERN REGISTER IS ZERO/
39	072277	123	105	105	EM50: .ASCIZ /SEEK INCOMPLETE ('SKI') ERROR/
40	072335	120	122	117	EM51: .ASCIZ /PROGRAM DETECTED POSITIONING ERROR/
41	072400	105	103	110	EM52: .ASCIZ /ECC ERROR - UNCORRECTABLE ECC ERROR/
42	072444	104	122	111	EM60: .ASCIZ /DRIVE UNSAFE ERROR/

CZRMAAO RM80 PERF EXER MACRO V04.00 14-JAN-82 15:16:58 PAGE 62
ERROR MESSAGES

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

1	073272	120	122	123	LIN2C:	.ASCIZ	/PRSNT COMMAND= /
2	073312	040	040	120	LIN2P:	.ASCIZ	/ PREV COMMAND= /
3	073333	052	040	105	LIN2S:	.ASCIZ	@* ERROR AT BAD TRACK/SECTOR@
4	073367	105	122	122	LINM3:	.ASCIZ	/ERROR AT C/
5	073402	040	124	000	T:	.ASCIZ	/ T/
6	073405	120	122	123	LINN3:	.ASCIZ	/PRSNT ADDR= C/
7	073423	040	123	000	S:	.ASCIZ	/ S/
8	073426	040	040	120	LINP3:	.ASCIZ	/ PREV ADDR= C/
9	073445	123	124	101	LINS3:	.ASCIZ	/START CYL= /
10	073461	040	040	105	LINEN3:	.ASCIZ	/ END CYL= /
11	073475	040	040	101	LINA3:	.ASCIZ	/ ACTUAL CYL= /
12	073514	040	040	124	LINT3:	.ASCIZ	/ TRK= /
13	073524	040	040	122	LINCA3:	.ASCIZ	/ RMDC= /
14	073535	122	115	104	LINDA3:	.ASCIZ	/RMDA= /
15	073544	122	115	102	LINB3:	.ASCIZ	/RMDA= /
16	073553	040	040	122	LINW3:	.ASCIZ	/ RMWC= /
17	073564	123	124	101	LINST3:	.ASCIZ	/START TRK= /
18	073600	123	124	101	LINSS3:	.ASCIZ	/START SEC= /
19	073614	102	125	106	LINM4:	.ASCIZ	/BUFFER ADDR= /
20	073632	040	040	127	LINS4:	.ASCIZ	/ WRD CNT= /
21	073646	040	040	101	LINX4:	.ASCIZ	/ ACTUAL NMBR WRDS XFRD= /
22	073700	105	130	120	LIND5:	.ASCIZ	/EXPCTD DATA= /
23	073716	040	040	122	LINB5:	.ASCIZ	/ RECEVD DATA= /
24	073736	040	040	127	LINP5:	.ASCIZ	/ WORD POS= /
25	073753	110	105	101	LINS5:	.ASCIZ	/HEADER FROM ERROR SECTOR= /
26	074006	122	115	105	LINEP5:	.ASCIZ	/RMEC1= /
27	074016	040	040	122	LINEO5:	.ASCIZ	/ RMEC2= /
28	074030	123	105	103	LINB6:	.ASCIZ	/SECTOR IS ECC CORRECTABLE /
29	074063	123	105	103	LINC6:	.ASCIZ	/SECTOR READ CORRECTLY /
30	074112	103	117	122	LING6:	.ASCIZ	/CORRECTED ON /
31	074130	040	040	122	LINR6:	.ASCIZ	/ RETRIES/
32	074142	125	116	103	LINU06:	.ASCIZ	/UNCORRECTABLE AFTER /
33	074167	040	040	124	LIN7M:	.ASCIZ	/ TOTAL MISPOS ERR= /
37	074214	124	117	124	LIN7P:	.ASCIZ	/TOTAL SEEKS= /
38	074232	040	040	124	LIN7S:	.ASCIZ	/ TOTAL SKI ERR= /
39	074254	124	117	124	LIN7T:	.ASCIZ	/TOTAL ERRORS:/
40	074272	040	040	127	LIN7OX:	.ASCIZ	/ WOFL:/
41	074302	040	127	122	LIN7X:	.ASCIZ	/ WRDS WRITN:/
42	074317	040	040	122	LIN7OR:	.ASCIZ	/ ROFL:/
43	074327	040	127	122	LIN7R:	.ASCIZ	/ WRDS READ:/
44	074343	105	122	122	LIN8Pi:	.ASCIZ	/ERROR DURING RETRY/
45	074366	104	101	124	LIN9B:	.ASCIZ	/DATA COMPARISON ERRORS/
46	074415	040	040	040	LIN9H:	.ASCII	/
47	074444	114	117	103		.ASCIZ	EXPCTD RECEVD/<CRLF>
48	074473	040	040	040	LIN9I:	.ASCII	/LOC DATA DATA/<CRLF>
49	074512	114	117	103		.ASCIZ	RECEVD/<CRLF>
50	074531	124	117	124	LIN9E:	.ASCIZ	/LOC DATA/<CRLF>
51	074560	124	110	105	LIN9G:	.ASCIZ	/TOTAL COMPARE ERRORS= /
52	074606	105	122	122	LIN10A:	.ASCIZ	/THE DATA COMPARED OK/<CRLF>
53	074642	040	111	116	LIN10B:	.ASCIZ	/ERROR BURST BEGINS AT WORD /
54	074702	105	122	122	LIN10C:	.ASCII	/ IN DATA FIELD OF ERROR SECTOR/<CRLF>
55	074744	105	103	103		.ASCIZ	/ERROR WAS NOT IN THE DATA READ - /<CRLF>
56	075006	105	103	103	LIN10H:	.ASCII	/ECC CORRECTION CAN'T BE PERFORMED/
57	075035	040	040	040		.ASCIZ	/ECC CORRECTION RESULTS/<CRLF>
58	075070	101	104	104		.ASCII	/
59	075116	103	117	116	LIN11H:	.ASCIZ	/ ADDR BAD CORRECTED /<CRLF>
60	075171	101	104	104	LIN11:	.ASCIZ	/ADDR DATA DATA/<CRLF>

ERROR MESSAGES

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

			.SBTTL TELETYPE MESSAGES	
1				
2				
3	075272	077	000	QUES: .ASCIZ /?/
4	075274	075	000	EQUAL: .ASCIZ /=/
5	075276	054	000	COMMA: .ASCIZ /./
6	075300	055	000	DASH: .ASCIZ /-/
7	075302	011	000	TAB: .ASCIZ <11>
8	075304	104	122	111 UNTMSG: .ASCIZ /DRIVE/
9	075312	040	117	106 UNTOFF: .ASCIZ / OFFLINE/
10	075323	040	117	116 UNTON: .ASCIZ / ONLINE/
11	075333	040	116	117 UNTNOT: .ASCIZ / NOT BEING TESTED/
12	075355	040	101	114 UNTASN: .ASCIZ / ALREADY BEING TESTED/
13	075403	040	116	117 NOTRM: .ASCIZ @ NOT AN RM80@
14	075420	040	116	117 NOTPRS: .ASCIZ / NOT PRESENT/
15	075435	040	116	117 NOTAVL: .ASCIZ / NOT AVAILABLE/
16	075454	040	125	116 NOTSAF: .ASCIZ / UNSAFE/
17	075464	040	114	117 LODEV: .ASCIZ / LOAD DEVICE/
18	075501	200	104	122 SYSTAT: .ASCIZ <CRLF>/DRIVE STATUS:/
19	075520	122	115	070 SRM80: .ASCIZ /RM80/
20	075525	200	012	052 REPHD: .ASCIZ <CRLF><LF>/***** PERFORMANCE REPORT *****/
21	075566	052	052	052 STAR30: .ASCII /*****
22	075617	052	052	052 STAR5: .ASCIZ /*****/<CRLF>
23	075626	200	104	122 SUMHD: .ASCIZ <CRLF>/DRIVE SUMMARY, (OFLOW= 2,147,483,647.)/<CRLF>
24	075700	007	077	106 DROPNG: .ASCIZ <07>/?FATAL OR EXCESSIVE ERRORS/
25	075734	200	105	116 ENDPAS: .ASCIZ <CRLF>/END OF PASS #/
26	075753	040	117	116 MSGON: .ASCIZ / ON /
27	075760	200	105	116 ENDTST: .ASCIZ <CRLF>/END OF TEST /
28	075776	106	117	122 MSGFOR: .ASCIZ /FOR /
29	076003	200	052	052 DEASSG: .ASCIZ <CRLF>/***** DRIVE DEASSIGNED/<CRLF>
30	076034	200	052	052 DRNUM: .ASCIZ <CRLF>/***** DRIVE #/
31	076053	040	123	124 ASGND: .ASCIZ / STARTED/
32	076064	200	007	077 NEDCLK: .ASCIZ <CRLF><07>/? 'L' OR 'P' CLOCK REQUIRED ON SYSTEM/<CRLF>
33	076135	116	000	N: .ASCIZ /N/
34	076137	131	000	Y: .ASCIZ /Y/
35	076141	056	000	PERIOD: .ASCIZ /./
36	076143	040	077	103 MSWRO: .ASCIZ / ?CAN'T WRITE IN 'READ ONLY' MODE/<CRLF>
37	076206	040	077	111 INVLD: .ASCIZ / ?INVALID COMMAND/<CRLF>
38	076231	200	105	116 ENTCOM: .ASCIZ <CRLF>/ENTER COMMAND: /
39	076252	200	105	116 ENTLMT: .ASCIZ <CRLF>/ENTER ADDRESS LIMITS:/<CRLF>
40	076302	200	105	116 ENTADR: .ASCIZ <CRLF>/ENTER BAD SECTR ADRS:/<CRLF>
41	076332	072	000	COLON: .ASCIZ /:/
42	076334	116	117	116 NONE: .ASCIZ /NONE/
43	076341	040	077	111 BADENT: .ASCIZ / ?INVALID ENTRY/<CRLF>
44	076362	200	106	101 MERR1: .ASCIZ <CRLF>/FAILED TO RETRIEVE BAD SECTOR FILE(DEC144) ON /
45	076442	200	111	116 MERR2: .ASCIZ <CRLF>/INVALID FILE(DEC144) STRUCTURE ON /
46	076506	052	040	102 MSFULL: .ASCIZ /* BAD SECTOR TABLE IS FULL */<CRLF>
47	076544	103	131	114 MSGCTS: .ASCIZ /CYL,TRK,SEC= /
48	076562	200	104	117 MESFE: .ASCIZ <CRLF>/DO YOU WISH TO EXERCISE ONLY FE CYLINDERS (L) Y ? /
49	076646	101	122	105 SURE: .ASCIZ /ARE YOU SURE (L) N ? /
50	076674	200	052	040 FEONLY: .ASCIZ <CRLF>/* EXERCISER WILL OPERATE ON FE CYLINDERS ONLY */<CRLF>
51	076756	007	007	OVRWRT: .ASCII <07><07>
52	076760	200	011	041 .ASCII <CRLF>/ ! CUSTOMER DATA WILL BE OVERWRITTEN !/
53	077027	200	011	055 .ASCII <CRLF>/ -----/
54	077076	007	007	200 .ASCIZ <07><07><CRLF><LF>
55	077103	200	012	116 NODRVS: .ASCIZ <CRLF><LF>/NO DRIVES ASSIGNED/<CRLF>
56	077131	200	120	122 MREAD: .ASCIZ <CRLF>/PROGRAM LOCKED IN 'READ ONLY' MODE/<CRLF>
57	077176	052	040	116 NOENTY: .ASCIZ /* NO ENTRIES */<CRLF>

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

```

1
2
3 077466 077632 017400 001462 PARLST: .WORD PAR1,7936.,WRDCNT
4 077474 077642 077777 001464 .WORD PAR2,32767.,INTRVL
5 077502 100012 077777 001470 .WORD PAR19,32767.,PASSES
6 077510 077652 000017 001472 .WORD PAR3,15.,PATTERN
7 077516 077752 000001 001474 .WORD PAR11,1.,RANDWC
8 077524 077762 000007 001476 .WORD PAR14,7.,RATIO
9 077532 100002 000001 001500 .WORD PAR16,1.,ENDING
10 077540 077772 000001 001502 .WORD PAR15,1.,WRTCHK
11 077546 100022 000001 001504 .WORD PAR20,1.,MESSAGE
12 077554 100032 000001 001506 .WORD PAR21,1.,RANDOM
13 077562 077742 000001 001510 .WORD PAR10,1.,BADBLK
14 077570 000000 .WORD 0 ;TABLE TERMINATOR
  
```

```

15
16 077572 040 057 040 SLASH: .ASCIZ @ / @
17 077576 200 103 110 ASKPAR: .ASCIZ <CRLF>/CHANGE PARAMETERS (L) N ? /
18 077632 127 122 104 PAR1: .ASCIZ /WRDCNT /
19 077642 111 116 124 PAR2: .ASCIZ /INTRVL /
20 077652 120 101 124 PAR3: .ASCIZ /PATTERN/
21 077662 115 101 130 PAR4: .ASCIZ /MAXCYL /
22 077672 115 111 116 PAR5: .ASCIZ /MINCYL /
23 077702 115 101 130 PAR6: .ASCIZ /MAXTRK /
24 077712 115 111 116 PAR7: .ASCIZ /MINTRK /
25 077722 115 101 130 PAR8: .ASCIZ /MAXSEC /
26 077732 115 111 116 PAR9: .ASCIZ /MINSEC /
27 077742 102 101 104 PAR10: .ASCIZ /BADBLK /
28 077752 122 101 116 PAR11: .ASCIZ /RANDWC /
29 077762 122 101 124 PAR14: .ASCIZ /RATIO /
30 077772 127 122 124 PAR15: .ASCIZ /WRTCHK /
31 100002 105 116 104 PAR16: .ASCIZ /ENDING /
32 100012 120 101 123 PAR19: .ASCIZ /PASSES /
33 100022 115 105 123 PAR20: .ASCIZ /MESSAGE/
34 100032 122 101 116 PAR21: .ASCIZ /RANDOM /
  
```

.EVEN

```

35
36
37
38 ;PARAMETER TABLE POINTERS FOR ADDRESS LIMITS
39 100042 100062 TABLE: .WORD TABLE0 ;PARAMETER TABLE FOR DRIVE 0
42 100044 100130 .WORD TABLE1 ;PARAMETER TABLE FOR DRIVE 1
100046 100176 .WORD TABLE2 ;PARAMETER TABLE FOR DRIVE 2
100050 100244 .WORD TABLE3 ;PARAMETER TABLE FOR DRIVE 3
100052 100312 .WORD TABLE4 ;PARAMETER TABLE FOR DRIVE 4
100054 100360 .WORD TABLE5 ;PARAMETER TABLE FOR DRIVE 5
100056 100426 .WORD TABLE6 ;PARAMETER TABLE FOR DRIVE 6
100060 100474 .WORD TABLE7 ;PARAMETER TABLE FOR DRIVE 7
  
```

```

43
44 ;PARAMETER TABLE FOR ADDRESS LIMITS
54 100062 077672 000000 045546 TABLE0: .WORD PAR5,0,MINCYL+DRIVE0
100070 077662 000000 045544 .WORD PAR4,0,MAXCYL+DRIVE0
100076 077712 000000 045552 .WORD PAR7,0,MINTRK+DRIVE0
100104 077702 000000 045550 .WORD PAR6,0,MAXTRK+DRIVE0
100112 077732 000036 045556 .WORD PAR9,30.,MINSEC+DRIVE0
100120 077722 000036 045554 .WORD PAR8,30.,MAXSEC+DRIVE0
100126 000000 .WORD 0 ;TERMINATOR

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

100136	077662	000000	047760	.WORD	PAR4,0,MAXCYL+DRIVE1
100144	077712	000000	047766	.WORD	PAR7,0,MINTRK+DRIVE1
100152	077702	000000	047764	.WORD	PAR6,0,MAXTRK+DRIVE1
100160	077732	000036	047772	.WORD	PAR9,30.,MINSEC+DRIVE1
100166	077722	000036	047770	.WORD	PAR8,30.,MAXSEC+DRIVE1
100174	000000			.WORD	0 ; TERMINATOR
100176	077672	000000	052176	TABLE2: .WORD	PAR5,0,MINCYL+DRIVE2
100204	077662	000000	052174	.WORD	PAR4,0,MAXCYL+DRIVE2
100212	077712	000000	052202	.WORD	PAR7,0,MINTRK+DRIVE2
100220	077702	000000	052200	.WORD	PAR6,0,MAXTRK+DRIVE2
100226	077732	000036	052206	.WORD	PAR9,30.,MINSEL+DRIVE2
100234	077722	000036	052204	.WORD	PAR8,30.,MAXSEC+DRIVE2
100242	000000			.WORD	0 ; TERMINATOR
100244	077672	000000	054412	TABLE3: .WORD	PAR5,0,MINCYL+DRIVE3
100252	077662	000000	054410	.WORD	PAR4,0,MAXCYL+DRIVE3
100260	077712	000000	054416	.WORD	PAR7,0,MINTRK+DRIVE3
100266	077702	000000	054414	.WORD	PAR6,0,MAXTRK+DRIVE3
100274	077732	000036	054422	.WORD	PAR9,30.,MINSEC+DRIVE3
100302	077722	000036	054420	.WORD	PAR8,30.,MAXSEC+DRIVE3
100310	000000			.WORD	0 ; TERMINATOR
100312	077672	000000	056626	TABLE4: .WORD	PAR5,0,MINCYL+DRIVE4
100320	077662	000000	056624	.WORD	PAR4,0,MAXCYL+DRIVE4
100326	077712	000000	056632	.WORD	PAR7,0,MINTRK+DRIVE4
100334	077702	000000	056630	.WORD	PAR6,0,MAXTRK+DRIVE4
100342	077732	000036	056636	.WORD	PAR9,30.,MINSEC+DRIVE4
100350	077722	000036	056634	.WORD	PAR8,30.,MAXSEC+DRIVE4
100356	000000			.WORD	0 ; TERMINATOR
100360	077672	000000	061042	TABLE5: .WORD	PAR5,0,MINCYL+DRIVE5
100366	077662	000000	061040	.WORD	PAR4,0,MAXCYL+DRIVE5
100374	077712	000000	061046	.WORD	PAR7,0,MINTRK+DRIVE5
100402	077702	000000	061044	.WORD	PAR6,0,MAXTRK+DRIVE5
100410	077732	000036	061052	.WORD	PAR9,30.,MINSEC+DRIVE5
100416	077722	000036	061050	.WORD	PAR8,30.,MAXSEC+DRIVE5
100424	000000			.WORD	0 ; TERMINATOR
100426	077672	000000	063256	TABLE6: .WORD	PAR5,0,MINCYL+DRIVE6
100434	077662	000000	063254	.WORD	PAR4,0,MAXCYL+DRIVE6
100442	077712	000000	063262	.WORD	PAR7,0,MINTRK+DRIVE6
100450	077702	000000	063260	.WORD	PAR6,0,MAXTRK+DRIVE6
100456	077732	000036	063266	.WORD	PAR9,30.,MINSEC+DRIVE6
100464	077722	000036	063264	.WORD	PAR8,30.,MAXSEC+DRIVE6
100472	000000			.WORD	0 ; TERMINATOR
100474	077672	000000	065472	TABLE7: .WORD	PAR5,0,MINCYL+DRIVE7
100502	077662	000000	065470	.WORD	PAR4,0,MAXCYL+DRIVE7
100510	077712	000000	065476	.WORD	PAR7,0,MINTRK+DRIVE7
100516	077702	000000	065474	.WORD	PAR6,0,MAXTRK+DRIVE7
100524	077732	000036	065502	.WORD	PAR9,30.,MINSEC+DRIVE7
100532	077722	000036	065500	.WORD	PAR8,30.,MAXSEC+DRIVE7
100540	000000			.WORD	0 ; TERMINATOR

1

.SBTTL ROUTINE TO SIZE MEMORY

```

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

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

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

.SBTTL BUSADR - GET BUS ADDRESS AND VECTOR ADDRESS

:THIS ROUTINE IS USED TO INSURE THE BUS ADDRESS OF THE RH/RM
 :IS SETUP FOR THE PROPER ADDRESS. IT WILL ALSO READ THE ADDRESS
 :FROM THE TTY IF REQUIRED.

:NOTE: THIS ROUTINE DESTROYS R0-R4
 :CALL:

:
 : JSR PC,BUSADR
 : RETURN
 :

```

BUSADR: TST   CHGADR      ;INPUT FROM TTY REQUESTED?
        BGE   3$        ;NO--BRANCH
        CLR   CHGADR      ;YES--CLEAR THE REQUEST FLAG
        TYPE  ,SCLF      ;TYPE A CR-LF
        INC   #-1        ;FIRST TIME THRU ?
        BEQ   1$        ;BR IF YES
        TYPE  ,SCLF      ;CR-LF
1$:     CLR   CFLAG      ;CLEAR CONTROL C FLAG
        MOV   #SRMADR,R0 ;FIRST ADDRESS
        TYPE  ,MRMCS1    ;'RMCS1='
        MOV   (R0),-(SP) ;PRESENT RMCS1 ADDRESS
        TYPOC ;TYPE IT
        TYPE  ,BLNKS2    ;TYPE 2 BLANKS
        RDLIN ;GET THE ENTRY
        MOV   (SP)+,R1   ;ADDRESS OF ASCII TEXT
        TST   CFLAG      ;WAS IT CONTROL C ?
        BNE   1$        ;BR IF YES
        JSR   R5,CK.NUM  ;ENTER AND STORE THE NEW ADDRESS
        BR    1$        ;ERROR EXIT
2$:     MOV   #SRMVEC,R0 ;VECTOR ADDRESS
        TYPE  ,MRMVEC    ;'RMVEC='
        MOV   (R0),-(SP) ;PRESENT RH/RM VECTOR ADDRESS ON THE STACK
        TYPOC ;TYPE IT
        TYPE  ,BLNKS2    ;TYPE 2 BLANKS
        RDLIN ;READ THE ENTRY
        MOV   (SP)+,R1   ;ASCII TEXT ADDRESS
        TST   CFLAG      ;WAS IT CONTROL C ?
        BNE   1$        ;BR IF YES
        JSR   R5,CK.NUM  ;ENTER AND STORE NEW ADDRESS
        BR    2$        ;ERROR EXIT
3$:     MOV   #SRMADR,R0 ;FIRST ADDRESS OF NEW PARAMETERS
        MOV   #RMADR,R1  ;FIRST ADDRESS OF WHERE TO PUT THEM
        MOV   (R0)+,(R1)+ ;BUS ADDRESS
        MOV   (R0)+,(R1)+ ;VECTOR ADDRESS
        RTS   PC        ;RETURN
    
```

```

49 101046 122 115 103 MRMCS1: .ASCIZ @RMCS1=@
50 101055 122 115 126 MRMVEC: .ASCIZ @RMVEC=@
    
```

```

1      .SBTTL  CK.NUM - CHECK NUMBER (OCTAL)
2
3      ;THIS ROUTINE CHECKS AN ASCIZ STRING FOR LEGAL CHARACTERS
4      ;AND FORMS AN OCTAL NUMBER IN R2
5      ;CALL:
6      ;      MOV      #ADR,R0      ;ADDRESS TO PLACE NEW NUMBER
7      ;      MOV      #ADR,R1      ;ADDRESS OF ASCIZ STRING
8      ;      JSR      R5,CK.NUM     ;R5 CHANGED
9      ;      RET      ;ERROR EXIT
10     ;      RET      ;NORMAL EXIT
11
12     CK.NUM: MOV      R2,-(SP)      ;SAVE R2
13     MOV      R3,-(SP)      ;SAVE R3
14     MOV      R4,-(SP)      ;SAVE R4
15     MOV      #6,R3         ;MAX OCTAL DIGITS IN THE NUMBER
16     CLR      R2            ;FINAL OCTAL VALUE
17     1$:  MOVB     (R1)+,R4      ;GET CURRENT POINTED BYTE
18     BEQ      3$           ;BRANCH,IF TERMINATOR DETECTED
19     CMPB     R4,#'0        ;SMALLER THAN ASCII-0 ?
20     BLO      5$           ;YES,ERROR EXIT
21     CMPB     R4,#'7        ;LARGER THAN ASCII-7 ?
22     BHI      5$           ;YES,ERROR EXIT
23     ASL      R2            ;SHIFT LEFT
24     BCS      5$           ;
25     ASL      R2            ;ONE
26     BCS      5$           ;
27     ASL      R2            ;OCTAL DIGIT
28     BCS      5$           ;ERROR IF CARRY BIT SET
29     BIC      #177770,R4    ;CHOP OFF HIGHER BITS
30     ADD      R4,R2         ;APPENDING CURRENT DIGIT TO NUMBER
31     DEC      R3            ;DECREMENT BYTE COUNT
32     BEQ      2$           ;BRANCH,IF LAST BYTE
33     BR       1$           ;LOOPING BACK
34     2$:  MOVB     (R1)+,R4      ;CHECK TERMINATOR
35     BNE      5$           ;ERROR EXIT
36     3$:  TST      R2            ;FINAL VALUE= 0
37     BFQ      4$           ;YES,THEN NOT REPLACE THE ORIGINAL VALUE
38     MUV     R2,(R0)        ;REPLACE THE ORIGINAL VALUE
39     4$:  TST      (R5)+        ;ADJUST FOR NORMAL RETURN
40     5$:  MOV      (SP)+,R4      ;RESTORE R4
41     MOV      (SP)+,R3      ;RESTORE R3
42     MOV      (SP)+,R2      ;RESTORE R2
43     RTS      R5            ;EXIT
44
45     CYLNDR: .LLKW 258.      ;ONE SECTOR WORD CTR MAX SIZE
46
47     ENDPGM=.
48
49     .END 200
    
```

SYMBOL TABLE

ABASE = 176700	ASGN6 026302	BIT5 = 000040	COMMA 075276	DRIVE4 056476
ABNRML 031264	ASGN7 026314	BIT6 = 000100	COMTBL 002076	DRIVE5 060712
ACDW1 = 000000	ASKPAR 077576	BIT7 = 000200	CPSAVE 035224	DRIVE6 063126
ACDW2 = 000000	ASNERR 031154	BIT8 = 000400	CR = 000015	DRIVE7 065342
ACK = 000123	ASNLST 001542	BIT9 = 001000	CRLF = 000200	DRNUM 076034
ACPUOP= 000000	ASNMSG 031200	BLKADR 002056	CTRAP 034574	DROP 031204
ACTDRV 040250	ASSIGN 025666	BLNKS1 075234	CYLIMT 001422	DROPNG 075700
ACTSTR 040251	ASWREG= 000000	BLNKS2 075233	CYLNR 101174	DRQ = 004000
ADDW0 = 000000	ATA = 100000	BLNKS3 075232	DASH 075300	DRVACT 040154
ADDW1 = 000000	ATABIT 040300	BLNKS4 075231	DATAPK 026600	DRVCLR= 000111
ADDW10= 000000	AATESTN= 000000	BPTVEC= 000014	DATA0 002370	DRVER 011200
ADDW11= 000000	ATTN 001322	BSE = 100000	DATA1 002430	DRVINT 040534
ADDW12= 000000	ATO = 000001	BUFTBL 001654	DATA10 003070	DRVNO 001324
ADDW13= 000000	AT1 = C00002	BUSADR 100674	DATA11 003130	DRVPAR 001442
ADDW14= 000000	AT2 = 000004	CFLAG 001340	DATA12 003170	DRVPRM 027054
ADDW15= 000000	AT3 = 000010	CHGADR 001336	DATA13 003250	DRVQUE 045344
ADDW2 = 000000	AT4 = 000020	CHKWC 020330	DATA14 003270	DRVSN 077332
ADDW3 = 000000	AT5 = 000040	CI1 041516	DATA15 003330	DRVSTA 040164
ADDW4 = 000000	AT6 = 000100	CI3 041650	DATA2 002470	DRVSTYP 040174
ADDW5 = 000000	AT7 = 000200	CI4 041756	DATA3 002530	DRY = 000200
ADDW6 = 000000	AUNIT = 000000	CI5 042312	DATA4 002570	DSWR = 177570
ADDW7 = 000000	AUSWR = 000000	CI8 042334	DATA5 002630	DTE = 010000
ADDW8 = 000000	AUTLST 032012	CKBUS 013154	DATA6 002670	DTEER 011772
ADDW9 = 000000	AVAIL 001610	CKCLK 023422	DATA7 002730	DTUW 040276
ADEVCT= 000000	AVECT1= 000254	CKCLK1 023516	DATA8 002770	DT00 = 000001
ADEVN = 000000	AVECT2= 000000	CKCLK2 023566	DATA9 003030	DT01 = 000002
AENV = 000000	A16 = 000400	CKCLK3 023620	DCK = 100000	DT02 = 000004
AENVN = 000000	A17 = 001000	CKERR 013062	DCKER 010072	DT03 = 000010
AFATAL= 000000	BADBLK 001510	CKFMT 011232	DCKER1 010254	DT04 = 000020
ALOST 077262	BADENT 076341	CKHCE 011406	DDISP = 177570	DT05 = 000040
AMADR1= 000000	BADSEC 001342	CKLMTS 020130	DDRVS 001544	DT06 = 000100
AMADR2= 000000	BADTMO 003450	CKSWR = 104407	DEASGN 026356	DT07 = 000200
AMADR3= 000000	BAI = 000010	CK.CHR 033020	DEASSG 076003	DT08 = 000400
AMADR4= 000000	BEGCOD 001514	CK.DEC 032772	DEC2 001436	DT1 073134
AMAMS1= 000000	BEGPAT 001512	CK.DIG 033072	DH1 072467	DT14 073204
AMAMS2= 000000	BEGWC 001516	CK.NUM 101064	DH14 072644	DT15 073226
AMAMS3= 000000	BIT0 = 000001	CK.OCT 032744	DH15 072746	DT16 073250
AMAMS4= 000000	BIT00 = 000001	CLKFLG 001312	DH16 073044	DT17 073264
AMSGAD= 000000	BIT01 = 000002	CLOCK 025110	DH17 073114	DT2 073140
AMSGLG= 000000	BIT02 = 000004	CLR = 000040	DH2 072474	DT3 073156
AMSGTY= 000000	BIT03 = 000010	CLRDPB 026630	DH3 072550	DT4 073166
AMTYP1= 000000	BIT04 = 000020	CLRQUE 045300	DH4 072576	DT6 073200
AMTYP2= 000000	BIT05 = 000040	CMCNT 001370	DH6 072635	DVA = 004000
AMTYP3= 000000	BIT06 = 000100	CMCYL 001372	DISPLA 001156	DVC = 000200
AMTYP4= 000000	BIT07 = 000200	CMDAT 013500	DISPLY= 104414	ECBADO 001412
AOE = 001000	BIT08 = 000400	CMHED 013424	DISPRE 000174	ECBAD1 001420
APASS = 000000	BIT09 = 001000	CMPAR 013240	DLT = 100000	ECBIT 001376
APRIOR= 000000	BIT1 = 000002	CMPARD 013244	DONE 007550	ECC 014472
APTCSU= 000040	BIT10 = 002000	CMPLMT 001460	DPE = 000010	ECCX 015230
APTENV= 000001	BIT11 = 004000	CMPRES 020750	DPINT 040204	ECC1 015074
APTSIZ= 000200	BIT12 = 010000	CMPT 014060	DPR = 000400	ECC2 015224
APTSPO= 000100	BIT13 = 020000	CMPRX 014052	DPRQS 040214	ECGD 001410
ASGND 076053	BIT14 = 040000	CMSEC 001374	DRIVE = 001220	ECGD1 001416
ASGN1 025764	BIT15 = 100000	CMSTR 013342	DRIVE0 045416	ECH = 000100
ASGN2 026040	BIT? = 000104	CMSTR2 013466	DRIVE1 047632	ECI = 004000
ASGN3 026130	BIT3 = 000000	CMTRK 001375	DRIVE2 052046	ECMSK0 001402
ASGN4 026300	BIT4 = 000020	COLON 076332	DRIVE3 054262	ECMSK1 001404

ECSEC	001400	EOP2	031370	INCMIS	024714	LINR6	074130	MDPE	= 000400
ECWRD	001406	EQUAL	075274	INCSKI	024670	LINSS3	073600	MERR1	076362
ECWRD1	001414	ERCTR	001364	INCSOF	024620	LINST3	073564	MERR2	076442
EMPTYQ	045332	ERPRC1	007356	INCTOT	024740	LINS3	073445	MESFE	076562
EMTVEC=	000030	ERPROC	007342	INTRVL	001464	LINS4	073632	MESSAG	001504
EM1	067646	ERR	= 040000	INVL	076206	LINS5	073753	MINCYL=	000130
EM1G	070160	ERROR	= 104000	IOTVEC=	000020	LINT3	073514	MINSEC=	000140
EM11	070223	ERRVEC=	000004	IP	= 000100	LINU06	074142	MINTRK=	000134
EM12	070256	FACTOR	016240	ISR	042452	LINW3	073555	MINUTE	001346
EM13	070307	FAIRNS	001332	IVC	= 010000	LINX4	073646	MNTBL	002124
EM14	070361	FEFLAG	001434	KSR	025254	LINX5	075236	MOH	= 020000
EM15	070404	FEONLY	076674	KSR1	025262	LIN10A	074606	MOL	= 010000
EM2	067720	FER	= 000020	LBC	= 002000	LIN10B	074642	MREAD	077131
EM20	070440	FE1	001430	LBT	= 002000	LIN10C	074702	MRMCS1	101046
EM21	070461	FE2	001432	LF	= 000012	LIN10H	075006	MRMVEC	101055
EM22	070512	FILBUF	016632	LIMIT	001366	LIN11	075171	MSFULL	076506
EM23	070565	FILLZ	032464	LINA3	073475	LIN11A	075212	MSGCTS	076544
EM24	070644	FILLO	032572	LINB3	073544	LIN11H	075116	MSGFOR	075776
EM25	070712	FMTER	012166	LINB5	073716	LIN2C	073272	MSGON	075753
EM26	070773	FMT16 =	010000	LINB6	074030	LIN2P	073312	MSPRM	077344
EM27	071020	FRSTER	001356	LINCA3	073524	LIN2S	073333	MSWAIT	077406
EM3	067756	F0	= 000002	LINC6	074063	LIN3.1	022120	MSWRO	076143
EM30	071055	F1	= 000004	LINDA3	073535	LIN3.3	022226	MXF	= 001000
EM31	071107	F2	= 000010	LINDEC	023234	LIN3.4	022260	MXWWDW	040320
EM32	071152	F3	= 000020	LIND5	073700	LIN6.2	022722	M.DPID	032106
EM33	071205	F4	= 000040	LINEN3	073461	LIN7M	074167	M.DP40	032144
EM34	071262	GENDPB	067556	LINE05	074016	LIN7OR	074317	M.DP41	032200
EM35	071324	GENPAR	017370	LINEP5	074006	LIN7OX	074272	M.DP42	032210
EM36	071362	GENRCG	067576	LINE1	021134	LIN7P	074214	M.DP44	032242
EM37	071413	GETADR	027650	LINE2	021214	LIN7R	074327	M.DP50	032254
EM4	070014	GETBUF	016242	LINE2A	021364	LIN7S	074232	N	076135
EM40	071475	GETID	027550	LINE2B	021402	LIN7T	074254	NED	= 010000
EM41	071544	GETLMT	027462	LINE3	021654	LIN7X	074302	NEDCLK	076064
EM42	071610	GETPAT	021070	LINE3A	021662	LIN8M	074343	NEM	= 004000
EM43	071671	GETREG=	000141	LINE3B	021670	LIN9B	074366	NEWASN	026556
EM44	071765	GETREM	032014	LINE3C	021702	LIN9E	074531	NEWUNT	001566
EM45	072062	GETREQ	045366	LINE3D	021712	LIN9G	074560	NOGRVS	077103
EM46	072150	GO	= 000001	LINE3E	021760	LIN9H	074415	NOENTY	077176
EM47	072222	GODRIV	016710	LINE3F	022046	LIN9I	074473	NOMTCH	012646
EM5	070051	GTSWR =	104406	LINE4	022324	LKPAR	005232	NONE	076334
EM50	072277	HCE	= 000200	LINE5	022414	LODEV	075464	NOTAVL	075435
EM51	072335	HCEER	012230	LINE5A	022556	LODPAR	020500	NOTPRS	075420
EM52	072400	HCI	= 002000	LINE5B	022624	LSC	= 000000	NOTRM	075403
EM6	070105	HCRC	= 000400	LINE6	022666	LSTAD	001334	NOTSAF	075454
EM60	072444	HCRCER	011054	LINE6A	022700	LSTHDR	077216	NSA	= 100000
ENDCMP	014334	HDASN	077320	LINE6C	022706	MAIN	006240	OFFDIR=	000001
ENDCON	001446	HOUR	001344	LINE6D	022714	MAINLA	006266	OFFON =	000001
ENDING	001500	HT	= 000011	LINE7	022746	MAIN1	006404	OFLIN	007446
ENDPAS	075734	HZ	001314	LINE7A	023076	MAIN2	006522	ONES	003272
ENDPGM=	102200	IAE	= 002000	LINE8	023170	MANTER	030254	ONESEC	001352
ENDSEK	001452	IAEER	012106	LING6	074112	MASK	001326	OPE	= 020000
FNDTST	075760	IBSAVE	035226	LINM3	073367	MATCH	014402	OPI	= 020000
ENTADR	076302	IDLE	007104	LINM4	073614	MAXCYL=	000126	OPIER	011666
ENTCOM	076231	IE	= 000100	LINN3	073405	MAXER	001456	OPIER1	011732
ENTLMT	076252	ILF	= 000001	LINOCT	023202	MAXSEC=	000136	OPT	041236
ENTPR	005542	ILR	= 000002	LINP3	073426	MAXTRK=	000132	OPTBL	002104
EOP1	031344	INCHRD	024644	LINP5	073736	MCPE	= 020000	OR	= 000200

SYMBOL TABLE

ORDERQ	001520	PWRFLG	037716	RMHR	= 000036	SPOTCK	020764	SW5	= 000040
OVRWRT	076756	PWRVEC	= 000024	RMINIT	040322	SRCHWT	040246	SW6	= 000100
PACK	001320	QDRV	040224	RMLA	= 000020	SSE	= 000040	SW7	= 000200
PAR	= 000010	QUES	075272	RMMR1	= 000024	SSEI	= 001000	SW8	= 000400
PARENT	031030	RANCYL	017600	RMMR2	= 000040	STA	006006	SW9	= 001000
PARER	012014	RANDOM	001506	RMOF	= 000032	STAC	= 001100	SYSTAT	075501
PARLST	077466	RANDWC	001474	RMR	= 000004	STAPI	003532	T	073402
PAR1	077632	RANPAT	020040	RMSN	= 000030	STAT1	003542	TAB	075302
PAR10	077742	RANSEC	017670	RMTMR	044152	START2	003560	TABLE	100042
PAR11	077752	RANSIZ	017746	RMVEC	040312	STAR30	075566	TABLE0	100062
PAR14	077762	RANTRK	017634	RMWC	= 000002	STAR5	075617	TABLE1	100130
PAR15	077772	RANXIT	020060	RM80	041000	STATIN	001316	TABLE2	100176
PAR16	100002	RATIO	001476	RNOP	= 000101	STATIS	016072	TABLE3	100244
PAR19	100012	RDCHR	= 104410	RTC	= 000117	STATPR	023626	TABLE4	100312
PAR2	077642	RDDAT	= 000171	RTNCTR	015510	STKLMT	= 177774	TABLE5	100360
PAR20	100022	RDHD	= 000173	RTURN	031766	STNDAT	002324	TABLE6	100426
PAR21	100032	RDLIN	= 104411	R6	=X000006	STO	044244	TABLE7	100474
PAR3	077652	RDONLY	001440	R7	=X000007	STO1	044274	TAB.XY	= 001114
PAR4	077662	RDY	= 000200	S	073423	STO2	044462	TAP	= 040000
PAR5	077672	RD.ADR	044706	SATPOW	037736	STO3	044532	TBITVE	= 000014
PAR6	077702	RD.RM	044670	SAVEFG	040252	STO5	044556	TD	042514
PAR7	077712	RD.RM1	044704	SAVER1	001360	STO6	044564	THEAD	017514
PAR8	077722	RD.RM3	044742	SAVER5	001362	STO7	044622	TIMER	040256
PAR9	077732	RD.RM4	044746	SAVER5	001362	STO8	044652	TKVEC	= 000060
PASSES	001470	RD.WRD	044710	SAVREG	= 104412	STO9	044662	TPVEC	= 000064
PAT	= 000020	READDR	023254	SC	043334	SUMARY	023714	TRAPVE	= 000034
PATTER	001472	READHD	015652	SCMND	026464	SUMHD	075626	TRE	= 040000
PCLOCK	001310	READIN	= 000121	SCOPE	= 000004	SUPRS	032300	TRFER	012272
PERFEX	= 000001	RECAL	= 000107	SC04	= 000400	SUPRSL	032264	TRKLMT	001226
PERIOD	076141	RECALT	015534	SC1	= 000100	SUPR1	032312	TRNSWT	040244
PFECH	035406	RECALO	015624	SC10	= 001000	SUPR2	032354	TRTVFC	= 000014
PFECH1	035416	REDAPK	026566	SC11	043672	SURE	076646	TST	003550
PFECH2	035500	RELBUF	016376	SC12	043750	SVRH70	045062	TY'RV	032622
PFECH3	035532	RELSE	= 000113	SC13	044012	SWR	001154	TY'IDA	032652
PFECH4	035542	REPHD	075525	SC2	= 000200	SWREG	000176	TYLIST	030650
PF TSTN	035546	REPLZ	032470	SC20	= 002000	SWTIM	007410	TYPDRV	032626
PGE	= 002000	RESREG	= 104413	SC3	043400	SW0	= 000001	TYPDS	= 104405
PGM	= 001000	RESVEC	= 000010	SC4	043404	SWU0	= 000001	TYPE	= 104401
PIP	= 020000	RETRY	001330	SC5	043416	SW01	= 000002	TYPHDA	032656
PIRQ	= 177772	RHEXT	040316	SC6	043504	SW02	= 000004	TYPOC	= 104402
PIRQVE	= 000240	RMADR	040310	SC8	043656	SW03	= 000010	TYPON	= 104404
POPQUE	045400	RMAS	= 000016	SDETAL	023752	SW04	= 000020	TYPOS	= 104403
POSER	011614	RMBA	= 000004	SEARCH	= 000131	SW05	= 000040	TYPRI4	032574
PROCES	007252	RMBAE	= 000050	SECLMT	001424	SW06	= 000100	UNS	= 040000
PRTBAD	015236	RMCS1	= 000000	SECOND	001350	SW07	= 000200	UNSAF	012532
PRTIM	007512	RMCS2	= 000010	SEEK	= 000105	SW08	= 000400	UNTSN	075355
PRO	= 000000	RMCS3	= 000052	SEEKFG	040254	SW09	= 001000	UNTMGS	075304
PR1	= 000040	RMDA	= 000006	SELDRV	= 000145	SW1	= 000002	UNTNOT	075333
PR2	= 000100	RMDB	= 000022	SETFMT	= 000143	SW10	= 002000	UNTOFF	075312
PR3	= 000140	RMDC	= 000034	SETVEC	005570	SW11	= 004000	UNTON	075323
PR4	= 000200	RMDS	= 000012	SET.IE	045226	SW12	= 010000	UPE	= 020000
PR5	= 000240	RMDT	= 000026	SHDTYP	023742	SW13	= 020000	US1	= 000001
PR6	= 000300	RMEC1	= 000044	SIZE70	004556	SW14	= 040000	US2	= 001002
PR7	= 000340	RMEC2	= 000046	SIJMEM	005024	SW15	= 100000	US4	= 000004
PS	= 177776	RMEPRS	040146	SKI	= 040000	SW2	= 000004	VV	= 000100
PSEL	= 002000	RMER1	= 000014	SKIER	012372	SW3	= 000010	WAIT	001632
PSW	= 177776	RMER2	= 000042	SKIP	042746	SW4	= 000020	WATPAK	026612
				SLASH	077572				

WCE = 040000	\$DBLK 036546	\$LKS 001304	\$PWRUP 037610	\$SUPR2 032454
WCF = 000020	\$DB2D 037222	\$LLVEC 001306	\$QUES 001202	\$SVPC = 000210
WCFER 012434	\$DB20 037416	\$LONUM 037124	\$RAND 037024	\$SWR = 122000
WCHKX = 000000	\$DECVL 037402	\$LPADR 001122	\$RDCHR 034170	\$SWREG 001230
WCKD = 000151	\$DEVCT 001216	\$LPERR 001124	\$RDLIN 034260	\$STATUS= 000016
WCKER 010546	\$DEVN 001264	\$LPVEC 001302	\$RDOFL= 000064	\$TERM = 000032
WCKHD = 000153	\$DIV 032040	\$LSTAD 100672	\$RDSZ = 000017	\$TESTN 001212
WC.HK 043246	\$DOAGN 031762	\$MADR1 001240	\$READ = 000066	\$TIME 024764
WLE = 004000	\$DRVSN= 002130	\$MADR2 001244	\$REG = 000014	\$TKB 001162
WLEER 012140	\$SDPLY 032716	\$MADR3 001250	\$RESRE 037164	\$TKCNT 033310
WRDCNT 001462	\$DTBL 036536	\$MADR4 001254	\$RETRY 015744	\$TKINT 033326
WRL = 004000	\$ENDAD 031752	\$MAIL 001206	\$RHEXT= 000001	\$TKQEN= 033325
WRTCHK 001502	\$ENDAT= 000036	\$MAMS1 001236	\$RMADR 001272	\$TKQIN 033312
WRTDAT= 000161	\$ENDCT 031736	\$MAMS2 001242	\$RMAS = 002156	\$TKQOU 033314
WRTHD = 000163	\$ENDSK= 000042	\$MAMS3 001246	\$RMBA = 002144	\$TKQSR 033316
WRTPK 017000	\$ENV 001226	\$MAMS4 001252	\$RMBAE= 002210	\$TKS 001160
WRT.AD 045032	\$ENVM 001227	\$MBADR 001102	\$RMCS1= 002140	\$TKSRV 033376
WRT.RM 044750	\$EOP 031312	\$MFLG 037020	\$RMCS2= 002150	\$TMP0 001174
WRT.R1 045026	\$EOPCT 031730	\$MISPO= 000102	\$RMCS3= 002212	\$TN = 000002
WRT.R3 045052	\$ERFLG 001117	\$MNEW 034563	\$RMDA = 002146	\$TNPWR 037332
WRT.R4 045056	\$ERMAX 001131	\$MSGAD 001222	\$RMDB = 002162	\$TOTAL= 000072
WRT.R5 045060	\$ERROR 034640	\$MSGLG 001224	\$RMDC = 002174	\$TPB 001166
WRT.WD 045030	\$ERRPC 001132	\$MSGTY 001206	\$RMDS = 002152	\$TPFLG 001173
XXDP 001444	\$ERRTB 003370	\$MSWR 034552	\$RMDT = 002166	\$TPS 001164
Y 076137	\$ERRTY 035230	\$MTYP1 001237	\$RMEC1= 002204	\$TRAP 040060
ZEROS 002370	\$ERTTL 001126	\$MTYP2 001243	\$RMEC2= 002206	\$TRAP2 040102
ZROIND 001354	\$ETABL 001226	\$MTYP3 001247	\$RMER1= 002154	\$TRK = 000011
\$APTHD 001100	\$ETEND 001272	\$MTYP4 001253	\$RMER2= 002202	\$TRP = 000015
\$ATYC 036602	\$FAIR = 000106	\$NCODE= 000114	\$RMHR = 002176	\$TRPAD 040114
\$ATY1 036556	\$FATAL 001210	\$NCYL = 000120	\$RMI.A = 002160	\$TSTM 001104
\$ATY3 036564	\$FFLG 037022	\$NEXT = 000122	\$RMR1= 002164	\$TSTNM 001116
\$ATY4 036574	\$FILLC 001172	\$NPATC= 000115	\$RMR2= 002200	\$TTYIN 034514
\$AUTOB 001150	\$FILLS 001171	\$NSEC = 000116	\$RMOF = 002172	\$TYPDS 036332
\$BASE 001262	\$FIRST= 000124	\$NTRK = 000117	\$RMSN = 002170	\$TYPE 035550
\$BDADR 001136	\$FMT = 000001	\$NULL 001170	\$RMVEC 001274	\$TYPEC 035762
\$BDDAT 001142	\$GDADR 001134	\$NWTST= 000000	\$RMWC = 002142	\$TYPEX 036102
\$BDSEC= 000146	\$GDDAT 001140	\$SOCNT 036326	\$RM80 075520	\$TYPOC 036130
\$BELL 001176	\$GET42 031742	\$SOCTVL 037520	\$RTNAD 031764	\$TYPON 036144
\$BUF = 000006	\$GTSWR 033716	\$SOMODE 036330	\$SAVRE 037126	\$TYPOS 036104
\$CDW1 001266	\$HARD = 000076	\$OPERC= 000046	\$SAVR6 037714	\$UNIT 001220
\$CDW2 001270	\$HD = 000000	\$PACK = 000026	\$SB2D 033230	\$UNITM 001110
\$CHARC 036100	\$HIBTS 001100	\$PASS 001214	\$SB20 033260	\$USWR 001232
\$CKSWR 033626	\$HINUM 037122	\$PASSC= 000104	\$SEC = 000010	\$VECT1 001256
\$CMTAG 001114	\$HLDWC= 000110	\$PASTM 001106	\$SETUP= 000156	\$VECT2 001260
\$CM3 = 000000	\$HSNL = 000142	\$PATTC= 000030	\$SIZE 100542	\$WCNT = 000004
\$CM4 = 000001	\$HSNM = 000144	\$POSIT= 000052	\$SKI = 000100	\$WRDL = 000020
\$CNTLC 034533	\$ICNT 001120	\$POWER 037720	\$SOFT = 000074	\$WRITN= 000060
\$CNTLG 034545	\$ILLUP 037710	\$PREVA= 000032	\$SSEC = 000022	\$WTOFL= 000056
\$CNTLU 034540	\$INTAG 001151	\$PREVO= 000027	\$SSENB= 000112	\$XOFF = 000023
\$CODE = 000024	\$ITEMB 001130	\$PSEL = 000003	\$STUP = 177777	\$XON = 000021
\$COMND= 000002	\$LF 001204	\$PWRAD 037704	\$SUPRL 032364	\$GET4= 000000
\$CPUOP 001234	\$LFLG 037021	\$PWRDN 037536	\$SUPRS 032400	\$OFILL 036327
\$CRLF 001203	\$LKCSB 001300	\$PWRMG 037700	\$SUPR1 032412	\$.X = 001100
\$CYL = 000012	\$LKCSR 001276			
. ABS. 102200 000				
000000 001				

ERRORS DETECTED: 0

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

\$RMADR	7-0#	9-73*	9-85	9-118	9-132	62-5	67-21	67-43						
\$RMAS	59-84#	62-7												
\$RMBA	12-739	13-247	13-281	13-303	13-341	13-380	14-9	22-103	22-153	22-258	22-269*	22-275*	22-278*	22-279
\$RMBAE	59-79#	62-7												
\$RMCS1	59-98#	62-10												
	12-6	12-8	12-715	13-413	15-24	20-11	22-40	56-143	59-77#	59-78	59-79	59-80	59-81	59-82
	59-83	59-84	59-85	59-86	59-87	59-88	59-89	59-90	59-91	59-92	59-93	59-94	59-95	59-96
	59-98	59-99	59-113	59-113	59-113	59-113	59-113	59-113	59-113	59-113	62-6			
\$RMCS2	12-132	12-332	12-617	12-717	59-81#	62-6								
\$RMCS3	22-276	29-27	59-99#	59-113	59-113	59-113	59-113	59-113	59-113	59-113	59-113	62-10		
\$RMDA	15-27	16-25	22-186	22-241	23-19	23-20	59-80#	62-7						
\$RMDB	22-285	59-86#	62-7											
\$RMDC	12-458	15-28	16-26	22-190	22-229	23-21	59-91#	62-8						
\$RMDS	12-10	12-136	12-174	59-82#	62-6									
\$RMDT	59-88#	62-7												
\$RMEC1	12-202	12-208	12-210	13-266	22-316	59-95#	62-6							
\$RMEC2	12-212	13-285	22-320	59-96#	62-6									
\$RMER1	12-128	12-140	12-144	12-148	12-152	12-156	12-160	12-164	12-168	12-172	12-178	12-182	12-200	12-242
	12-261	12-263	12-311	12-341	12-355	12-357	12-410	12-449	12-719	13-502	13-505	59-83#	62-6	
\$RMER2	12-186	12-188	12-669	12-680	12-721	59-94#	62-6							
\$RMHR	59-92#	62-8												
\$RMLA	59-85#	62-7												
\$RMR1	59-87#	62-7												
\$RMR2	59-93#	62-6												
\$RMOF	15-30	16-28	29-145	59-90#	62-8									
\$RMSN	30-14	59-89#	62-8											
\$RMVEC	7-0#	9-70*	9-86	67-32										
\$RMWC	12-734	13-12	13-248	13-343	22-100	22-156	22-287	59-78#	62-7					
\$RTNAD	35-1#													
\$SAVR6	51-1	51-1#	51-1*	51-1*	51-1*									
\$SAVRE	48-1#	52-1	52-1											
\$SB2D	13-277	22-478	24-82	26-14	26-19	26-24	32-101	32-107	32-115	39-226#				
\$SB2O	22-462	39-243#												
\$SEC	13-18	13-425*	13-468*	15-19	15-25	15-27*	15-32*	15-36	15-39*	15-50	16-30*	20-16	20-36*	22-175
	29-127*	31-41*	31-52*	31-53	31-66	31-82	31-84*	53-207	56-77*	56-145	59-10#			
\$SETUP	4-267	4-267	4-267	4-267	4-267	4-267#	4-267#	4-267#	4-267#	4-267#	4-267#	9-25	9-25	9-25
	9-25	9-25	9-25	9-25	9-25	9-25	9-25	9-25	9-25	9-30	9-30	9-30	35-1	35-1
	40-1	40-1	40-1	40-1	40-1	41-1	41-1	41-1	41-1	41-1	41-1			
\$SIZE	9-156	66-1#												
\$SKI	22-440	24-138	24-147	25-18	25-18*	59-36#								
\$SOFT	24-132	24-147	25-16	25-16*	59-34#									
\$SSEC	13-24	13-29	13-30	13-254	13-348	15-60*	20-38*	20-41*	22-290	29-43*	29-47*	59-19#		
\$SSENB	11-173*	21-25	29-147	56-124*	59-41#									
\$STUP	4-267	4-267	4-267	4-267	4-267	4-267#	4-267#	4-267#	4-267#	4-267#	4-267#	4-267#	4-267#	4-267#
	4-267#													
\$SUPR1	38-19	38-24#												
\$SUPR2	38-18*	38-23*	38-33	38-35*	38-37#									
\$SUPRL	13-278	22-406	22-414	22-435	22-479	38-16#								
\$SUPRS	38-21#													
\$SVPC	5-5	5-5#												
\$SWR	4-47#	4-57	4-58	4-58	4-58	4-58	4-58	4-58	4-58	4-58	6-0	6-0	6-0	9-19
	9-25	35-1	35-1	35-1	35-1	35-1	41-1	41-1	41-1	41-1	41-1	41-1	41-1	41-1
	41-1	41-1	41-1	51-1										
\$SWREG	6-0#	9-25												
\$STATUS	11-167	12-4	12-24	12-39	12-41	12-43	12-124	12-252	12-255	12-345	13-418	13-438	13-476	13-492
	13-498	14-7	22-97	22-249	29-20	31-49	53-211	56-67*	56-142*	59-14#	62-8			

A16	4-84#	9-136	9-140			
A17	4-85#	9-136	9-140			
ABASE	4-264#	6-0	6-0			
ABNRML	11-175	34-36#				
ACDW1	6-0	6-0				
ACDW2	6-0	6-0				
ACK	4-252#					
ACPUOP	6-0	6-0				
ACTDRV	53-118#	55-13*	55-57*	56-3*	56-11*	57-6
ACTSTR	53-124#	57-8*	57-23*			
ADDW0	6-0					
ADDW1	6-0					
ADDW10	6-0					
ADDW11	6-0					
ADDW12	6-0					
ADDW13	6-0					
ADDW14	6-0					
ADDW15	6-0					
ADDW2	6-0					
ADDW3	6-0					
ADDW4	6-0					
ADDW5	6-0					
ADDW6	6-0					
ADDW7	6-0					
ADDW8	6-0					
ADDW9	6-0					
ADEVCT	6-0	6-0				
ADEVN	6-0	6-0				
AENV	6-0	6-0				
AENVN	6-0	6-0				
AFATAL	6-0	6-0				
AHOST	32-50	64-59#				
AMADR1	6-0	6-0				
AMADR2	6-0	6-0				
AMADR3	6-0	6-0				
AMADR4	6-0	6-0				
AMAMS1	6-0	6-0				
AMAMS2	6-0	6-0				
AMAMS3	6-0	6-0				
AMAMS4	6-0	6-0				
AMSGAD	6-0	6-0				
AMSGLG	6-0	6-0				
AMSGTY	6-0	6-0				
AMTYP1	6-0	6-0				
AMTYP2	6-0	6-0				
AMTYP3	6-0	6-0				
AMTYP4	6-0	6-0				
AOE	4-156#					
APASS	6-0	6-0				
APRIOR	6-0					
APTCSU	43-1	46-1#				
APTENV	41-1	43-1	46-1	46-1#		
APTSIZ	9-25	46-1#				
APTSP0	43-1	46-1	46-1#			
ASGN1	27-113#					
ASGN2	27-111	27-126#				

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

DH16	22-85	61-10#												
DH17	22-93	61-12#												
DH2	8-12	8-33	61-2#											
DH3	8-19	61-3#												
DH4	8-26	61-4#												
DH6	8-40	61-5#												
DISPLA	6-0#	9-25*	9-25*	41-1*										
DISPLY	12-86	12-98	12-109	12-205	12-215	12-218	12-234	12-258	12-305	12-314	12-336	12-374	12-401	12-423
	12-467	12-494	12-510	12-532	12-538	12-557	12-567	12-576	12-594	12-612	12-633	12-645	12-665	12-691
	12-696	12-697	12-698	12-706	12-706	12-706	12-706	12-706	12-706	12-706	12-706	12-724	12-742	13-158
	13-163	13-164	13-165	13-181	13-184	13-187	13-196	13-199	13-274	13-279	13-316	13-321	13-321	13-321
	13-325	13-330	13-330	13-330	13-333	13-334	13-360	13-361	13-364	13-367	13-370	13-373	13-375	13-379
	13-382	13-387	13-389	13-390	13-515	22-13	22-14	22-16	22-17	22-19	22-36	22-62	22-64	22-70
	22-71	22-72	22-73	22-74	22-77	22-82	22-85	22-93	22-105	22-106	22-116	22-119	22-125	22-131
	22-138	22-152	22-155	22-158	22-164	22-167	22-168	22-172	22-173	22-177	22-185	22-188	22-189	22-192
	22-202	22-207	22-209	22-212	22-216	22-220	22-225	22-228	22-235	22-239	22-243	22-251	22-254	22-257
	22-262	22-268	22-284	22-293	22-295	22-302	22-307	22-307	22-308	22-309	22-315	22-318	22-319	22-322
	22-328	22-329	22-343	22-357	22-363	22-382	22-383	22-396	22-399	22-402	22-407	22-410	22-415	22-431
	22-436	22-439	22-442	22-451	22-465	38-36	52-2#							
DISPRE	5-1#	9-25												
DLT	4-114#													
DONE	12-26	12-124#												
DPE	4-224#													
DPINT	53-79#	54-46	54-49	55-21	56-178	56-191	56-275	56-277*	57-44	57-80	57-89	57-97*		
DPR	4-136#													
DPRQS	53-88#	55-27	55-105*	56-181	56-289*	57-46	57-82	57-91	57-106*					
DRIVE	7-0#	41-1*	62-2	62-3	62-4									
DRIVE0	7-0	9-103*	9-155*	59-113#	65-54	65-54	65-54	65-54	65-54	65-54	65-54	65-54	65-54	65-54
DRIVE1	7-0	9-103*	9-155*	59-113#	65-54	65-54	65-54	65-54	65-54	65-54	65-54	65-54	65-54	65-54
DRIVE2	7-0	9-103*	9-155*	59-113#	65-54	65-54	65-54	65-54	65-54	65-54	65-54	65-54	65-54	65-54
DRIVE3	7-0	9-103*	9-155*	59-113#	65-54	65-54	65-54	65-54	65-54	65-54	65-54	65-54	65-54	65-54
DRIVE4	7-0	9-103*	9-155*	59-113#	65-54	65-54	65-54	65-54	65-54	65-54	65-54	65-54	65-54	65-54
DRIVE5	7-0	9-103*	9-155*	59-113#	65-54	65-54	65-54	65-54	65-54	65-54	65-54	65-54	65-54	65-54
DRIVE6	7-0	9-103*	9-155*	59-113#	65-54	65-54	65-54	65-54	65-54	65-54	65-54	65-54	65-54	65-54
DRIVE7	7-0	9-103*	9-155*	59-113#	65-54	65-54	65-54	65-54	65-54	65-54	65-54	65-54	65-54	65-54
DRNUM	27-159	64-30#												
DROP	12-104	34-20#	34-40											
DROPNG	34-27	64-24#												
DRQ	4-192#													
DRVACT	53-50#	55-32	55-237*	55-243	55-253*	56-16*	56-184	56-222*	56-237*	57-55*	57-60	57-67*		
DRVCLR	4-245#													
DRVER	12-190	12-400#												
DRVINT	54-37	54-71#	55-23	55-76	56-256	56-278	57-58							
DRVNO	7-0#	12-3*	22-75	35-1	35-1	35-1*								
DRVPAR	7-0#	27-93*	27-109*	27-157	29-64	29-86	32-13							
DRVPRM	10-33	27-167	29-58#											
JRVQUE	55-30	55-38	56-140	58-162#										
DRVSN	39-60	64-61#												
DRVSTA	9-276	10-24	27-151	53-60#	54-27*	54-28*	54-29*	54-30*	54-40*	54-72*	54-80*	54-103*	54-108*	55-19
	55-25	55-44	55-78	55-82	55-259*	56-280	57-98*							
DRVSTYP	9-280	27-177	53-70#	54-73*	54-85*	54-90*	55-47	55-260*						
DRY	4-135#													
DSWR	4-76#	6-0	9-25											
DT00	4-183#													
DT01	4-184#													
DT02	4-185#													

C
C
W
X
Y
Z

MANTER	27-168	32-10#												
MASK	7-0#	12-241*	12-261	12-315*	12-386*	12-435*	12-479*	12-515*	12-543*	12-620*	12-652*	12-671*	13-500	13-505
MATCH	13-55	13-220#												
MAXCYL	17-3	18-37	19-28	29-71*	29-98	29-101*	29-121	29-123*	59-55#	59-56	59-57	59-58	59-59	59-60
	59-64	65-54	65-54	65-54	65-54	65-54	65-54	65-54	65-54					
MAXER	7-0#	34-38												
MAXSEC	17-25	18-28	19-19	19-34	29-73*	29-110	29-113*	59-59#	65-54	65-54	65-54	65-54	65-54	65-54
	65-54	65-54												
MAXTRK	17-14	18-32	19-23	19-30	29-72*	29-104	29-107*	59-57#	65-54	65-54	65-54	65-54	65-54	65-54
	65-54	65-54												
MCPE	4-87#													
MDPE	4-107#													
MERR1	31-94	64-44#												
MERR2	31-92	64-45#												
MESFE	9-192	64-48#												
MESSAG	7-0#	21-50	65-11											
MINCYL	15-41	16-47	17-4	17-6	17-9	18-16	18-18	29-76*	29-78*	29-97	29-102*	29-118	29-120*	29-125
	59-56#	65-54	65-54	65-54	65-54	65-54	65-54	65-54	65-54					
MINSEC	15-39	16-45	17-26	17-28	17-31	18-22	18-24	18-30	18-34	19-17	29-80*	29-109	29-114*	29-127
	59-60#	65-54	65-54	65-54	65-54	65-54	65-54	65-54	65-54					
MINTRK	15-40	16-46	17-15	17-17	17-20	18-19	18-21	18-35	19-21	29-79*	29-103	29-108*	29-126	59-58#
	65-54	65-54	65-54	65-54	65-54	65-54	65-54	65-54	65-54					
MINUTE	7-0#	9-96*	11-87*	26-18	26-39*	26-40	26-42*							
MNTBL	7-0#	22-60												
MOH	4-193#													
MOL	4-140#													
MREAD	9-232	64-56#												
MRMCS1	67-22	67-49#												
MRMVEC	67-33	67-50#												
MSFULL	32-26	64-46#												
MSGCTS	32-33	64-47#												
MSGFOR	35-1	64-28#												
MSGON	34-28	35-1	64-26#											
MSPRM	27-94	64-62#												
MSWAIT	51-21	64-63#												
MSWRO	27-79	64-36#												
MXF	4-108#													
MXWINDW	53-173#	55-159												
N	64-33#													
NED	4-111#													
NEDCLK	23-70	64-32#												
NEM	4-110#													
NEWASN	27-42	28-49#												
NEWUNT	7-0#	10-32*	11-53	11-69	11-70*	27-169*								
NODRVS	11-90	64-55#												
NOENTY	32-125	64-57#												
NOMTCH	12-688#	13-57												
NONE	39-81	64-42#												
NOTAVL	64-15#													
NOTPRS	9-286	27-182	64-14#											
NOTRM	9-283	27-180	64-13#											
NOTSAF	9-292	27-174	64-16#											
NSA	4-195#													
OFFDIR	4-210#													
OFFQI	4-133#													
OFFSET	7-0	12-247	12-268	12-285	13-442	22-332	22-366							

PR3	4-76#																	
PR4	4-76#	27-13																
PR5	4-76#																	
PR6	4-76#	9-28																
PR7	4-76#																	
PROCES	11-172	12-3#																
PRTBAD	12-240	12-282	12-650	13-339#														
PRTIM	12-44	12-108#																
PS	4-76	4-76#	9-93*	27-13*	27-86*	35-1*	39-43	39-44*	39-98*	51-14*	54-15	54-16*	54-45*	54-53*				
	55-11	55-12*	55-58*	55-68	55-114*													
PSEL	4-86#																	
PSW	4-76#																	
PWRFLG	9-179	9-267	10-35	10-48*	29-11	51-1*	51-2#											
PWRVEC	4-76#	9-25*	9-25*	51-1*	51-1*	51-1*	51-1*	51-1*	51-1*	51-1*								
QDRV	53-95#	58-131	58-149*	58-163	58-165*	58-179	58-191	58-192*										
QUES	34-8	64-3#																
R6	4-76#	9-25	9-25*	9-25*														
R7	4-76#																	
RANCYL	16-36	17-3#	17-40															
RANDOM	7-0#	16-35	65-12															
RANDWC	7-0#	17-46	65-7															
RANPAT	17-68#																	
RANSEC	17-25#																	
RANSIZ	16-49	17-45#	17-53															
RANTRK	17-14#																	
RANXIT	17-69	17-72#																
RATIO	7-0#	10-8*	16-15	65-8														
RD.ADR	58-11*	58-12*	58-14#	62-3														
RD.RM	54-83	54-96	54-100	55-144	55-187	55-198	55-208	56-25	56-44	56-47	56-50	56-74	56-76	56-78				
	56-99	56-121	56-155	56-208	56-212	57-41	57-86	58-10#	58-42	58-81								
RD.RM1	58-13#																	
RD.RM3	58-20	58-23#																
RD.RM4	58-22	58-25#																
RD.WRD	58-15#	58-16	62-3	62-4														
RDCHR	40-1	52-1#																
RDDAT	4-261#	7-0	15-65	20-32	31-43													
RDHD	4-262#	7-0	13-471															
RDLIN	9-193	9-212	9-238	27-22	27-95	32-34	33-19	52-1#	67-26	67-37								
RONLY	7-0#	9-104*	9-107*	9-207	9-230	15-67	16-9	27-54	27-66									
RDY	4-83#																	
READDR	12-371	12-414	12-452	13-420	20-18	21-28	22-200	23-14#										
READHD	12-372	12-415	12-453	13-467#														
READIN	4-251#																	
RECAL	4-244#	13-414	13-434															
RECALO	10-28	13-434#	27-149															
RECALT	12-502	12-639	12-683	13-412#														
REDAPK	27-62	28-54#																
RELBUF	11-183	14-86#																
RELSE	4-246#																	
REPHD	11-192	64-20#																
REPLZ	24-83	39-13#																
RESREG	14-164	27-83	36-49	49-1	50-1	52-1#	54-54	55-52	55-55	55-115	55-262	56-10	57-22	58-98				
	58-139																	
RESVEC	4-76#																	
RETRY	7-0#	12-221*	12-246*	12-265*	12-266	12-266	12-320*	12-343*	12-348*	12-352*	12-353	12-353	12-387*	12-436*				
	12-480*	12-516*	12-544*	12-619*	12-651*	12-672*	13-495*	13-507*	13-508	13-508	22-380							

SSCMRE	5-586#													
SSCMTM	5-586#	6-0												
SSESCA	4-76#													
SSNEWT	4-76#	9-19												
SSSET	52-1	52-1	52-1	52-1	52-1	52-1	52-1	52-1	52-1	52-1	52-1	52-1#	52-2	
SSSETM	9-25	9-25#												
SSSKIP	4-76#													
.SACT1	4-50#	5-5												
.SAPT8	4-53#	6-0	6-0#											
.SAPTH	4-53#	5-8												
.SAPTY	4-53#	46-1												
.SCATC	4-51#	5-1												
.SCMTA	4-51#	5-586												
.\$DB2D	4-52#	49-1												
.\$DB2O	4-52#	50-1												
.\$EOP	4-53#	35-1												
.\$ERRO	4-51#	41-1												
.\$ERRT	4-51#	42-1												
.\$POWE	4-53#	51-1												
.\$RAND	4-52#	47-1												
.\$READ	4-50#	40-1												
.\$SAVE	4-52#	48-1												
.\$SIZE	4-52#	66-1												
.\$STRAP	4-52#	52-1												
.\$STYPD	4-51#	45-1												
.\$STYPE	4-50#	43-1												
.\$STYPO	4-51#	44-1												
.\$EQUAT	4-50#	4-76												
.\$HEADE	4-50#	4-57												
.\$SETUP	4-50#	4-267												
.\$SWRHI	4-50#	4-58												
.\$SWRLO	4-50#	4-58#	4-59	4-60	4-61	4-63	4-65	4-70	4-73	4-74				
A	25-2#	25-16	25-17	25-18	25-19	25-20								
CKCHR	4-11#	39-189	39-197											
CKDIG	4-21#	32-53	32-60	32-67	33-23									
CKNUM	4-34#													
COMMEN	4-76#													
ENDCOM	4-76#													
ERENTR	40-13#	41-1												
ERRCAL	53-1#	56-158	56-196	56-198										
ERROR	4-76#													
ESCAPE	4-76#													
GETPFI	4-76#	66-1												
GETSWR	4-76#	9-30	9-30#											
MORETA	5-11#	6-0												
MULT	4-76#													
NEWTST	4-76#	9-19												
POP	4-76#	12-461	12-497	13-150	21-54	22-109	24-33	28-44	29-48	29-128	30-29	31-98	31-107	32-80
	32-123	39-34	45-1	46-1	46-1	47-1	48-1	51-1	51-1					
PUSH	4-76#	12-456	13-143	21-21	22-35	22-400	22-408	23-45	24-7	28-28	29-10	30-10	31-33	31-37
	32-10	32-93	45-1	46-1	46-1	46-1	47-1	48-1	51-1	51-1				
READ	53-8#	54-83	54-96	54-100	55-144	55-187	55-198	55-25	56-44	56-47	56-50	56-74	56-76	56-78
	56-99	56-121	56-155	56-208	56-212	57-41	57-86	58-42						
REPORT	4-76#													
SETPRI	4-76#	40-1												
SETTRA	52-1	52-1	52-1	52-1	52-1	52-1	52-1	52-1	52-1	52-1	52-1	52-1#	52-2	

