

RM03

DRIVE COMPATIBILITY TEST
MD-11-DZRMI-A

EP-DZRMI-A-DL-A

OCT 1977

COPYRIGHT © 1977

digital

FICHE 1 OF 1

MADE IN USA

[Frame 1: Test parameters]	[Frame 2: Test results]	[Frame 3: Test results]	[Frame 4: Test results]	[Frame 5: Test results]	[Frame 6: Test results]	[Frame 7: Test results]	[Frame 8: Test results]	[Frame 9: Test results]	[Frame 10: Test results]	[Frame 11: Test results]	[Frame 12: Test results]
[Frame 13: Test parameters]	[Frame 14: Test results]	[Frame 15: Test results]	[Frame 16: Test results]	[Frame 17: Test results]	[Frame 18: Test results]	[Frame 19: Test results]	[Frame 20: Test results]	[Frame 21: Test results]	[Frame 22: Test results]	[Frame 23: Test results]	[Frame 24: Test results]
[Frame 25: Test parameters]	[Frame 26: Test results]	[Frame 27: Test results]	[Frame 28: Test results]	[Frame 29: Test results]	[Frame 30: Test results]	[Frame 31: Test results]	[Frame 32: Test results]	[Frame 33: Test results]	[Frame 34: Test results]	[Frame 35: Test results]	[Frame 36: Test results]
[Frame 37: Test parameters]	[Frame 38: Test results]	[Frame 39: Test results]	[Frame 40: Test results]	[Frame 41: Test results]	[Frame 42: Test results]	[Frame 43: Test results]	[Frame 44: Test results]	[Frame 45: Test results]	[Frame 46: Test results]	[Frame 47: Test results]	[Frame 48: Test results]
[Frame 49: Test parameters]	[Frame 50: Test results]	[Frame 51: Test results]	[Frame 52: Test results]	[Frame 53: Test results]	[Frame 54: Test results]	[Frame 55: Test results]	[Frame 56: Test results]	[Frame 57: Test results]	[Frame 58: Test results]	[Frame 59: Test results]	[Frame 60: Test results]
[Frame 61: Test parameters]	[Frame 62: Test results]	[Frame 63: Test results]	[Frame 64: Test results]	[Frame 65: Test results]	[Frame 66: Test results]	[Frame 67: Test results]	[Frame 68: Test results]	[Frame 69: Test results]	[Frame 70: Test results]	[Frame 71: Test results]	[Frame 72: Test results]
[Frame 73: Test parameters]	[Frame 74: Test results]	[Frame 75: Test results]	[Frame 76: Test results]	[Frame 77: Test results]	[Frame 78: Test results]	[Frame 79: Test results]	[Frame 80: Test results]	[Frame 81: Test results]	[Frame 82: Test results]	[Frame 83: Test results]	[Frame 84: Test results]
[Frame 85: Test parameters]	[Frame 86: Test results]	[Frame 87: Test results]	[Frame 88: Test results]	[Frame 89: Test results]	[Frame 90: Test results]	[Frame 91: Test results]	[Frame 92: Test results]	[Frame 93: Test results]	[Frame 94: Test results]	[Frame 95: Test results]	[Frame 96: Test results]
[Frame 97: Test parameters]	[Frame 98: Test results]	[Frame 99: Test results]	[Frame 100: Test results]	[Frame 101: Test results]	[Frame 102: Test results]	[Frame 103: Test results]	[Frame 104: Test results]	[Frame 105: Test results]	[Frame 106: Test results]	[Frame 107: Test results]	[Frame 108: Test results]
[Frame 109: Test parameters]	[Frame 110: Test results]	[Frame 111: Test results]	[Frame 112: Test results]	[Frame 113: Test results]	[Frame 114: Test results]	[Frame 115: Test results]	[Frame 116: Test results]	[Frame 117: Test results]	[Frame 118: Test results]	[Frame 119: Test results]	[Frame 120: Test results]
[Frame 121: Test parameters]	[Frame 122: Test results]	[Frame 123: Test results]	[Frame 124: Test results]	[Frame 125: Test results]	[Frame 126: Test results]	[Frame 127: Test results]	[Frame 128: Test results]	[Frame 129: Test results]	[Frame 130: Test results]	[Frame 131: Test results]	[Frame 132: Test results]
[Frame 133: Test parameters]	[Frame 134: Test results]	[Frame 135: Test results]	[Frame 136: Test results]	[Frame 137: Test results]	[Frame 138: Test results]	[Frame 139: Test results]	[Frame 140: Test results]	[Frame 141: Test results]	[Frame 142: Test results]	[Frame 143: Test results]	[Frame 144: Test results]

B01

CO1

.REM 2

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

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DZRM1-A-D
PRODUCT NAME: RMO3 DRIVE COMPATIBILITY TEST
DATE CREATED: AUGUST 1977
MAINTAINER: DIAGNOSTIC ENGINEERING
AUTHOR: C. CHEN

COPYRIGHT (C) 1977 DIGITAL EQUIPMENT CORP., MAYNARD, MASS.

THE INFORMATION IN THIS STATEMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.

ACTUAL DISTRIBUTION OF THE SOFTWARE DESCRIBED IN THIS DOCUMENT WILL BE SUBJECT TO TERMS AND CONDITIONS TO BE ANNOUNCED ON SOME FUTURE DATE BY DIGITAL EQUIPMENT CORPORATION.

DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.

THIS SOFTWARE IS FURNISHED TO PURCHASER UNDER A LICENSE TO USE ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH INCLUSION OF DEC'S COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY OTHERWISE BE PROVIDED IN WRITING BY DEC.

TABLE OF CONTENTS

49	
50	
51	
52	
53	
54	
55	
56	
57	
58	
59	
60	
61	1.0 INTRODUCTION
62	
63	2.0 HARDWARE REQUIREMENTS
64	
65	3.0 PRELIMINARY PROGRAM REQUIREMENTS
66	
67	4.0 GENERAL PROGRAM CONSIDERATIONS
68	4.1 SYSMAC
69	4.2 XXDP
70	4.3 ACT
71	4.4 APT
72	4.5 DUAL-ACCESS
73	4.6 MEMORY MANAGEMENT
74	4.7 MEMORY PARITY OPTION
75	4.8 BAD SECTORS
76	4.9 EXECUTION TIME
77	
78	5.0 PROGRAM LOAD MEDIA
79	
80	6.0 PROGRAM OPTIONS
81	6.1 STARTING ADDRESSES
82	6.2 SWITCH REGISTER OPTIONS USED
83	
84	7.0 RUNNING THE PROGRAM
85	
86	8.0 OPERATIONAL DIALOGUE
87	8.1 DIALOGUE FOR ADDRESS 200 START
88	8.2 DIALOGUE FOR ADDRESS 204 START
89	8.3 DIALOGUE FOR ADDRESS 220 START
90	8.4 PASS 1 DIALOGUE
91	8.5 PASS 2 DIALOGUE
92	
93	9.0 DESCRIPTION OF TESTS
94	9.1 DESCRIPTION OF PASS 1 TESTS
95	9.2 DESCRIPTION OF PASS 2 TESTS
96	
97	10.0 PRINTOUT OF TEST RESULTS
98	10.1 OVERWRITE AND DRIVE COMPATIBILITY DATA TEST RESULTS
99	
100	11.0 ERROR REPORTING
101	11.1 COMMON ERRORS
102	11.2 ERROR HANDLING
103	11.3 ERROR PRINTOUT EXAMPLE
104	

EO1

MAINDEC-11-DZRM1 AQ/00, RMO3 DRIVE COMPATIBILITY TEST MACY11 30(1046) 21-JUL-77 16:51 PAGE 4
DZRMIA.P11 21-JUL-77 15:44

SEQ 0003

105
106
107

TABLE A - BASIC READ/WRITE TEST SECTORS

TABLE B - WORSE CASE DATA PATTERN

108
109
110
111
112
113
114
115
116
117
118
119
120
121
122

TABLE C - CYLINDER BLOCK ASSIGNMENT FOR A GIVEN SURFACE

TABLE D - BASIC CYLINDER BLOCK LAYOUT EXAMPLE

TABLE E - OVERWRITE CYLINDERS

TABLE F - SELF-TEST CYLINDERS

TABLE G - PSEUDO-RANDOM DATA PATTERN

12.0 RM03 SOFTWARE DRIVER DOCUMENT

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

1.0 INTRODUCTION

THE PURPOSE OF THIS PROGRAM IS TO VERIFY THE COMPATIBILITY OF UP TO 16 RMD3S DRIVES WHICH MAY RESIDE ON ONE / MORE RH11(70)/RMD3 SUBSYSTEMS.

COMPATIBILITY IS DEFINED HERE AS THE ABILITY OF A DRIVE TO WRITE DATA WHICH CAN BE READ SUCCESSFULLY BY ALL OTHER DRIVES, AND ADDITIONALLY THE ABILITY OF A DRIVE TO COMPLETELY OVER-WRITE DATA WRITTEN BY ALL OTHER DRIVES.

THE PROGRAM IS DESIGNED TO DETECT THE FOLLOWING CONDITIONS WHICH MOST COMMONLY CAUSE INCOMPATIBILITY BETWEEN DRIVES:

1. HEAD MIS-ALIGNMENT
2. POSITIONER LATERAL MISALIGNMENT
3. SPINDLE-CARTRIDGE INTERFACE RUNOUT
4. IMPROPER LEVELS OF WRITE CURRENT
5. INCORRECT ADDRESSING OF READ/WRITE HEADS

THE TESTING IS DONE IN TWO PASSES. IN PASS 1, COMPATIBILITY DATA PATTERNS ARE WRITTEN BY ALL THE DRIVES UPON THE SAME DISK CARTRIDGE, AND THE BASIC READ/WRITE CAPABILITY OF EACH DRIVE IS DEMONSTRATED. IN PASS 2, THE COMPATIBILITY DATA FROM ALL DRIVES IS READ BY EACH DRIVE, WITH HEAD OFFSET, AND THIS IS COMPARED WITH EACH DRIVE'S ABILITY TO READ ITS OWN DATA. IN ADDITION, EACH DRIVE'S CAPABILITY TO OVERWRITE DATA WRITTEN BY ALL OTHER DRIVES IS TESTED ON THE SECOND PASS. (FOR THE REMAINDER OF THIS SPECIFICATION, THE ABOVE DEFINITIONS OF THE FIRST AND SECOND PASS SHALL APPLY).

IN BOTH PASSES, THE PROGRAM DIRECTS THE OPERATOR IN THE LOADING AND UNLOADING OF DRIVES AND THE MOVEMENT OF THE CARTRIDGE FROM DRIVE TO DRIVE, THROUGH MESSAGES AT THE CONSOLE TERMINAL. AT THE COMPLETION OF TESTING ON EACH DRIVE DURING THE SECOND PASS A SUMMARY IS PRINTED OF COMPATIBILITY TEST RESULTS FOR THAT DRIVE.

WITHIN THE VARIOUS TESTS OF BOTH PASSES, THE CAPABILITY IS PROVIDED TO LOOP ON CURRENT OPERATIONS, AND SWITCH REGISTER OPTIONS ARE PROVIDED, FOR A VARIETY OF LOOPING, RUNNING, AND REPORTING MODES (SEE SECTION 6.2).

UNEXPECTED ERRORS WILL BE REPORTED AS THEY OCCUR. THE REPORT WILL INCLUDE DESCRIPTION AND APPLICABLE DEVICE REGISTER CONTENTS.

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

2.0 HARDWARE REQUIREMENTS

THE FOLLOWING HARDWARE IS REQUIRED TO RUN THE RMO3 DRIVE COMPATIBILITY PROGRAM, FOR EACH SUBSYSTEM WHICH MAY BE PRESENT:

PDP-11/04, (05,10 MFG. ONLY), 20,30,34,35,40,45,50,70
16K MEMORY
CONSOLE TERMINAL
RH11(70) CONTROLLER
1 TO 8 RMO3 DISK DRIVES PER CONTROLLER

IN ADDITION, A SINGLE RMO3 DISK CARTRIDGE IS REQUIRED WHICH MUST BE FORMATTED IN 32 SECTOR FORMAT, ON A RELIABLE WELL-ALIGNED(REFERENCE PACK) RMO3 DRIVE.
THIS CARTRIDGE WILL BE MOVED FROM DRIVE TO DRIVE, (ON UP TO 16 DRIVES) ON EACH OF THE TWO PASSES.

3.0 PRELIMINARY PROGRAM REQUIREMENTS

BEFORE RUNNING THE RMO3 DRIVE COMPATIBILITY PROGRAM, THE SUBSYSTEM(S) UNDER TEST SHOULD BE CAPABLE OF PASSING THE CONTROLLER DIAGNOSTICS AND THE DRIVE DIAGNOSTICS. IN ADDITION, THE CARTRIDGE MUST BE FORMATTED IN 32 SECTOR FORMAT USING THE PACK FORMATTER.

4.0 GENERAL PROGRAM CONSIDERATIONS

4.1 SYSMAC

THIS PROGRAM USES PORTIONS OF THE SYSMAC DIAGNOSTIC SYSTEM MACRO PACKAGE.

4.2 XXDP

THIS PROGRAM MAY BE LOADED UNDER XXDP, AND MAY BE RUN IN DUMP MODE ONLY. DUE TO MANUAL INTERVENTION AND LACK OF END-OF-PASS HOOKS, THE PROGRAM IS NOT XXDP CHAINABLE.

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

4.3 ACT

THIS PROGRAM MAY BE LOADED UNDER ACT AND MAY BE RUN IN DUMP MODE ONLY. IT IS NOT CHAINABLE UNDER ACT.

4.4 APT

THIS PROGRAM MAY BE LOADED BY THE APT SYSTEM, BUT MAY BE RUN IN PROGRAM (DUMP) MODE ONLY. IT CANNOT BE RUN IN APT SCRIPT MODE.

4.5 DUAL-ACCESS

THIS PROGRAM DOES NOT UTILIZE THE DUAL-ACCESS OPTION IN ANY WAY, AND ALL DRIVES UNDER TEST SHOULD BE DE-SELECTED THROUGH THE PORT WHICH IS NOT IN USE, OR LOCKED ON THE PORT BEING TESTED.

4.6 MEMORY MANAGEMENT

MEMORY MANAGEMENT IS NOT UTILIZED IN THIS PROGRAM. IF IT IS INSTALLED, IT IS DISABLED BY THE PROGRAM.

4.7 MEMORY PARITY OPTION

IF PARITY MEMORY IS INSTALLED, MEMORY PARITY TRAPS ARE DISABLED BY THE PROGRAM.

4.8 BAD SECTORS

THE LIST OF BAD SECTORS ON THE CARTRIDGE IS OBTAINED FROM THE FIRST DRIVE TO BE TESTED ON THE CURRENT SUBSYSTEM. ACCORDING TO A SWITCH REGISTER OPTION (SEE SECTION 6.2) THIS LIST MAY BE TYPED AT THE CONSOLE AT THE START OF THE FIRST PASS. ALL DATA ERRORS OCCURING IN THE PROGRAM ARE IGNORED IF THEY OCCUR IN SECTORS DESIGNATED AS BAD.

4.9 EXECUTION TIME

THE TOTAL TIME REQUIRED TO RUN THE DRIVE COMPATIBILITY PROGRAM IS DIRECTLY PROPORTIONAL TO THE NUMBER OF DRIVES TO BE TESTED AND REQUIRES ABOUT 2 MINUTES PER DRIVE, EXCLUDING OPERATOR INTERVENTION.

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

5.0 PROGRAM LOAD MEDIA

THIS PROGRAM CAN BE LOADED FROM PAPER TAPE USING THE ABSOLUTE LOADER OR FROM THE ACT OR APT SYSTEMS OR FROM ANY MEDIA SUPPORTED BY XXDP.

6.0 PROGRAM OPTIONS

6.1 STARTING ADDRESSES

200 - THIS IS THE STARTING ADDRESS FOR DEFAULT PARAMETERS AND RUNNING OF PASS 1 AND PASS 2 ON A SINGLE SUBSYSTEM. THE PROGRAM WILL USE DEFAULT RH11(70) BASE ADDRESS, INTERRUPT VECTOR AND PRIORITY.
 THE PROGRAM WILL ASSUME ALL DRIVES TO BE TESTED RESIDE ON ONE RH11(70)/RMO3 SUBSYSTEM ONLY.

204 - THIS IS THE STARTING ADDRESS TO RUN PASS 1 ON ALL RH11(70)/RMO3 SUBSYSTEMS WHICH RESIDE ON THIS PDP-11 SYSTEM. THE PROGRAM WILL ASK FOR THE RH11(70) BASE ADDRESS, INTERRUPT VECTOR, AND PRIORITY FOR EACH SUBSYSTEM ON THIS SYSTEM, AND IT ASKS FOR THE LETTER NAMES (A THRU H) ASSIGNED TO ALL OTHER SUBSYSTEMS, AND THE DRIVE(S) WHICH WILL BE TESTED ON EACH.

220 - THIS IS THE STARTING ADDRESS TO RUN PASS 2 ON ALL RH11(70)/RMO3 SUBSYSTEMS WHICH RESIDE ON THIS PDP-11 SYSTEM. THE PROGRAM WILL ASK FOR THE RH11(70) BASE ADDRESS, INTERRUPT VECTOR FOR EACH SUBSYSTEM ON THIS SYSTEM, AND IT ASKS FOR THE LETTER NAMES (A THRU H) ASSIGNED TO ALL OTHER SUBSYSTEMS, AND THE DRIVE(S) WHICH WILL BE TESTED ON EACH.

6.2 SWITCH REGISTER OPTIONS USED

THIS PROGRAM IS DESIGNED TO ALLOW THE USE OF THE HARDWARE SWITCH REGISTER IF PRESENT, OR THE SYSMAC-SUPPORTED SOFTWARE SWITCH REGISTER (IF HARDWARE SWR IS NOT PRESENT, OR IS SET TO ALL ONES). IN EITHER CASE, THE FOLLOWING OPTIONS ARE IMPLEMENTED WHEN THE APPROPRIATE BITS ARE SET TO 1:

BIT	OPTION
---	-----
15	HALT ON ERROR
14	LOOP ON CURRENT TEST

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

13	INHIBIT ERROR REPORTS
12	REPORT DESCRIPTION ONLY, ON ERRORS
11	UNUSED
10	BELL ON ERROR
09	LOOP ON ERROR
08	APPLY RANDOM STALL BETWEEN OPERATIONS
07	TYPE BAD SECTOR FILES (BSF'S) AT START
06-00	UNUSED

7.0 RUNNING THE PROGRAM

ONCE THE PROGRAM HAS BEEN LOADED INTO CORE (IN A GIVEN SYSTEM, IF THERE ARE MULTIPLE SYSTEMS) THE FOLLOWING STEPS MUST BE TAKEN TO RUN THE PROGRAM:

1. INSURE THAT ALL DRIVES TO BE TESTED ARE POWERED UP AND SINGLE PORT SELECTED.
2. LOAD THE DESIRED START ADDRESS.
3. SET ANY DESIRED BITS IN THE HARDWARE SWITCH REGISTER (IF PRESENT).
4. START THE PROGRAM.
5. FOLLOW ALL INSTRUCTIONS TYPED BY THE PROGRAM PERTAINING TO THE MANUAL INTERVENTION REQUIRED, AND THE ALTERNATE USE OF MULTIPLE SYSTEMS (IF THERE ARE ANY).

8.0 OPERATIONAL DIALOGUE

THIS SECTION DESCRIBES THE CONSOLE TERMINAL DIALOGUE THROUGH WHICH THE PROGRAM DIRECTS THE OPERATOR, IN THE SELECTION OF OPTIONS AND THE LOADING AND UNLOADING OF DRIVES, AND THE MOVEMENT OF THE TEST CARTRIDGE. THE EXACT DIALOGUE WHICH IS USED DEPENDS UPON THE STARTING ADDRESS WHICH WAS CHOSEN (SEE SECTION 6.1).

IN THE FOLLOWING DISCUSSION AND IN THE PRINTOUT OF TEST RESULTS, DRIVES TO BE TESTED WILL BE REFERRED TO BY A LETTER AND A NUMBER. THE LETTER IS THE SUBSYSTEM LETTER NAME (OPERATOR ASSIGNED), AND THE NUMBER IS THE DRIVE NUMBER ON THAT SUBSYSTEM. FOR EXAMPLE, DRIVE C6 REFERS TO DRIVE 6 ON SUBSYSTEM C.

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

8.1 DIALOGUE FOR ADDRESS 200 START

THIS STARTING ADDRESS MAY BE USED FOR DEFAULTING PARAMETERS ON ONE SUB-SYSTEM.

THE PROGRAM FIRST IDENTIFIES ITSELF AS FOLLOWS:

(MAINDEC DZRMI-A) - RMO3 DRIVE COMPATIBILITY TEST

THEN, THE PROGRAM ASKS THE DRIVES TO BE TESTED.

THE PROGRAM TYPES THE DRIVE LIST, AS IN THE FOLLOWING EXAMPLE:

DRIVES = 2,5,7<CR>

THE PROGRAM NOW PROCEEDS WITH PASS 1, AND DIRECTS THE OPERATOR IN THE MOUNTING OF THE PACK, AS DESCRIBED IN SECTION 8.4.

PLEASE NOTE THAT THERE IS ONLY ONE SUBSYSTEM ON AN ADR. 200 START, AND IT IS NAMED SUBSYSTEM A. THE DRIVES IN THE ABOVE EXAMPLE WOULD BE REFERRED TO AS A2,A5,A7 IN THE TEST RESULTS PRINTOUT AT THE END OF PASS 2.

8.2 DIALOGUE FOR ADDRESS 204 START

THIS STARTING ADDRESS MUST BE USED ON EACH SYSTEM, WHEN THERE IS MORE THAN ONE SUBSYSTEM, BUT IT MAY ALSO BE USED WHEN THERE IS JUST ONE SUBSYSTEM (TOTAL), TO SPECIFY DRIVES TO TEST AND NON-DEFAULT PARAMETER VALUES, FOR PASS 1.

THE PROGRAM IDENTIFIES ITSELF AS FOLLOWS:

(MAINDEC DZRMI-A)- RMO3 DRIVE COMPATIBILITY TEST

THEN, THE PROGRAM ASKS THE OPERATOR FOR THE DRIVES TO BE TESTED ON EACH OF THE POSSIBLE SUBSYSTEMS (STARTING WITH A - THE NAMES RANGE FROM SUBSYS A TO SUBSYS H. THERE COULD BE UP TO 8 SUBSYSTEMS, WITH A DRIVE ON EACH) :

SUBSYS A DRIVE(S) =

THE OPERATOR THEN TYPES THE DESIRED DRIVE NUMBERS, AS IN THE FOLLOWING EXAMPLE:

SUBSYS A DRIVE(S) = 2,5,7<CR>

THE PROGRAM THEN VERIFIES THE DRIVE NUMBERS BY TYPING :

WILL TEST DRIVE(S) 2,5,7 ON SUBSYS A.

NEXT, THE PROGRAM ASKS THE FOLLOWING QUESTION:

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

IS THERE ANOTHER SUBSYS (Y OR N <CR>)?

THE OPERATOR TYPES Y<CR> OR N<CR>. (IF JUST <CR> IS TYPED, THE PROGRAM ASSUMES THAT N<CR> WAS TYPED). IF THE OPERATOR TYPED N, THE PROGRAM PROCEEDS WITH PASS 1, AND DIRECTS THE OPERATOR IN THE MOUNTING OF THE PACK, AS DESCRIBED IN SECTION 8.4. IF Y WAS TYPED, THE PROGRAM ASKS FOR THE NUMBERS OF THE DRIVES TO BE TESTED ON THE NEXT SUBSYSTEM (SUBSYS B) AS FOLLOWS:

SUBSYS B DRIVE(S) =

THE OPERATOR TYPES THE DRIVE NUMBERS, AS IN THE FOLLOWING EXAMPLE:

SUBSYS B DRIVE(S) = 2,3<CR>

THE PROGRAM THEN VERIFIES THE DRIVE NUMBERS. BY TYPING :

WILL TEST DRIVE(S) 2,3 ON SUBSYS B.

NEXT, THE PROGRAM WILL ASK :

IS THERE ANOTHER SUBSYS (Y OR N <CR>)?

AND IN THE SAME MANNER, THE OPERATOR SPECIFIES THE DRIVES ON EACH OF THE REMAINING SUBSYSTEMS, UNTIL ALL HAVE BEEN SPECIFIED.

ALL SUBSYSTEMS MUST BE TESTED IN THE ORDER IN WHICH THE LETTERS ARE ASSIGNED (A THRU H). NEXT, THE PROGRAM ALLOWS THE OPERATOR TO ALTER THE RH11(70) BUS ADDRESS, VECTOR ADDRESS FOR THIS SUBSYSTEM. FOR EACH PARAMETER THE CURRENT VALUE IS TYPED, AND THE OPERATOR IS GIVEN THE OPPORTUNITY TO TYPE IN A NEW VALUE, PLUS <CR>. IF JUST <CR> IS TYPED, THE PARAMETER IS NOT CHANGED. WHEN THE PROGRAM IS FIRST LOADED, THE FOLLOWING DEFAULT VALUES ARE ASSIGNED: RH11(70) BUS ADDRESS = 177670, VECTOR ADDRESS = 240, (IF 200 START). THE FOLLOWING EXAMPLE SHOWS A PRINTOUT IN WHICH BUS ADDRESS AND VECTOR WERE CHANGED:

RMCS1 = 000000 177670
RMVEC = 000 254

THEN THE PROGRAM PROCEEDS WITH PASS 1, AND DIRECTS THE OPERATOR IN THE MOUNTING OF THE PACK, AS DESCRIBED IN SECTION 8.4. AT THE COMPLETION OF PASS 1 ON THE SUBSYSTEM, THE PROGRAM WILL INFORM THE OPERATOR HOW

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

TO PERFORM PASS 1 ON THE NEXT SUBSYSTEM.

8.3 DIALOGUE FOR ADDRESS 220 START

THIS STARTING ADDRESS MUST BE USED ON EACH SYSTEM, WHEN THERE IS MORE THAN 1 SUBSYSTEM, BUT IT MAY ALSO BE USED WHEN THERE IS JUST ONE SUBSYSTEM (TOTAL) TO SPECIFY DRIVES TO TEST AND NON-DEFAULT PARAMETER VALUES, FOR PASS 2. THE PROGRAM IDENTIFIES ITSELF, AS FOLLOWS :

(MAINDEC DZAMI-A) - RMO3 DRIVE COMPATIBILITY TEST

THE DIALOGUE FOR 220 START IS IDENTICAL TO THE DIALOGUE FOR THE 204 START DESCRIBED ABOVE (SECTION 8.2), FOR THE SELECTION OF SUBSYSTEM PARAMETERS AND THE SPECIFICATION OF ALL DRIVES TO BE TESTED ON THE VARIOUS SUBSYSTEMS. HOWEVER, AFTER THIS DIALOGUE IS COMPLETED, THE PROGRAM PROCEEDS WITH PASS 2, AND DIRECTS THE OPERATOR IN THE MOVEMENT OF THE PACK, AS DESCRIBED IN SECTION 8.5.

NOTE THAT SINCE THE APPROPRIATE PROCESSOR MUST BE STARTED AT THE STARTING ADDRESS FOR EACH SUBSYSTEM TO BE TESTED, THE COMPATIBILITY TEST MAY BE PERFORMED IN STEPS, AT VARIOUS TIMES AND BETWEEN VARIOUS DISTANT LOCATIONS, BY MOVING THE TEST PACK AND SAVING THE PRINTOUT FROM EACH PASS ON EACH PDP-11 SYSTEM INVOLVED.

8.4 PASS 1 DIALOGUE

AFTER THE SELECTION OF PARAMETERS AND DRIVES HAS BEEN COMPLETED ON THE CURRENT SUBSYSTEM (SECTIONS 8.1-8.2), THE PROGRAM INDICATES THE START OF PASS 1 AS FOLLOWS:

** STARTING PASS 1 **

NEXT, THE PROGRAM SELECTS THE FIRST DRIVE TO BE TESTED ON THIS SUBSYSTEM, AND INSTRUCTS THE OPERATOR TO MOUNT THE TEST CARTRIDGE AND LOAD THE HEADS ON THAT DRIVE, AS IN THE FOLLOWING EXAMPLE:

MOUNT PACK ON DRIVE A2 AND LOAD.
TYPE R<CR> WHEN DRIVE READY:

THE OPERATOR PERFORMS THIS TASK AND TYPES R<CR> WHEN THE DRIVE READY LIGHT IS ON. THE PROGRAM PERFORMS PASS 1 FUNCTIONS ON THIS DRIVE (SEE SECTION 9.1) AND THEN INSTRUCTS THE OPERATOR TO UNLOAD THE DRIVE AND REMOVE THE PACK AS FOLLOWS:

UNLOAD DRIVE A2 AND REMOVE PACK.
TYPE R<CR> WHEN DONE:

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

THE OPERATOR PERFORMS THESE FUNCTIONS AND TYPES R<CR> AFTER THE PACK HAS BEEN REMOVED.

IN THE SAME MANNER, THE PROGRAM INSTRUCTS THE OPERATOR IN THE MOVEMENT OF THE PACK THROUGHOUT THE REST OF THE DRIVES ON THE CURRENT SUBSYSTEM. WHEN THIS HAS BEEN COMPLETED, THE PROGRAM DOES ONE OF THREE THINGS: (1) IF THERE IS ONLY ONE SUBSYSTEM (FROM ADR 200 START) THE PROGRAM BEGINS PASS 2 (SEE SECTION 8.5). (2) IF THERE IS ANOTHER SUBSYSTEM, THE PROGRAM DIRECTS THE OPERATOR TO PERFORM PASS 1 ON THE NEXT SUBSYS AS FOLLOWS :

STARTING PASS 1 ON SUBSYS B

(3) IF THERE ARE NO MORE DRIVES TO TEST IN PASS 1 ON ANY SUBSYS, THE PROGRAM DIRECTS THE OPERATOR TO BEGIN PASS 2 ON THE FIRST SUBSYS (SEE SECT. 8.5) AS FOLLOWS :

STARTING PASS 2 ON SUBSYS A

8.5 PASS 2 DIALOGUE

THE OPERATOR RETURNS TO THE FIRST SUBSYSTEM TO PERFORM PASS 2 EITHER THROUGH THE DIALOGUE OF THE ADR 200 START, OR AFTER THE SELECTION OF PARAMETERS AND DRIVES HAS BEEN COMPLETED IN ACCORDANCE WITH THE DIALOGUE OF THE ADR 220 START (SEE SECTION 8.3). IN EITHER CASE, THE PROGRAM INDICATES THE START OF PASS 2 BY TYPING :

** STARTING PASS 2 **

THE PROGRAM THEN DIRECTS THE OPERATOR IN THE UNLOADING, PACK MOVEMENT, AND LOADING OF ALL DRIVES ON THE FIRST SUBSYSTEM, IN THE SAME MANNER AS DESCRIBED FOR PASS 1 (SEE SECTION 8.4).

HOWEVER, AFTER PASS 2 TESTING (SEE SECTION 9.2) IS COMPLETED ON A GIVEN DRIVE, THE ENTIRE TEST RESULTS FOR THAT DRIVE ARE TYPED. THE DETAILS OF THIS PRINTOUT ARE DESCRIBED IN SECTION 10, AFTER THE DETAILS OF THE TESTING ARE DESCRIBED.

WHEN PASS 2 HAS BEEN COMPLETED FOR ALL DRIVES ON THE FIRST SUBSYSTEM, THE PROGRAM DOES ONE OF TWO THINGS: (1) IF THERE IS ONLY ONE SUBSYSTEM (FROM ADR 200 START) OR IF ALL DRIVES ON ALL SUBSYSTEMS HAVE BEEN TESTED IN PASS 2 (FROM ADR 220 START), THE ENTIRE TESTING AND REPORTING HAVE BEEN COMPLETED, AND THE PROGRAM TYPES:

** END OF TESTING **

C02

MAINDEC-11-DZRM1 AO/00, RMO3 DRIVE COMPATIBILITY TEST MACY11 30(1046) 21-JUL-77 16:51 PAGE 15
DZRMIA.P11 21-JUL-77 15:44

SEQ 0014

599
600
601
602
603
604
605
606

(2) IF THERE IS ANOTHER SUBSYSTEM, HOWEVER, THE PROGRAM DIRECTS THE
OPERATOR TO PERFORM PASS 2 ON THE NEXT SUBSYSTEM AS FOLLOWS
:

STARTING PASS 2 ON SUBSYS B

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

9.0 DESCRIPTION OF TESTS

THE MAIN FUNCTIONAL BLOCKS OF CODE IN THE PROGRAM ARE ASSIGNED TEST NUMBERS, FOR THE PURPOSE OF IDENTIFICATION IN ERROR PRINTOUTS. TEST 0 REFERS TO THE OPERATOR INPUT DIALOGUE ROUTINES DESCRIBED IN SECTIONS 8.1-8.3. THE OTHER TEST NUMBERS ARE ASSIGNED BELOW, IN THE DESCRIPTION OF PASS 1 AND PASS 2 TESTING.

IN THE FOLLOWING SECTIONS, TABLES A-G ARE REFERRED TO. IN THESE TABLES, DRIVES ARE NAMED FROM 0-7 FOR ILLUSTRATIVE PURPOSES, ALTHOUGH THE DRIVES ARE NAMED THE FOLLOWING WAY IN AN ACTUAL SITUATION : AO,A1, A2,...BO,B1,B2,...CO,C1,C2,... ETC. (SEE SECTION 8.0).

9.1 DESCRIPTION OF PASS 1 TESTS

IN PASS 1, THE BASIC READ/WRITE CAPABILITY OF EACH DRIVE IS DEMONSTRATED, AND COMPATIBILITY DATA PATTERNS ARE WRITTEN BY ALL DRIVES UPON THE SAME TEST CARTRIDGE.

THE SEQUENCE OF OPERATIONS PERFORMED ON EACH DRIVE IS AS FOLLOWS:

1. TEST 1 - MOUNTING OF TEST CARTRIDGE FOR PASS 1 - THE OPERATOR MOUNTS THE PACK ON THIS DRIVE AND MANUALLY LOADS THE HEADS, AS DIRECTED BY THE PROGRAM (SEE SECTION 8.4).
2. TEST 2 - BASIC READ/WRITE DATA TEST - THE PROGRAM PERFORMS A WRITE AND WRITE CHECK OPERATION USING A "WORST CASE" DATA PATTERN, AT THE APPROPRIATE SECTOR FOR THIS DRIVE (SEE TABLE A) ON ALL SURFACES. THE PURPOSE OF THIS OPERATION IS TO VERIFY THE BASIC READ/WRITE CAPABILITY OF THE DRIVE ON PASS 1. THE ENTIRE SECTOR IS WRITTEN WITH THE REPETITION OF THE DATA PATTERN SHOWN IN TABLE B.
3. TEST 3 - THE PROGRAM WRITES ALL SECTORS FOR THIS DRIVE WITHIN THE CYLINDER BLOCKS SHOWN IN TABLE C ON ALL SURFACES USING A SINGLE REPEATED WORD OF THE PATTERN IN TABLE G. DRIVE 0 USES WORD 0, DRIVE 1 USES WORD 1, DRIVE 7 USES WORD 7, ETC. THUS, THE DATA FROM EACH DRIVE IS UNIQUE. TABLE C HAS BEEN DETERMINED AS FOLLOWS:

IN EACH OF THE SEVEN WRITE CURRENT ZONES ON EACH SURFACE, SECTORS ARE WRITTEN WITHIN TWO DISTINCT CYLINDER BLOCKS. THE FIRST 16 CYLINDERS OF EACH WRITE CURRENT ZONE IS THE FIRST BLOCK USED FOR WRITE TEST IN PASS 2. THE LAST 16 CYLINDERS OF EACH CURRENT

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

ZONE (EXCEPT THE INNERMOST ZONE) IS THE SECOND BLOCK USED FOR READ TEST IN PASS 2. WITHIN EACH CURRENT ZONE, THESE TWO BLOCKS ARE IDENTICALLY WRITTEN. HOWEVER, THE SECTORS DESIGNATED TO EACH DRIVE ARE ROTATED FROM ZONE TO ZONE SO THAT THE DATA APPEARS AT VARIOUS ANGULAR POSITIONS ON THE PACK.

WITHIN EACH CYLINDER BLOCK, UP TO 32 SECTORS ARE WRITTEN (DEPENDING ON THE NUMBER OF DRIVES BEING TESTED) ON EACH CYLINDER.

THE BASIC LAYOUT OF A TYPICAL CYLINDER BLOCK IS SHOWN IN TABLE D, WHERE THE BLOCK SHOWN IS THE READ TEST BLOCK FOR ZONE 1, WHICH STARTS ON CYLINDER 112, AND HAS THE ROTATING STARTING SECTOR = SECTOR 0. EACH NUMBER INSIDE THE BLOCK IS THE NUMBER OF THE DRIVE WHICH WRITES THAT SECTOR. TABLE D SHOWS THE BLOCKS WRITTEN BY EACH OF 16 DRIVES. IF ANY OF THE DRIVES SHOWN ARE NOT PRESENT, HOWEVER, THE BLOCKS RESERVED FOR THE MISSING DRIVES ARE SIMPLY NOT WRITTEN.

THE ABOVE PATTERN OF SECTOR WRITES INSURES THAT DATA FROM EACH DRIVE IS WRITTEN ON ADJACENT CYLINDERS TO DATA FROM EVERY OTHER DRIVE, IN BOTH DIRECTIONS. IN ADDITION, THE ROTATION OF THE ABOVE SECTORS FROM CURRENT ZONE TO CURRENT ZONE INSURES THAT WRITE TEST AND READ TEST ARE DONE AT SEVERAL DIFFERENT ANGULAR POSITIONS WITH RESPECT TO THE CARTRIDGE.

4. TEST 4 - DISMOUNTING OF TEST CARTRIDGE IN PASS 1 - THE OPERATOR UNLOADS THE DRIVE AND DISMOUNTS THE PACK, AS DIRECTED BY THE PROGRAM (SEE SECTION 8.4), TO PROCEED WITH THE ABOVE STEPS ON THE NEXT DRIVE.

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

9.2 DESCRIPTION OF PASS 2 TESTS

IN PASS 2, THE ABILITY OF EACH DRIVE TO COMPLETELY OVERWRITE DATA WRITTEN BY ALL OTHER DRIVES AND TO READ DATA WRITTEN BY ALL OTHER DRIVES, IS TESTED.

THE SEQUENCE OF OPERATIONS PERFORMED BY EACH DRIVE IS AS FOLLOWS:

1. TEST 5 - MOUNTING OF TEST CARTRIDGE FOR PASS 2 - THE OPERATOR MOUNTS THE PACK ON THIS DRIVE AND MANUALLY LOADS THE HEADS, AS DIRECTED BY THE PROGRAM (SEE SECTION 8.5).
2. TEST 6 - WRITE TEST - NEXT, THE PROGRAM PROCEEDS TO TEST THIS DRIVE'S OVERWRITE CAPABILITY. FIRST, THE APPROPRIATE CYLINDERS IN TABLE E FOR THIS DRIVE ARE OVERWRITTEN, ON EACH SURFACE. THE DATA USED IS A REPETITION OF A SINGLE WORD OF THE PATTERN IN TABLE G. DRIVE 0 USES WORD 0, DRIVE 1 USES WORD 1, DRIVE 7 USES WORD 7, ETC.

THEN, EACH CYLINDER OVERWRITTEN IS READ BACK BY THIS DRIVE IN EACH OFFSET DIRECTION (+ AND -). THE PROGRAM SCANS FOR READ ERRORS (DCK, HCRC, ETC.) DURING THIS READ, AND IF ONE OCCURS, THE PROGRAM DETERMINES WHICH DRIVE'S DATA HAS NOT BEEN CORRECTLY OVERWRITTEN, AND A SCORE FOR THAT DRIVE IS DECREMENTED. THEN, THE TRANSFER IS CONTINUED AT THE NEXT SECTOR, WITH THAT OFFSET VALUE. THE READS ARE DONE WITH ALL OF THE ABOVE OFFSETS APPLIED, AND A SEPARATE SCORE FOR EACH DRIVE IS KEPT WHILE THE CURRENT DRIVE IS PERFORMING THE OVERWRITES. FOR EACH TRACK, SCORES ARE AVERAGED OVER ALL CYLS TESTED, IN EACH OFFSET DIRECTION. AT THE COMPLETION OF THE OVERWRITE TEST ON THIS DRIVE, THE SCORES OF ALL THE DRIVES ARE CONVERTED AND STORED, FOR PRINTING AT THE END OF PASS 2 (AS DESCRIBED IN SECTION 10.2). EACH SCORE PROPORTIONAL TO THE OFFSET IN A GIVEN DIRECTION BY THE CURRENT DRIVE WHILE SUCCESSFULLY READING THE DATA IT WROTE OVER ONE OF THE OTHER DRIVE'S DATA. THUS, THE PRINTOUT REVEALS WHICH DRIVES ARE INVOLVED, IN A SITUATION IN WHICH A DRIVE CANNOT OVERWRITE ONE OR SEVERAL OTHER DRIVE'S DATA.

3. TEST 7 - DRIVE SELF-TEST - THE PROGRAM NEXT EVALUATES THE DRIVE'S ABILITY TO WRITE AND READ ITS OWN DATA, AT VARIOUS POSITIONS ON THE PACK. FIRST, ALL SECTORS OF THE APPROPRIATE CYLINDERS SHOWN IN TABLE F FOR THIS DRIVE ARE WRITTEN WITH THE DATA PATTERN SHOWN IN TABLE B, FOR ALL SURFACES. THEN, THE SECTORS ARE READ WITH OFFSET IN EACH DIRECTION. THE PROGRAM SCANS FOR READ ERRORS DURING EACH READ, AND IT COMPUTES A SCORE WHICH IS PROPORTIONAL TO THE FAILING OFFSET.

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

THEN, THE SCORES FOR ALL SECTORS READ IN THIS CYLINDER BLOCK ARE AVERAGED, TO COME UP WITH A DRIVE SELF-TEST SCORE FOR EACH SURFACE FOR EACH OFFSET DIRECTION. THIS SCORE IS SAVED FOR LATER USE, TO BECOME THE STANDARD FOR THE READS WHICH ARE TO FOLLOW.

4. TEST 10(OCTAL) - COMPATIBILITY DATA READ TEST - HAVING ESTABLISHED A SELF-TEST SCORE FOR THIS DRIVE, THE PROGRAM PROCEEDS TO PERFORM THE COMPATIBILITY DATA READS OF THE PATTERNS WRITTEN BY ALL THE DRIVES IN EACH CYLINDER BLOCK (ON EACH SURFACE). EACH COMPATIBILITY CYLINDER BLOCK SHOWN IN TABLE C IS READ, A CYLINDER AT A TIME IN EACH OFFSET DIRECTION. THE PROGRAM SCANS FOR READ ERRORS DURING EACH READ AND IF ONE OCCURS, THE PROGRAM DETERMINES WHICH DRIVE'S DATA WAS BEING READ AT THAT INSTANT AND A SCORE FOR THAT DRIVE IS DECREMENTED. THEN, THE TRANSFER IS CONTINUED AT THE NEXT SECTOR, WITH THAT OFFSET VALUE. THE READS ARE DONE WITH OFFSETS IN EACH DIRECTION. AND A SEPARATE SCORE FOR EACH DRIVE IS KEPT, WHILE THE CURRENT DRIVE IS READING THE COMPATIBILITY DATA. THEN, EACH SCORE IS APPROPRIATELY ADJUSTED TO REFLECT THE SELF-TEST SCORE FOR THE CURRENT DRIVE AT THAT PARTICULAR CYLINDER BLOCK. THE SCORES ARE THEN AVERAGED OVER ALL CYLINDER BLOCKS. EACH SCORE IS PROPORTIONAL TO THE CAPABILITY OF THE CURRENT DRIVE TO SUCCESSFULLY READ THE DATA WRITTEN BY ONE OF THE OTHER DRIVES, AND SCORES ARE COMPUTED SEPARATELY FOR EACH SURFACE (TRACK), FOR EACH OFFSET DIRECTION. THUS, THE PRINTOUT REVEALS WHICH DRIVES ARE INVOLVED IN A SITUATION IN WHICH A PARTICULAR DRIVE HAS DIFFICULTY IN READING THE DATA OF ONE OR SEVERAL OTHER DRIVES.
5. TEST 11(OCTAL) - TYPE TEST SCORES AND DISMOUNT PACK IN PASS 2 - THE OVERWRITE AND COMPATIBILITY DATA READ TEST SCORES FOR THIS DRIVE ARE CONVERTED AND TYPED. THEN, THE OPERATOR UNLOADS THE DRIVE AND DISMOUNTS THE PACK AS DIRECTED BY THE PROGRAM (SEE SECTION B.5), TO PROCEED WITH THE ABOVE STEPS ON THE NEXT DRIVE.

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

10.0 PRINTOUT OF TEST RESULTS

THE TEST RESULTS ARE PRINTED AT THE END OF PASS 2 ON EACH DRIVE BEING TESTED. THESE RESULTS PERTAIN TO THE OVERWRITE TEST AND THE COMPATIBILITY DATA READ TEST.

10.1 TEST RESULTS

THE RESULTS OF BOTH THE OVERWRITE AND OF THE COMPATIBILITY DATA READ ARE PRINTED, REGARD OF DEGREE OF SUCCESS. IF THE TEST IS SUCCESSFUL, THE MESSAGES :

** ALL DRIVES ARE COMPATIBLE **

IS PRINTED. IF THE TEST IS FAILURE, THE TEST RESULTS ARE TABULAR IN FORM AS SHOWN.

IN THE FOLLOWING EXAMPLE, THERE ARE 2 SYSTEMS, AND THE DRIVES BEING TESTED ARE A0,A1,A2,B0, AND B5. THE TEST RESULTS FOR DRIVE A1 ARE SHOWN BELOW:

SCORES FOR DRIVE A1 :

TRACK NO.	DRIVE READ	OVRWRT OFST-	OVRWRT OFST+	READ OFST-	READ OFST+
0	A2	* 0	* 0		

THE ABOVE EXAMPLE REVEALS A POSSIBLE COMPATIBILITY PROBLEM EXISTS BETWEEN DRIVES A1 AND A2. NOTICE THAT ON TRACK 0, THAT THE OVERWRITE SCORES WERE UNACCEPTABLY LOW (0), AND THE PROGRAM NOTED THESE BAD SCORES WITH AN ASTERISK (*). ALL ACCEPTABLE TEST RESULTS ARE NOT PRINTED.

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

11.0 ERROR REPORTING

11.1 COMMON ERRORS

THE FOLLOWING IS A LIST OF COMMON ERROR MESSAGES WHICH ACCOMPANY ERROR TYPEOUTS FROM THE RMD3 DRIVE COMPATIBILITY PROGRAM. THE ERRORS ARE SELF-EXPLANATORY.

ADDRESS PLUG CHANGE BIT SET
RH11(70) DIDN'T RESPOND TO ADDRESSING
UNCORRECTABLE MASSBUS PARITY ERROR
FATAL MASSBUS PARITY ERROR
PERSISTENT DEVICE UNSAFE
OPERATION NOT COMPLETED WITHIN TIME LIMIT
DRIVE WENT OFFLINE
NO RESPONSE TO PORT REQUEST
HEADER CRC ERROR
DATA CHECK 'DCK' ERROR
WRITE CHECK ERROR - DATA CHECK 'DCK' SET
WRITE CHCKE ERROR - DATA CHECK 'DCK' NOT SET
HEADER READ ERROR - 'FMT' BIT DROPPED
HEADER READ ERROR - HEADER COMPARE 'HCE' ERROR
FORMAT ERROR 'FER'
HEADER COMPARE 'HCE' ERROR
MISCELLANEOUS DRIVE ERROR
OPERATION INCOMPLETE 'OPI' ERROR
DRIVE TIMING 'DTE' ERROR
PARITY 'PAR' ERROR AFTER OPERATION STARTED

J02

MAINDEC-11-DZRMI AO/OO, RMO3 DRIVE COMPATIBILITY TEST MACY11 30(1046) 21-JUL-77 16:51 PAGE 22
DZRMI.A.P11 21-JUL-77 15:44

SEQ 0021

884	WRITE CLOCK FAILURE 'WCF' ERROR
885	
886	INVALID ADDRESS 'IAE' ERROR
887	
888	WRITE LOCK 'WLE' ERROR
889	
890	DATA CHECK 'DCK' SET DURING WRITE CHECK COMMAND
891	
892	RH11(70) OR UNIBUS TRANSFER ERROR
893	
894	BUS ADDRESS OR WORD COUNT INCORRECT
895	
896	DATA COMPARE ERRORS - NO OTHER ERROR(S) DETECTED
897	
898	CAN'T MATCH DATA READ WITH A PATTERN
899	
900	ERROR BIT(S) SET, BUT NO ERROR SIGNALLED BY THE RH11(70)
901	
902	ECC LOGIC FAILURE - POSITION REGISTER VALUE NOT VALID
903	
904	BUS ADDRESS AND WORD COUNT NOT CONSISTENT
905	
906	SEEK INCOMPLETE 'SKI' ERROR
907	
908	PROGRAM DETECTED POSITIONING ERROR
909	
910	DRIVE UNSAFE ERROR
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

11.2 ERROR HANDLING

ERRORS REPORTED BY THE PROGRAM CONSIST OF COMMON FAILURES RESULTING FROM ATTEMPTED SUBSYSTEM FUNCTIONS, AS WELL AS CERTAIN ERRORS UNIQUE TO PARTICULAR TESTS. EACH ERROR PRINTOUT CONSISTS OF AN ERROR DESCRIPTION AND TEST NUMBER, POSSIBLY FOLLOWED BY HEADER LINES, COLUMN HEADINGS, AND COLUMNS OF REGISTER CONTENTS IN OCTAL. AS MUCH MEANINGFUL REGISTER DATA AS POSSIBLE (FOR EXAMPLE, RH11(70) REGISTERS) ARE REPORTED IN A GIVEN ERROR. OTHER ERROR REPORTS MAY CONSIST OF A SINGLE DESCRIPTIVE LINE.

11.3 ERROR PRINTOUT EXAMPLE

```

RH11(70) OR UNIBUS TRANSFER ERROR
DRIVE  RMCS1  RMC  RMBA  RMDA
000001 144250 174400 0055030 000431

RMCS2  RMD5  RMER1  RMAS  RMD8
000100 010700 000000 000000 000000

RMMR1  RMDT  RMOF  RMDC  RMMR2
000050 024024 010000 000716 011777

RMER2  RMEC1  RMEC2
000000 004066 000000

```

L02

TABLE A

BASIC READ/WRITE TEST SECTORS

ADDRESS OF SECTOR ON EACH SURFACE

<u>DRIVE NO.</u>	<u>CYLINDER</u>	<u>SECTORS</u>
0	620	0
1	620	1
2	620	2
3	620	3
4	620	4
5	620	5
6	620	6
7	620	7
8	620	8
9	620	9
10	620	10
11	620	11
12	620	12
13	620	13
14	620	14
15	620	15

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

TABLE B

WORST CASE DATA PATTERN (REPEATS EVERY 16 WORDS)

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

<u>WORD NO.</u>	<u>DATA (OCTAL)</u>
0	066666
1	155554
2	133331
3	066663
4	155546
5	133315
6	066633
7	155466
8	133155
9	066333
10	154666
11	131555
12	063333
13	146666
14	115555
15	033333

1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053

TABLE C

CYLINDER BLOCK ASSIGNMENT FOR A GIVEN SURFACE

	CURRENT ZONE - RANGE	OVERWRITE CYL BLK RANGE	COMPATIBILITY CYL BLK RANGE
	-----	-----	-----
1	- CYL 0-127	CYL 0-15	CYL 112-127
2	- 128-255	128-143	240-255
3	- 256-383	256-271	368-383
4	- 384-511	384-399	496-511
5	- 512-639	512-527	624-639
6	- 640-767	640-655	752-767
7	- 768-822	768-783	----

1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105

TABLE D

BASIC CYLINDER BLOCK LAYOUT EXAMPLE

CYLINDER NUMBERS	SECTOR NUMBERS																															
	0	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
112	0	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
113	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	0
114	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	1	
115	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	2	0	
116	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	3	1	0	
117	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	4	2	1	0	
118	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	5	3	2	1	0	
119	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	6	4	3	2	1	0	
120	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	7	5	4	3	2	1	0	
121	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	8	6	5	4	3	2	1	0	
122	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	9	7	6	5	4	3	2	1	0	
123	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	10	8	7	6	5	4	3	2	1	0	
124	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	11	9	8	7	6	5	4	3	2	1	0	
125	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	12	10	9	8	7	6	5	4	3	2	1	0	
126	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	13	11	10	9	8	7	6	5	4	3	2	1	0	
127	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	14	12	11	10	9	8	7	6	5	4	3	2	1	0	

1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136

TABLE E

OVERWRITE CYLINDERS

DRIVE #	CYLINDERS OVERWRITTEN
0	0, 128, 256, 384, 512, 640, 768
1	1, 129, 257, 385, 513, 641, 769
2	2, 130, 258, 386, 514, 642, 770
3	3, 131, 259, 387, 515, 643, 771
4	4, 132, 260, 388, 516, 644, 772
5	5, 133, 261, 389, 517, 645, 773
6	6, 134, 262, 390, 518, 646, 774
7	7, 135, 263, 391, 519, 647, 775
8	8, 136, 264, 392, 520, 648, 776
9	9, 137, 265, 393, 521, 649, 777
10	10, 138, 266, 394, 522, 650, 778
11	11, 139, 267, 395, 523, 651, 779
12	12, 140, 268, 396, 524, 652, 780
13	13, 141, 269, 397, 525, 653, 781
14	14, 142, 270, 398, 526, 654, 782
15	15, 143, 271, 399, 527, 655, 783

1137
1138
1139
1140
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

TABLE F

SELF-TEST CYLINDERS

DRIVE #	CYLINDERS
0	17, 145, 273, 401, 529, 657, 785
1	18, 146, 274, 402, 530, 658, 786
2	19, 147, 275, 403, 531, 659, 787
3	20, 148, 276, 404, 532, 660, 788
4	21, 149, 277, 405, 533, 661, 789
5	22, 150, 278, 406, 534, 662, 790
6	23, 151, 279, 407, 535, 663, 791
7	24, 152, 280, 408, 536, 664, 792
8	25, 153, 281, 409, 537, 665, 793
9	26, 154, 282, 410, 538, 666, 794
10	27, 155, 283, 411, 539, 667, 795
11	28, 156, 284, 412, 540, 668, 796
12	29, 157, 285, 413, 541, 669, 797
13	30, 158, 286, 414, 542, 670, 798
14	31, 159, 287, 415, 543, 671, 799
15	32, 160, 288, 416, 544, 672, 800

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
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

TABLE G

PSEUDO-RANDOM DATA PATTERN

WORD #	DATA (OCTAL)
-----	-----
0	040135
1	177070
2	070414
3	064531
4	174473
5	062422
6	114352
7	036620
8	010031
9	052336
10	017310
11	011347
12	102367
13	152567
14	001246
15	160073

12.1 RH70(11)/RMO3 DRIVER

THIS DOCUMENT IS THE USER'S GUIDE FOR THE RH70/RMO3 DRIVER.

12.2 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

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
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

EIGHT BYTES; ONE BYTE PER DRIVE. THE STATE OF EACH DRIVE WILL BE INDICATED AS FOLLOWS:

DRVSTA	DRIVE STATE
>0	ONLINE RMO3
=0	OFFLINE RMO3, DRIVE IS NOT AN RMO3, OR NONEXISTENT DRIVE
<0	UNSAFE RMO3

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

DRVTYP	CONDITION
0	NONEXISTENT DRIVE
4	RMO3
-1	NOT AN RMO3

THE 'RMINIT' ROUTINE WILL DO A READIN PRESET AND WILL SET FMT22.

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

CALL: JSR RO,RMO3 ;MAKE THE CALL
 PNTDPB ;ADDRESS OF DPB*
 RETURN1 ;RETURN IF QUEUE IS FULL
 RETURN2 ;RETURN IF REQUEST IS IN
 ;QUEUE OR THERE IS AN
 ;ERROR CONDITION

*DPB (DATA PARAMETER BLOCK)

PNTDPB: .BYTE 0 ;(0) DRIVE NUMBER
 .BYTE 0 ;(1) OFFSET VALUE OR FMT22, ECT, AND HCI
 .BYTE 0 ;(2) COMMAND
 .BYTE 0 ;(3) PSEL AND A17 AND A16
 .WORD 0 ;(4) WORD COUNT (MUST BE NEG.)
 .WORD 0 ;(6) BUFFER ADDRESS OR
 ;REGISTER TABLE POINTER
 .BYTE 0 ;(10) SECTOR ADDRESS OR
 ;FIRST REG. INDEX
 .BYTE 0 ;(11) TRACK ADDRESS OR
 ;LAST REG. INDEX
 .WORD 0 ;(12) CYLINDER ADDRESS
 .WORD 0 ;(14) ERROR TABLE POINTER
 ;POINTS TO THE FIRST OF TWENTY
 ;LOCATIONS OF WHERE THE DRIVER
 ;IS TO STORE THE RH70/RMO3
 ;REGISTERS ON AN ERROR. IF LEFT
 ;ZERO REGISTERS ARE NOT SAVED.


```

1284                                     .WORD 0                               ;(16) STATUS/ERROR INDICATOR
1285                                     ;BIT15=1=>ERROR OCCURRED
1286                                     ;BIT07=1=>DONE
1287                                     ;BIT14-BIT09 AND BIT06-BIT03
1288                                     ;INDICATE TYPE OF ERROR
1289

```

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

```

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

```

IT SHOULD BE NOTED THAT YOU MUST PROVIDE THE CODE TO DRIVE THE CLOCK. AND THE ELAPSED TIME MUST BE IN MILLISECONDS. THE DRIVER WILL SET THE TIMEOUT TO 1 SECOND FOR ALL POSITIONING AND DATA TRANSFER OPERATIONS AND WILL SET THE TIMEOUT TO 30 SECONDS FOR ERROR RECOVERY OPERATIONS.

12.4.1 EXAMPLE - WRITE 1000. WORDS

```

1S: JSR RO,RMO3 ;CALL THE DRIVER
WRTDPB ;DPB ADDRESS
BR 1S ;WAIT FOR QUEUE IF FULL
2S: TST WRTDPB+16 ;WAIT FOR COMMAND TO COMPLETE
BEQ 2S
BMI ERROR1 ;ERROR OCCURRED
.
.
.

```

```

WRTDPB: .BYTE 5 ;DRIVE #5
        .BYTE 0
        .BYTE 161 ;WRITE COMMAND
        .BYTE 0
        .WORD -1000. ;WORD COUNT
        .WORD WRTBUF ;BUFFER ADDRESS
        .BYTE 3 ;SECTOR
        .BYTE 5 ;TRACK
        .WORD 400 ;CYLINDER
        .WORD ERRTBS ;ERROR TABLE
        .WORD 0 ;STATUS/ERROR INDICATOR

```

ALTERNATE DPB SETUP

```

WRTDPB: .WORD 5 ;THIS SETUP ACHIEVED
        .WORD WRITE ;EVERYTHING THE
        .WORD -1000. ;ABOVE TABLE DID, BUT
        .WORD WRTBUF ;IN A CLEANER FORMAT
        .BYTE 3,5
        .WORD 400,ERRTBS,0

```

12.5 RH70/RMO3 REGISTERS

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
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339

	<u>MNEMONIC</u>	<u>INDEX</u>
1340		
1341		
1342		
1343		
1344		
1345		
1346		
1347		
1348		
1349		
1350		
1351		
1352		
1353		
1354		
1355		
1356		
1357		
1358		
1359		
1360		
1361		
1362		
1363		
1364		
1365		
1366		
1367		
1368		
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		

RMCS1	0
RMWC	2
RMBA	4
RMDA	6
RMCS2	10
RMD5	12
RMR1	14
RMA5	16
RMLA	20
RMD8	22
RMR1	24
RMDT	26
RMSN	30
RMOF	32
RMDC	34
RMR	36
RMR2	40
RMR2	42
RMEC1	44
RMEC2	46

12.6 COMMANDS PERFORMED BY THE DRIVER

	<u>COMMAND</u>	<u>CODE</u>	<u>COMMAND TYPE</u>
1364			
1365			
1366			
1367			
1368			
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			

NO OPERATION	101	N
UNLOAD	103	N
SEEK	105	P
RECALIRATE	107	P
DRIVE CLEAR	111	N
RELEASE	113	N
OFFSET	115	P
RETURN TO CENTER	117	P
READIN PRESET	121	N
PACK ACKNOWLEDGE	123	N
SEARCH	131	P
GET REGISTER(S)	141	S
SET FORMAT	143	S
SELECT DRIVE	145	S

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
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

WRITE CHECK DATA	151	D
WRITE CHECK HEADER AND DATA	153	D
WRITE DATA	161	D
WRITE HEADER & DATA	163	D
READ DATA	171	D
READ HEADER & DATA	173	D

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

12.7 DPB STATUS/ERROR INDICATOR WORD

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

<u>BIT NO.</u>	<u>MEANING IF ON A "1"</u>
15	ERROR OCCURRED DONE (BIT07=0); BITS 14-10 SPECIFIES TYPE DONE (BIT07=1); BITS 06-03 SPECIFIES TYPE
14(1)	USER MADE A REQUEST FOR A FUNCTION TO BE PERFORMED ON AN OFFLINE OR UNSAFE DRIVE
13(1)	USER MADE A REQUEST FOR A FUNCTION TO BE PERFORMED ON A DRIVE THAT HAS AN UNLOAD REQUEST IN QUEUE.
12(2)	PERSISTENT UNSAFE CONDITION EXIST.
11(2)	UNCORRECTABLE PARITY ERROR OCCURRED
10(2)(4)	FATAL PARITY ERROR. A MASSBUS CLEAR WAS PERFORMED, ALL QUEUES WERE EMPTIED, AND ALL DRVACT'S SET TO THE IDLE STATE
9(3)(4)	SOFTWARE TIMEOUT OCCURRED ON THIS DRIVE
8(4)	SOFTWARE TIMEOUT OCCURRED ON ANOTHER DRIVE
7	DONE
6(2)	ERROR OCCURRED DURING AN I/O OPERATION

1452	5(2)	ERROR OCCURRED DURING AN OPERATION OTHER THAN I/O.
1453		
1454		
1455	4(2)	CORRECTABLE UNSAFE CONDITION OCCURRED
1456		
1457	3(2)	DRIVE ERROR OCCURRED THAT CAUSED AN AUTOMATIC "RECALIBRATE" SEQUENCE
1458		
1459		
1460	2	PORT REQUEST TIMEOUT. THE DRIVER REQUESTED THE DRIVE BUT THE OPPOSITE PORT DID NOT RELEASE THE DRIVE WITHIN 20 SECONDS.
1461		
1462		
1463		
1464	1	NON-EXISTENT DRIVE REQUESTED. USER MADE A REQUEST FOR A NON-EXISTENT DRIVE.
1465		
1466		
1467	(1) =>	REQUEST WASN'T PUT IN QUEUE. (RH70/RMO3 REGISTERS WERE NOT SAVED)
1468		
1469		
1470	(2) =>	REQUEST QUEUE HAS BEEN EMPTIED. THE DRIVER ISSUED A "DRIVE CLEAR" TO THE DRIVE. NOTE: ALL RH70/RMO3 REGISTERS ARE SAVED AS PER DPB+14 BEFORE THE "DRIVE CLEAR".
1471		
1472		
1473		
1474		
1475	(3) =>	REQUEST QUEUE HAS BEEN EMPTIED. THE DRIVER ISSUED A MASSBUS INIT. ALL RH70/RMO3 REGISTERS FOR THE DRIVE WERE SAVED AS PER DPB+14 BEFORE THE INIT.
1476		
1477		
1478		
1479		
1480	(4) =>	A "RECALIBRATE" SHOULD BE ISSUED BEFORE ANY OTHER COMMAND.
1481		
1482		

12.8 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=RMER2
3	MASSBUS PARITY	RD.ADR= ADDRESS OF REG. READ

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
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

K03

MAINDEC-11-DZRM1 AO/00, RM03 DRIVE COMPATIBILITY TEST MACY11 30(1046) 21-JUL-77 16:51 PAGE 36
DZRMIA.P11 21-JUL-77 15:44

SEQ 0035

1508		ERROR (MCPE=1)	RD.WRD= WORD READ
1509			
1510			
1511	4	MASSBUS PARITY	WRT.AD= ADDRESS OF REG. WRITTEN
1512		ERROR (PAR=1)	WRT.WD= WORD WRITTEN
1513			RD.WRD= WORD READ BACK
1514			
1515	5	ADDRESS PLUG CHANGE	R1= DRIVE NUMBER
1516		BIT SET ('OPE' ERROR)	R3= ATA BIT
1517			*R4= RMCS1'S ADDRESS
1518			R5= (RMAS)
1519			RMERRS =RMS
1520			RMERRS+2=RMR1
1521			RMERRS+4=RMR2
1522			RMERRS+6=RMMR2
1523			
1524			
1525			

* THIS IS THE ACTUAL UNIBUS ADDRESS (176700)

2

```

1526
1527 .TITLE MAINDEC-11-DZRMI AD/00, RMO3 DRIVE COMPATIBILITY TEST
1528 ;*COPYRIGHT (C) 1976
1529 ;*DIGITAL EQUIPMENT CORP.
1530 ;*MAYNARD, MASS. 01754
1531 ;*
1532 ;*PROGRAM BY C. CHEN
1533 ;*
1534 ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
1535 ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
1536 ;*
1537 .SBTTL OPERATIONAL SWITCH SETTINGS
1538 ;*
1539 ;* SWITCH USE
1540 ;* -----
1541 ;* 15 HALT ON ERROR
1542 ;* 14 LOOP ON TEST
1543 ;* 13 INHIBIT ERROR TYPEOUTS
1544 ;* 12 INHIBIT TRACE TRAP
1545 ;* 11 INHIBIT ITERATIONS
1546 ;* 10 BELL ON ERROR
1547 ;* 9 LOOP ON ERROR
1548 ;* 8 LOOP ON TEST IN SWR<7:0>
1549 ;* 7 TYPE THE BAD SECTOR FILE
1550 .SBTTL BASIC DEFINITIONS
1551 ;*
1552 ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
1553 001100 STACK= 1100
1554 .EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL
1555 .EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL
1556 ;*
1557 ;*MISCELLANEOUS DEFINITIONS
1558 000011 HT= 11 ;;CODE FOR HORIZONTAL TAB
1559 000012 LF= 12 ;;CODE FOR LINE FEED
1560 000015 CR= 15 ;;CODE FOR CARRIAGE RETURN
1561 000200 CRLF= 200 ;;CODE FOR CARRIAGE RETURN-LINE FEED
1562 177776 PS= 177776 ;;PROCESSOR STATUS WORD
1563 .EQUIV PS,PSW
1564 177774 STKLMT= 177774 ;;STACK LIMIT REGISTER
1565 177772 PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
1566 177570 DSWR= 177570 ;;HARDWARE SWITCH REGISTER
1567 177570 DDISP= 177570 ;;HARDWARE DISPLAY REGISTER
1568 ;*
1569 ;*GENERAL PURPOSE REGISTER DEFINITIONS
1570 000000 R0= %0 ;;GENERAL REGISTER
1571 000001 R1= %1 ;;GENERAL REGISTER
1572 000002 R2= %2 ;;GENERAL REGISTER
1573 000003 R3= %3 ;;GENERAL REGISTER
1574 000004 R4= %4 ;;GENERAL REGISTER
1575 000005 R5= %5 ;;GENERAL REGISTER
1576 000006 R6= %6 ;;GENERAL REGISTER
1577 000007 R7= %7 ;;GENERAL REGISTER
1578 000006 SP= %6 ;;STACK POINTER
1579 000007 PC= %7 ;;PROGRAM COUNTER
1580 ;*
1581 ;*PRIORITY LEVEL DEFINITIONS

```


1582	000000	PR0=	0	:: PRIORITY LEVEL 0
1583	000040	PR1=	40	:: PRIORITY LEVEL 1
1584	000100	PR2=	100	:: PRIORITY LEVEL 2
1585	000140	PR3=	140	:: PRIORITY LEVEL 3
1586	000200	PR4=	200	:: PRIORITY LEVEL 4
1587	000240	PR5=	240	:: PRIORITY LEVEL 5
1588	000300	PR6=	300	:: PRIORITY LEVEL 6
1589	000340	PR7=	340	:: PRIORITY LEVEL 7

```
1591 ;*SWITCH REGISTER* SWITCH DEFINITIONS
```

1592	100000	SW15=	100000
1593	040000	SW14=	40000
1594	020000	SW13=	20000
1595	010000	SW12=	10000
1596	004000	SW11=	4000
1597	002000	SW10=	2000
1598	001000	SW09=	1000
1599	000400	SW08=	400
1600	000200	SW07=	200
1601	000100	SW06=	100
1602	000040	SW05=	40
1603	000020	SW04=	20
1604	000010	SW03=	10
1605	000004	SW02=	4
1606	000002	SW01=	2
1607	000001	SW00=	1
1608		.EQUIV	SW09, SW9
1609		.EQUIV	SW08, SW8
1610		.EQUIV	SW07, SW7
1611		.EQUIV	SW06, SW6
1612		.EQUIV	SW05, SW5
1613		.EQUIV	SW04, SW4
1614		.EQUIV	SW03, SW3
1615		.EQUIV	SW02, SW2
1616		.EQUIV	SW01, SW1
1617		.EQUIV	SW00, SW0

```
1619 ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
```

1620	100000	BIT15=	100000
1621	040000	BIT14=	40000
1622	020000	BIT13=	20000
1623	010000	BIT12=	10000
1624	004000	BIT11=	4000
1625	002000	BIT10=	2000
1626	001000	BIT09=	1000
1627	000400	BIT08=	400
1628	000200	BIT07=	200
1629	000100	BIT06=	100
1630	000040	BIT05=	40
1631	000020	BIT04=	20
1632	000010	BIT03=	10
1633	000004	BIT02=	4
1634	000002	BIT01=	2
1635	000001	BIT00=	1
1636		.EQUIV	BIT09, BIT9
1637		.EQUIV	BIT08, BIT8

BASIC DEFINITIONS

```

1638 .EQUIV BIT07,BIT7
1639 .EQUIV BIT06,BIT6
1640 .EQUIV BIT05,BIT5
1641 .EQUIV BIT04,BIT4
1642 .EQUIV BIT03,BIT3
1643 .EQUIV BIT02,BIT2
1644 .EQUIV BIT01,BIT1
1645 .EQUIV BIT00,BIT0
    
```

;;#BASIC "CPU" TRAP VECTOR ADDRESSES

```

1647 000004 ERRVEC= 4 ; TIME OUT AND OTHER ERRORS
1648 000010 RESVEC= 10 ; RESERVED AND ILLEGAL INSTRUCTIONS
1649 000014 TBITVEC=14 ; "T" BIT
1650 000014 TRTVEC= 14 ; TRACE TRAP
1651 000014 BPTVEC= 14 ; BREAKPOINT TRAP (BPT)
1652 000014 IOTVEC= 20 ; INPUT/OUTPUT TRAP (IOT) **SCOPE**
1653 000020 PMRVEC= 24 ; POWER FAIL
1654 000024 EMTVEC= 30 ; EMULATOR TRAP (EMT) **ERROR**
1655 000030 TRAPVEC=34 ; "TRAP" TRAP
1656 000034 TKVEC= 60 ; TTY KEYBOARD VECTOR
1657 000060 TPVEC= 64 ; TTY PRINTER VECTOR
1658 000064 PIRQVEC=240 ; PROGRAM INTERRUPT REQUEST VECTOR
1659 176700 ABASE=176700
1660 000254 AVECT1=254
    
```

.SBTTL RM03 REGISTERS

.SBTTL RM03 DRIVER COMMANDS

;;*****

```

1671 000101 RNOP = 101 ; NO OPERATION
1672 000105 SEEK = 105 ; SEEK
1673 000107 RECAL = 107 ; RECALIBRATE
1674 000111 DRVCLR = 111 ; DRIVE CLEAR
1675 000113 RELSE = 113 ; RELEASE
1676 000115 OFFSET = 115 ; OFFSET
1677 000117 RTC = 117 ; RETURN TO CENTER LINE
1678 000121 READIN = 121 ; READ IN PRESET
1679 000123 ACK = 123 ; PACK ACKNOWLEDGE
1680 000131 SEARCH = 131 ; SEARCH
1681 000141 GETREG = 141 ; GET REGISTERS
1682 000143 SETFMT = 143 ; SET FORMAT (& ECI OR HCI)
1683 000145 SELDRV = 145 ; SELECT DRIVE
1684 000151 WCKD = 151 ; WRITE CHECK DATA
1685 000153 WCKHD = 153 ; WRITE CHECK HEADER & DATA
1686 000161 WRTDAT = 161 ; WRITE DATA
1687 000163 WRTHD = 163 ; WRITE HEADER & DATA
1688 000171 RDDAT = 171 ; READ DATA
1689 000173 RDHD = 173 ; READ HEADER & DATA
    
```

.SBTTL TRAP CATCHER

```

1691
1692 000000
1693 . = 0
    
```



```

1694 ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
1695 ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
1696 ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
1697      .=174
1698 000174 000000  DISPREG: .WORD 0          ;; SOFTWARE DISPLAY REGISTER
1699 000176 000000  SWREG:   .WORD 0          ;; SOFTWARE SWITCH REGISTER
1700      .SBTTL  STARTING ADDRESS(ES)
1701 000200 000137 003062  JMP      @#START ;; JUMP TO STARTING ADDRESS OF PROGRAM
1702 000204 000137 003072  JMP      @#START1 ;; CHANGE THE RH11 UNIBUS ADDRESS
1703                                     ;; AFTER INITIAL START
1704
1705      .=220                                     ; SECOND PASS STARTING ADDRESS
1706 000220 000137 003102  JMP      @#START2
1707      .SBTTL  ACT11 HOOKS
1708
1709 ;;*****
1710 ; HOOKS REQUIRED BY ACT11
1711      $SVPC=.                                     ; SAVE PC
1712      .=46
1713 000046 017312  SENDAD                                     ;; 1) SET LOC.46 TO ADDRESS OF SENDAD IN .SEOP
1714      .=52
1715 000052 040000  .WORD 40000                               ;; 2) SET LOC.52 TO 40000
1716      .=$SVPC                                     ;; RESTORE PC
1717      .=1100
1718      .SBTTL  APT PARAMETER BLOCK
1719
1720 ;;*****
1721 ; SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
1722 ;;*****
1723      .SX=.                                     ;; SAVE CURRENT LOCATION
1724      .=24                                     ;; SET POWER FAIL TO POINT TO START OF PROGRAM
1725 000024 000200  200                                     ;; FOR APT START UP
1726      .=44                                     ;; POINT TO APT INDIRECT ADDRESS PNTR.
1727 000044 001100  $APTHDR ;; POINT TO APT HEADER BLOCK
1728      .=.SX                                     ;; RESET LOCATION COUNTER
1729 ;;*****
1730 ; SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
1731 ; INTERFACE SPEC.
1732
1733 001100  $APTHD:
1734 001100 000000  $HIBTS: .WORD 0          ;; TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
1735 001102 001210  $MBAOR: .WORD $MAIL     ;; ADDRESS OF APT MAILBOX (BITS 0-15)
1736 001104 000764  $STSM:  .WORD 500.     ;; RUN TIM OF LONGEST TEST
1737 001106 000764  $PASTM: .WORD 500.     ;; RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
1738 001110 000764  $UNITM: .WORD 500.     ;; ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
1739 001112 000032  .WORD  SETEND-$MAIL/2 ;; LENGTH MAILBOX-ETABLE(WORDS)
1740      TAB.XY=.

```


1741
1742
1743
1744
1745
1746
1747 001114
1748 001114 000000
1749 001114 000000
1750 001116 000
1751 001117 000
1752 001120 000000
1753 001122 000000
1754 001124 000000
1755 001126 000000
1756 001130 000
1757 001131 001
1758 001132 000000
1759 001134 000000
1760 001136 000000
1761 001140 000000
1762 001142 000000
1763 001144 000000
1764 001146 000000
1765 001150 000
1766 001151 000
1767 001152 000000
1768 001154 177570
1769 001156 177570
1770 001160 177560
1771 001162 177562
1772 001164 177564
1773 001166 177566
1774 001170 000
1775 001171 002
1776 001172 012
1777 001173 000
1778 001174 000000
1779 001176 000000
1780 001200 177607 000377
1781 001204 077
1782 001205 015
1783 001206 000012
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793 001210
1794 001210 000000
1795 001212 000000
1796 001214 000000

.SBTTL COMMON TAGS

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

.=TAB.XY

SCMTAG: .WORD 0 ; ; START OF COMMON TAGS
\$TSTNM: .BYTE 0 ; ; CONTAINS THE TEST NUMBER
\$ERFLG: .BYTE 0 ; ; CONTAINS ERROR FLAG
\$ICNT: .WORD 0 ; ; CONTAINS SUBTEST ITERATION COUNT
\$LPADR: .WORD 0 ; ; CONTAINS SCOPE LOOP ADDRESS
\$LPERR: .WORD 0 ; ; CONTAINS SCOPE RETURN FOR ERRORS
\$ERTTL: .WORD 0 ; ; CONTAINS TOTAL ERRORS DETECTED
\$ITEMB: .BYTE 0 ; ; CONTAINS ITEM CONTROL BYTE
\$ERMAX: .BYTE 1 ; ; CONTAINS MAX. ERRORS PER TEST
\$ERRPC: .WORD 0 ; ; CONTAINS PC OF LAST ERROR INSTRUCTION
\$GDADR: .WORD 0 ; ; CONTAINS ADDRESS OF 'GOOD' DATA
\$BDADR: .WORD 0 ; ; CONTAINS ADDRESS OF 'BAD' DATA
\$GDAT: .WORD 0 ; ; CONTAINS 'GOOD' DATA
\$BDAT: .WORD 0 ; ; CONTAINS 'BAD' DATA
 ; ; RESERVED--NOT TO BE USED
\$AUTOB: .BYTE 0 ; ; AUTOMATIC MODE INDICATOR
\$INTAG: .BYTE 0 ; ; INTERRUPT MODE INDICATOR
\$SWR: .WORD DSWR ; ; ADDRESS OF SWITCH REGISTER
\$DISPLAY: .WORD DDISP ; ; ADDRESS OF DISPLAY REGISTER
\$TKS: 177560 ; ; TTY KBD STATUS
\$TKB: 177562 ; ; TTY KBD BUFFER
\$TPS: 177564 ; ; TTY PRINTER STATUS REG. ADDRESS
\$TPB: 177566 ; ; TTY PRINTER BUFFER REG. ADDRESS
\$NULL: .BYTE 0 ; ; CONTAINS NULL CHARACTER FOR FILLS
\$FILLS: .BYTE 2 ; ; CONTAINS # OF FILLER CHARACTERS REQUIRED
\$FILLC: .BYTE 12 ; ; INSERT FILL CHARS. AFTER A "LINE FEED"
\$TPFLG: .BYTE 0 ; ; "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
\$TIMES: 0 ; ; MAX. NUMBER OF ITERATIONS
\$ESCAPE: 0 ; ; ESCAPE ON ERROR ADDRESS
\$BELL: .ASCIZ <207><377><377> ; ; CODE FOR BELL
\$QUES: .ASCII /?/ ; ; QUESTION MARK
\$CRLF: .ASCII <15> ; ; CARRIAGE RETURN
\$LF: .ASCIZ <12> ; ; LINE FEED

.SBTTL APT MAILBOX-ETABLE

.NLIST ME
;
;

.EVEN
\$MAIL: ; ; APT MAILBOX
\$MSGTY: .WORD AMSGTY ; ; MESSAGE TYPE CODE
\$FATAL: .WORD AFATAL ; ; FATAL ERROR NUMBER
\$TESTN: .WORD ATESTN ; ; TEST NUMBER

1797	001216	000000	SPASS:	.WORD	APASS	::	PASS COUNT
1798	001220	000000	SDEVCT:	.WORD	ADEVCT	::	DEVICE COUNT
1799	001222	000000	SUNIT:	.WORD	AUNIT	::	I/O UNIT NUMBER
1800	001224	000000	SMSGAD:	.WORD	AMSGAD	::	MESSAGE ADDRESS
1801	001226	000000	SMSGLG:	.WORD	AMSGLG	::	MESSAGE LENGTH
1802	001230		SETABLE:			::	APT ENVIRONMENT TABLE
1803	001230	000	SENV:	.BYTE	AENV	::	ENVIRONMENT BYTE
1804	001231	000	SENVH:	.BYTE	AENVH	::	ENVIRONMENT MODE BITS
1805	001232	000000	SSHREG:	.WORD	ASWREG	::	APT SWITCH REGISTER
1806	001234	000000	SUSWR:	.WORD	AUSWR	::	USER SWITCHES
1807	001236	000000	SCPUOP:	.WORD	ACPUOP	::	CPU TYPE, OPTIONS
1808			*			::	BITS 15-11=CPU TYPE
1809			*			::	11/04=01, 11/05=02, 11/20=03, 11/40=04, 11/45=05
1810			*			::	11/70=06, PDQ=07, Q=10
1811			*			::	BIT 10=REAL TIME CLOCK
1812			*			::	BIT 9=FLOATING POINT PROCESSOR
1813			*			::	BIT 8=MEMORY MANAGEMENT
1814	001240	000	SMAMS1:	.BYTE	AMAMS1	::	HIGH ADDRESS, M.S. BYTE
1815	001241	000	SMTYP1:	.BYTE	AMTYP1	::	MEM. TYPE, BLK#1
1816			*			::	MEM. TYPE BYTE -- (HIGH BYTE)
1817			*			::	900 NSEC CORE=001
1818			*			::	300 NSEC BIPOLAR=002
1819			*			::	500 NSEC MOS=003
1820	001242	000000	SMADR1:	.WORD	AMADR1	::	HIGH ADDRESS, BLK#1
1821			*			::	MEM. LAST ADDR. =3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE
1822	001244	000	SMAMS2:	.BYTE	AMAMS2	::	HIGH ADDRESS, M.S. BYTE
1823	001245	000	SMTYP2:	.BYTE	AMTYP2	::	MEM. TYPE, BLK#2
1824	001246	000000	SMADR2:	.WORD	AMADR2	::	MEM. LAST ADDRESS, BLK#2
1825	001250	000	SMAMS3:	.BYTE	AMAMS3	::	HIGH ADDRESS, M.S. BYTE
1826	001251	000	SMTYP3:	.BYTE	AMTYP3	::	MEM. TYPE, BLK#3
1827	001252	000000	SMADR3:	.WORD	AMADR3	::	MEM. LAST ADDRESS, BLK#3
1828	001254	000	SMAMS4:	.BYTE	AMAMS4	::	HIGH ADDRESS, M.S. BYTE
1829	001255	000	SMTYP4:	.BYTE	AMTYP4	::	MEM. TYPE, BLK#4
1830	001256	000000	SMADR4:	.WORD	AMADR4	::	MEM. LAST ADDRESS, BLK#4
1831	001260	000254	SVECT1:	.WORD	AVECT1	::	INTERRUPT VECTOR#1, BUS PRIORITY#1
1832	001262	000000	SVECT2:	.WORD	AVECT2	::	INTERRUPT VECTOR#2, BUS PRIORITY#2
1833	001264	176700	SBASE:	.WORD	ABASE	::	BASE ADDRESS OF EQUIPMENT UNDER TEST
1834	001266	000000	SDEVH:	.WORD	ADEVH	::	DEVICE MAP
1835	001270	000000	SCDW1:	.WORD	ACDW1	::	CONTROLLER DESCRIPTION WORD#1
1836	001272	000000	SCDW2:	.WORD	ACDW2	::	CONTROLLER DESCRIPTION WORD#2
1837	001274		SETEND:				
1838			.MEXIT				
1839		000015	CR	=	15		
1840		000012	LF	=	12		
1841	001274	176700	SRMADR:	.WORD	176700		:: FIRST ADDRESS OF RH11/RMD3 REGISTERS
1842	001276	000254	SRMVEC:	.WORD	254		:: RMD3 VECTOR ADDRESS
1843	001300	172540	SLKCSR:	.WORD	172540		:: ADDR OF KW11-P STATUS REGISTER
1844	001302	172542	SLKCSB:	.WORD	172542		:: ADDR OF KW11-P COUNTER BUFFER
1845	001304	000104	SLPVEC:	.WORD	104		:: ADDR OF KW11-P VECTOR
1846	001306	177546	SLKS:	.WORD	177546		:: ADDR OF KW11-L STATUS REGISTER
1847	001310	000100	SLLVEC:	.WORD	100		:: ADDR OF KW11-L VECTOR
1848	001312	177777	PCLOCK:	.WORD	-1		:: '0' IF KW11-P IS ON SYSTEM
1849	001314	177777	CLKFLG:	.WORD	-1		:: '0' IF A CLOCK IS AVAILABLE
1850	001316	000074	HZ:	.WORD	74		:: 74 (8) IF 60 HZ SYSTEM; 62 (8) IF 50 HZ SYSTEM
1851	001320	000000	STATIN:	.WORD	0		:: 'TYPE STATISTICS' INDICATOR
1852	001322	000000	PACK:	.WORD	0		:: ENTRY TO THE TABLE D

1853		001222	DRIVE	=	SUNIT		;DRIVE # STORAGE: ERRORS 1-5 & 10
1854	001324	000000	ATTN:	.WORD	0		;ATTN REG STORAGE: ERRORS 1-5 & 10
1855	001326	000000	UNIT:	.WORD	0		;DRIVE # STORAGE FOR PRINTOUT
1856							;RETRY COUNT IN THE UPPER BYTE
1857	001330	000000	LSTAD:	.WORD	0		;STORE LAST MEMORY ADDRESS HERE
1858	001332	000000	CHGADR:	.WORD	0		;CHANGE RH11 UNIBUS ADDRESS FLAG
1859	001334	000000	CFLAG:	.WORD	0		;CONTROL C' FLAG
1860	001336	000000	TSTNM:	.WORD	0		;TEST NUMBER FOR PRINT AND SCORE RT.
1861	001340	000000	BADSEC:	.WORD	0		;BAD SECTOR/TRACK FLAG
1862	001342	000000	HOUR:	.WORD	0		;HOUR COUNT STORED HERE (MAXIMUM - 999.)
1863	001344	000000	MINUTE:	.WORD	0		;MINUTE'S COUNT STORED HERE
1864	001346	000000	SECOND:	.WORD	0		;SECOND'S COUNT STORED HERE
1865	001350	000000	SIXTEE:	.WORD	0		;TIMER ROUTINE COUNTER (FOR ONE SECOND)
1866	001352	000000	CMCNT:	.WORD	0		;ZONE COUNT
1867	001354	000000	CMCYL:	.WORD	0		;CYLINDER ADDRESS
1868	001356	000000	STARSC:	.WORD	0		;STARTING SECTOR (FOR TEST 6,8)
1869	001360	000000	CMSEC:	.WORD	0		;DALTA CYLINDER COUNT
1870	001362	000000	CMTRK:	.WORD	0		;TRACK ADDRESS
1871	001364	000000	MULINE:	.WORD	0	;NEW LINE	FLAG AND COLUMN CTR
1872	001366	000037	SECLMT:	.WORD	31.		;SECTOR ADDRESS LIMIT
1873	001370	000004	TRKLMT:	.WORD	4.		;TRACK ADDRESS LIMIT
1874	001372	001466	CYLINT:	.WORD	822.		;CYLINDER ADDRESS LIMIT FOR RMD3
1875	001374	000000	FAULT:	.WORD	0		;=1, IF ALL DRIVES NOT COMPATIBLE

1876
1877
1878 .SBTTL COMMON PARAMETERS
1879

1880
1881 .SBTTL TABLES, CONSTANTS, AND VARIABLE LOCATIONS
1882

1883 ;*****
1884 ;TABLE D
1885 ;TABLE LISTED BELOW SPECIFIES THE SECTORS TO BE WRITTEN
1886 ;BY A LOGICAL DRIVE. EACH LOGICAL DRIVE WRITES TWO SECTORS IN ONE
1887 ;CYLINDER, 16 CYLINDERS IN ONE BLOCK, 2 BLOCKS IN ONE WRITE-CURRENT
1888 ;ZONE AND 7 CURRENT ZONES IN A PACK.
1889 ;
1890 ;
1891 ;

1892	001376	000	017	015	LOG0:	.BYTE	0,15.,13.,10.,6,1,11.,4,12.,13.,9.,14.,2,5,7,8.	;DRIVE 0
1893	001401	012	006	001				
1894	001404	013	004	014				
1895	001407	015	011	016				
1896	001412	002	005	007				
1897	001415	010						
1898	001416	001	000	016	LOG1:	.BYTE	1,0,14.,11.,7,2,12.,5,13.,4,10.,15.,3,6,8.,9.	;DRIVE 1
1899	001421	013	007	002				
1900	001424	014	005	015				
1901	001427	004	012	017				
1902	001432	003	006	010				
1903	001435	011						
1904	001436	002	001	017	LOG2:	.BYTE	2,1,15.,12.,8.,3,13.,6,14.,5,11.,0,4,7,9.,10.	;DRIVE 2
1905	001441	014	010	003				
1906	001444	015	006	016				
1907	001447	005	013	000				
1908	001452	004	007	011				

F04

1909	001455	012							
1910	001456	003	002	000	LOG3:	.BYTE	3,2,0,13.,9.,4,14.,7,15.,6,12.,1,5,8.,10.,11.		
1911	001461	015	011	004					
1912	001464	016	007	017					
1913	001467	006	014	001					
1914	001472	005	010	012					
1915	001475	013							
1916	001476	004	003	001	LOG4:	.BYTE	4,3,1,14.,10.,5,15.,8.,0,7,13.,2,6,9.,11.,12.		
1917	001501	016	012	005					
1918	001504	017	010	000					
1919	001507	007	015	002					
1920	001512	006	011	013					
1921	001515	014							
1922	001516	005	004	002	LOG5:	.BYTE	5,4,2,15.,11.,6,0,9.,1,8.,14.,3,7,10.,12.,13.		
1923	001521	017	013	006					
1924	001524	000	011	001					
1925	001527	010	016	003					
1926	001532	007	012	014					
1927	001535	015							
1928	001536	006	005	003	LOG6:	.BYTE	6,5,3,0,12.,7,1,10.,2,9.,15.,4,8.,11.,13.,14.		
1929	001541	000	014	007					
1930	001544	001	012	002					
1931	001547	011	017	004					
1932	001552	010	013	015					
1933	001555	016							
1934	001556	007	006	004	LOG7:	.BYTE	7,6,4,1,13.,8.,2,11.,3,10.,0,5,9.,12.,14.,15.		
1935	001561	001	015	010					
1936	001564	002	013	003					
1937	001567	012	000	005					
1938	001572	011	014	016					
1939	001575	017							
1940	001576	010	007	005	LOG8:	.BYTE	8.,7,5,2,14.,9.,3,12.,4,11.,1,6,10.,13.,15.,0		
1941	001601	002	016	011					
1942	001604	003	014	004					
1943	001607	013	001	006					
1944	001612	012	015	017					
1945	001615	000							
1946	001616	011	010	006	LOG9:	.BYTE	9.,8.,6,3,15.,10.,4,13.,5,12.,2,7,11.,14.,0,1		
1947	001621	003	017	012					
1948	001624	004	015	005					
1949	001627	014	002	007					
1950	001632	013	016	000					
1951	001635	001							
1952	001636	012	011	007	LOG10:	.BYTE	10.,9.,7.,4,0,11.,5,14.,6,13.,3,8.,12.,15.,1,2		
1953	001641	004	000	013					
1954	001644	005	016	006					
1955	001647	015	003	010					
1956	001652	014	017	001					
1957	001655	002							
1958	001656	013	012	010	LOG11:	.BYTE	11.,10.,8.,5,1,12.,6,15.,7,14.,4,9.,13.,0,2,3		
1959	001661	005	001	014					
1960	001664	006	017	007					
1961	001667	016	004	011					
1962	001672	015	000	002					
1963	001675	003							
1964	001676	014	013	011	LOG12:	.BYTE	12.,11.,9.,6,2,13.,7,0,8.,15.,5,10.,14.,1,3,4		

1965	001701	006	002	015	
1966	001704	007	000	010	
1967	001707	017	005	012	
1968	001712	016	001	003	
1969	001715	004			
1970	001716	015	014	012	LOG13: .BYTE 13.,12.,10.,7,3,14.,8.,1,9.,0,6,11.,15.,2,4,5
1971	001721	007	003	016	
1972	001724	010	001	011	
1973	001727	000	006	013	
1974	001732	017	002	004	
1975	001735	005			
1976	001736	016	015	013	LOG14: .BYTE 14.,13.,11.,8.,4,15.,9.,2,10.,1,7,12.,0,3,5,6
1977	001741	010	004	017	
1978	001744	011	002	012	
1979	001747	001	007	014	
1980	001752	000	003	005	
1981	001755	006			
1982	001756	017	016	014	LOG15: .BYTE 15.,14.,12.,9.,5,0,10.,3,11.,2,8.,13.,1,4,6,7
1983	001761	011	005	000	
1984	001764	012	003	013	
1985	001767	002	010	015	
1986	001772	001	004	006	
1987	001775	007			
1988					
1989	001776	000000			ASNLST: .WORD 0 ;A BIT SET IS AN ASSIGNED LOGICAL DRIVE
1990					
1991	002000	000000			ASSGN1: .WORD 0 ;A BIT SET IS AN ASSIGNED LOGICAL DRIVE FOR PASS 1
1992					
1993	002002	000000			ASSGN2: .WORD 0 ;A BIT SET IS AN ASSIGNED LOGICAL DRIVE FOR PASS 2
1994					
1995	002004	000020			SYSADR: .BLKW 16. ;SUB SYSTEM ADDRESS TABLE
1996					
1997	002044	000000			TABLEX: .WORD 0 ;CURRENT SELECTED SCORE BOARD
1998					
1999					;SCORE BOARD TABLES
2000					;
2001					;
2002					TABLE OF OVERWRITE SCORE,NEGATIVE OFFSET SCORE
2003					
2004	002046	000	000	000	OVWNO: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2005	002051	000	000	000	
2006	002054	000	000	000	
2007	002057	000	000	000	
2008	002062	000	000	000	
2009	002065	000			
2010	002066	000	000	000	OVWN1: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2011	002071	000	000	000	
2012	002074	000	000	000	
2013	002077	000	000	000	
2014	002102	000	000	000	
2015	002105	000			
2016	002106	000	000	000	OVWN2: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2017	002111	000	000	000	
2018	002114	000	000	000	
2019	002117	000	000	000	
2020	002122	000	000	000	

```

2021 002125 000
2022 002126 000 000 000 OVWN3: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2023 002131 000 000 000
2024 002134 000 000 000
2025 002137 000 000 000
2026 002142 000 000 000
2027 002145 000
2028 002146 000 000 000 OVWN4: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2029 002151 000 000 000
2030 002154 000 000 000
2031 002157 000 000 000
2032 002162 000 000 000
2033 002165 000
2034
2035 ;TABLE OF OVERWRITE SCORE, POSITIVE OFFSET SCORE
2036
2037
2038 002166 000 000 000 OVWP0: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2039 002171 000 000 000
2040 002174 000 000 000
2041 002177 000 000 000
2042 002202 000 000 000
2043 002205 000
2044
2045 002206 000 000 000 OVWP1: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2046 002211 000 000 000
2047 002214 000 000 000
2048 002217 000 000 000
2049 002222 000 000 000
2050 002225 000
2051
2052 002226 000 000 000 OVWP2: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2053 002231 000 000 000
2054 002234 000 000 000
2055 002237 000 000 000
2056 002242 000 000 000
2057 002245 000
2058
2059 002246 000 000 000 OVWP3: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2060 002251 000 000 000
2061 002254 000 000 000
2062 002257 000 000 000
2063 002262 000 000 000
2064 002265 000
2065
2066 002266 000 000 000 OVWP4: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2067 002271 000 000 000
2068 002274 000 000 000
2069 002277 000 000 000
2070 002302 000 000 000
2071 002305 000
2072
2073 ;TABLE OF READ SCORE, NEGATIVE OFFSET SCORE
2074
2075 002306 000 000 000 RDND: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2076 002311 000 000 000

```


2077	002314	000	000	000	
2078	002317	000	000	000	
2079	002322	000	000	000	
2080	002325	000			
2081	002326	000	000	000	RDN1: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2082	002331	000	000	000	
2083	002334	000	000	000	
2084	002337	000	000	000	
2085	002342	000	000	000	
2086	002345	000			
2087	002346	000	000	000	RDN2: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2088	002351	000	000	000	
2089	002354	000	000	000	
2090	002357	000	000	000	
2091	002362	000	000	000	
2092	002365	000			
2093	002366	000	000	000	RDN3: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2094	002371	000	000	000	
2095	002374	000	000	000	
2096	002377	000	000	000	
2097	002402	000	000	000	
2098	002405	000			
2099	002406	000	000	000	RDN4: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2100	002411	000	000	000	
2101	002414	000	000	000	
2102	002417	000	000	000	
2103	002422	000	000	000	
2104	002425	000			
2105					
2106					;TABLE OF READ SCORE, POSITIVE OFFSET SCORE
2107					
2108	002426	000	000	000	RDP0: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2109	002431	000	000	000	
2110	002434	000	000	000	
2111	002437	000	000	000	
2112	002442	000	000	000	
2113	002445	000			
2114	002446	000	000	000	RDP1: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2115	002451	000	000	000	
2116	002454	000	000	000	
2117	002457	000	000	000	
2118	002462	000	000	000	
2119	002465	000			
2120	002466	000	000	000	RDP2: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2121	002471	000	000	000	
2122	002474	000	000	000	
2123	002477	000	000	000	
2124	002502	000	000	000	
2125	002505	000			
2126	002506	000	000	000	RDP3: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2127	002511	000	000	000	
2128	002514	000	000	000	
2129	002517	000	000	000	
2130	002522	000	000	000	
2131	002525	000			
2132	002526	000	000	000	RDP4: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

2133 002531 000 000 000
 2134 002534 000 000 000
 2135 002537 000 000 000
 2136 002542 000 000 000
 2137 002545 000

;TABLE OF SELF TEST SCORE

2140
 2141 002546 000 000
 2142 002550 000 000
 2143 002552 000 000
 2144 002554 000 000
 2145 002556 000 000

SELF0: .BYTE 0,0
 SELF1: .BYTE 0,0
 SELF2: .BYTE 0,0
 SELF3: .BYTE 0,0
 SELF4: .BYTE 0,0

;THE START LOGICAL DRIVE # TO WRITE ON EACH CYLINDER OF A BLOCK
 ;16 CYLINDERS,2 BLOCKS,TOTAL 32 CYLINDERS IN ONE ZONE

2146
 2147
 2148
 2149
 2150 002560 000 001 003
 2151 002563 006 012 017
 2152 002566 005 014 004
 2153 002571 015 007 002
 2154 002574 016 013 011
 2155 002577 010

INDST: .BYTE 0,1,3,6,10.,15.,5,12.,4,13.,7,2,14.,11.,9.,8.

2156
 2157
 2158
 2159
 2160
 2161
 2162

;BUFTBL: ;BUFFER ALLOCATION TABLE ENTRY COUNT
 ; .IRP X,<0,1,2,3,4,5,6,7,10,11,12,13,14,15,16,17>
 ; .WORD #ENDPGM+{258.*X} ;BUFFER ADDRESS OF DRIVE X
 ; .ENDM

2163 002600 036506
 2164 002602 036530
 2165 002604 036552
 2166 002606 036574
 2167 002610 036616
 2168 002612 036640
 2169 002614 036662
 2170 002616 036704
 2171 002620 036726
 2172 002622 036750
 2173 002624 036772
 2174 002626 037014
 2175 002630 037036
 2176 002632 037060
 2177 002634 037102
 2178 002636 037124

BLKADR: .WORD DRIV0 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 0
 .WORD DRIV1 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 1
 .WORD DRIV2 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 2
 .WORD DRIV3 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 3
 .WORD DRIV4 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 4
 .WORD DRIV5 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 5
 .WORD DRIV6 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 6
 .WORD DRIV7 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 7
 .WORD DRIV10 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 10
 .WORD DRIV11 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 11
 .WORD DRIV12 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 12
 .WORD DRIV13 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 13
 .WORD DRIV14 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 14
 .WORD DRIV15 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 15
 .WORD DRIV16 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 16
 .WORD DRIV17 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 17

2179
 2180

.EVEN

2181
 2182 002640 000000

OFFCOD: .WORD 0 ;OFFSET CODE TABLE
 ;NUMBER FOR NEGATIVE OFFSET (DIR = OUT)
 ;NUMBER FOR POSITIVE OFFSET (DIR = IN)

2183
 2184
 2185

.EVEN

2186
 2187
 2188 002642 066666

STNDAT: .WORD 066666

K04

2189	002644	155554	.WORD	155554
2190	002646	133331	.WORD	133331
2191	002650	066663	.WORD	066663
2192	002652	155546	.WORD	155546
2193	002654	133315	.WORD	133315
2194	002656	066633	.WORD	066633
2195	002660	155466	.WORD	155466
2196	002662	133155	.WORD	133155
2197	002664	066333	.WORD	066333
2198	002666	154666	.WORD	154666
2199	002670	131555	.WORD	131555
2200	002672	063333	.WORD	063333
2201	002674	146666	.WORD	146666
2202	002676	115555	.WORD	115555
2203	002700	033333	.WORD	033333
2204	002702	040135	PSEUDO: .WORD	040135
2205	002704	177070	.WORD	177070
2206	002706	070414	.WORD	070414
2207	002710	064531	.WORD	064531
2208	002712	174473	.WORD	174473
2209	002714	062422	.WORD	062422
2210	002716	114352	.WORD	114352
2211	002720	036620	.WORD	036620
2212	002722	010031	.WORD	010031
2213	002724	052336	.WORD	052336
2214	002726	017310	.WORD	017310
2215	002730	011347	.WORD	011347
2216	002732	102367	.WORD	102367
2217	002734	152567	.WORD	152567
2218	002736	001246	.WORD	001246
2219	002740	160073	.WORD	160073

2220
2221
2222
2223

;*****

2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279

.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 SITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
 ;*NOTE1: IF SITEMB IS 0 THE ONLY PERTINENT DATA IS (SERRPC).
 ;*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

\$ERRTB:

;ERROR 1

;RH70 INTERRUPT OCCURRED (RMAS = 0)

;ERROR 2

;UNEXPECTED ATTENTION OCCURRED

;ERROR 3

;MASSBUS PARITY ERROR (MCPE=1)

;ERROR 4

;MASSBUS PARITY ERROR (PAR=1)

;ERROR 5

;ADDRESS PLUG BIT CHANGED

;ERROR 6

;RH11 DIDN'T RESPOND TO ADDRESSING

002742

002742 031616
 002744 034244
 002746 034666
 002750 035012

002752 031665
 002754 034253
 002756 034672
 002760 035013

002762 031723
 002764 034335
 002766 034710
 002770 035021

002772 031761
 002774 034366
 002776 034720
 003000 035024

003002 032016
 003004 034253
 003006 034672
 003010 035013

003012 032052
 003014 034430

EM1
 DH1
 DT1
 DF1

EM2
 DH2
 DT2
 DF2

EM3
 DH3
 DT3
 DF3

EM4
 DH4
 DT4
 DF4

EM5
 DH2
 DT2
 DF2

EM6
 DH6

```

2280 003016 034732 DT6
2281 003020 035012 DF1
2282
2283 ;ERROR 7
2284
2285 003022 000000 0
2286 003024 034441 DH7
2287 003026 034736 DT7
2288 003030 000000 0
2289
2290 ;ERROR 10
2291
2292 003032 000000 0
2293 003034 034512 DH10
2294 003036 034752 DT10
2295 003040 000000 0
2296
2297 ;ERROR 11
2298
2299 003042 000000 0
2300 003044 034563 DH11
2301 003046 034766 DT11
2302 003050 000000 0
2303
2304 ;ERROR 12
2305
2306 003052 000000 0
2307 003054 034634 DH12
2308 003056 035002 DT12
2309 003060 000000 0
2310
2311
2312 .SBTTL SETUP AND INITIALIZATION ROUTINE
2313
2314 ; START ADDRESS = 200
2315 ; ADDRESS TO CHANGE RH11 UNIBUS ADDRESS = 204
2316
2317
2318
2319 003062 012737 000400 001332 START: MOV #400,CHGADR ;200 START ADDRESS FLAG
2320 003070 000406 BR START3
2321 003072 012737 177777 001332 START1: MOV #-1,CHGADR ;204 START ADDRESS FLAG
2322 003100 000402 BR START3
2323 003102 005037 001332 START2: CLR CHGADR ;220 START ADDRESS FLAG
2324 ;COMMON BRANCH POINT
2325 003106 000005 START3: RESET ;CLEAR THE BUS
2326 .SBTTL INITIALIZE THE COMMON TAGS
2327 ;;CLEAR THE COMMON TAGS (SCMTAG) AREA
2328 003110 012706 001114 MOV #SCMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
2329 003114 005026 CLR (R6)+ ;;CLEAR MEMORY LOCATION
2330 003116 022706 001154 CMP #SWR,R6 ;;DONE?
2331 003122 001374 BNE -6 ;;LOOP BACK IF NO
2332 003124 012706 001100 MOV #STACK,SP ;;SETUP THE STACK POINTER
2333 ;;INITIALIZE A FEW VECTORS
2334 003130 012737 022362 000020 MOV #SSCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
2335 003136 012737 000340 000022 MOV #340,@#IOTVEC+2 ;;LEVEL 7

```



```

2336 003144 012737 017366 000030      MOV      #ERRVEC, @EMTVEC      ;; EMT VECTOR FOR ERROR ROUTINE
2337 003152 012737 000340 000032      MOV      #340, @EMTVEC+2      ;; LEVEL 7
2338 003160 012737 022642 000034      MOV      @STRAP, @TRAPVEC      ;; TRAP VECTOR FOR TRAP CALLS
2339 003166 012737 000340 000036      MOV      #340, @TRAPVEC+2      ;; LEVEL 7
2340 003174 012737 020452 000024      MOV      @SPWRDN, @PWRVEC      ;; POWER FAILURE VECTOR
2341 003202 012737 000340 000026      MOV      #340, @PWRVEC+2      ;; LEVEL 7
2342 003210 013737 017260 017252      MOV      SENDCT, SEOPCT      ;; SETUP END-OF-PROGRAM COUNTER
2343 003216 005037 001174      CLR      $TIMES                ;; INITIALIZE NUMBER OF ITERATIONS
2344 003222 005037 001176      CLR      $ESCAPE                ;; CLEAR THE ESCAPE ON ERROR ADDRESS
2345 003226 112737 000001 001131      MOV      #1, $ERMAX            ;; ALLOW ONE ERROR PER TEST
2346                                     ;; INITIALIZE THE "1-BIT" TRAP VECTOR. THEN LOAD LOCATION "SRTRN", IN
2347                                     ;; THE "END-OF-PASS" (SEOP) ROUTINE, WITH A "RTI" OR "RTT".
2348 003234 012737 017356 000014      MOV      @SRTRN, @TBITVEC      ;; SET "T" BIT VECTOR TO $SRTRN
2349 003242 012737 000340 000016      MOV      #340, @TBITVEC+2      ;; LEVEL 7
2350 003250 012737 000002 017356      MOV      @RTI, $SRTRN          ;; SET $SRTRN TO A RTI
2351 003256 012737 003304 000010      MOV      #655, @RESVEC        ;; TRY TO DO A RTT
2352 003264 005046      CLR      -(SP)                 ;; DUMMY PS
2353 003266 012746 003274      MOV      #645, -(SP)          ;; AND PC
2354 003272 000006      RTT                          ;; TRY THE RTT
2355 003274 012737 000006 017356 645:    MOV      @RTT, $SRTRN          ;; RTT IS LEGAL--SET $SRTRN TO A RTT
2356 003302 000402      BR      665                    ;;
2357 003304 062706 000010 655:    ADD      #10, SP              ;; RTT ILLEGAL--CLEAN OFF THE STACK
2358 003310 012737 000012 000010 665:    MOV      @RESVEC+2, @RESVEC    ;; RESTORE TRAP CATCHER
2359 003316 005037 017364      CLR      $TBIT                ;; CLEAR "T" BIT SWITCH
2360 003322 012737 003322 001122      MOV      #, $SLPADR            ;; INITIALIZE THE LOOP ADDRESS FOR SCOPE
2361 003330 012737 003330 001124      MOV      #, $LPERR            ;; SETUP THE ERROR LOOP ADDRESS
2362                                     ;; SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
2363                                     ;; EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
2364 003336 013746 000004      MOV      @ERRVEC, -(SP)        ;; SAVE ERROR VECTOR
2365 003342 012737 003376 000004      MOV      #675, @ERRVEC        ;; SET UP ERROR VECTOR
2366 003350 012737 177570 001154      MOV      @DSWR, $SWR           ;; SETUP FOR A HARDWARE SWICH REGISTER
2367 003356 012737 177570 001156      MOV      @DDISP, $DISPLAY      ;; AND A HARDWARE DISPLAY REGISTER
2368 003364 022777 177777 175562      CMP      #-1, @SWR            ;; TRY TO REFERENCE HARDWARE SWR
2369 003372 001012      BNE     695                    ;; BRANCH IF NO TIMEOUT TRAP OCCURRED
2370                                     ;; AND THE HARDWARE SWR IS NOT = -1
2371 003374 000403      BR      685                    ;; BRANCH IF NO TIMEOUT
2372 003376 012716 003404 675:    MOV      #685, (SP)           ;; SET UP FOR TRAP RETURN
2373 003402 000002      RTI                          ;;
2374 003404 012737 000176 001154 685:    MOV      @SWREG, $SWR          ;; POINT TO SOFTWARE SWR
2375 003412 012737 000174 001156      MOV      @DISPREG, $DISPLAY    ;;
2376 003420 012637 000004 695:    MOV      (SP)+, @ERRVEC       ;; RESTORE ERROR VECTOR
2377
2378 003424 005037 001216      CLR      $PASS                ;; CLEAR PASS COUNT
2379 003430 132737 000200 001231      BITB    @APTSIZE, $ENVM        ;; TEST USER SIZE UNDER APT
2380 003436 001403      BEQ     705                    ;; YES, USE NON-APT SWITCH
2381 003440 012737 001232 001154      MOV      @SSWREG, $SWR        ;; NO, USE APT SWITCH REGISTER
2382 003446      705:
2383 003446 012737 000240 000032      MOV      #240, @EMTVEC+2      ;; CHANGE EMT PRIORITY TO 5
2384 003454 012737 000240 000036      MOV      #240, @TRAPVEC+2     ;; CHANGE TRAP PRIORITY TO 5
2385 003462 005227 177777      INC      #-1                  ;; FIRST START ?
2386 003466 001002      BNE     1$                    ;; BR IF NOT
2387 003470 104401 037262      TYPE    $TITLE                ;; NAME AND MANDEC NUMBER
2388 003474 004737 016462 1$:    JSR     PC, $STKINT            ;; TURN ON THE KEYBOARD INTERRUPT
2389 .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
2390 003500 005737 000042      TST     @#42                  ;; ARE WE RUNNING UNDER XXDP/ACT?
2391 003504 001012      BNE     715                    ;; BRANCH IF YES
    
```



```

2392 003506 123727 001230 000001      CMPB      $ENV, #1      ;; ARE WE RUNNING UNDER APT?
2393 003514 001406                      BEQ       71$          ;; BRANCH IF YES
2394 003516 023727 001154 000176      CMP       SWR, #SWREG  ;; SOFTWARE SWITCH REG SELECTED?
2395 003524 001005                      BNE      72$          ;; BRANCH IF NO
2396 003526 104406                      GTSWR                      ;; GET SOFT-SWR SETTINGS
2397 003530 000403                      BR       72$
2398 003532 112737 000001 001150 71$:      MOVB     #1, SAUTOB   ;; SET AUTO-MODE INDICATOR
2399 003540                      72$:
2400 003540 013737 001274 023076      MOV      $RMADR, RMADR ;; RH11 ADDRESS
2401 003546 013737 001276 023100      MOV      $RMVEC, RMVEC ;; RH11 VECTOR ADDRESS
2402 003554 012705 001776                      MOV      #ASNLST, RS  ;; START OF AREA TO CLEAR
2403 003560 005025                      2$:      CLR      (R5)+
2404 003562 022705 002560                      3$:      CMP      #INDST, RS   ;; LOOK FOR END OF CLEAR AREA
2405 003566 001374                      BNE      3$          ;; BR IF NOT FINISHED
2406 003570 012706 001100                      MOV      #STACK, SP  ;; SETUP THE STACK POINTER
2407 003574 005037 177776                      CLR      PS          ;; CLEAR THE PROCESSOR STATUS WORD
2408 003600 013737 001316 001350      MOV      HZ, SIXTEE  ;; 1/60 TH OR 1/50 TH SECOND COUNTER VALUE
2409 003606 005037 001342                      CLR      HOUR        ;; CLEAR THE HOUR'S COUNTER
2410 003612 005037 001344                      CLR      MINUTE      ;; CLEAR THE MINUTE'S COUNTER
2411 003616 005037 001346                      CLR      SECOND      ;; CLEAR THE SECOND'S COUNTER
2412 003622 005037 001334                      CLR      CFLAG       ;; CLEAR THE 'CONTROL C' FLAG
2413
2414
2415                      ; ROUTINE TO DETERMINE BUFFER AREA SIZE
2416 003626 005227 177777      SIZMEM:  INC      #-1      ;; SEE IF TIME TO SIZE MEMORY
2417 003632 001002                      BNE      1$          ;; BR IF NOT
2418 003634 004737 031216                      JSR      PC, $SIZE   ;; SEE HOW MUCH MEMORY ON SYSTEM
2419 003640 013737 031312 001330 1$:      MOV      $LSTAD, LSTAD ;; SAVE THE LAST ADDRESS
2420 003646 023727 001330 160000      CMP      LSTAD, #160000 ;; OVER 28K ?
2421 003654 101403                      BLOS    2$          ;; NO, THEN DON'T SET THE NEW LIMIT
2422 003656 012737 160000 001330      MOV      #160000, LSTAD ;; SET NEW LIMIT
2423 003664 162737 005670 001330 2$:      SUB      #1500.*2, LSTAD ;; SAVE XXDP LOADER AND ABSOLUTE LOADER
2424
2425                      ;
2426                      ;   SET UP THE OTHER SYSTEM DEVICES THAT
2427                      ;   THE PROGRAM WILL USE
2428
2429 003672 004737 014740      SETVEC:  JSR      PC, CKCLK ;; START THE CLOCK
2430 003676 012737 177777 023036      MOV      #-1, SAVEFG  ;; SET THE SAVE REGISTERS FLAG
2431
2432                      ; SETUP IF 'XXDP' OR 'ACT11' OPERATION
2433
2434 003704 005001      MONTR:   CLR      R1      ;; DRIVE #
2435 003706 005002                      CLR      R2          ;; AVAIL TABLE INDEX
2436 003710 005003                      CLR      R3          ;; DRIVE# X 2
2437 003712 016300 002600 1$:      MOV      BLKADR(R3), R0 ;; LOAD DPB ADDRESS
2438 003716 004737 015426                      JSR      PC, CLRDPB  ;; CLEAR DPB BLOCK
2439 003722 022322      2$:      CMP      (R3)+, (R2)+  ;; INCREMENT INDEX
2440 003724 005201                      INC      R1          ;; NEXT DRIVE
2441 003726 022701 000007      CMP      #7, R1      ;; ALL DRIVE ASSIGN ?
2442 003732 002367                      BGE      1$          ;; NO
2443
2444                      ;
2445                      ;   ASSIGN LOGICAL DRIVES TO BE TEST IN THE PASS1 AND PASS 2
2446                      ;   THREE WORDS ARE USED IN THE BIT MAPS:
2447                      ;   ASNLST:  SPECIFIES THE LOGICAL DRIVES ASSIGNED
2448                      ;   ASSGN1:  SPECIFIES THE LOGICAL DRIVES WILL BE TESTED IN PASS1.

```

2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503

```

:      ASSGN2:  SPECIFIES THE LOGICAL DRIVES WILL BE TESTED IN PASS2.
:
:      EACH LOGICAL DRIVE HAS A HISTORY FILE LABELED DIRV'Z (Z=0 TO 17)
:      THE LOCATIONS LABELED $SYSNM AND $PHYRD IN THE HISTORY FILE
:      STORE THE SYSTEM NAME (A TO H) AND PHYSICAL DRIVE NUMBER
:      (0 TO 7).
:      THE SUB-SYSTEM ADDRESS AND INTERRUPT VECTOR ARE STOREED
:      IN THE TABLE LABELED "SYSADR:".
:
:      THE LOCATIONS SYSADR AND SYSADR+2 FOR SUB-SYSTEM A, SYSADR+4
:      AND SYSADR+6 FOR SUBSYSTEM B , ETC, THE FIRST WORD
:      IS THE SUB SYSTEM ADDRESS WHILE THE SECOND WORD IS THE VECTOR
:
:      THE LOGICAL DRIVES ARE ASSIGNED FROM CONSOLE KEYBOARD.
    
```

```

MOD00:  MOV      #ASNLST,RO      ;ADDRESS OF 1ST BIT MAPS IN RO
        MOV      #INDST,R1      ;LAST ADDRESS TO CLEAR
1$:     CLR      (RO)+          ;CLEAR CURRENT POINTED ADDRESS
        CMP      R1,RO          ;ALL DONE ?
        BHI     1$             ;NO, THEN BRANCH BACK
        MOV      #'A,$CDW2      ;TEMP STORATE OF SYS NAME

MOD21:  CLR      DRIVE          ;TEM STORAGE OF PHYSICAL DRIVE BIT MAP
        TYPE    ,SCRLF          ;SUB-SYSTEM
        TYPE    ,MESG1          ;A TO H (ONE OF THEM)
        TYPE    ,SCDW2
        TYPE    ,LINSF
        TYPE    ,MESG2
        TYPE    ,COLON
        RDLIN
        MOV      (SP)+,R1       ;READ IN THE DRIVE NUMBERS
        TSTB    (R1)           ;GET THE INPUT LINE ADDRESS
1$:     BEQ      2$             ;END OF STRING ?
        JSR     R5,CK.OCT      ;CHECK THE DIGIT MUST 0 TO 7, RETURN VALUE IN R2
        BR     MOD21           ;INCORRECT DRIVE NUMBER, ENTER AGAIN
        BISB    ATABIT(R2),DRIVE ;SET THE ASSIGNED PHYSICAL DRIVE BIT
        ;R2 THE DRIVE NUMBER FROM CK.OCT RT.
        INC     R1             ;NEXT BYTE OF INPUT LINE
        TSTB    (R1)           ;END OF STRING ?
        BEQ     2$             ;YES
        CMPB   #' (R1)+        ;MUST BE A
        BNE    MOD21          ;ENTER AGAIN, IF NOT
        BR     1$             ;LOCATE NEXT DRIVE
2$:     TST     CHGADR          ;START AT 200 ?
        BGT     MOD22          ;BRANCH IF SO
        MOV     $CDW2,R1       ;SYS NAME ASCII FROM A TO H
        BIC    #177760,R1     ;LEFT ONLY 4 BITS
        DEC     R1             ;ADJUST INDEX VALUE
        ASL     R1             ;2 WORD INDEX VALUE
        ASL     R1
        ADD     #SYSADR,R1     ;SYS ADDRESS TABLE ADDRESS
    
```



```

2504 004106 010137 001270      MOV      R1,SCDW1      ; SYS ADDRESS TABLE'S ENTRY TO CDW1
2505 004112 000400      3$: BR      MOD23      ;
2506
2507 004114 005737 001222      MOD23: TST      DRIVE      ;CHECK IF ANY PHYSICAL DRIVE(S) ASSIGNED
2508 004120 001416      BEQ      2$           ;BRANCH IF NONE
2509 004122 013700 001270      1$: MOV      SCDW1,R0      ;SYS ADDRESS TABLE ENTRY
2510 004126 011037 001274      MOV      (R0),SRMADR      ; SYS ADDRESS
2511 004132 016037 000002 001276      MOV      2(R0),SRMVEC      ; SYS VECTOR
2512 004140 004737 031314      JSR      PC,BUSADR      ;CHECK THE ADDRESS WITH THE OPERATOR
2513 004144 013710 001274      MOV      SRMADR,(R0)      ;NEW RH11/70 ADDRESS INTO TABLE
2514 004150 013760 001276 000002      MOV      SRMVEC,2(R0)      ;NEW VECTOR OF RH11/70 INTO TABLE
2515 004156 000407      2$: BR      MOD11      ;BRANCH TO NEXT MODULE
2516
2517
2518 004160      MOD22:
2519 004160 013737 001274 002004 3$: MOV      SRMADR,SYSADR ;LOAD THE SYSTEM ADDRESS TABLE
2520 004166 013737 001276 002006 4$: MOV      SRMVEC,SYSADR+2 ;LOAD THE VECTOR
2521 004174 000400      BR      MOD11
2522
2523
2524
2525
2526
2527      ;
2528      SCDW2-- ASCII NAME OF SUB SYSTEM (A TO H)
2529      SCDW1--ENTRY TO THE SYS ADDRESS TABLE
2530      DRIVE--PHYSICAL DRIVES TO BE ASSIGNED
2531      THIS SECTION OF CODING USES THE ABOVE PARAMETERS
2532      TO SET UP THE BIT MAP OF ASNLST,ASSGN1,ASSGN2
2533
2534
2535
2536
2537 004176 005737 001222      MOD11: TST      DRIVE      ;ANY DRIVE ASSIGN ?
2538 004202 001002      BNE      MOD30      ;BRANCH IF ANY
2539 004204 000137 004360      JMP      MOD12      ;BRANCH, IF NONE
2540 004210 022737 177777 001776  MOD30: CMP      #-1,ASNLST      ; ALL 16 LOGICAL DRIVES HAS BEEN ASSIGNED ?
2541 004216 001457      BEQ      5$           ; YES, THEN DON'T ASSIGN NEW DRIVES
2542 004220 013700 001776      MOV      ASNLST,R0      ; FOUND THE AVAILABLE LOGICAL DRIVE LOCATION
2543 004224 012701 000001      MOV      #1,R1      ; START FROM LOGICAL DRIVE 0
2544 004230 005002      CLR      R2           ; INDEX VALUE
2545 004232 030100      1$: BIT      R1,R0      ; IS THE LOGICAL DRIVE AVAILABLE ?
2546 004234 001406      BEQ      2$           ; YES
2547 004236 000241      CLC
2548 004240 006101      ROL      R1           ; NEXT LOGICAL DRIVE
2549 004242 103445      BCS      5$           ; BRANCH IF NONE IS AVAILABLE
2550 004244 062702 000002      ADD      #2,R2      ; INCREMENT INDEX VALUE
2551 004250 000770      BR      1$           ; LOCATE NEXT LOGICAL DRIVE
2552 004252 016204 002600 000014 2$: MOV      BLKADR(R2),R4 ; GET THE LOGICAL DRIVE'S HISTORY FILE
2553 004256 113764 001272      MOV      SCDW2,$SYSNM(R4) ; LOAD THE ASCII SYS NAME
2554 004264 050137 001776      BIS      R1,ASNLST      ; SET LOGICAL DRIVE ASSIGN BIT
2555 004270 050137 002000      BIS      R1,ASSGN1      ; SET PASS 1 BIT
2556 004274 050137 002002      BIS      R1,ASSGN2      ; SET PASS 2 BIT
2557 004300 013700 001222      MOV      DRIVE,R0      ; LOAD THE ASCII PHYSICAL DRIVE #
2558 004304 012701 000001      MOV      #1,R1      ; START FROM DRIVE 0
2559 004310 005002      CLR      R2           ; DRIVE #
2559 004312 030100      3$: BIT      R1,R0      ; IS THE PHYSICAL DRIVE ASSIGNED
    
```


E05

MAINDEC-11-DZRM1 AO/00, RMO3 DRIVE COMPATIBILITY TEST MACY11 30(1046) 21-JUL-77 16:51 PAGE 56
 DZRM1A.P11 21-JUL-77 15:44 GET VALUE FOR SOFTWARE SWITCH REGISTER

SEQ 0055

2560	004314	001005			BNE	4\$; YES, THEN BRANCH
2561	004316	000241			CLC			; LOCATE THE NEXT DRIVE
2562	004320	006101			ROL	R1		; SHIFT ONE BIT LEFT
2563	004322	103415			BCS	5\$; BRANCH IF NO MORE DRIVES
2564	004324	005202			INC	R2		; PHYSICAL DRIVE NUMBER
2565	004326	000771			BR	3\$; LOOPING BACK
2566	004330	110214		4\$:	MOVB	R2, (R4)		; LOAD THE PHYSICAL DRIVE # INTO HISTORY FILE
2567	004332	062702	000060		ADD	#'0, R2		; ASCII CODE OF DRIVE #
2568	004336	110264	000015		MOVB	R2, \$PHYDR(R4)		; PHYSICAL DRIVE NUMBER
2569	004342	112764	000011	000016	MOVB	#HT, \$GAP(R4)		; MAKE UP FOR SCORE TYPE
2570	004350	040137	001222		BIC	R1, DRIVE		; RESET THE ASSIGNED DRIVE BIT
2571	004354	001315			BNE	MOD30		; BRANCH IF NOT ALL DONE
2572	004356	000400		5\$:	BR	MOD12		; CHECK OTHER SUB SYSTEM
2573								
2574								
2575	004360	005737	001332		MOD12:	TST	CHGADR ;200 START ?	
2576	004364	003066			BGT	6\$; YES, THEN EXIT
2577	004366	022737	177777	001776	1\$:	CMP	#-1, ASNLST	; FULL HOUSE
2578	004374	001462			BEQ	6\$; YES, THEN EXIT
2579	004376	104401	035737		2\$:	TYPE	, MSG5	; WILL TEST
2580	004402	104401	035041			TYPE	, MSG2	; DRIVES
2581	004406	005003			CLR	R3		; INDEX TO LOGICAL DRIVE HISTORY FILE
2582	004410	012704	000001		MOV	#1, R4		; BIT MAP OF ASNLST
2583	004414	030437	001776		3\$:	BIT	R4, ASNLST	; ASSIGNED LOGICAL DRIVE ?
2584	004420	001417			BEQ	5\$; NO
2585	004422	016305	002600		MOV	BLKADR(R3), R5		; LOAD THE HISTORY FILE ADDRESS
2586	004426	126537	000014	001272	CMPB	\$SYSNM(R5), \$CDW2		; ON THE SAME SYSTEM ?
2587	004434	001003			BNE	4\$; NO
2588	004436	111546			MOVB	(R5), -(SP)		; TYPE THE PHYSICAL DRIVE #
2589	004440	104403			TYPOS			
2590	004442	002			.BYTE	2		
2591	004443	000			.BYTE	0		
2592	004444	062703	000002		4\$:	ADD	#2, R3	; INCREMENT TO NEXT LOGICAL DRIVE
2593	004450	000241			CLC			
2594	004452	006104			ROL	R4		; BIT MAP OF THE NEXT LOGICAL DRIVE
2595	004454	103401			BCS	5\$; BRANCH IF ALL LOGICAL DRIVE'S CHECKED
2596	004456	000756			BR	3\$; BRANCH BACK
2597	004460	104401	036121		5\$:	TYPE	, LINSF	
2598	004464	104401	035752			TYPE	, MSG6	; ON
2599	004470	104401	035030			TYPE	, MSG1	; SUB SYSTEM
2600	004474	104401	001272			TYPE	, \$CDW2	; A TO H
2601								
2602	004500	005237	001272		INC	\$CDW2		; CHECK NEXT SUB SYSTEM
2603	004504	122737	000110	001272	CMPB	#'H, \$CDW2		; ALL EIGHT SUB SYSTEMS CHECKED?
2604	004512	103413			BLO	6\$; YES
2605	004514	104401	001205		TYPE	, \$CRLF		
2606	004520	104401	035756		TYPE	, MSG7		; ASK FOR OTHER SUB SYSTEM
2607	004524	104411			RDLIN			; READ IN A LINE
2608	004526	012605			MOV	(SP)+, R5		; CHECK THE FIRST CHARACTER
2609	004530	122715	000131		CMPB	#'Y, (R5)		; ENTER Y ?
2610	004534	001002			BNE	6\$; BRANCH IF NOT
2611	004536	000137	003760		JMP	MOD21		; SET UP OTHER SUB SYSTEM
2612	004542	005737	001332		6\$:	TST	CHGADR	; START AT 220
2613	004546	001402			BEQ	7\$; YES, EXECUTE PASS2 ONLY
2614	004550	000137	004564		JMP	XPASS1		; TO PASS1
2615	004554	005037	002000		7\$:	CLR	ASSGN1	; CLEAR THE PASS1 BIT MAP

F05

2616 004560 000137 006772

JMP XPASS2 ;BRANCH TO PASS2

2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671

```

: PASS ONE , THE VARIABLES ARE ASSIGNED AS FOLLOWS:
: SCDW1 : ADDRESS OF THE CURRENT LOGICAL DRIVE HISTORY FILE
: SCDW2 : SYSTEM NAME A THROUGH H
: SDEVN : CURRENT LOGICAL DRIVE #
: ASNLST : ASSIGNED LOGICAL DRIVES
: ASSGN1 : ASSIGNED LOGICAL DRIVES IN THIS PASS
: RO : ADDRESS OF DPB BLOCK FEEDED INTO DRIVER-HANDLER
: R1 : TEMP STORAGE OF ADDRESS OF THE LOGICAL BLOCK
  
```

```

XPASS1: TST ASSGN1 ;ANY DRIVE ASSIGNED FOR PASS1
        BNE IS ;BRANCH IF THEY ARE
        JMP ENDX1 ;END OF PASS 1
IS:     CLR SDEVN ;INDEX OF LOGICAL BLOCK
        MOV BLKADR,SCDW1 ;ADDRESS OF LOGICAL BLOCK 0
        MOVB #'A,SCDW2 ;SYS NAME STARTS FROM A
        MOV #IDLEX,ARMVEC ;RESET ALL INTERRUPT VERTOR
        CLR ARMVEC+2 ;CLEAR THE INTERRUPT LEVEL
        MOV SYSADR,RMADR ;LOAD SYS-TEM A INTO
        MOV SYSADR+2,RMVEC ;DIRVER-HANDLER
        MOV SCDW1,R1 ;R1=ADDRESS OF LOGICAL BLOCK 1
        MOV #IDLEX,ARMVEC ;RESET ALL INTERRUPT VECTOR
        CLR ARMVEC+2 ;CLEAR THE INTERRUPT LEVEL
        TYPE ,SCRLF
        TYPE ,MSG3
        TYPE ,SCRLF
        TYPE ,MSG4
        TYPE ,SCDW2
        MOVB (R1),DRIVE ;LOAD THE PHYSICAL DRIVE #
        MOVB (R1),-(SP) ;TYPE THE DRIVE #
        .BYTE 2
        .BYTE 0
        TYPE ,LINSF
        TYPE ,MSG8
        TYPE ,SCRLF
        TYPE ,MSG9
        TYPE ,LINSF

2S:     RDLIN ;CHECK IF O.P. READY
        MOV (SP)+,R2 ;LOCATE THE INPUT LINE
        CMPB #'R,(R2)+ ;FIRST CHARACTER IS A "R" ?
        BNE 2S ;BRANCH IF NOT
        TSTB (R2) ;FOLLOW BY <CR> ?
        BNE 2S ;BRANCH IF NOT
  
```

004564 005737 002000
 004570 001002
 004572 000137 005432
 004576 005037 001266
 004602 013737 002600 001270
 004610 112737 000101 001272
 004616 012777 013256 016254
 004624 005077 016252
 004630 013737 002004 023076
 004636 013737 002006 023100
 004644 013701 001270
 004650 012777 013256 016222
 004656 005077 016220
 004662 104401 001205
 004666 104401 035053
 004672 104401 001205
 004676 104401 035101
 004702 104401 001272
 004706 111137 001222
 004712 111146
 004714 104403
 004716 002
 004717 000
 004720 104401 036121
 004724 104401 035127
 004730 104401 001205
 004734 104401 035142
 004740 104401 036121
 104411
 012602
 122722 000122
 001373
 105712
 001371

G05

MAINDEC-11-DZRM1 AO/00, RMO3 DRIVE COMPATIBILITY TEST MACY11 30(1046) 21-JUL-77 16:51 PAGE 58
 DZRM1A.P11 21-JUL-77 15:44 GET VALUE FOR SOFTWARE SWITCH REGISTER

SEQ 0057

```

2672 004762 004737 023114          JSR    PC,RMINIT      ;INITIALIZE THE SUB SYSTEM
2673 004766 012737 177777 023036  MOV    #-1,SAVEFG    ;SAVE ALL RH/RM REGISTERS
2674 004774 012737 177777 023040  MOV    #-1,SEEKFG    ;DON'T DO IMPLY SEEK
2675 005002 013700 001222          MOV    DRIVE,RO      ;RO=PHYSICAL DRIVE # OF THE SUB SYSTEM
2676 005006 105760 022750          TSTB   DRVSTA(RO)    ;DIRVE EXIST AND ON LINE ?
2677 005012 003452          BLE    5$           ;BRANCH IF NOT
2678 005014 105760 022760          TSTB   DRVTP(RO)    ;DRIVE IS AN RMO3 ?
2679 005020 003447          BLE    5$           ;BRANCH IF NOT
2680                                ;SET UP TO RETRIEVE THE BAD SPOT FILE
2681                                ;FROM THE LAST TRACK
2682 005022 012700 037146          MOV    #FMDPB,RO     ;DPB ADDRESS
2683 005026 113710 001222          MOVVB  DRIVE,(RO)    ;LOAD THE DRIVE NUMBER
2684 005032 012760 037262 000006  MOV    #ENDPGM,$BUF(RO) ;LOAD BUFFER ADDRESS
2685 005040 012760 037166 000014  MOV    #RM.REG,14(RO) ;AREA TO SAVE ALL RH/RMO3 REG'S
2686 005046 012760 177400 000004  MOV    #-256,$WRDM(RO) ;WORD COUNT (NEG)
2687 005054 013760 001372 000012  MOV    CYLINT,$CYL(RO) ;CYLINDER 822.
2688 005062 113760 001370 000011  MOVVB  TRKLMT,$TRK(RO) ;TRACK ADDRESS 4.
2689 005070 105060 000010          CLRB   $SEC(RO)     ;SEC 0
2690 005074 112760 000171 000002  MOVVB  #RDDAT,$COMND(RO) ;READ DATA COMMAND
2691
2692 005102 004037 023620          3$:   JSR    RO,RMO3    ;CALL THE DRIVER-HANDLER
2693 005106 037146          FMDPB  ;PARAMETER ADDRESS
2694 005110 000774          BR     3$           ;LOOPING IF QUEUE IS NOT SUCCESSFUL
2695 005112 005737 037164          4$:   TST    FMDPB+16  ;COMMAND DONE ?
2696 005116 001775          BEQ    4$           ;BRANCH IF NOT
2697 005120 100013          BPL    6$           ;BRANCH IF DONE, WITHOUT ERROR
2698 005122 062737 000002 037156  ADD    #2,FMDPB+10  ;TRY NEXT SECTOR (0,2,4,6,8)
2699 005130 122737 000010 037156  CMPB   #8.,FMDPB+10 ;ALL FIVE SECTORS CHECKED ?
2700 005136 101361          BHI    3$           ;NO, THEN TRY AGAIN
2701
2702 005140 104401 035201          5$:   TYPE   ,MSG10    ;DRIVE IS NOT READY
2703 005144 000137 005432          JMP    ENDX1        ;STOP THE TEST
2704
2705
2706                                ;BAD SPOT FILE IS RETRIVED, IS STORED
2707                                ;FROM ENDPGM+4 TO ENDPGM+256.
2708                                ;FIRST WORD CYLINDER # SECOND WORD
2709                                ;TRK AND SEC NUMBERS, FILE IS TERMINATED
2710                                ;BY A -1 IN THE CYLINDER NUMBER
2711
2712
2713 005150 012704 037272          6$:   MOV    #ENDPGM+10,R4 ;R4 ADDRESS OF BAD SPOT FILE
2714 005154 022714 177777          7$:   CMP    #-1,(R4)    ;END OF BAD SPOT FILE ?
2715 005160 001411          BEQ    8$           ;YES
2716 005162 022704 040262          CMP    #ENDPGM+1000,R4 ;END OF BAD SPOT FILE ?
2717 005166 101406          BLOS  8$           ;BRANCH IF IT IS
2718 005170 004537 005264          JSR    R5,SPOTX     ;CHECK THE CYLINDER POINTED BY (R4)
2719 005174 000415          BR     10$          ;BRANCH IF BAD SPOT IN THE TEST ZONES
2720 005176 062704 000004          ADD    #4,R4        ;NEXT BAD SPOT ADDRESS
2721 005202 000764          BR     7$           ;LOOPING BACK
2722
2723 005204 032777 000200 173742  8$:   BIT    #BIT7,$SWR   ;SWITCH 7 SET ?
2724 005212 001404          BEQ    9$           ;BRANCH IF NOT SET
2725 005214 012700 037146          MOV    #FMDPB,RO     ;DPB ADDRESS
2726 005220 004737 014406          JSR    PC,PRTBAD    ;PRINT THE BAD SPOT FILE
2727 005224 000137 005450          9$:   JMP    TST1        ;PACK IS ACCEPTABLE

```


H05

```

2728 005230 104401 035245      10$: TYPE      ,MESG11      ;PACK NOT ACCEPTABLE
2729 005234 104401 001205      TYPE      ,SCRLF
2730 005240 032777 000200 173706  BIT      #BIT7,2SWR    ;SWITCH 7 SET ?
2731 005246 001404      BEQ      11$        ;BRANCH IF NOT SET
2732 005250 012700 037146      MOV      #FMTDPB,R0  ;DPB ADDRESS
2733 005254 004737 014406      JSR      PC,PRTBAD   ;TYPE THE BAD SPOT FILE
2734 005260 000137 004564      11$: JMP      XPASS1   ;RESTART PASS 1
2735
2736
2737      ;
2738      SUBROUTINE SPOTX
2739      CHECK THE CYLINDER POINTED BY (R4) IS IN THE TESTING ZONES
2740      (0-15,128-143,256-271,384-399,512-527,640-655,768-783
2741      112-127,240-255,368-383,496-511,624-639,752-767
2742      17-32,145-160,273-288,401-416,529-544,657-672,785-800
2743      AND 620 )
2744      CALLING SEQ:
2745      JSR      R5,SPOTX ;R4=POINT TO CYLINDER NUMBER
2746      RET1
2747      RET2 ;NORMAL RET1
2748
2749 005264 010146      SPOTX: MOV      R1,-(SP)  ;SAVE R1 THROUGH R3
2750 005266 010246      MOV      R2,-(SP)
2751 005270 010346      MOV      R3,-(SP)
2752
2753 005272 005003      CLR      R3 ;ERROR FLAG
2754 005274 005046      CLR      -(SP) ;DUMMY PAIR
2755 005276 005046      CLR      -(SP) ;DUMMY PAIR
2756 005300 005046      CLR      -(SP) ;ZONE STARTING ADDRESS
2757 005302 012746 000007      MOV      #7,-(SP) ;SEGMENT NUMBER
2758 005306 012746 000021      MOV      #17,-(SP) ;ZONE STARTING ADDRESS
2759 005312 012746 000007      MOV      #7,-(SP) ;SEGMENT NUMBER
2760 005316 012701 000160      MOV      #12.,R1 ;R1=ZONE STARTING ADDRESS
2761 005322 012702 000006      MOV      #6,R2 ;R2=SEGMENT NUMBER
2762 005326 021401      1$: CMP      (R4),R1 ;CYL IN THE ZONE ?
2763 005330 103410      BLO      3$ ;BRANCH IF NOT
2764 005332 062701 000017      ADD      #15.,R1 ;CHECK WITH THE UPPER BOND
2765 005336 021401      CMP      (R4),R1 ;CYL IN THE ZONE ?
2766 005340 101002      BHI      2$ ;BRANCH IF NOT
2767 005342 052703 000002      BIS      #BIT1,R3 ;SET THE ERROR FLAG
2768
2769 005346 162701 000017      2$: SUB      #15.,R1 ;RESTORE TO THE LOWER BOND
2770 005352 005302      3$: DEC      R2 ;DECREMENT THE SEGMENT COUNT
2771 005354 001403      BEQ      4$ ;ALL SEGMENT CHECKED ?
2772 005356 062701 000200      ADD      #128.,R1 ;ADJUST ZONE STARTING ADDRESS
2773 005362 000761      BR      1$ ;LOOPING BACK UNTIL ALL SEGMENTS ARE CHECKED
2774
2775 005364 012602      4$: MOV      (SP)+,R2 ;POP THE NEXT SET OF ZONE PARAMETERS
2776 005366 012601      MOV      (SP)+,R1
2777 005370 005702      TST      R2 ;DUMMY PAIR
2778 005372 001355      BNE      1$ ;BRANCH IF NOT
2779 005374 005701      TST      R1 ;DUMMY PAIR
2780 005376 001353      BNE      1$ ;BRANCH IF NOT
2781
2782 005400 021427 001154      5$: CMP      (R4),#620. ;ON CYLDER 620 ?
2783 005404 001002      BNE      6$ ;NO
  
```

2784 005406 052703 000002
2785 005412 005703
2786 005414 001002
2787 005416 062705 000002
2788 005422 012603
2789 005424 012602
2790 005426 012601
2791 005430 000305

6S: BIS #BIT1,R3 ;SET ERROR FLAG
TST R3 ;ANY ERROR ?
BNE 7S ;YES
7S: ADD #2,R5 ;ADJUST FOR NORMAL RETURN
MOV (SP)+,R3 ;RESTORE R3 THROUGH R1
MOV (SP)+,R2
MOV (SP)+,R1
RTS R5 ;EXIT

2792
2793
2794 005432 104401 001205
2795 005436 104401 036424
2796 005442 000000
2797 005444 000137 000200
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823

ENDX1: TYPE ,SCRLF ;
TYPE ,MSG21 ;DRIVE NOT ONLINE OR NOT ASSIGNED
HALT ;TEMP HALT FOR DEBUG
JMP @#200 ;RESTART IF DESIRED

THE FOIIOWING CODING FOR TEST 1 THROUGH TEST 4
PARAMETER IN TST 1
SCDW1 : ADDRESS OF LOGICAL DRIVE BLOCK
DRIVE : PHYSICAL DRIVE #
SDEVN : LOGICAL DRIVE # 0-17
ASSGN1 : ASSIGN LOGICAL DRIVE BIT MAP
ASNLST : ASSIGNED LOGICAL DRIVE MAP INDICATOR
SCDW2: SYS-NAME
SCDW2,DRIVE ARE ONLY CHANGED IN TST1 DURING PASS 1
SDEVN,ASSGN1,SCDW1 ARE CHANGED BY THE TEST 4

TST 1:DIRECT OPERATOR TO MOUNT AND LOAD PACKS

2824 005450 000004
2825 005452 012737 000001 001174
2826 005460 012737 000001 001336
2827 005466 012706 001100
2828 005472 023737 001270 002600
2829 005500 001524
2830 005502 013701 001270
2831 005506 126137 000014 001272
2832 005514 001426
2833 005516 116137 000014 001272
2834 005524 012777 013256 015346
2835 005532 005077 015344
2836 005536 104401 001205
2837 005542 104401 035053
2838 005546 104401 001205
2839 005552 104401 035752

TST1: SCOPE
MOV #1,\$TIMES ;DO 1 ITERATION
MOV #1,TSTNM ;LOAD THE TEST NUMBER
MOV #STACK,SP ;INITIALIZE THE STACK POINT
CMP \$CDW1,BLKADR ;LOGICAL DRIVE 0
BEQ 3S ;THEN EXIT
MOV \$CDW1,R1 ;R1=LOGICAL DRIVE BLOCK ADDRESS
CMPB \$\$SYSNM(R1),\$CDW2 ;STILL ON THE SAME SYSTEM ?
BEQ 1S ;THEN EXIT
MOVB \$\$SYSNM(R1),\$CDW2 ;LOAD THE NEW SUB SYSTEM NAME
MOV #IDLEX,@RMVEC ;RESET THE INTERRUPT VECTOR
CLR @RMVEC+2 ;CLEAR THE INTERRUPT LEVEL
TYPE ,SCRLF
TYPE ,MSG3 ;** STARTING PASS 1 **
TYPE ,SCRLF
TYPE ,MSG6 ;ON--

J05

```

2840 005556 104401 035030          TYPE      ,MSG1          ;SUB-SYSTEM
2841 005562 104401 001272          TYPE      ,SCDW2          ;A TO H
2842 005566 104401 036121          TYPE      ,LINSF          ;
2843 005572 111137 001222          1$:      MOV      (R1),DRIVE ;LOAD THE PHYSICAL DRIVE #
2844 005576 104401 001205          TYPE      ,SCRLF          ;
2845 005602 104401 035101          TYPE      ,MSG4          ;MOUNT PACK ON THE DRIVE
2846 005606 104401 001272          TYPE      ,SCDW2          ;SYS-NAME
2847 005612 113746 001222          MOV      DRIVE,-(SP) ;THE PHYSICAL DRIVE #
2848 005616 104403          TYPOS
2849 005620          .BYTE    2
2850 005621          .BYTE    0
2851 005622 104401 036121          TYPE      ,LINSF          ;
2852 005626 104401 035127          TYPE      ,MSG8          ;AND LOAD
2853 005632 104401 001205          TYPE      ,SCRLF          ;
2854 005636 104401 035142          TYPE      ,MSG9          ;
2855 005642 104411          2$:      RDLIN
2856 005644 012605          MOV      (SP)+,R5      ;LOCATE THE READIN LINE
2857 005646 122725 000122          CMPB    #'R,(R5)+    ;CHECK IF O.P. READY ?
2858 005652 001373          BNE     2$           ;IF NOT
2859 005654 105715          TSTB   (R5)         ;NOT CORRECT INPUT LINE FORMAT
2860 005656 001371          BNE     2$           ;BRANCH IF NOT
2861 005660 113701 001272          MOV      $CDW2,R1    ;LOCATE THE SYSTEM ADDRESS TABLE
2862 005664 042701 177760          BIC     #177760,R1   ;LEFT ON 4 BITS
2863 005670 005301          DEC     R1           ;ADJUST THE INDEX VALUE
2864 005672 006301          ASL     R1           ;FOUR WORD INDEX VALUE
2865 005674 006301          ASL     R1
2866 005676 016137 002004 023076          MOV      SYSADR(R1),RMDR ;LOAD THE SYSTEM ADDRESS
2867 005704 016137 002006 023100          MOV      SYSADR+2(R1),RMVEC ;LOAD THE SYSTEM INTERRUPT VECTOR
2868 005712 004737 023114          JSR     PC,RINIT     ;INITIALIZE THE SYSTEM
2869 005716 012737 177777 023036          MOV      #-1,SAVEFG ;SAVE ALL RH/RM REGISTER
2870 005724 012737 177777 023040          MOV      #-1,SEEKFG ;DON'T DO ANY IMPLY SEEK
2871 005732 013700 001222          MOV      DRIVE,RD   ;RD=PHYSICAL DRIVE #
2872 005736 105760 022750          TSTB   DRVSTA(RD)   ;ON-LINE ?
2873 005742 003404          BLE     4$           ;BRANCH IF NOT
2874 005744 105760 022760          TSTB   DRVTP(RD)   ;AN RMO3
2875 005750 003401          BLE     4$           ;BRANCH IF NOT AN RMO3
2876 005752 000403          3$:      BR      TST2    ;ALL DONE PROCEED TO TEST 2
2877 005754 104401 035201          4$:      TYPE      ,MSG10   ;DRIVE NOT READY
2878 005760 000633          BR      TST1        ;TRY AGAIN

```

```

;TEST 2
;BASIC READ AND WRITE TEST
;ALL LOGICAL DRIVE ACCESS CYLINDER 620
;AND SECTOR ADDRESS IS COORESPONDING TO THE LOGICAL DRIVE #
;EACH LOGICAL DRIVE PERFORM WRITE AND WRITE CHECK ON ALL FIVE SURFACES(TRK0 - TRK4)

```

```

;*****
TST2:  SCOPE
      MOV      #1,$TIMES ;DO 1 ITERATION
      MOV      #2,TSTNM  ;LOAD TEST NUMBER
      MOV      #STACK,SP ;INITIAL THE STACK POINTER
      MOV      #FMTDPB,RD ;DPB BLOCK ADDRESS
      MOV      $CDW1,R1  ;ADDRESS OF THE LOGICAL DRIVE BLOCK

```

```

2890 005762 000004          000001 001174
2891 005764 012737          000002 001336
2892 005772 012737          001100
2893 006000 012706          037146
2894 006004 012700          001270
2895 006010 013701

```


K05

```

2896 006014 113710 001222          MOVB   DRIVE,(R0)      ;PHYSICAL DRIVE #
2897 006020 013760 001266 000010    MOV    $DEVH,$SEC(R0) ;LOAD THE SECTOR # FROM THE LOGICAL DRIVE NUMBER.
2898 006026 012760 001154 000012    MOV    #620,$CYL(R0)  ;LOAD CYLINDER NUMBER
2899 006034 012760 177400 000004    MOV    #-256,$WRDM(R0);LOAD NEG WORD COUNT
2900 006042 012760 037262 000006    MOV    #ENDPGH,$BUF(R0);LOAD BUFFER ADDRESS
2901 006050 012760 037166 000014    MOV    #RM.REG,14(R0) ;ADDRESS TO SAVE ALL RM/RM03 REG'S
2902 006056 105060 000011          CLRB   $TRK(R0)       ;START FROM TRACK 0
2903 006062 112760 000161 000002    MOVB   $WRTDAT,$COMND(R0);WRITE DATA COMMAND
2904 006070 004737 014546          JSR    PC,FILBUF      ;FILL THE BUFFER WITH STANDARD PATTERN
2905 006074 004037 023620          JSR    R0,RM03       ;CALL THE DRIVER
2906 006100 037146          FMTDPB
2907 006102 000774          BR     2$            ;BRANCH IF NOT QUEUE SUCCESSFULLY
2908 006104 005737 037164          3$:  TST    FMTDPB+16
2909 006110 001775          BEQ    3$            ;BRANCH IF NOT DONE
2910 006112 012700 037146          MOV    #FMTDPB,R0    ;R0=FMTDPB ADDRESS
2911 006116 004737 013264          JSR    PC,PROCESS    ;CHECK THE TERMINATION
2912 006122 112760 000151 000002    MOVB   #WCKD,$COMND(R0);CHANGE TO THE WRITE CHECK DATA COMMAND
2913 006130 004037 023620          4$:  JSR    R0,RM03       ;CALL THE DRIVER
2914 006134 037146          FMTDPB
2915 006136 000774          BR     4$            ;BRANCH IF NOT QUEU SUCCESSFULLY
2916 006140 005737 037164          5$:  TST    FMTDPB+16
2917 006144 001775          BEQ    5$            ;BRANCH IF NOT DONE
2918 006146 012700 037146          MOV    #FMTDPB,R0
2919 006152 004737 013264          JSR    PC,PROCESS    ;PROCESS IF ANY ERROR HAPPENS ?
2920 006156 112760 000161 000002    MOVB   $WRTDAT,$COMND(R0);RESET TO WRITE DATA COMMAND
2921 006164 105260 000011          INCB   $TRK(R0)      ;INCREMENT TO THE NEXT TRACK
2922 006170 122760 000005 000011    CMPB   #5,$TRK(R0)   ;ALL TRACKS DONE ?
2923 006176 101336          BHI    2$            ;NO
2924
2925
2926
2927          ;TEST 3
2928          ;WRITE 7 ZONES FOR WRITE TEST IN PASS 2
2929          ;WRITE 6 ZONES FOR READ TEST IN PASS2
2930          ;SDEVH : LOGICAL DRIVE #
2931          ;SCDW1 : ADDRESS OF LOGICAL DRIVE HISTORY BLOCK FILE
2932          ;SCDW2 : SYS NAME
2933          ;DRIVE : PHYSICAL DRIVE # OF THIS LOGICAL DRIVE
2934          ;PACK : ENTRY OF TABLE-D
2935          ;CMCNT : ZONE COUNT
2936          ;CMSEC : DELTA CYLINDER COUNT
2937          ;R4 : ENTRY POINTER OF TABLE-D,CANNOT BE DESTORIED.
2938          ;R0 : ADDRESS OF FMTDPB
2939
2940
2941
2942          ;*****
2943          ;TST3: SCOPE
2944          ;DO 1 ITERATION
2945          ;LOAD THE TEST NUMBER
2946          ;INITIAL THE STACK POINTER
2947          ;ADDRESS OF TABLE-D
2948          ;LOGICAL DRIVE #
2949          ;BRANCH IF LOGICAL DRIVE 0
2950          ;EACH LOGICAL DRIVE TAKES 16 BYTES IN THE TABLE
2951          ;DECREMENT THE DRIVE # COUNT,
    
```

2952	006242	001374				BNE	15		;BRANCH UNTIL THE ENTRY IS LOCATED
2953	006244	010437	001322		25:	MOV	R4,PACK		;SAVE THE TABLE-D ENTRY IN PACK
2954									;SET UP THE FMTDPB BLOCK
2955	006250	012700	037146			MOV	#FMTDPB,R0		;ADDRESS OF FMTDPB
2956	006254	113710	001222			MOV	DRIVE,(R0)		;PHYSICAL DRIVE NUMBER
2957	006260	112760	000161	000002		MOV	#WRDAT,\$COMND(R0)		;WRITE DATA COMMAND
2958	006266	012760	037262	000006		MOV	#ENDPGM,\$BUF(R0)		;BUFFER ADDRESS
2959	006274	012760	037166	000014		MOV	#RM.REG,14(R0)		;ADDRESS TO SAVE ALL RH/RMO3 REG'S
2960	006302	012760	177400	000004		MOV	#-256,\$WDM(R0)		;NEG WORD COUNT
2961	006310	013701	001266			MOV	\$DEVN,R1		;LOGICAL DRIVE #
2962	006314	006301				ASL	R1		;WORD INDEX
2963	006316	016104	002702			MOV	PSEUDO(R1),R4		;LOAD THE DATA PATTERN
2964	006322	016002	000006			MOV	\$BUF(R0),R2		;BUFFER ADDRESS IN R2
2965	006326	012703	000400			MOV	#256,R3		;POS WORD COUNT
2966	006332	010422			35:	MOV	R4,(R2)+		;FULL THE BUFFER WITH SIGLE WORD PATTERN
2967	006334	005303				DEC	R3		;DECREMENT THE WORD COUNT
2968	006336	001375				BNE	35		;BRANCH,UNTIL IT IS FULL
2969									
2970									
2971	006340	005046				CLR	-(SP)		;DUMY PAIR OF ZONE COUNT
2972	006342	005046				CLR	-(SP)		;DUMY STARTING CYLINDER NUMBER
2973	006344	012746	000006			MOV	#6,-(SP)		;ZONE COUNT
2974	006350	012746	000160			MOV	#112,-(SP)		;STARTING CYLINDER NUMBER
2975	006354	012737	000007	001352		MOV	#7,CMCNT		;ZONE COUNT
2976	006362	005037	001354			CLR	CMCYL		;STARTING CYLINDER NUMBER
2977	006366	012737	000020	001360	45:	MOV	#16,CMSEC		;DELTA CYLINDER NUMBER
2978	006374	012700	037146			MOV	#FMTDPB,R0		;R0 DPB ADDRESS
2979	006400	013704	001322			MOV	PACK,R4		;R4 ENTRY TO TABLE-D
2980	006404	013760	001354	000012	55:	MOV	CMCYL,\$CYL(R0)		;STARTING CYLINDER
2981	006412	105060	000011			CLRB	\$TRK(R0)		;STARTS FROM TRACK 0
2982	006416	111460	000010		65:	MOV	(R4),\$SEC(R0)		;LOAD SECTOR NUMBER FROM TABLE-D
2983	006422	004037	023620		75:	JSR	R0,RMO3		;CALL THE DRIVER
2984	006426	037146				FMTDPB			
2985	006430	000774				BR	75		;BRANCH IF QUEU IS NOTSUCCESSFUL
2986	006432	005737	037164		85:	TST	FMTDPB+16		;DONE ?
2987	006436	001775				BEQ	85		;BRANCH IF NOT
2988	006440	012700	037146			MOV	#FMTDPB,R0		
2989	006444	004737	013264			JSR	PC,PROCES		;PROCESS TO CHECK IF ANY ERROR
2990	006450	062760	000020	000010		ADD	#16,\$SEC(R0)		;WRITE TWO SECTORS ON ONE CYLINDER
2991	006456	122760	000037	000010		CMPB	#31,\$SEC(R0)		;ALL DONE-TWO SECTORS
2992	006464	103356				BHIS	75		;BRANCH IF NOT
2993	006466	111460	000010			MOV	(R4),\$SEC(R0)		;RESTORE SECTOR #
2994	006472	105260	000011			INCB	\$TRK(R0)		;INCREMENT TO NEXT TRACK
2995	006476	122760	000004	000011		CMPB	#4,\$TRK(R0)		;LAST TRACK ?
2996	006504	103346				BHIS	75		;NO, THEN BRANCH
2997	006506	105060	000011			CLRB	\$TRK(R0)		;RESTORE TO TRACK-0
2998	006512	005260	000012			INC	\$CYL(R0)		;INCREMENT CYLINDER NUMBER
2999	006516	005204				INC	R4		;INCREMENT TABLE-D ENTRY
3000	006520	005337	001360			DEC	CMSEC		;DECREMENT THE DELTA CYLINDER COUNT
3001	006524	001334				BNE	65		;BRANCH IF NOT END OF THIS BLOCK
3002	006526	062760	000160	000012		ADD	#112,\$CYL(R0)		;INCREMENTTHE CYLINDER NUMBER TO NEXT ZONE
3003	006534	016037	000012	001354		MOV	\$CYL(R0),CMCYL		;INITAIL THE STARTING CYLINDER IN THE BLOCK
3004	006542	005337	001352			DEC	CMCNT		;DECREMENT THE ZONE COUNT
3005	006546	001307				BNE	45		;LOOPING IF NOT END OF ZONE
3006	006550	012637	001354			MOV	(SP)+,CMCYL		;LOAD NEW PAIR OF STARTING CYLINDER
3007	006554	012637	001352			MOV	(SP)+,CMCNT		;AND ZONE COUNT

M05

```
3008 006560 005737 001354
3009 006564 001300
3010 006566 005737 001352
3011 006572 001275
```

```
TST CMCYL ;NOT END YET ?
BNE 4$ ;BRANCH IF NOT
TST CMCNT ;BRANCH IF NOT END
BNE 4$ ;LOOPING BACK
```

```
;TEST 4
;UPDATE THE PARAMETERS,SCDW1,SDEVN,ASSGN1
;DIRECT THE OPERATOR TO DISMOUNT PACK AND LOAD TO OTHER DRIVE
;SCDW2,DRIVE ARE CHANGED BY TEST ONE ONLY AFTER THE TEST LOOPING TO TEST1
```

```
3021
3022 006574 000004
3023 006576 012737 000001 001174
3024 006604 012737 000004 001336
3025 006612 012706 001100
3026 006616 012777 013256 014254
3027 006624 005077 014252
3028 006630 104401 001205
3029 006634 104401 035323
3030 006640 104401 001272
3031 006644 013746 001222
3032 006650 104403
3033 006652 002
3034 006653 000
3035 006654 104401 036121
3036 006660 104401 035341
3037 006664 012701 000001
3038 006670 005002
3039 006672 020237 001266 1$:
3040 006676 001404
3041 006700 000241
3042 006702 006101
3043 006704 005202
3044 006706 000771
3045 006710 040137 002000 2$:
3046 006714 001410
3047 006716 005202 3$:
3048 006720 006302
3049 006722 016237 002600 001270
3050 006730 006202
3051 006732 010237 001266
3052 006736 104411 4$:
3053 006740 012605
3054 006742 122725 000122
3055 006746 001373
3056 006750 105715
3057 006752 001371
3058 006754 005737 002000
3059 006760 001002
3060 006762 000137 006772
3061 006766 000137 005450 5$:
```

```
*****
1$T4: SCOPE
MOV #1,STIMES ;DO 1 ITERATION
MOV #4,TSTNM ;LOAD THE TEST NUMBER
MOV #STACK,SP ;LOAD THE STACK POINTER
MOV #IDLEX,ARMVEC ;RESET THE INTERRUPT VECTOR
CLR ARMVEC+2 ;CLEAR THE INTERRUPT LEVEL
TYPE ,SCRLF
TYPE ,F 312
TYPE ,W2
MOV DRIVE,-(SP)
TYPOS
.BYTE 2
.BYTE 0
TYPE ,LINSR
TYPE ,MESG13
MOV #1,R1
CLR R2
1$: CMP R2,SDEVN ;LOCATE THE COORESPONDING BIT MAP
BEQ 2$ ;BRANCH IF LOCATED
CLC ;LOCATE NEXT DRIVE
ROL R1
INC R2 ;NEXT DRIVE #
BR 1$ ;LOCATE THE BIT MAP
2$: BIC R1,ASSGN1 ;DEASSIGN THE LOGICAL DRIVE FOR PASS 1
BEQ 4$ ;NO MORE DRIVES
3$: INC R2 ;GET NEXT LOGICAL DRIVE #
ASL R2 ;WORD INDEX
MOV BLKADR(R2),SCDW1 ;LOAD THE NEW DPB ADDRESS
ASR R2 ;RESTORE R2
MOV R2,SDEVN ;LOAD THE NEW LOGICAL DRIVE #
4$: RDLIN ;WAIT UNTIL IT IS DONE
MOV (SP)+,R5 ;LOCATE THE INPUT LINE
CMPB #'R,(R5)+
BNE 4$
TSTB (R5) ;TERMINATOR ?
BNE 4$ ;BRANCH IF NOT
TST ASSGN1 ;OTHER DRIVES ?
BNE 5$ ;BRANCH IF MORE DRIVES IN TEST
JMP XPASS2 ;BRANCH TO PASS 2
5$: JMP TST1 ;JUMP TO TEST 1
```

3062
3063

N05

```

3064 ;XPASS2      INITILIZE FOR PASS 2 TEST
3065 ;SCDW1      : ADDRESS OF THE CURRENT LOGICAL DRIVE HISTORY FILE
3066 ;SCDW2      : SYSTEM NAME A THROUGH H
3067 ;SDEVN      : CURRENT LOGICAL DRIVE # 0 TO 15.
3068 ;ASSGN2     : ASSIGNED LOGICAL DRIVE FOR PASS 2
3069 ;ASNLST     : ASSIGNED LOGICAL DRIVE
3070 ;DRIVE      : PHYSICAL DRIVE # OF CURRENT RH/RM SYSTEM
    
```

```

3074 006772 005737 002002 XPASS2: TST      ASSGN2      ;ANYTHING IN TEST FOR PASS 2
3075 006776 001002          BNE        1$          ;YES, THEN GO ON
3076 007000 000137 012762          JMP        XEND2       ;JUMP TO END OF PASS 2
3077 007004 005037 001266          CLR        SDEVN      ;START FROM LOGICAL DRIVE 0
3078 007010 013737 002600 001270 1$:      MOV        BLKADR,SCDW1 ;ADDRESS OF LOGICAL BLOCK DRIVE 0
3079 007016 112737 000101 001272          MOVB      #'A,SCDW2   ;LOAD SYSTEM NAME "A"
3080 007024 013737 002004 023076          MOV        SYSADR,RMADR ;LOAD SYSTEM-A ADDRESS TO DRIVER
3081 007032 013737 002006 023100          MOV        SYSADR+2,RMVEC ;LOAD SYSTEM-A VECTOR TO DRIVER
3082 007040 013701 001270          MOV        SCDW1,R1   ;R1=ADDRESS OF LOGICAL BLOCK
3083 007044 104401 001205          TYPE     ,SCRLF
3084 ;           ;
3085 007050 005037 001374          TYPE     MMSG14 ;STARTING PASS 2
3086 007054 000400          CLR        FAULT     ;RESET THE NOT COMPATIBLE FLAG
3087 ;           ;
3088 ;           ;
3089 ;           ;
3090 ;           ;
3091 ;           ;
3092 ;           ;
3093 ;           ;
3094 ;           ;
3095 ;           ;
3096 ;           ;
3097 ;           ;
3098 ;           ;
3099 ;           ;
    
```

```

;SCDW1      : ADDRESS OF CURRENT LOGICAL DRIVE BLOCK, ONLY CHANGED BY TST9
;SCDW2      : SYSTEM-NAME , ONLY CHANGED BY TEST 5
;SDEVN      : CURRENT LOGICAL DRIVE #
;ASSGN2     : ASSIGNED LOGICAL DRIVE'S BIT MAP FOR PASS 2
;DRIVE      : PHYSICAL DRIVE #
;ASNLST     : ASSIGNED LOGICAL DRIVE IN THE TEST
;IN TEST 5 DIRECT OPERATOR TO CHANGE, LOAD THE PACK
    
```

```

3101 007056 000004          TST5:  SCOPE
3102 007060 012737 000001 001174          MOV        #1,$TIMES  ;DO 1 ITERATION
3103 007066 012737 000005 001336          MOV        #5,TSTNM   ;LOAD THE TEST NUMBER
3104 007074 012706 001100          MOV        #STACK,SP  ;LOAD THE STACK POINTER
3105 007100 012777 013256 013772          MOV        #IDLEX,RMVEC ;RESET THE INTERRUPT VECTOR
3106 007106 005077 013770          CLR        RMVEC+2    ;CLEAR THE INTERRUPT LEVEL
3107 007112 013701 001270          MOV        SCDW1,R1   ;ADDRESS OF THE HISTORY FILE
3108 007116 126137 000014 001272          CMPB      $$SYSNM(R1),SCDW2 ;ON THE SAME SUB-SYSTEM
3109 007124 001420          BEQ        1$        ;THEN DON'T UPDATE SYSTEM ADDRESS
3110 007126 116137 000014 001272          MOVB      $$SYSNM(R1),SCDW2
3111 007134 013700 001272          MOV        SCDW2,RO   ;LOCATE SYSTEM ADDRESS TABLE
3112 007140 005300          DEC        RO         ;ADJUST FOR INDEX FORM 0
3113 007142 042700 177760          BIC        #177760,RO ;LEFT ON FOUR BITS
3114 007146 006300          ASL        RO
3115 007150 006300          ASL        RO         ;INDEX FOR TWO WORD
3116 007152 016037 002004 023076          MOV        SYSADR(RO),RMADR ;SYSTEM ADDRESS
3117 007160 016037 002006 023100          MOV        SYSADR+2(RO),RMVEC ;SYSTEM INTERRUPT VECTOR
3118 ;           ;
3119 ;           ;
    
```

```

3120 007166 104401 001205      1S:  TYPE      ,SCRLF
3121 007172 104401 035414      TYPE      ,MSG14      ;START THE PASS 2
3122 007176 104401 001205      TYPE      ,SCRLF
3123 007202 104401 035101      TYPE      ,MSG4      ;MOUNT PACK ON DRIVE
3124 007206 104401 001272      TYPE      ,SCDW2     ;SYSTEM NAME
3125 007212 111137 001222      MOV      (R1),DRIVE ;LOCATE THE PHYSICAL DRIVE #
3126 007216 111146      MOV      (R1),-(SP)
3127 007220 104403      TYPOS
3128 007222      .BYTE    2
3129 007223      .BYTE    0
3130 007224 104401 036121      TYPE      ,LINSR
3131 007230 104401 035127      TYPE      ,MSG8      ;AND LOAD
3132 007234 104401 001205      TYPE      ,SCRLF
3133 007240 104401 035142      TYPE      ,MSG9      ;TYPE R<CR> WHEN DRIVE IS READY
3134 007244 104411      2S:  RDLIN
3135 007246 012602      MOV      (SP)+,R2   ;LOCATE THE READ IN LINE
3136 007250 122722 000122      CMPB     #'R,(R2)+ ;FIRST CHAR IS A "R"
3137 007254 001373      BNE      2S        ;BRANCH IF NOT
3138 007256 105712      TSTB     (R2)      ;FOLLOWED BY A TERMINATOR
3139 007260 001371      BNE      2S        ;BRANCH IF NOT
3140 007262 004737 023114      JSR      PC,RMINIT
3141 007266 012737 177777 023036      MOV      #-1,SAVEFG ;SAVE ALL RH/RM REGISTERS
3142 007274 012737 177777 023040      MOV      #-1,SEEKFG ;DON'T DO IMPLY SEEK
3143 007302 013700 001222      MOV      DRIVE,RO  ;LOAD THE PHYSICAL DRIVE NUMBER
3144 007306 105760 022750      TSTB     DRVSTA(RO);DRIVE EXISTS AND ON-LINE ?
3145 007312 003405      BLE      3S        ;BRANCH IF NOT
3146 007314 105760 022760      TSTB     DRVTP(RO) ;AN RMO3 ?
3147 007320 003402      BLE      3S        ;NOT AN RMO3
3148 007322 000137 007340      JMP      TST6      ;BRANCH IF IT IS,(NEXT TEST)
3149 007326 104401 001205      3S:  TYPE      ,SCRLF
3150 007332 104401 035201      TYPE      ,MSG10   ;DRIVE IS NOT READY
3151 007336 000647      BR       TST5      ;TRY AGAIN
    
```

```

;DRIVE : PHYSICAL DRIVE NUMBER
;SCDW1 : DPB BLOCK OF THIS LOGICAL DRIVE
;SDEVN : LOGICAL DRIVE #
;SCDW2 : SUB-SYSTEM NAME
;RMADR : SUB-SYSTEM BASE REGISTER ADDRESS
;RMVEC : SUB-SYSTEM INTERRUPT VECTOR
    
```

```

;THE ABOVE PARAMETERS CANNOT BE MODIFIED BY THIS TEST (TST6)
;THE FOLLOWING REG'S ARE ASSIGNED AS:
;RO : ADDRESS OF DPB FEEDED INTO DRIVER HANDLER=FMTDPB
;R1 : ADDRESS OF THE HISTORY FILE BLOCK OF THE LOGICAL DRIVE=SCDW1
    
```

```

;IN TEST 6, EACH LOGICAL DRIVE WRITES 7 CYLINDERS ON EACH SURFACE WITH
;AN UNIQUE DATA PATTERN. THESE 7 CYLINDERS HAS BEEN
;WRITTEN BY OTHER DRIVES IN PASS 1.
;THEN, THIS LOGICAL DRIVE EXECUTES "WRITE CHECK DATA" TO SEE IF
;THIS LOGICAL DRIVE CAN OVER-WRITE ALL DATA WRITTEN BY OTHER
;DRIVES.
    
```

```

;THESE 7 CYLINDERS ARE SPECIFIED AS SDEVN, SDEVN+128, SDEVN+256
    
```

3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3170
3171
3172
3173
3174
3175

3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231

007340 000004
007342 012737 000001 001174
007350 012737 000006 001336
007356 012706 001100
007362 012703 002046
007366 012704 002306
007372 005023
007374 020403
007376 101375
007400 012700 037146
007404 013701 001270
007410 113710 001222
007414 012760 160000 000004
007422 005060 000010
007426 112760 000161 000002
007434 013760 001266 000012
007442 012760 037262 000006
007450 012760 037166 000014
007456 013702 001266
007462 006302
007464 016202 002702
007470 016003 000006
007474 012704 020000
007500 010223
007502 005304
007504 001375
007506 004037 023620
007512 037146
007514 000774
007516 005737 037164
007522 001775
007524 012700 037146
007530 004737 013264
007534 105260 000011
007540 122760 000005 000011
007546 101357
007550 105060 000011
007554 062760 000200 000012
007562 022760 001417 000012
007570 103346

;SDEV+384,SDEV+512,SDEV+640 AND SDEV+768.
;THE OVER-WRITE TEST IS PERFORMED WITH OFFSETS IN BOTH DIRECTIONS.
;:
;*****
TST6: SCOPE
MOV #1,STIMES ;DO 1 ITERATION
MOV #6,TSTNM ;LOAD TEST NUMBER
MOV #STACK,SP ;INITIAL THE STACK POINT
MOV #OVIND,R3 ;1 ST ADDRESS TO CLEAR
MOV #RDNO,R4 ;LAST ADDRESS+2
15: CLR (R3)+ ;RESET ALL SCORE BOARD
CMP R4,R3 ;ALL LOCATIONS ARE CLEARED ?
BHI 15 ;BRANCH IF NOT
MOV #FMTDPB,RO ;SET UP PARAMETER BLOCK
MOV #CDW1,R1 ;HISTORY FILE BLOCK
MOVB DRIVE,(RO) ;LOAD THE PHYSICAL DRIVE #
MOV #8192,SWRDM(RO) ;LOAD THE WORD COUNT
CLR #SEC(RO) ;START FROM TRKO,SECO
MOVB #WRTDAT,SCOMND(RO) ;WRITE DATA COMMAND
MOV SDEV,SCYL(RO) ;LOAD THE STARTING CYLINDER
MOV #ENDPGH,SBUF(RO) ;LOAD THE BUFFER ADDRESS
MOV #RM.REG,14(RO) ;REG'S SAVE ADDRESS
MOV SDEV,R2 ;LOCATE THE DATA PATTERN
ASL R2 ;WORD INDEX
MOV PSEUDO(R2),R2 ;LOAD R2 THE DATA PATTERN POINTED
;BY ITSELF
MOV SBUF(RO),R3 ;BUFFER ADDRESS
MOV #8192,R4 ;WORD COUNT
25: MOV R2,(R3)+ ;FILL THE BUFFER
DEC R4 ;DECREMENT WORD CTR
BNE 25 ;BRANCH IF NOT DONE
;START TO WRITE ONE CYLINDER
;ON EACH SURFACE OF EACH WRITE
;CURRENT ZONE (7 ZONES ON ONE PACK)
35: JSR RO,RMO3 ;CALL THE DRIVER
FMTDPB ;PARAMETER BLOCK
BR 35 ;BRANCH IF QUEUE FAIL
45: TST FMTDPB+16 ;WRITE COMMAND DONE ?
BEQ 45 ;NO, THEN WAIT
MOV #FMTDPB,RO ;PROCESS IF ANY ERROR
JSR PC,PROCES
INCB STRK(RO) ;NEXT TRACK
CMPB #5,STRK(RO) ;LAST TRACK IS DONE ?
BHI 35 ;NO, THEN BRANCH
CLRB STRK(RO) ;RESET TRACK NUMBER
ADD #128,SCYL(RO) ;MOVE TO NEXT ZONE
CMP #783,SCYL(RO) ;LAST ZONE IS DONE ?
BHIS 35 ;BRANCH IF NOT
;RESET THE FMTDPB BLOCK AND
;EXECUTE WRITE-CHECK COMMAND
;TO DETECT ANY COMPATIBLE PROBLEM


```

3232                                     ;THE FOLLOWING SUBROUTINE ARE CALLED
3233                                     ;SCORE,OFFST,MAKEUP.
3234
3235 007572 005037 002640                CLR    OFFCOD                ;SET NEGATIVE OFFSET DIRECTION FLAG
3236 007576 012700 037146                MOV    #FMTDPB,RO           ;RO=FMTDPB ADDRESS
3237 007602 005060 000010                CLR    $SEC(RO)            ;START FROM SECTOR 0,SURFACE 0
3238 007606 013760 001266 000012        MOV    $DEVH,$CYL(RO)      ;STARTING CYLINDER NUMBER
3239                                     ;TOTAL 7 CYLINDERS ON ONE SURFACE
3240 007614 112760 000151 000002        LOOP2: MOVB  #WCKD,$COMND(RO) ;WRITE-CHECK-DATA COMMAND
3241 007622 012760 037262 000006        MOV    #ENDPGM,$BUF(RO)   ;RESET BUFFER ADDRESS
3242 007630 012760 037166 000014        MOV    #RM.REG,14(RO)     ;ADDRESS TO SAVE RM/RMO3 REG'S
3243 007636 004037 023620                1$:   JSR    RO,RMO3        ;CALL THE DRIVER
3244 007642 037146                        FMTDPB ;PARAMETER BLOCK ADDRESS
3245 007644 000774                        BR     1$                  ;BRANCH IF QUEUE FAILURE
3246 007646 005737 037164                2$:   TST    FMTDPB+16     ;TEST IF COMMAND IS DONE ?
3247 007652 001775                        BEQ    2$                  ;BRANCH IF NOT
3248 007654 012700 037146                MOV    #FMTDPB,RO         ;LOAD THE PARAMETER BLOCK ADDRESS
3249 007660 004737 013264                JSR    PC,PROCÉS         ;REPORT IF ANY ERROR
3250 007664 004737 010732                JSR    PC,LABAD          ;LOCATE STARTING AND ENDING SECTORS
3251 007670 005760 000016                TST    16(RO)            ;ANY ERROR ?
3252 007674 100006                        BPL    3$                  ;BRANCH IF NONE
3253 007676 023737 001360 001356        CMP    CMSEC,STARSC      ;ON THE SAME SECTOR FOR STARTING AND ENDING
3254 007704 101404                        BLOS   4$                  ;DON'T INCREMENT SCORES
3255 007706 005337 001360                DEC    CMSEC              ;DECREMENT THE SECTOR ADDRESS
3256 007712 004737 010512                3$:   JSR    PC,SCORE      ;INCREMENT SCORE
3257 007716 005760 000022                4$:   TST    $RMWC(RO)     ;WORD COUNT = 0 /
3258 007722 001407                        BEQ    5$                  ;BRANCH IF WORD COUNT IS 0
3259 007724 116060 000026 000010        MOVB  $RMDA(RO),$SEC(RO) ;UPDATE STARTING SECTOR
3260 007732 016060 000022 000004        MOV    $RMWC(RO),$WRDM(RO);UPDATE WORD COUNT
3261 007740 000736                        BR     1$                  ;TO READ THE REST SECTORS
3262
3263
3264                                     ;THE FOLLOWING CODING TEST THE
3265                                     ;COMPATIBLE PROBLEM IN OFFSET MODE
3266                                     ;OFFCOD=0,NEGATIVE ;OFFCOD=1 ,POSITIVE
3267
3268 007742 012700 037146                5$:   MOV    #FMTDPB,RO     ;RESET THE DPB BLOCK
3269 007746 012760 160000 000004        MOV    #-8192,$WRDM(RO)  ;FULL TRACK WORD COUNT
3270 007754 105060 000010                CLRB  $SEC(RO)           ;RESET TO SECTOR 0,SURFACE NOT CHANGED
3271 007760 012760 037262 000006        MOV    #ENDPGM,$BUF(RO) ;BUFFER ADDRESS
3272 007766 012760 037166 000014        MOV    #RM.REG,14(RO)   ;ADDRESS TO SAVE ALL RM/RMO3 REG'S
3273 007774 004537 010176                6$:   JSR    K5,OFFST     ;CALL OFFSET
3274 010000 000440                        BR     10$                ;BRANCH TO NEXT CYLINDER,IF OFFSET FAILS
3275 010002 004737 010400                JSR    PC,MAKEUP         ;CALL WRITE CHECK IN OFFSET MODE
3276 010006 005737 037164                7$:   TST    FMTDPB+16   ;OFFSET WRITE CHECK IS DONE ?
3277 010012 001775                        BEQ    7$                  ;BRANCH IF NOT
3278
3279 010014 012700 037146                MOV    #FMTDPB,RO         ;LOAD THE DPB ADDRESS
3280 010020 004737 013264                JSR    PC,PROCÉS         ;REPORT IF ANY ERROR
3281 010024 004737 010732                JSR    PC,LABAD          ;LOCATE STARTING AND ENDING SECTORS
3282 010030 005760 000016                TST    16(RO)            ;ANY ERROR ?
3283 010034 100006                        BPL    8$                  ;BRANCH IF NONE
3284 010036 023737 001360 001356        CMP    CMSEC,STARSC      ;STARTING AND ENDING ADDRESSES O.K ?
3285 010044 101404                        BLOS   9$                  ;BRANCH IF NOT
3286 010046 005337 001360                DEC    CMSEC              ;EXCLUDING THE BAD SECTOR
3287 010052 004737 010512                8$:   JSR    PC,SCORE      ;UPDATE THE TEST SCORE
    
```

E06

```

3288 010056 005760 000022 9S: TST SRMWC(R0) ;WORD CTR = 0 ?
3289 010062 001407 BEQ 10S ;IF WORD CTR=0 BRANCH TO NEXT OP
3290 010064 116060 000026 000010 MOV SBMWA(R0),SSEC(R0) ;UPDATE THE NEW STARTING ADDRESS
3291 010072 016060 000022 000004 MOV SRMWC(R0),SWORD(R0) ;UPDATE THE NEW WORD COUNT
3292 010100 000735 BR 6S
3293 010102 062760 000200 000012 10S: ADD #128.,SCYL(R0) ;ADJUST CYLINDER ADDRESS TO NEXT ZONE
3294 010110 022760 001417 000012 CMP #783.,SCYL(R0) ;ALL 7 ZONES HAVE BEEN TESTED ?
3295 010116 103402 BLO 11S ;BRANCH, IF ALL DONE
3296 010120 000137 007614 JMP LOOP2 ;TO NEXT WRITE CURRENT ZONE
3297 010124 013760 001266 000012 11S: MOV SDEVN,SCYL(R0) ;RESET CYLINDER ADDRESS
3298 010132 105260 000011 INCB STRK(R0) ;INCREMENT TO NEXT SURFACE
3299 010136 122760 000005 000011 CMPB #5,STRK(R0) ;ALL FIVE SURFACES ARE TESTED ?
3300 010144 101402 BLOS 12S ;BRANCH IF ALL DONE
3301 010146 000137 007614 JMP LOOP2 ;TO NEXT SURFACE
3302 010152 005737 002640 12S: TST OFFCOD ;FINISHING TEST THE POSITIVE OFFSET ?
3303 010156 001005 BNE 13S ;BRANCH IF ALL DONE
3304 010160 012737 000001 002640 MOV #1,OFFCOD ;SET POSITIVE OFFSET DIRECTION FLAG
3305 010166 000137 007576 JMP LOOP1 ;RESTART LOCATION
3306 010172 000137 011030 13S: JMP TST7 ;BRANCH TO NEXT TEST
    
```

```

3307
3308
3309
3310
3311 ;OFFST ROUTINE
3312 ;OFFSET THE HEAD IN THE DIRECTION TOWARD SPINDLE OR AWAY FROM THE SPINDLE
3313 ;OFFCOD = 1 TOWARD SPINDLE (POSITIVE)
3314 ;OFFCOD = 0 AWAY FROM SPINDLE (NEGATIVE)
3315 ;DRIVE PHYSICAL DRIVE NUMBER
3316 ;R0 DPB ADDRESS
3317 ;RETRY 3 TIMES
3318 ;CALLING SEQUENCE
3319 ; JSR R5,OFFST
3320 ; RET1 OFFSET FAIL RETURN ADDRESS
3321 ; RET2 OFFSET SUCCESSFUL RETURN
3322
    
```

```

3323 010176 010146 OFFST: MOV R1,-(SP) ;SAVE ALL REGISTERS
3324 010200 010246 MOV R2,-(SP)
3325 010202 010346 MOV R3,-(SP)
3326 010204 010446 MOV R4,-(SP)
3327 010206 013704 023076 MOV RMADR,R4 ;RH70/RM BASE ADDRESS
3328 010212 013701 001222 MOV DRIVE,R1 ;PHYSICAL DRIVE # R1 FOR INDEXING
3329 010216 012702 000003 1S: MOV #3,R2 ;TRY TO OFFSET 3 TIMES
3330 010222 005003 CLR R3 ;DECREMENT COUNT
3331 010224 010164 000010 MOV R1,RMCS2(R4) ;LOAD DRIVE ADDRESS
3332 010230 012764 000011 000000 MOV #11,RMCS1(R4) ;CLEAR DRIVE WITH INTERRUPT DISABLE
3333 010236 016064 000012 000034 MOV SCYL(R0),RMDC(R4) ;LOAD DESIRED CYLINDER ADDRESS
3334 010244 016064 000010 000006 MOV SSEC(R0),RMDA(R4) ;LOAD DESIRED ADDRESS
3335 010252 016064 000004 000002 MOV SWORD(R0),RMWC(R4) ;LOAD WORD COUNT
3336 010260 016064 000006 000004 MOV SBUF(R0),RMBR(R4) ;LOAD BUFFER ADDRESS
3337 010266 005737 002640 TST OFFCOD ;WHICH DIRECTION ?
3338 010272 001404 BEQ 2S ;BRANCH IF NEGATIVE
3339 010274 012764 010200 000032 MOV #BIT12:BIT7,RMOF(R4) ;LOAD RMOF WITH FMT AND OFT
3340 010302 000403 BR 3S
3341 010304 012764 010000 000032 2S: MOV #BIT12,RMOF(R4) ;LOAD FMT ONLY
3342 010312 012764 000015 000000 3S: MOV #15,RMCS1(R4) ;ISSUE OFFSET COMMAND
3343 010320 136164 023064 000016 4S: BITB ATABIT(R1),RMA5(R4) ;ATA= 1
    
```



```

3344 010326 001005      BNE      5$      ; OFFSET IS DONE
3345 010330 005303      DEC      R3      ; COUNT = 0
3346 010332 001372      BNE      4$      ; BRANCH IF NOT
3347 010334 005302      DEC      R2      ; DECREMENT RETRY COUNT
3348 010336 003413      BLE      6$      ; ERROR EXIT
3349 010340 000730      BR       1$      ; RETRY
3350 010342 032764 040000 000012 5$: BIT      #BIT14,RMDS(R4) ; ANY ERROR DURING OFFSET ?
3351 010350 001324      BNE      1$      ; BRANCH IF ERROR
3352 010352 032764 000001 000012 BIT      #BIT0,RMDS(R4) ; OFFSET MODE BIT SET ?
3353 010360 001720      BEQ      1$      ; BRANCH IF NOT
3354 010362 062705 000002      ADD      #2,R5    ; ADJUST RETURN ADDRESS
3355 010366 012604      6$: MOV      (SP)+,R4 ; RESTORE REG
3356 010370 012603      MOV      (SP)+,R3
3357 010372 012602      MOV      (SP)+,R2
3358 010374 012601      MOV      (SP)+,R1
3359 010376 000205      RTS      R5      ;EXIT
3360
3361
3362      ;MAKEUP ROUTINE
3363      ;THIS ROUTINE ISSUES A WRITE CHECK OR READ  COMMAND TO THE SELECTED DRIVE
3364      ;IN OFFSET MODE.
3365      ;AND SET UP THE FOLLOWING PARAMETERS
3366      ;
3367      ; DTUM = PHYSICAL DRIVE NUMBER
3368      ; TRNSWT = FMTDPB
3369      ; DRVACT (DRIVE) = 1
3370      ; TIMER (DRIVE) = 1 SECOND
3371
3372      ;CALLING SEQ
3373
3374      JSR      PC,MAKEUP
3375      RET
3376      ;MAIN PURPOSE OF THIS ROUTINE, TO EXECUTE A COMMAND WHILE ASSURE THAT
3377      ;THIS READ OR WRITE-CHECK COMMAND BEING EXECUTED IN OFFSET MODE.
3378      ;ROUTINES USED TD,SC,STO,( IN DRIVE HANDLER )
3379
3380
3381
3382 010400 010146      MAKEUP: MOV      R1,-(SP)
3383 010402 010246      MOV      R2,-(SP)
3384 010404 010446      MOV      R4,-(SP)
3385 010406 012702 000151  MOV      #WCKD,R2 ; WRITE CHECK DATA IN TEST 6
3386 010412 022737 000010 001336  CMP      #10,TSTNM ; ON TEST 8 ?
3387 010420 001002      BNE      1$      ; BRANCH IF NOT
3388 010422 012702 000171  MOV      #RDDAT,R2 ; READ DATA COMMAND IN TEST 8
3389 010426 005037 037164      1$: CLR      FMTDPB+16 ; CLEAR THE STATUS WORD
3390 010432 013737 001222 023062  MOV      DRIVE,DTUM ; ACTIVE DRIVE NUMBER
3391 010440 012737 037146 023010  MOV      #FMTDPB,TRNSWT ; TRANSFER UNDERWAY FLAG
3392 010446 013701 001222      MOV      DRIVE,R1 ; DRIVE NUMBER
3393 010452 013704 023076      MOV      RMADR,R4 ; RH/RM BASE ADDRESS
3394 010456 112761 000001 022740  MOVB     #1,DRVACT(R1) ; ACTIVE DRIVE FLAG
3395 010464 006301      ASL      R1      ; WORD INDEX
3396 010466 012761 001750 023042  MOV      #1000.,TIMER(R1) ; ONE SECOND TIMER
3397 010474 006201      ASR      R1
3398 010476 010264 000000      MOV      R2,RMDS1(R4) ; ISSURE WRITE CHECK OR READ COMMAND
3399 010502 012604      MOV      (SP)+,R4 ; RESTORE REG 4, 1

```



```

3400 010504 012602          MOV    (SP)+,R2
3401 010506 012601          MOV    (SP)+,R1
3402 010510 000207          RTS    PC                ;EXIT
3403
3404
3405          ; SCORE ROUTINE
3406          ; ROUTINE TO UPDATE THE TEST SCORE
3407          ; (TABLEX): ADDRESS OF CRRENT SCORE BOARD
3408          ; SDEVN: LOGICAL DRIVE #
3409          ; DRIVE: PHYSICAL DRIVE NUMBER
3410          ; CMSEC: END SECTOR ADDRESS
3411          ; STARSC: START SECTOR ADDRESS
3412          ; CALL SEQ
3413
3414          ;
3415          ; JSR    PC, SCORE
3416          ; RET
3417
3418          SCORE:
3419 010512 010146          MOV    R1, -(SP)        ;: PUSH R1 ON STACK
3420 010514 010246          MOV    R2, -(SP)        ;: PUSH R2 ON STACK
3421 010516 010346          MOV    R3, -(SP)        ;: PUSH R3 ON STACK
3422 010520 010446          MOV    R4, -(SP)        ;: PUSH R4 ON STACK
3423 010522 023737 001360 001356  CMP    CMSEC, STARSC    ;: CORRECT START AND STOP ADDRESSES
3424 010530 003473          BLE    9$              ;: BRANCH IF NOT
3425 010532 022737 000010 001336  CMP    #10, TSTNM       ;: ON TEST #
3426 010540 001011          BNE    2$              ;: BRANCH IF NOT (MUST BE TEST 6)
3427 010542 005737 002640          TST    OFFCOD           ;: NEGATIVE OFFSET ?
3428 010546 001403          BEQ    1$              ;: BRANCH IF NEGATIVE OFFSET
3429 010550 012703 002426          MOV    #RDPO, R3        ;: SCORE BOARD ADDRESS
3430 010554 000413          BR     4$
3431 010556 012703 002306          1$: MOV    #RDND, R3        ;: SCORE BOARD ADDRESS
3432 010562 000410          BR     4$
3433 010564 005737 002640          2$: TST    OFFCOD           ;: NEGATIVE OFFSET
3434 010570 001403          BEQ    3$              ;: BRANCH IF NEGATIVE OFFSET
3435 010572 012703 002166          MOV    #OVWPO, R3       ;: SCORE BOARD ADDRESS
3436 010576 000402          BR     4$
3437 010600 012703 002046          3$: MOV    #OVWNO, R3     ;: SCORE BOARD ADDRESS
3438 010604 113702 037157          4$: MOVB   2#FMTDPB+STRK, R2 ;: LOAD THE SURFACE NUMBER
3439 010610 005702          TST    R2              ;: ON SURFACE 0
3440 010612 001404          BEQ    6$              ;: BRANCH IF IT IS
3441 010614 062703 000020          5$: ADD    #16., R3       ;: EACH SCORE BOARD TAKES 16 BYTES
3442 010620 005302          DEC    R2              ;: LOCATED ?
3443 010622 001374          BNE    5$              ;: BRANCH IF NOT
3444 010624 010337 002044          6$: MOV    R3, TABLEX    ;: STORE THE TABLE STARTING ADDRESS
3445 010630 010301          MOV    R3, R1          ;: RE ASSIGN REGISTERS
3446 010632 013702 001266          MOV    SDEVN, R2       ;: LOGICAL DRIVE #
3447 010636 013703 001356          MOV    STARSC, R3      ;: START SECTOR
3448 010642 116204 002560          MOVB   INDST(R2), R4   ;: LOCATE THE STARTING PRONT FOR SCORE BOARD
3449 010646 060304          ADD    R3, R4          ;: UPDATE POINTER
3450 010650 022704 000017          11$: CMP    #15., R4       ;: SHOULD ADJUST POINTER ?
3451 010654 003403          BLE    7$              ;: NO
3452 010656 162704 000020          SUB    #16., R4        ;: ENTRY POINT AT THE SCORE BOARD
3453 010662 000772          BR     11$
3454 010664 060401          7$: ADD    R4, R1
3455 010666 023703 001360          8$: CMP    CMSEC, R3    ;: ENDING SECTOR REACHED ?

```

```

3456 010672 002412          BLT    9$          ;BRANCH IF IT IS
3457 010674 105221          INCB   (R1)+       ;INCREMENT SCORE AND POINT TO NEXT
3458                                ;LOGICAL DRIVE
3459 010676 005204          INC    R4          ;INCREMENT LOGICAL DRIVE #
3460 010700 005203          INC    R3          ;INCREMENT SECTOR COUNT
3461 010702 022704 000017  CMP    #15.,R4     ;TIME TO RESET TABLE ?
3462 010706 002367          BGE    8$          ;BRANCH IF NOT
3463 010710 013701 002044  MOV    TABLEX,R1 ;RESET TABLE ADDRESS
3464 010714 005004          CLR    R4          ;LOGICAL DRIVE 0
3465 010716 000763          BR     8$          ;LOOPING BACK
3466 010720          9$:
3467 010720 012604          MOV    (SP)+,R4   ;:POP STACK INTO R4
3468 010722 012603          MOV    (SP)+,R3   ;:POP STACK INTO R3
3469 010724 012602          MOV    (SP)+,R2   ;:POP STACK INTO R2
3470 010726 012601          MOV    (SP)+,R1   ;:POP STACK INTO R1
3471 010730 000207          RTS    PC
    
```

```

3472
3473
3474          ;LABAD ROUTINE
3475          ;LOCATE THE START SECTOR AND THE TERMINATE SECTOR OF THE PREVIOUS
3476          ;OPERATION
3477
3478          ;STARSC :      STARTING SECTOR
3479          ;CMSEC  :      ENDING SECTOR
3480          ;INFORMATION FROM $RMDA(FMTDPB), $RMWC(FMTDPB), $SEC(FMTDPB)
3481          ;          $SEC(FMTDPB), $WRDM(FMTDPB)
3482          ;CALLING SEQ
3483
3484          ;          JSR    PC,LABAD
3485          ;          RET
    
```

```

3486
3487
3488          LABAD:
3489          010732 010046          MOV    R0,-(SP)   ;:PUSH R0 ON STACK
3490          010734 010246          MOV    R2,-(SP)   ;:PUSH R2 ON STACK
3491          010736 012700 037146  MOV    #FMTDPB,R0 ;:DPB ADDRESS
3492          010742 116002 000026  MOV    $RMDA(R0),R2 ;:HARDWARE TERMINATING SECTOR
3493          010746 116037 000010 001356  MOV    $SEC(R0),STARSC ;:STARTING SECTOR ADDRESS
3494          010754 005702          TST    R2          ;:TERMINATOR AT 0 SECTOR
3495          010756 001404          BEQ    1$          ;:BRANCH IF IT IS
3496          010760 005302          DEC    R2          ;:DECREMENT ONE SECTOR COUNT
3497          010762 010237 001360  MOV    R2,CMSEC   ;:ENDING SECTOR ADDRESS
3498          010766 000415          BR     3$          ;:EXIT
3499          010770 005760 000022  1$:    TST    $RMWC(R0)  ;:WORD COUNT = 0
3500          010774 001407          BEQ    2$          ;:BRANCH IF IT IS
3501          010776 026060 000022 000004  CMP    $RMWC(R0), $WRDM(R0) ;:WORD COUNT CHANGED AT ALL ?
3502          011004 001003          BNE    2$          ;:BRANCH IF CHANGED
3503          011006 116037 000010 001360  MOV    $SEC(R0),CMSEC ;:END SECTOR = START SECTOR
3504          011014 112737 000037 001360  2$:    MOV    #31.,CMSEC ;:END AT SECTOR 31
3505          011022          3$:
3506          011022 012602          MOV    (SP)+,R2   ;:POP STACK INTO R2
3507          011024 012600          MOV    (SP)+,R0   ;:POP STACK INTO R0
3508          011026 000207          RTS    PC
    
```

```

3509
3510          ;TEST 7
3511          ;SELF TEST:WRITE CYLINDERS $DEV+17,$DEV+17+128,$DEV+17+128X2,ETC.
    
```


3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3560
3561
3562
3563
3564
3565
3566
3567

; THEN EXECUTE WRITE CHECK TO DETECT ANY DATA OR HEADER PROBLEM
; FOR THE SELECTED LOGICAL DRIVE
; SDEVN: LOGICAL DRIVE #
; DRIVE: PHYSICAL DRIVE #
;

::*****

TST: SCOPE
MOV #1,STIMES ; DO 1 ITERATION
MOV #7,TSTNM ; LOAD TEST NUMBER
MOV #STACK,SP ; INITIAL THE STACK
MOV #SELF0,R2 ; CLEAR THE SELF TEST SCORE BOARDS
1S: CLR (R2)+
CMP #SELF0+12,R2 ; ALL DONE ?
BHI 1S ; BRANCH IF NOT
MOV #FMTDPB,R0 ; SET UP DPB
MOV #-8192,SWRDM(R0) ; LOAD FULL TRACK WORD COUNT
MOVB #WRTDAT,SCOMND(R0) ; WRITE DATA COMMAND
CLR \$SEC(R0) ; SECTOR 0,SURFACE 0
MOVB DRIVE,(R0) ; LOAD PHY. DRIVE
MOV SDEVN,R2 ; LOCATE THE STARTING CYLINDER
ADD #17,R2
MOV R2,\$CYL(R0) ; CYLINDER ADDRESS
MOV #ENDPGM,\$BUF(R0) ; RESET BUFFER ADDRESS
MOV #RM.REG,14(R0) ; ADDRESS TO SAVE ALL RH/RMO3 REG'S
JSR PC,FILBUF ; FILL THE BUFFER WITH WORSE CASE PATTERN
2S: JSR R0,RMO3 ; CALL DRIVER HANDLER
FMTDPB
BR 2S ; BRANCH IF QUEUE FAILS
3S: TST FMTDPB+16 ; ALL DONE ?
BEQ 3S ; BRANCH IF NOT
MOV #FMTDPB,R0 ; REPORT IF ANY ERROR
JSR PC,PROCÉS
INCB \$TRK(R0) ; NEXT SURFACE
CMPB #5,\$TRK(R0) ; ALL FIVE SURFACE ARE DONE ?
BHI 2S ; BRANCH IF NOT
CLR \$SEC(R0) ; RESET SECTOR AND TRACK
ADD #128,\$CYL(R0) ; ADVANCE TO NEXT ZONE
CMP #800,\$CYL(R0) ; ALL 7 ZONES ARE DONE ?
BHIS 2S ; BRANCH IF NOT
; EXECUTE WRITE CHECK
; TO DETECT ANY DATA OR HEADER
; PROBLEM
; CHANGE TO WRITE CHECK COMMAND
; R2 POINTS TO SCORE BOARD
; CAN NOT BE DESTORIED
; LOACATE STARTING ADDRESS
MOV #WCKD,SCOMND(R0)
MOV #SELF0,R2
MOV SDEVN,R3
ADD #17,R3
MOV R3,\$CYL(R0) ; STARTING CYLINDER
CLR OFFCOD ; SET NEGATIVE OFFSET FLAG
4S: JSR R0,RMO3 ; CALL DRIVE HANDLER
FMTDPB
BR 4S ; BRANCH IF QUEUE FAILS
5S: TST FMTDPB+16 ; ALL DONE ?

```

3568 011304 001775          BEQ      5$                ;BRANCH IF NOT
3569 011306 012700 037146   MOV      #FMTDPB,R0
3570 011312 004737 013264   JSR     PC,PROCÉS
3571 011316 005760 000016   TST     16(R0)          ;ANY ERROR
3572 011322 100401          BMI     6$                ;YES, THEN DON'T INCREMENT SCORE
3573 011324 105212          INCB   (R2)             ;INCREMENT SCORE
3574 011326 004537 010176   6$:    JSR     R5,OFFST   ;OFFSET
3575 011332 000415          BR     8$                ;BRANCH IF OFFSET FAILS
3576 011334 004737 010400   JSR     PC,MAKEUP      ;EXECUTE WRITE CHECK IN OFFSET MODE
3577 011340 005737 037164   7$:    TST     FMTDPB+16 ;ALL DONE /
3578 011344 001775          BEQ     7$                ;BRANCH IF NOT
3579 011346 012700 037146   MOV     #FMTDPB,R0
3580 011352 004737 013264   JSR     PC,PROCÉS
3581 011356 005760 000016   TST     16(R0)          ;REPORT IF ANY ERROR
3582 011362 100401          BMI     8$                ;ERROR BIT SET ?
3583 011364 105212          INCB   (R2)             ;DON'T INCREMENT SCORE
3584 011366 005737 002640   8$:    TST     OFFCOD     ;INCREMENT SCORE
3585 011372 001006          BNE    9$                ;POSITIVE OFFSET IS DONE ?
3586 011374 005202          INC    R2               ;BRANCH IF DONE
3587 011376 012737 000001 002640  MOV     #1,OFFCOD      ;INCREMENT SCORE BOARD POINTER
3588 011404 000137 011270   JMP     4$                ;SET POSITIVE OFFSET FLAG
3589 011410 105260 000011   9$:    INCB   STRK(R0)   ;POSITIVE OFFSET TEST
3590 011414 005202          INC    R2               ;INCREMENT TO NEXT SURFACE
3591 011416 122760 000005 000011  CMPB   #5,STRK(R0)    ;ADVANCE SCORE BOARD POINTER
3592 011424 101404          BLOS   10$              ;ALL FIVE SURFACES ARE DONE ?
3593 011426 005037 002640   CLR    OFFCOD           ;BRANCH IF ALL DONE
3594 011432 000137 011270   JMP     4$                ;RESET OFFSET DIRECTION
3595 011436 005060 000010   10$:  CLR    $SEC(R0)        ;TRY THE NEXT SURFACE
3596 011442 012702 002546   MOV     #SELFO,R2       ;SURFACE 0, SECTOR 0
3597 011446 062760 000200 000012  ADD    #128,$CYL(R0)    ;RESET SCORE BOARD
3598 011454 022760 001440 000012  CMP    #800,$CYL(R0)   ;ADVANCE TO NEXT ZONE
3599 011462 103404          BLO    11$              ;ALL 7 ZONES ARE DONE ?
3600 011464 005037 002640   CLR    OFFCOD           ;BRANCH IF ALL DONE
3601 011470 000137 011270   JMP     4$                ;RESET OFFSET FLAG
3602 011474 000240          11$:  NOP                    ;TO NEXT ZONE
3603                                     ;TO NEXT TEST
    
```

```

3604                                     ;TEST10 TEST 8 READ COMPATIBLE TEST
3605                                     ;CYLINDERS TESTED : $DEVM+112,$DEVM+112+128XN N=1 TO 5
3606                                     ;THIS TEST SELECT ONE CYLINDER FOR EACH LOGICAL DRIVE FROM
3607                                     ;ZONE 1 TO ZONE 6
3608                                     ;$DEVM : LOGICAL DRIVE #
3609                                     ;DRIVE : PHYSICAL DRIVE #
    
```

```

3610                                     ;*****
3611                                     ;*****
3612                                     ;*****
3613                                     ;*****
3614 011476 000004          †ST10: SCOPE
3615 011500 012737 000001 001174  MOV     #1,STIMES      ;DO 1 ITERATION
3616 011506 012737 000010 001336  MOV     #10,TSTNM     ;LOAD TEST NUMBER
3617 011514 012706 001100          MOV     #STACK,SP     ;INITAIL THE STACK POINTER
3618 011520 012702 002306          MOV     #RDNO,R2      ;CLEAR THE SCORE BOARD OF READ TEST
3619 011524 005022          1$:    CLR    (R2)+
3620 011526 022702 002546          CMP     #RDP4+16.,R2 ;ALL DONE
3621 011532 101374          BHI    1$                ;BRANCH IF NOT
3622 011534 005037 002640          CLR    OFFCOD         ;SET NEGATIVE OFFSET FLAG
3623 011540 012700 037146  LOOP3: MOV     #FMTDPB,R0 ;SET UP DPB BLOCK
    
```


K06

3624	011544	012760	160000	000004		MOV	#-8192.,SWRDM(RO) ;LOAD THE WORD CTR,FULL TRACK
3625	011552	005060	000010			CLR	\$SECT(RO) ;SURFACE 0 AND SECTOR 0
3626	011556	112760	000171	000002		MOV	\$RDATA,\$COMND(RO) ;LOAD THE READ DATA COMMAND
3627	011564	012760	037262	000006		MOV	\$ENDPGM,\$BUF(RO) ;RESET BUFFER ADDRESS
3628	011572	012760	037166	000014		MOV	\$RM.REG.14(RO) ;ADDRESS TO SAVE ALL RH/RMO3 ADDRESS
3629	011600	113710	001222			MOV	DRIVE,(RO) ;LOAD PHY. DRIVE ADDRESS
3630	011604	013703	001266			MOV	\$DEVN,R3 ;LOCATE STARTING CYL
3631	011610	062703	000160			ADD	#112.,R3
3632	011614	010360	000012			MOV	R3,\$CYL(RO) ;STARTING CYL ADDRESS
3633	011620	004037	023620		1\$:	JSR	RO,RMO3 ;CALL THE DRIVE HANDLER
3634	011624	037146				FMTDPB	
3635	011626	000774				BR	1\$
3636	011630	005737	037164		2\$:	TST	FMTDPB+16 ;COMMAND DONE ?
3637	011634	001775				BEQ	2\$;BRANCH IF NOT
3638	011636	012700	037146			MOV	#FMTDPB,RO
3639	011642	004737	013264			JSR	PC,PROCES ;REPORT IF ANY ERROR
3640	011646	004737	010732			JSR	PC,LABAD ;LOCATE START AND STOP ADDRESS
3641	011652	005760	000016			TST	16(RO) ;ANY ERROR ?
3642	011656	100006				BPL	3\$;BRANCH,IF NONE
3643	011660	023737	001360	001356		CMP	CMSEC,STARSC ;ON THE SAME SECTOR ADDRESS
3644	011666	101404				BLOS	4\$
3645	011670	005337	001360			DEC	CMSEC ;DECREMENT ONE SECTOR COUNT
3646	011674	004737	010512		3\$:	JSR	PC,SCORE ;UPDATE THE SCORE
3647	011700	005760	000022		4\$:	TST	\$RMWC(RO) ;WORD COUNT=0
3648	011704	001407				BEQ	5\$;BRANCH IIF IT IS
3649	011706	116060	000026	000010		MOV	\$RMDA(RO),\$SECT(RO) ;UPDATE THE NEW STARTING SECTOR
3650	011714	016060	000022	000004		MOV	\$RMWC(RO),\$SWRDM(RO) ;UPDATE WORD COUNT
3651	011722	000736				BR	1\$;CONTINUE
3652	011724	012700	037146		5\$:	MOV	#FMTDPB,RO ;RESET THE DPB BLOCK
3653	011730	012760	160000	000004		MOV	#-8192.,SWRDM(RO) ;FULL TRACK WORD COUNT
3654	011736	105060	000010			CLRB	\$SECT(RO) ;RESET TO SECTOR 0,TRACK NOT CHANGED
3655	011742	004537	010176		6\$:	JSR	R5,OFFST ;CALL OFFSET
3656	011746	000440				BR	10\$;BRANCH IF OFFSET FAILS
3657	011750	004737	010400			JSR	PC,MAKEUP ;EXECUTE READ DATA IN OFFSET MODE
3658	011754	005737	037164		7\$:	TST	FMTDPB+16 ;OFFSET READ IS DONE ?
3659	011760	001775				BEQ	7\$;BRANCH IF NOT
3660	011762	012700	037146			MOV	#FMTDPB,RO ;REPORT IF ANY ERROR
3661	011766	004737	013264			JSR	PC,PROCES
3662	011772	004737	010732			JSR	PC,LABAD ;LOCATE THE START AND STOP ADDRESSES
3663	011776	005760	000016			TST	16(RO) ;ANY ERROR ?
3664	012002	100006				BPL	8\$;BRANCH,IF NONE
3665	012004	023737	001360	001356		CMP	CMSEC,STARSC ;START AND STOP SECTOR ADDRESSES IS O.K ?
3666	012012	101404				BLOS	9\$;BRANCH IF NOT
3667	012014	005337	001360			DEC	CMSEC ;END SECTOR IS ERROR SECTOR
3668	012020	004737	010512		8\$:	JSR	PC,SCORE ;UPDATE THE SCORE
3669	012024	005760	000022		9\$:	TST	\$RMWC(RO) ;WORD COUNT IS 0
3670	012030	001407				BEQ	10\$;BRANCH IF IT IS
3671	012032	116060	000026	000010		MOV	\$RMDA(RO),\$SECT(RO) ;UPDATE SECTOR ADDRESS
3672	012040	016060	000022	000004		MOV	\$RMWC(RO),\$SWRDM(RO) ;UPDATE WORD COUNT
3673	012046	000735				BR	6\$;LOOPING UNTIL CURRENT TRACK IS DONE
3674	012050	062760	000200	000012	10\$:	ADD	#128.,\$CYL(RO) ;ADJUST CYLINDER TO NEXT ZONE
3675	012056	022760	001377	000012		CMP	#767.,\$CYL(RO) ;ALL 6 ZONES ARE DONE ?
3676	012064	103402				BLO	11\$;BRANCH IF ALL DONE
3677	012066	000137	011620			JMP	1\$
3678	012072	013703	001266		11\$:	MOV	\$DEVN,R3 ;LOCATE STARTING CYLINDER
3679	012076	062703	000160			ADD	#112.,R3 ;

LOG

MAINDEC-11-DZRM1 AD/00
DZRMIA.P11 21-JUL-77 15:44

RM03 DRIVE COMPATIBILITY TEST MACY11 30(1046) 21-JUL-77 16:51 PAGE 76
GET VALUE FOR SOFTWARE SWITCH REGISTER

SEQ 0075

```

3680 012102 010360 000012      MOV      R3, SCYL(R0)      ; STARTING CYLINDER
3681 012106 105260 000011      INCB    STRK(R0)          ; TO NEXT SURFACE
3682 012112 122760 000005 000011  CMPB    #5, STRK(R0)      ; ALL 5 SURFACE ARE DONE
3683 012120 101402          BLOS    12$              ; BRANCH IF ALL DONE
3684 012122 000137 011620      JMP     1$                ; LOOPING UNTIL CURRENT TRACK IS DONE
3685 012126 005737 002640      12$:    TST     OFFCOD      ; OFFSET CODE = POSITIVE ?
3686 012132 001005          BNE     13$              ; IF EQUAL THEN ALL DONE
3687 012134 012737 000001 002640  MOV     #1, OFFCOD        ; SET POSITIVE OFFSET FLAG
3688 012142 000137 011540      JMP     LOOP3
3689 012146 000240      13$:    NOP                    ; TO NEXT TEST
3690
3691
3692
3693
3694          ; TEST11 TEST 9
3695          ; REPORT THE TEST SCORES
3696          ; DIRECT THE OPERATOR TO CHANGE PACK AND MOUNT TO OTHER DRIVE
3697
3698          ; SDEVM : LOGICAL DRIVE #, SHOULD NOT BE UPDATED BEFORE THE REPORT IS COMPLETED.
3699
3700          ; BADSEC: ACCEPTANCE FLAG, BADSEC = 0, ALL DRIVES ARE COMPATIBLE
3701          ; BADSEC = -1, DRIVES ARE NOT COMPATIBLE
3702          ; CMTRK : TRACK (HEAD) NUMBER FOR CONTROLLING THE SCORE TYPING.
3703
3704
3705
3706
3707          ; *****
3708 012150 000004      ; TEST11: SCOPE
3709 012152 012737 000001 001174  MOV     #1, STIMES        ; DO 1 ITERATION
3710 012160 012737 000011 001336  MOV     #11, TSTNM       ; LOAD THE TEST NUMBER
3711 012166 012706 001100      MOV     #STACK, SP       ; LOAD THE STACK POINTER
3712          ; ADJUST THE READ COMPATIBLE TEST SCORE
3713          ; IF THE SELF READ TEST SCORE IS TOO LOW
3714          ; DUMMY SCORE BOARD ADDRESSES
3715          ;
3716          ; POSITIVE OFFSET READ TEST SCORE ADDRESS
3717          ; POSITIVE OFFSET TEST SCORE
3718          ; NEGATIVE OFFSET READ TEST SCORE
3719          ; NEGATIVE OFFSET READ COMPATIBLE TEST SCORE
3720          ; R1= SURFACE (TRACK) NUMBER
3721          ; SELF READ TEST SCORE IS TOO LOW
3722          ; BRANCH IF NOT
3723          ; INCREMENT READ COMPATIBLE SCORE FOR ALL 16 DIRVES
3724          ; ADJUST SCORE BY ADDING 6
3725          ;
3726          ; UPDATE SCORE AND POINTS TO NEXT DRIVE
3727          ; ALL DRIVES ARE UPDATED ?
3728          ; BRANCH IF NOT
3729          ;
3730          ; ADJUST POINTER OF SCORE BOARD ADDRESS
3731          ; UPDATE THE SELF TEST SCORE ADDRESS
3732          ; ALL FIVE SURFACES ARE DONE ?
3733          ; BRANCH IF NOT
3734          ; GET NEXT PAIR
3735          ; EXIT THE THEY ARE 0

```


3736	012272	001351				BNE	1\$		
3737									
3738									: SET ACCEPTANCE FLAG, INITIALIZE THE
3739									: TABLE POINTERS AND TRACK NUMBER
3740	012274	005037	001340			CLR	BADSEC		: SET ACCEPTANCE FLAG
3741	012300	005037	001362			CLR	CMTRK		: START FROM TRACK 0
3742									
3743	012304	005001				CLR	R1		: START FROM LOG DRV 0
3744	012306	012702	002046			MOV	#OVWNO, R2		: SCORE BOARD BASE ADDRESS
3745	012312	012703	000001			MOV	#BIT0, R3		: BIT POSITION FOR LOG DRV 0
3746	012316	030337	001776			BIT	R3, ASNLST	LOOP4:	: IS THE LOG DRV UNDER TEST ?
3747	012322	001502				BEQ	LOOPS		: BRANCH IF NOT
3748	012324	005037	001364			CLR	NULINE		: NEW LINE INDICATOR
3749	012330	123701	001266			CMPB	\$DEVN, R1		: SELF SCORE ?
3750	012334	001434				BEQ	SPATH		: BRANCH IF IT IS
3751	012336	122712	000016			CMPB	#14., (R2)	KPATH:	: OVERWRITE NEG OFFSET < 14 ?
3752	012342	101403				BLOS	1\$: BRANCH IF NOT
3753	012344	004537	012766			JSR	R5, PRINT		: REPORT EXCEPTIONS
3754	012350	000001				000001			: COLUMN POSITION ON REPORT
3755	012352	122762	000016	000120	1\$:	CMPB	#14., 80. (R2)		: OVERWRITE POS OFFSET < 14 ?
3756	012360	101403				BLOS	2\$: BRANCH IF NOT
3757	012362	004537	012766			JSR	R5, PRINT		: REPORT EXCEPTIONS
3758	012366	000002				000002			: COLUMN POSITION ON REPORT
3759	012370	122762	000014	000240	2\$:	CMPB	#12., 160. (R2)		: READ NEG OFFSET < 6
3760	012376	101403				BLOS	3\$: BRANCH IF NOT
3761	012400	004537	012766			JSR	R5, PRINT		: REPORT EXCEPTIONS
3762	012404	000003				000003			: COLUMN POSITION ON REPORT
3763	012406	122762	000014	000360	3\$:	CMPB	#12., 240. (R2)		: READ POS OFFSET < 12 ?
3764	012414	101445				BLOS	LOOPS		: BRANCH IF NOT
3765	012416	004537	012766			JSR	R5, PRINT		: REPORT EXECPTIONS
3766	012422	000004				000004			: COLUMN POSITION ON REPORT
3767	012424	000441				BR	LOOPS		: EXIT
3768	012426	122712	000016			CMPB	#14., (R2)	SPATH:	: OVERWRITE NEG OFF < 14
3769	012432	101403				BLOS	1\$: BRANCH IF NOT
3770	012434	004537	012766			JSR	R5, PRINT		: REPORT EXECPTION
3771	012440	000001				000001			: COLUMN POSITION ON REPORT
3772	012442	122762	000016	000120	1\$:	CMPB	#14., 80. (R2)		: OVERWRITE POS OFFSET < 14
3773	012450	101403				BLOS	2\$: BRANCH IF NOT
3774	012452	004537	012766			JSR	R5, PRINT		: REPORT EXCEPTION
3775	012456	000002				000002			: COLUMN POSITION ON REPORT
3776	012460	013704	001362		2\$:	MOV	CMTRK, R4		: LOCATE THE SELF TEST SCORE
3777	012464	006304				ASL	R4		: WORD INDEX
3778	012466	122764	000006	002546		CMPB	#6, SELFO(R4)		: SELF READ NEG OFFSET < 6 ?
3779	012474	101403				BLOS	3\$: BRANCH IF NOT
3780	012476	004537	012766			JSR	R5, PRINT		
3781	012502	000003				000003			: REPORT EXCEPTIONS
3782	012504	013704	001362		3\$:	MOV	CMTRK, R4		: LOCATE THE SELF TEST SCORE
3783	012510	006304				ASL	R4		: WORD INDEX
3784	012512	122764	000006	002547		CMPB	#6, SELFO+1(R4)		: SELF READ POS OFFSET < 6 ?
3785	012520	101403				BLOS	LOOPS		: BRANCH IF NOT
3786	012522	004537	012766			JSR	R5, PRINT		: REPORT EXCEPTIONS
3787	012526	000004				000004			: COLUMN POSITION FOR REPORT PRINTING
3788	012530	005201				INC	R1	LOOPS:	: INCREASE THE LOGICAL DRIVE #
3789	012532	000241				CLC			: ADJUST THE BIT POSITION
3790	012534	006103				ROL	R3		: POINTS TO NEXT DRIVE
3791	012536	005202				INC	R2		: UPDATE SCORE BOARD BASE ADDRESS

3792	012540	022701	000017		CMP	#15.,R1	: ALL DRIVES ARE CHECK ?
3793	012544	103264			BHIS	LOOP4	: BRANCH IF NOT ALL DONE
3794	012546	005237	001362	LOOP6:	INC	CMTRK	: NEXT SURFACE
3795	012552	012702	002046		MOV	#OVWNO,R2	: RESET SCORE BOARD BASE ADDRESS
3796	012556	005001			CLR	R1	: LOGICAL DRIVE START FROM 0
3797	012560	013703	001362		MOV	CMTRK,R3	: LOCATE THE SCORE BOARD BASE ADDRESS
3798	012564	001404			BEQ	2\$: BRANCH IF ON SURFACE 0
3799	012566	062702	000020	1\$:	ADD	#16.,R2	: FOR EACH SURFACE, 16 BYTES
3800	012572	005303			DEC	R3	: ALL SURFACE DONE ?
3801	012574	001374			BNE	1\$: BRANCH IF NOT
3802	012576	012703	000001	2\$:	MOV	#BIT0,R3	: BIT MAP POSITION FOR DRIVE 0
3803							: AT THIS POINT:
3804							: R1=0, LOGICAL DRIVE # START FROM 0
3805							: R2 ADDRESS OF SCORE BOARD BASE ADDRESS
3806							: R3=1 BIT MAP INDICATE LOGICAL DRIVE 0
3807	012602	122737	000005	001362	CMPB	#5,C,TRK	: ALL SURFACES ARE DONE ?
3808	012610	101242			BHI	LOOP4	: CONTINUE
3809	012612	005737	001340		TST	BADSEC	: ACCEPTANCE FLAG SET ?
3810	012616	100402			BMI	DISMNT	: NO, THEN BRANCH
3811	012620	104401	001205		TYPE	,SCRLF	
3812					TYPE	,MSG15 ;** ALL	: DRIVES ARE COMPATIBLE **
3813	012624	104401	001205		TYPE	,SCRLF	: CR-LF
3814	012630	012777	013256	010242	MOV	#IDLEX,ARMVEC	: RESET THE INTERRUPT VECTOR
3815	012636	005077	010240		CLR	ARMVEC+2	: CLEAR THE INTERRUPT VECTOR
3816	012642	104401	035323		TYPE	,MSG12	: UNLOAD AND DISMOUNT MESSAGE
3817	012646	104401	001272		TYPE	,SCDW2	: SYSTEM NAME
3818	012652	013746	001222		MOV	DRIVE,-(SP)	: PHYSICAL DRIVE NUMBER
3819	012656	104403			TYPOS		
3820	012660	002			.BYTE	2	
3821	012661	000			.BYTE	0	
3822	012662	104401	035341		TYPE	,MSG13	: MESSAGE TYPE R<CR> WHEN READY
3823	012666	104411		1\$:	RDLIN		: READ IN ONE LINE
3824	012670	012605			MOV	(SP)+,R5	: LOCATE THE READ IN LINE
3825	012672	122527	000122		CMPB	(R5)+,#'R	: READY ?
3826	012676	001373			BNE	1\$: TRY AGAIN IF NOT
3827	012700	105715			TSTB	(R5)	: TERMINATOR ?
3828	012702	001371			BNE	1\$: BRANCH IF NOT
3829	012704	005002		DEASG:	CLR	R2	: LOGICAL DRIVE NUMBER
3830	012706	012703	000001		MOV	#BIT0,R3	: BIT MAP OF ASSIGNED DRIVE
3831	012712	020237	001266	1\$:	CMP	R2,\$DEV	: IS THE LOGICAL DRIVE UNDER TEST ?
3832	012716	001404			BEQ	2\$: BRANCH IF IT IS
3833	012720	000241			CLC		: SHIFT THE BIT MAP FOR
3834	012722	006103			ROL	R3	: NEXT DRIVE
3835	012724	005202			INC	R2	: INCREMENT THE LOGICAL DRIVE #
3836	012726	000771			BR	1\$: LOOPING, UNTIL THE LOGICAL DRIVE LOCATED
3837	012730	040337	002002	2\$:	BIC	R3,ASSGN2	: CLEAR THE ASSIGNED BIT
3838	012734	001412			BEQ	XEND2	: BRANCH IF NO MORE DRIVES
3839	012736	005237	001266		INC	\$DEV	: INCREMENT THE LOGICAL DRIVE #
3840	012742	013702	001266		MOV	\$DEV,R2	: UPDATE SYSTEM BLOCK ADDRESS
3841	012746	006302			ASL	R2	
3842	012750	016237	002600	001270	MOV	BLKADR(R2),\$CDW1	: SYSTEM BLOCK ADDRESS
3843	012756	000137	007056		JMP	TST5	: JUMP TO TEST 5 FOR OTHER DRIVE
3844							
3845							
3846	012762	000137	017224	XEND2:	JMP	SEOP	: END OF PASS
3847							

3848
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3900
3901
3902
3903

012766
012766 010146
012770 010246
012772 010346
012774 010446
012776 005737 001340
013002 100432

013004 104401 001205
013010 104401 035503
013014 104401 001272
013020 013746 001222
013024 104403
013026 002
013027 000
013030 104401 001205
013034 104401 035526
013040 104401 001205
013044 104401 035622
013050 104401 001205
013054 012737 177777 001340
013062 012737 000001 001374
013070 005737 001364
013074 001045
013076 104401 001205
013102 104401 001205
013106 013746 001362
013112 104403
013114 002
013115 000
013116 104401 035725
013122 123701 001266
013126 001005

:PRINT ROUTINE
:NAME PRINT
:PRINT EXCEPTIONS FOR TEST SCORE
:PARAMETER USED
NULINE = 0 A NEW LINE TO PRINT
= 1 - 4 COLUMN NUMBER TO PRINT THE EXCEPTION MARK

BADSEC = 0 FIRST EXCEPTION DETECTED
BADSEC = -1 NOT FIRST EXCEPTION (DON'T PRINT THE TITLE)

CALLING SEQ:
JSR R5,PRINT
NUMBER COLUMN NUMBER
RET

R1: LOGICAL DRIVE #
SDEV: LOGICAL DRIVE UNDER TEST
R3: BIT POSITION OF LOGICAL DRIVE IN R1

PRINT:
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
TST BADSEC ;:FIRST EXCEPTION
BMI 1\$;:BRANCH IF NOT,DON'T HAVE TO PRINT
TITLE
TYPE ,SCRLF ;:CRLF
TYPE ,MESG16 ;:SCORES FOR DRIVE --
TYPE ,SCDW2 ;:SYSTEM NAME
MOV DRIVE,-(SP) ;:TYPE THE PHYSICAL DRIVE #
TYPOS
.BYTE 2
.BYTE 0
TYPE ,SCRLF ;:CR-LF
TYPE ,MESG17 ;:SUB TITLE 1
TYPE ,SCRLF
TYPE ,MESG18 ;:SUB TITLE 2
TYPE ,SCRLF
MOV #-1,BADSEC ;:RESET THE ACCEPTANCE FLAG
MOV #1,FAULT ;:NOT COMPATIBLE FLAG
TST NULINE ;:NEW LINE ?
BNE 5\$;:BRANCH IF NOT
TYPE ,SCRLF
TYPE ,SCRLF
MOV CMTRK,-(SP) ;:TYPE THE SURFACE NUMBER
TYPOS
.BYTE 2
.BYTE 0
TYPE TAB ;:TYPE "TABLE" CHARACTER
CMPB \$DEV, R1 ;:SCORE FOR \$DEV ITSELF ?
BNE 2\$;:BRANCH IF NOT

```

3904 013130 104401 035717 TYPE ,SELF ;MESSAGE: SELF .
3905 013134 104401 035725 TYPE ,TAB
3906 013140 000420 BR 4$ ;NEXT STEP
3907 013142 006301 ASL R1 ;LOCATE THE SYS HISTORY FILE
3908 013144 016137 002600 013174 2$: MOV BLKADR(R1),2#3$ ;LOCATE THE SYSTEM NAME AND DRIVE #
3909 013152 006201 ASR R1 ;RESTORE DRIVE #
3910 013154 062737 000014 013174 ADD #SSYSNM,2#3$ ;LOCATE THE SYSTEM NAME AND DRIVE #
3911 013162 104401 036121 TYPE ,LINS
3912 013166 104401 036121 TYPE ,LINS
3913 013172 104401 TYPE ;TYPE THE SYSTEM AND DRIVE
3914 013174 000000 3$: .WORD 0 ;ADDRESS FOR TYPING MESSAGE
3915 013176 104401 035725 TYPE ,TAB
3916 013202 012737 000001 001364 4$: MOV #1,NULINE ;INDICATE PRINTER STOPS AT COULMN 1
3917 013210 012504 5$: MOV (R5)+,R4 ;RETRIEVE THE DESIRED COLUMN # FROM
3918 ;CALLING ROUTINE
3919 013212 020437 001364 6$: CMP R4,NULINE ;ON THE RIGHT COLUMN ?
3920 013216 103412 BLO 8$ ;IF LOW NOT PRINT
3921 013220 001405 BEQ 7$ ;BRANCH IF LOCATED
3922 013222 104401 035725 TYPE ,TAB ;ADVANCE TO NEXT COLUMN
3923 013226 005237 001364 INC NULINE ;INCREMENT COLUMN CTR
3924 013232 000767 BR 6$ ;CHECK AGAIN
3925 013234 104401 035727 7$: TYPE ,MARKX ;THE EXCEPTION MARK * 0
3926 013240 005237 001364 INC NULINE ;NEXT COLUMN
3927 013244 8$:
3928 013244 012604 MOV (SP)+,R4 ;:POP STACK INTO R4
3929 013246 012603 MOV (SP)+,R3 ;:POP STACK INTO R3
3930 013250 012602 MOV (SP)+,R2 ;:POP STACK INTO R2
3931 013252 012601 MOV (SP)+,R1 ;:POP STACK INTO R1
3932 013254 000205 RTS R5 ;EXIT
3933
3934
3935
3936 013256 000240 IDLEX: NOP ;RESET ALL RH VECTOR FOR MOUNT AND DISMOUNT
3937 013260 000240 NOP
3938 013262 000002 RTI ;EXIT
3939 ;PROCESS THE ORDER TERMINATION
3940
3941 013264 111037 001326 PROCES: MOVB (R0),UNIT ;DRIVE NUMBER FOR ANY ERROR MESSAGES
3942 013270 005760 000016 TST STATUS(R0) ;SEE IF DRIVER SIGNALLED AN ERROR
3943 013274 100421 BMI ERPROC ;BR IF ERROR
3944 013276 032760 100000 000020 BIT #BIT15,SRMCS1(R0) ;SEE IF 'SC' SET
3945 013304 001410 BEQ 1$ ;BR IF NOT SET
3946 013306 032760 040000 000020 BIT #BIT14,SRMCS1(R0) ;SEE IF 'TRE' SET
3947 013314 001011 BNE ERPROC ;BR IF SET
3948 013316 032760 040000 000032 BIT #BIT14,SRMDS(R0) ;SEE IF 'ERR' SET
3949 013324 001005 BNE ERPROC ;BR IF SET
3950 013326 004737 014276 1$: JSR PC,CKERR ;NO ERROR, CHECK ERROR BITS ANYWAY
3951 013332 004737 014344 JSR PC,CKBUS ;NO ERROR, CHECK BUS ADDR & WC
3952 013336 000207 2$: RTS PC ;RETURN
3953
3954 ;ORDER TERMINATED WITH AN ERROR - PROCESS THE ERROR
3955
3956 013340 032760 000200 000016 ERPROC: BIT #BIT07,STATUS(R0) ;DONE BIT SET ?
3957 013346 001402 BEQ ERPC1 ;BR IF ORDER DIDN'T COMPLETE NORMALLY
3958 013350 000137 013516 JMP DONE ;PROCESS ERROR WITH 'DONE' BIT SET
3959

```



```

3960 ;PROCESS ORDER COMPLETION WITH 'ERROR' & 'DONE NOT' BITS
3961
3962 013354 032760 010000 000016 ERPRC1: BIT #BIT12,STATUS(RO) ;SEE IF DRIVE WAS UNSAFE
3963 013362 001025 BNE PUNSAF ;BR IF YES
3964 013364 032760 004000 000016 BIT #BIT11,STATUS(RO) ;PARITY ERROR OCCURRED
3965 013372 001025 BNE UCPAR ;BR IF IT DID
3966 013374 032760 002000 000016 BIT #BIT10,STATUS(RO) ;FATAL PARITY ERROR?
3967 013402 001025 BNE FALPAR ;BR IF THERE IS ONE
3968 013404 032760 001000 000016 BIT #BIT09,STATUS(RO) ;TIMEOUT?
3969 013412 001025 BNE SWTIM ;BR IF YES
3970 013414 032760 040002 000016 BIT #BIT14!BIT01,STATUS(RO) ;DRIVE WENT OFFLINE ?
3971 013422 001025 BNE OFLIN ;BR IF IT DID
3972 013424 032760 000004 000016 BIT #BIT2,STATUS(RO) ;PORT REQUEST TIME OUT ?
3973 013432 001025 BNE PRTIM ;BR IF IT DID
3974 013434 000207 RTS PC ;ERROR. RETURN
3975
3976 ;DRIVE IS PERSISTENTLY UNSAFE
3977
3978 013436 PUNSAF:
3979 013436 104414 032216 DISPLY EM12
3980 013442 000137 014264 JMP DUMP
3981
3982 ;UNCORRECTABLE MASSBUS PARITY ERROR OCCURRED
3983
3984 013446 UCPAR:
3985 013446 104414 032120 DISPLY EM10
3986 013452 000137 014264 JMP DUMP
3987
3988 ;'FATAL' MASSBUS PARITY ERROR OCCURRED
3989
3990 013456 FALPAR:
3991 013456 104414 032163 DISPLY EM11
3992 013462 000137 014264 JMP DUMP
3993
3994 ;SOFTWARE TIMEOUT OCCURRED
3995
3996 013466 SWTIM:
3997 013466 104414 032247 DISPLY EM13
3998 013472 000137 014264 JMP DUMP
3999
4000 ;DRIVE WENT OFFLINE
4001
4002 013476 OFLIN:
4003 013476 104414 032321 DISPLY EM14
4004 013502 000137 014264 JMP DUMP
4005
4006 ;PORT REQUEST TIMEOUT ERROR
4007
4008 013506 PRTIM:
4009 013506 104414 032344 DISPLY EM15
4010 013512 000137 014264 JMP DUMP
4011
4012 ;PROCESS ORDER COMPLETION WITH 'ERROR' & 'DONE' BITS SET
4013 013516 032760 000030 000016 DONE: BIT #BIT04!BIT03,STATUS(RO) ;UNSAFE OCCURRED
4014 013524 001402 BEQ +6 ;BR IF NOT
4015 013526 000137 014254 JMP UNSAF ;REPORT UNSAFE

```

E07

```

4016 013532 032760 040000 000030 BIT #BIT14,SRMCS2(RO) ;IS 'MCE' SET ?
4017 013540 001402 BEQ .+6 ;BRANCH IF NOT SET
4018 013542 000137 JMP WCKER ;WRITE CHECK ERROR
4019 013546 032760 040000 000032 BIT #BIT14,SRMDS(RO) ;CHECK 'ERR'
4020 013554 001002 BNE 1$ ;BR IF SET
4021 013556 000137 JMP TRFER ;PROCESS 'TRE'
4022 013562 032760 000400 000034 1$: BIT #BIT08,SRMER1(RO) ;'HCRC' SET?
4023 013570 001402 BEQ .+6 ;BR IF NOT
4024 013572 000137 JMP HCRCER ;PROCESS 'HCRC'
4025 013576 032760 000020 000034 BIT #BIT04,SRMER1(RO) ;'FMT' SET?
4026 013604 001402 BEQ .+6 ;BR IF NOT SET
4027 013606 000137 JMP CKFMT ;CHECK FORMAT ERROR
4028 013612 032760 000200 000034 BIT #BIT07,SRMER1(RO) ;'HCE' SET?
4029 013620 001402 BEQ .+6 ;BR IF NOT SET
4030 013622 000137 JMP CKHCE ;CHECK 'HCE' ERROR
4031 013626 032760 020000 000034 BIT #BIT13,SRMER1(RO) ;'OPI' SET?
4032 013634 001402 BEQ .+6 ;BR IF NOT SET
4033 013636 000137 JMP OPIER ;REPORT 'OPI'
4034 013642 032760 000010 000034 BIT #BIT3,SRMER1(RO) ;'PAR' SET?
4035 013650 001402 BEQ .+6 ;BR IF NOT SET
4036 013652 000137 JMP PARER ;REPORT 'PAR'
4037 013656 032760 000040 000034 BIT #BIT5,SRMER1(RO) ;'WCF' SET?
4038 013664 001402 BEQ .+6 ;BR IF NOT SET
4039 013666 000137 JMP WCFER ;REPORT 'WCF'
4040 013672 032760 002000 000034 BIT #BIT10,SRMER1(RO) ;'IAE' SET?
4041 013700 001402 BEQ .+6 ;BR IF NOT SET
4042 013702 000137 JMP IAEER ;REPORT 'IAE'
4043 013706 032760 004000 000034 BIT #BIT11,SRMER1(RO) ;'WLE' SET?
4044 013714 001402 BEQ .+6 ;BR IF NOT SET
4045 013716 000137 JMP WLEER ;REPORT 'WLE'
4046 013722 032760 001000 000034 BIT #BIT9,SRMER1(RO) ;'AOE' SET?
4047 013730 001405 BEQ 2$ ;BR IF NOT SET
4048 013732 032760 002000 000032 BIT #BIT10,SRMDS(RO) ;'LST' SET?
4049 013740 001401 BEQ 2$ ;BR IF NOT SET
4050 013742 000207 RTS PC ;'AOE' & 'LST' SET, EXIT
4051 013744 032760 010000 000034 2$: BIT #BIT12,SRMER1(RO) ;SEE IF 'DTE' SET
4052 013752 001402 BEQ .+6 ;BR IF NOT
4053 013754 000137 JMP DTEER ;REPORT 'DTE' ERROR
4054 013760 005760 000034 TST SRMER1(RO) ;SEE IF 'DCK' SET
4055 013764 100002 BPL .+6 ;BR IF NOT
4056 013766 000137 JMP DCKER ;PROCESS 'DCK'
4057 013772 032760 060000 000062 BIT #BIT14:BIT13,SRMER2(RO) ;'SKI' OR 'OCYL' SET
4058 014000 001006 BNE 3$ ;BR IF IT IS
4059 014002 032760 100000 000062 BIT #BIT15,SRMER2(RO) ;BAD SPOT ?
4060 014010 001004 BNE 4$ ;BRANCH IF SO
4061 014012 000137 JMP DRVER ;REPORT ERROR
4062 014016 000137 014234 3$: JMP SKIER ;REPORT DRIVE ERROR
4063 014022 000207 4$: RTS PC ;
4064
4065 ;PROCESS DATA ('DCK') CHECK ERROR
4066
4067 014024 DCKER:
4068 014024 104414 032421 DISPLY EM21
4069 014030 000137 014264 JMP DUMP
4070
4071 ;WRITE CHECK ERROR PROCESSING
    
```


4072						
4073	014034			000034	WCKER:	
4074	014034	032760	100000		BIT	#BIT15,SRMER1(RO) ;DCK BIT SET ?
4075	014042	001004			BNE	1\$;BRANCH IF SET
4076	014044	104414	032525		DISPLY	EM23
4077	014050	000137	014264		JMP	DUMP
4078	014054	104414	032452		1\$: DISPLY	EM22
4079	014060	000137	014264		JMP	DUMP
4080						
4081					;REPORT 'HCRC' ERROR	
4082						
4083	014064				HRCRER:	
4084	014064	104414	032400		DISPLY	EM20
4085	014070	000137	014264		JMP	DUMP
4086						
4087					;REPORT DRIVE ERROR	
4088						
4089	014074				DRVER:	
4090	014074	104414	033015		DISPLY	EM30
4091	014100	000137	014264		JMP	DUMP
4092						
4093					;PROCESS FORMAT ('FER') ERROR	
4094						
4095	014104				CKFMT:	
4096	014104	104414	032604		DISPLY	EM24
4097	014110	000137	014264		JMP	DUMP
4098						
4099					;PROCESS HEADER COMPARE ('HCE') ERROR	
4100						
4101	014114				CKHCE:	
4102	014114	104414	032652		DISPLY	EM25
4103	014120	000137	014264		JMP	DUMP
4104					;POSSIBLE POSITIONING ERROR	
4105						
4106	014124	104414	034153		POSER:	
4107	014130	000137	014264		DISPLY	EM51
4108					JMP	DUMP
4109	014134				;REPORT 'OPI' ERROR	
4110	014134	104414	033047		OPIER:	
4111	014140	000137	014264		DISPLY	EM31
4112					JMP	DUMP
4113					;REPORT 'DTE' ERROR	
4114	014144				DTEER:	
4115	014144	104414	033112		DISPLY	EM32
4116	014150	000137	014264		JMP	DUMP
4117					;REPORT 'PAR' ERROR	
4118						
4119	014154				PARER:	
4120	014154	104414	033145		DISPLY	EM33
4121	014160	000137	014264		JMP	DUMP
4122					;REPORT 'IAE' ERROR	
4123						
4124	014164				IAEER:	
4125	014164	104414	033264		DISPLY	EM35
4126	014170	000137	014264		JMP	DUMP
4127						

```

4128 ;REPORT WLE ERROR
4129
4130 014174 WLEER:
4131 014174 104414 033322 DISPLY EM36
4132 014200 000137 014264 JMP DUMP
4133 ;REPORT FORMAT ERROR
4134
4135 014204 FMTER:
4136 014204 104414 032733 DISPLY EM26
4137 014210 000137 014264 JMP DUMP
4138 ;REPORT HEADER COMPARE ERROR
4139
4140 014214 HCEER: DISPLY EM27
4141 014220 000137 014264 JMP DUMP
4142
4143 ;PROCESS CONTROL/INTERFACE TRANSFER ERROR
4144
4145 014224 TRFER:
4146 014224 104414 033435 DISPLY EM40
4147 014230 000137 014264 JMP DUMP
4148
4149 ;PROCESS 'SKI' OR 'OCYL'
4150
4151 014234 SKIER:
4152 014234 104414 034115 DISPLY EM50
4153 014240 000137 014264 JMP DUMP
4154 ;REPORT WRITE CLOCK FAILURE
4155
4156 014244 WCFER:
4157 014244 104414 033222 DISPLY EM34
4158 014250 000137 014264 JMP DUMP
4159
4160 ;REPORT DRIVE UNSAFE ERROR
4161
4162 014254 UNSAF:
4163 014254 104414 034216 DISPLY EM60
4164 014260 000137 014264 JMP DUMP
4165
4166
4167 014264 DUMP:
4168 014264 104007 ERROR 7
4169 014266 104010 ERROR 10
4170 014270 104011 ERROR 11
4171 014272 104012 ERROR 12
4172 014274 000207 RTS PC
4173
4174
4175 ;CHECK ERROR BITS IN THE RH11 AND RMO3 REGSTERS
4176
4177 014276 032760 060000 000020 CKERR: BIT #60000, $RMCS1(RO) ;SEE IF 'TRE' OR 'MCPE'
4178 014304 001012 BNE 1$ ;YES
4179 014306 032760 177400 000030 BIT #177400, $RMCS2(RO) ;ERROR BITS IN CS2 /
4180 014314 001006 BNE 1$ ;YES
4181 014316 005760 000034 TST $RMER1(RO) ;ANY ERROR IN ER1
4182 014322 001003 BNE 1$ ;YES
4183 014324 005760 000062 TST $RMER2(RO) ;ANY ERROR IN ER2

```


H07

```

4184 014330 001404
4185 014332 104414 033671
4186 014336 000137 014264
4187 014342 000207
4188
4189
4190
4191 014344 005760 000022
4192 014350 001011
4193 014352 016046 000004
4194 014356 005416
4195 014360 006316
4196 014362 066016 000006
4197 014366 022660 000024
4198 014372 001404
4199 014374 104414 033477
4200 014400 000137 014264
4201 014404 000207
4202
4203
4204 014406 104401 001205
4205 014412 104401 036351
4206 014416 104401 001205
4207 014422 016001 000024
4208 014426 016046 000004
4209 014432 005416
4210 014434 066016 000022
4211 014440 005046
4212 014442 012746 000400
4213 014446 004737 015472
4214 014452 005716
4215 014454 001403
4216 014456 006316
4217 014460 161601
4218 014462 000402
4219 014464 162701 001000
4220 014470 062706 000004
4221 014474 012702 000007
4222 014500 010146
4223 014502 004737 014666
4224 014506 020160 000024
4225 014512 001412
4226 014514 104414 036121
4227 014520 012146
4228 014522 004737 014666
4229 014526 005302
4230 014530 001366
4231 014532 104414 001205
4232 014536 000756
4233 014540 104414 001205
4234 014544 000207
4235
4236
4237
4238 014546 104412
4239 014550 016001 000006

;BRANCH IF NO ERROR
1$:  BEQ 2$ ;TYPE ALL REGISTERS
    DISPLY ,EM44
    JMP DUMP
2$:  RTS PC ;CHECK BUS ADDRESS REGISTER AND WORD COUNT REGISTER

CKBUS: TST $RMWC(RO) ;WORD COUNT = 0
       BNE 1$ ;NO
       MOV $WRDM(RO),-(SP) ;WORD LENGTH
       NEG (SP) ;GET THE POSITIVE NUMBER OF WORD COUNT
       ASL (SP) ;BYTE COUNT
       ADD $BUF(RO),(SP)
       CMP (SP)+,$RMB(RO)
1$:  DISPLY ,EM41 ;TYPE ALL REGISTERS
    JMP DUMP
2$:  RTS PC ;ROUTINE TO DISPLAY THE SECTOR WHICH GAVE THE HARD ERROR

PRTBAD: TYPE ,SCRLF
        TYPE ,MSG19
        TYPE ,SCRLF
        MOV $RMB(RO),R1 ;PUT THE END ADDRESS INTO R1
        MOV $WRDM(RO),-(SP) ;FIND THE BEGINNING OF THE SECTOR
        NEG (SP) ;GET THE POSITIVE NUMBER OF WORD COUNT
        ADD $RMWC(RO),(SP) ;SUBTRACT THE WORDS NOT TRANSFERED
        CLR -(SP) ;MAKE THE UPPER DIVIDEND 0
        MOV #256,-(SP) ;DIVIDE THE WORDS TRANSFERED BY THE SECTOR SIZE
        JSR PC,LINKDV ;DIVIDE
        TST (SP) ;REMAINDER = 0 ?
        BEQ 1$ ;BR IF IT IS - COMPLETE SECTOR TRANSFERED
        ASL (SP) ;CONVERT THE RESIDUAL SECTOR SIZE INTO BYTE COUNT
        SUB (SP),R1 ;SUBTRACT IT FROM THE END ADDRESS
        BR 2$ ;FINISH THE SIZING
1$:  SUB #1000,R1 ;SUBTRACT FULL SECTOR SIZE FROM END ADDR
2$:  ADD #4,SP ;RESTORE THE STACK POINTER
3$:  MOV #7,R2 ;R2 CONTAINS THE WORDS/LINE COUNT
    MOV R1,-(SP) ;PUT THE ADDRESS ON THE STACK
    JSR PC,LINOC ;TYPE THE ADDRESS
4$:  CMP R1,$RMB(RO) ;PRINTED ALL THE SECTOR ?
    BEQ 5$ ;BR IF ALL PRINTED
    DISPLY ,LINS ;SPACES
    MOV (R1)+,-(SP) ;PUT THE DATA ON THE STACK
    JSR PC,LINOC ;TYPE THE DATA
    DEC R2 ;DECREMENT THE HORIZONTAL COUNT
    BNE 4$ ;BR IF NOT AT THE END OF THE LINE
    DISPLY ,SCRLF ;CR-LF
    BR 3$ ;RESTORE THE WORDS/LINE COUNT
5$:  DISPLY ,SCRLF ;PRINT WHAT REMAINS IN THE BUFFER
6$:  RTS PC ;RETURN

FILBUF: SAVREG ;SAVE THE REGISTERS
        MOV $BUF(RO),R1 ;BUFFER ADDRESS

```

```

4240 014554 016002 000004      MOV     SWRDM(R0),R2      ; POSITIVE WORD COUNT
4241 014560 005402              NEG     R2                ;
4242 014562 012705 002642      MOV     #STNDAT,R5       ; PATTERN ADDRESS
4243 014566 012703 000020      MOV     #20,R3           ; PATTERN COUNT
4244 014572 012521      3$:    MOV     (R5)+,(R1)+   ; MOVE THE PATTERN INTO THE BUFFER
4245 014574 005302              DEC     R2                ; DECREMENT THE WORD COUNT
4246 014576 003407              BLE     4$                ; BR IF DONE (WORD COUNT = 0)
4247 014600 005303              DEC     R3                ; DECREMENT THE PATTERN COUNT
4248 014602 001373              BNE     3$                ; BR IF MORE PATTERN
4249 014604 012703 000020      MOV     #20,R3           ; RESTORE PATTERN COUNT
4250 014610 012705 002642      MOV     #STNDAT,R5       ; RESTORE THE ADDRESS
4251 014614 000766              BR      3$                ; CONTINUE DISTRIBUTING THE PATTERN
4252 014616 104413      4$:    RESREG              ; RESTORE THE REGISTERS
4253 014620 000207              RTS     PC                ; RETURN
4254
4255      ;*****
4256
4257      .SBTTL  ERROR MESSAGE GENERATION ROUTINES
4258
4259      ;*****
4260
4261      ;PRINT LINE 1 OF ERROR MESSAGE:
4262      ;'HH:MM:SS'
4263
4264 014622 032777 002000 164324  LINE1:  BIT     #SW10,2SWR      ; SWITCH 10 SET ?
4265 014630 001402              BEQ     1$                ; BR IF NOT
4266 014632 104401 001200              TYPE   ,SBELL            ; RING THE BELL
4267 014636 032777 020000 164310  1$:    BIT     #SW13,2SWR      ; INHIBIT TYPEOUT ?
4268 014644 001403              BEQ     2$                ; BR IF NOT
4269 014646 104414 001205              DISPLY ,SCRLF            ; CR-LF
4270 014652 000404              BR      3$                ; EXIT
4271 014654 004737 015136      2$:    JSR     PC,$TIME      ; TYPE THE TIME
4272 014660 104414 036332              DISPLY ,LINSPO           ; SPACES
4273 014664 000207      3$:    RTS     PC                ; RETURN & TYPE DESCRIPTION
4274
4275      ;OCTAL TYPEOUT ROUTINE
4276      ;CALL:
4277      ;
4278      ;
4279      ;
4280
4281 014666 016646 000002      LINOCT: MOV     2(SP),-(SP)   ; PUT NUMBER IN PROPER LOCATION ON STACK
4282 014672 004737 016432              JSR     PC,$$B20         ; CONVERT THE NUMBER TO OCTAL
4283 014676 012637 014712              MOV     (SP)+,1$         ; GET THE ADDRESS OF THE ASCII STRING
4284 014702 062737 000005 014712      ADD     #5.,1$           ; ADDRESS THE LAST 6 ASCII DIGITS
4285 014710 104414              DISPLY              ; TYPE IT
4286 014712 000000      1$:    .WORD 0                ; ADDRESS
4287 014714 012616              MOV     (SP)+,(SP)       ; CORRECT THE STACK
4288 014716 000207              RTS     PC                ; RETURN
4289
4290      ;ROUTINE TO CONVERT THE INPUT NUMBER TO DECIMAL AND TYPE IT WITH
4291      ;LEADING ZERO SUPPRESSION
4292      ;CALL:
4293      ;
4294      ;
4295      ;

```



```

4296
4297 014720 016646 000002 LINDEC: MOV 2(SP),-(SP) ;SET UP STACK FOR CONVERT
4298 014724 004737 016402 JSR PC,$SB2D ;CONVERT IT TO DECIMAL
4299 014730 004737 016006 JSR PC,$SUPRS ;TYPE IT (WITH LEADING ZEROS SUPRESSED)
4300 014734 012616 MOV (SP)+,(SP) ;RESTORE STACK POINTER
4301 014736 000207 RTS PC
4302
4303 ;*****
4304
4305 .SBTTL GENERAL SUPPORT SUBROUTINES
4306
4307 ;*****
4308
4309 ;ROUTINE TO CHECK FOR KW11-L OR KW11-P CLOCKS
4310
4311
4312 014740 012737 177777 001314 CKCLK: MOV #-1,CLKFLG ;CLEAR CLOCK AVAILABILITY FLAG
4313 014746 012737 177777 001312 MOV #-1,PCLOCK ;CLEAR KW11-P CLOCK AVAILABILITY FLAG
4314 014754 012737 015034 000004 MOV #CKCLK1,ERRVEC ;SET UP VECTOR FOR CLOCK CHECK
4315 014762 005037 000006 CLR @ERRVEC+2 ;NEW PSW
4316 014766 005777 164306 TST @SLKCSR ;CHECK FOR KW11-P
4317 014772 005037 001314 CLR CLKFLG ;SET CLOCK AVAILABILITY FLAG
4318 014776 005037 001312 CLR PCLOCK ;SET KW11-P CLOCK FLAG
4319 015002 013701 001304 MOV $LPVEC,R1 ;KW11-P VECTOR ADDRESS
4320 015006 012721 015234 MOV #CLOCK,(R1)+ ;SET UP KW11-P VECTOR
4321 015012 012711 000300 MOV #300,(R1) ;PSW - PRI 6
4322 015016 012777 174575 164256 MOV #-1667,@SLKCSB ;LOAD COUNTER BUFFER WITH 16.67
4323 015024 012777 000131 164246 MOV #131,@SLKCSR ;SET CLOCK - CNT UP, 10US, CONT INT
4324 015032 000435 BR CKCLK3
4325 015034 062706 000004 CKCLK1: ADD #4,SP ;RESTORE THE STACK POINTER
4326 015040 012737 015102 000004 MOV #CKCLK2,@ERRVEC ;CHANGE ERROR VECTOR TO CHECK FOR KW11-L
4327 015046 005777 164234 TST @SLKS ;LOOK FOR KW11-L
4328 015052 005037 001314 CLR CLKFLG ;SET CLOCK FLAG
4329 015056 013701 001310 MOV $LLVEC,R1 ;KW11-L VECTOR ADDRESS
4330 015062 012721 015234 MOV #CLOCK,(R1)+ ;SET UP KW11-L VECTOR
4331 015066 012711 000300 MOV #300,(R1) ;PSW - PRI 6
4332 015072 012777 000100 164206 MOV #100,@SLKS ;SET KW11-L INTERRUPT
4333 015100 000412 BR CKCLK3
4334 015102 062706 000004 CKCLK2: ADD #4,SP ;RESTORE THE STACK POINTER
4335 015106 104401 036130 TYPE ,NEDCLK ;'P OR L CLOCK MUST BE ON SYSTEM'
4336 015112 005737 000042 TST 42 ;UNDER MONITOR CONTROL ?
4337 015116 001400 BEQ 15 ;BR IF NOT
4338 015120 000000 15: HALT ;HALT
4339 015122 000137 003062 JMP START ;TRY AGAIN
4340 015126 012737 000006 000004 CKCLK3: MOV #6,@ERRVEC ;RESTORE THE ERROR VECTOR
4341 015134 000207 RTS PC
4342
4343
4344 ;ROUTINE TO TYPE THE TIME
4345
4346 015136 005737 001314 $TIME: TST CLKFLG ;CLOCK ON THE SYSTEM ?
4347 015142 001033 BNE 15 ;BR IF NOT
4348 015144 104401 001205 TYPE ,$CRLF ;CR-LF
4349 015150 013746 001342 MOV HOUR,-(SP) ;PUT 'HOURS' ON THE STACK
4350 015154 004737 016402 JSR PC,$SB2D ;CONVERT TO DECIMAL
4351 015160 004537 015716 JSR RS,REPLZ ;TYPE IT
    
```

K07

MAINDEC-11-DZRM1 AO/00,
DZRMIA.P11 21-JUL-77RMO3 DRIVE COMPATIBILITY TEST
15:44 GENERAL SUPPORTMACY11 30(1046)
SUBROUTINES

21-JUL-77 16:51 PAGE 88

SEQ 0087

```

4352 015164 000002          .WORD      2          ;TYPE 2 DIGITS
4353 015166 104401 036201  TYPE      ,COLON    ;'.'
4354 015172 013746 001344  MOV      MINUTE,-(SP) ;PUT 'MINUTES' ON THE STACK
4355 015176 004737 016402  JSR      PC,$SB2D    ;CONVERT TO DECIMAL
4356 015202 004537 015716  JSR      R5,REPLZ   ;TYPE IT
4357 015206 000002          .WORD      2          ;TYPE 2 DIGITS
4358 015210 104401 036201  TYPE      ,COLON    ;'.'
4359 015214 013746 001346  MOV      SECOND,-(SP);PUT SECONDS ON THE STACK
4360 015220 004737 016402  JSR      PC,$SB2D    ;CONVERT TO DECIMAL
4361 015224 004537 015716  JSR      R5,REPLZ   ;TYPE IT
4362 015230 000002          .WORD      2          ;TYPE 2 DIGITS
4363 015232 000207          .WORD      2          ;TYPE 2 DIGITS
4364
4365          ;CLOCK HANDLER ROUTINE
4366
4367 015234 005337 001350  CLOCK:   DEC      SIXTEE ;INCREMENT THE 1/60 SECOND COUNTER
4368 015240 001033          BNE      1$          ;BR IF A SECOND NOT COUNTED
4369 015242 013737 001316 001350  MOV      HZ,SIXTEE  ;RESTORE THE VALUE
4370 015250 005237 001346          INC      SECOND     ;COUNT THE SECOND
4371 015254 022737 000074 001346  CMP      #60.,SECOND ;AT MAXIMUM ?
4372 015262 001022          BNE      1$          ;BR IF NOT
4373 015264 005037 001346          CLR      SECOND     ;CLEAR THE SECOND'S COUNTER
4374 015270 005237 001344          INC      MINUTE     ;COUNT THE MINUTE
4375 015274 022737 000074 001344  CMP      #60.,MINUTE ;AT MAXIMUM ?
4376 015302 001012          BNE      1$          ;BR IF NOT
4377 015304 005037 001344          CLR      MINUTE     ;CLEAR THE MINUTE'S COUNTER
4378 015310 005237 001342          INC      HOUR       ;COUNT THE HOURS
4379 015314 022737 001747 001342  CMP      #999.,HOUR ;AT MAXIMUM
4380 015322 103002          BHS     1$          ;BR IF NOT
4381 015324 005037 001342          CLR      HOUR       ;CLEAR THE HOURS
4382 015330 012746 000021 1$:     MOV      #17,-(SP)  ;17 MS ON THE STACK
4383 015334 004737 027266          JSR      PC,RPTMR   ;DRIVER TIMER ROUTINE
4384 015340 000002          2$:     RTI
4385
4386          ;COMMAND DECODE ROUTINE
4387          ;CALL:
4388          ;MOV      #-1,CFLAG ;'CFLAG' IS NORMALLY SET BY THE TTY SERVICE
4389          ;JSR      PC,KSR    ;ROUTINE IN INTERRUPT MODE
4390          ;RETURN1
4391          ;RETURN2
4392          ;SYSTEM BUSY RETURN
4393          ;RETURN AFTER KEYBOARD SERVICED
4394 015342 104412          KSR:   SAVREG      ;SAVE THE REGISTERS
4395 015344 012737 000200 177776  MOV      #PR4,PS    ;SET PRIORITY TO 4
4396 015352 005037 001334          CLR      CFLAG     ;CLEAR THE 'CONTROL C' FLAG
4397 015356 004737 015136          JSR      PC,$TIME  ;TYPE THE TIME
4398 015362 005777 163574          TST     @STKB      ;CLEAR ANY GARBAGE IN THE TTY BUFFER
4399 015366 005737 001334          TST     CFLAG     ;CHECK THE CONTROL C FLAG
4400 015372 001002          BNE     7$        ;EXIT IF 'CONTROL C' ENTERED
4401 015374 000240          NOP
4402 015376 000240          NOP
4403 015400 104413          7$:   RESREG      ;RESTORE R0 - R5
4404 015402 062716 000002          ADD     #2,(SP)    ;INCREMENT THE RETURN ADDRESS
4405 015406 005777 163550          TST     @STKB      ;CLEAR THE TTY BUFFER
4406 015412 052777 000100 163540  BIS     #BIT06,@STKS ;SET TTY INTERRUPT ENABLE
4407 015420 005037 177776          CLR     PS        ;SET PRIORITY BACK TO ZERO

```



```

4408 015424 000207          RTS      PC          ;RETURN
4409
4410          ;ROUTINE TO CLEAR THE DPB FOR THE ASSIGNED DRIVE
4411          ;CALL:
4412          ;      MOV      #DPB,R0          ;DPB ADDRESS
4413          ;      JSR      PC,CLRDPB
4414          ;      RETURN
4415
4416          CLRDPB:
4417 015426 010146          MOV      R1,-(SP)          ;;PUSH R1 ON STACK
4418 015426 010346          MOV      R3,-(SP)          ;;PUSH R3 ON STACK
4419 015432 010446          MOV      R4,-(SP)          ;;PUSH R4 ON STACK
4420 015434 010546          MOV      R5,-(SP)          ;;PUSH R5 ON STACK
4421 015436 010004          MOV      R0,R4          ;;GET THE DPB ADDRESS
4422 015440 062704 000002          ADD      #2,R4          ;;ADDRESS OF FIRST LOCN TO BE CLEARED
4423 015444 012703 000020          MOV      #SEMTAB-2,R3    ;;NUMBER OF LOCATIONS TO CLEAR
4424 015450 005024          1S:      CLR      (R4)+          ;;CLEAR THE STORAGE LOCATION
4425 015452 162703 000002          SUB      #2,R3          ;;DECREMENT THE BYTE COUNT
4426 015456 001374          BNE     1S              ;;LOOPING BACK
4427 015460 012605          MOV      (SP)+,R5        ;;POP STACK INTO R5
4428 015462 012604          MOV      (SP)+,R4        ;;POP STACK INTO R4
4429 015464 012603          MOV      (SP)+,R3        ;;POP STACK INTO R3
4430 015466 012601          MOV      (SP)+,R1        ;;POP STACK INTO R1
4431 015470 000207          RTS      PC          ;RETURN
4432
4433 015472 104412          LINKDV: SAVREG          ;STORE R0 - R5
4434 015474 016605 000026          MOV      26(SP),R5      ;DIVISOR
4435 015500 005004          CLR      R4              ;OTHER DIVISOR WORD
4436 015502 016602 000030          MOV      30(SP),R2      ;UPPER DIVIDEND WORD
4437 015506 016603 000032          MOV      32(SP),R3      ;LOWER DIVIDEND WORD
4438 015512 005000          CLR      R0              ;CLEAR OTHER DIVIDEND REGISTERS
4439 015514 005001          CLR      R1
4440 015516 004737 015540          JSR      PC,M.DPID      ;GO TO THE DIVIDE ROUTINE
4441 015522 010166 000030          MOV      R1,30(SP)      ;REMAINDER ON THE STACK
4442 015526 010366 000032          MOV      R3,32(SP)      ;QUOTIENT ON THE STACK
4443 015532 104413          RESREG          ;RESTORE R0 - R5
4444 015534 012616          MOV      (SP)+,(SP)      ;MOVE RETURN UP THE STACK
4445 015536 000207          RTS      PC
4446
4447          ;
4448          ;
4449          ;
4450          ;
4451          ;
4452          ;
4453          ;
4454 015540 012746 000040          M.DPID: MOV      #40,-(SP)    ;COUNTER FOR DIVISION CYCLES
4455 015544 010446          MOV      R4,-(SP)      ;HIGH ORDER
4456 015546 010546          MOV      R5,-(SP)      ;LOW ORDER DIVISOR TO THE STACK
4457 015550 005466 000002          NEG      2(SP)          ;FORM NEGATIVE
4458 015554 005416          NEG      2SP            ;VERSION OF THE DIVISOR
4459 015556 005666 000002          SBC      2(SP)
4460 015562 061601          ADD      2SP,R1
4461 015564 005500          ADC      R0              ;PERFORM THE INITIAL SUBTRACTION
4462 015566 066600 000002          ADD      2(SP),R0
4463 015572 103445          BCS     M.DPS0          ;IF CARRY THEN OVERFLOW HAS OCCURRED

```

```

4464 015574 005046          CLR      -(SP)          ;THIS IS A LONGER LASTING CARRY BIT
4465 015576 006103      M.DP40: ROL      R3
4466 015600 006102          ROL      R2
4467 015602 006101          ROL      R1
4468 015604 006100          ROL      R0
4469 015606 005716          TST      @SP          ;TEST "CARRY" INDICATOR
4470 015610 001410          BEQ      M.DP41        ;IF NO "CARRY" THEN ADD ELSE SUBTRACT
4471 015612 005016          CLR      @SP          ;CLEAR UP FOR NEXT TIME
4472 015614 066601 000002    ADD      2(SP),R1
4473 015620 005500          ADC      R0          ;ADD -(DIVISOR)
4474 015622 005516          ADC      @SP          ;SET "CARRY"
4475 015624 066600 000004    ADD      4(SP),R0; <- I
4476 015630 000404          BR       M.DP42
4477 015632 060501      M.DP41: ADD      R5,R1
4478 015634 005500          ADC      R0          ;ADD +(DIVISOR)
4479 015636 005516          ADC      @SP          ;SET "CARRY"
4480 015640 060400          ADD      R4,R0      ;<-
4481 015642 005516      M.DP42: ADC      @SP
4482 015644 005716          TST      @SP          ;SET "CARRY"
4483 015646 001401          BEQ      .+4          ;TEST THE UPDATE INDICATOR
4484 015650 005203          INC      R3          ;IF ZERO FORGET IT
4485 015652 005366 000006    DEC      6(SP)        ;NO CARRY POSSIBLE HERE
4486 015656 003347          BGT      M.DP40      ;DECREMENT COUNTER
4487 015660 006003          ROR      R3          ;BRANCH IF MORE TO DO
4488 015662 103404          BCS      M.DP44
4489 015664 060501          ADD      R5,R1
4490 015666 005500          ADC      R0
4491 015670 060400          ADD      R4,R0
4492 015672 000241          CLC
4493 015674 006103      M.DP44: ROL      R3
4494 015676 062706 000010    ADD      @10,SP      ;ADJUST STACK BY 4 WORDS
4495 015702 000242          CLV
4496 015704 000207          RTS      PC
4497 015706 062706 000006    M.DP50: ADD      @6,SP
4498 015712 000262          SEV
4499 015714 000207          RTS      PC

4500
4501
4502          ;ROUTINE TO REPLACE LEADING ZEROS IN A NUMERIC STRING WITH SPACES
4503          ;CALL
4504          ;      MOV      #ADR, -(SP)          ;ADDRESS OF NUMBER (IN ASCII)
4505          ;      JSR      R5, REPLZ          ;
4506          ;      .WORD   N                  ;'N' IS NUMBER OF DIGITS TO BE TYPED
4507
4508 015716 010046      REPLZ: MOV      R0, -(SP)          ;SAVE R0
4509 015720 012746 000012    MOV      @10, -(SP)      ;MAXIMUM NUMBER OF DIGITS TO BE TYPED
4510 015724 162516          SUB      (R5)+, (SP)     ;SUBTRACT DIGITS TO FORM INDEX
4511 015726 016600 000006    MOV      6(SP), R0      ;ADDRESS OF NUMBER TO R0
4512 015732 122710 000060    1$:  CMPB   #'0', (R0)     ;BYTE EQUAL TO ASCII '0' ?
4513 015736 001004          BNE     2$             ;BR IF NOT
4514 015740 112710 000040    MOVB   @40, (R0)      ;REPLACE THE ZERO WITH A SPACE
4515 015744 005200          INC     R0            ;INCREMENT THE BYTE ADDRESS
4516 015746 000771          BR     1$             ;GO BACK AND LOOK FOR MORE LEADING ZEROS
4517 015750 105710      2$:  TSTB   (R0)          ;SEE IF ZERO BYTE TERMINATOR
4518 015752 001003          BNE     3$             ;BR IF NOT
4519 015754 005300          DEC     R0            ;BACKUP STRING POINTER

```



```

4520 015756 112710 000060      MOV      #'0,(R0)      ;PUT A ZERO BACK IN
4521 015763 016637 000006 015776 3$:  MOV      6(SP),4$      ;PUT ADDRESS IN LOCATION FOR TYPEOUT
4522 015770 062637 015776      ADD      (SP)+,4$      ;BEGINNING OF SIGNIFICANT DIGITS
4523 015774 104401      TYPE                                ;TYPE THE NUMBER
4524 015776 000000      .WORD   0              ;ADDRESS OF NUMBER
4525 016000 012600      MOV      (SP)+,R0      ;RESTORE R0
4526 016002 012616      MOV      (SP)+,(SP)    ;MOVE RETURN ADDRESS
4527 016004 000205      RTS      R5            ;RETURN
4528
4529      ;TYPE NUMERICAL ASCIZ STRING SUPPRESS LEADING ZEROS
4530
4531      ;CALL
4532      ;
4533      ;
4534      ;
4535 016006 010046      $$SUPRS: MOV      R0,-(SP)     ;SAVE R0
4536 016010 016600 000004      MOV      4(SP),R0     ;PICKUP THE POINTER
4537 016014 105710 1$:  TSTB     (R0)          ;TERMINATOR ?
4538 016016 001403      BEQ      2$           ;BR IF YES
4539 016020 122720 000060      CMPB    #'0,(R0)+    ;IS THIS AN ASCII '0' ?
4540 016024 001773      BEQ      1$           ;BR IF YES
4541 016026 005300 2$:  DEC      R0           ;BACKUP BY '1'
4542 016030 010037 016036      MOV      R0,3$        ;SAVE FOR TYPING
4543 016034 104414      DISPLY                                ;GO PRINT
4544 016036 000000 3$:  .WORD   0              ;ASCIZ POINTER GOES HERE
4545 016040 012600      MOV      (SP)+,R0     ;RESTORE R0
4546 016042 012616      MOV      (SP)+,(SP)   ;RESTORE THE STACK
4547 016044 000207      RTS      PC           ;RETURN
4548
4549      ;ROUTINE TO TYPE AT PRIORITY 4
4550
4551 016046 013746 177776 177776  TYPRI4: MOV      2#PS,-(SP)  ;SAVE THE PRESENT STATUS
4552 016052 012737 000200      MOV      200,2#PS    ;CHANGE THE PRIORITY TO 4
4553 016060 012537 016070      MOV      (R5)+,1$    ;MESSAGE ADDRESS
4554 016064 004737 017722      JSR     PC,$TYPE     ;TYPE THE MESSAGE
4555 016070 000000 1$:  .WORD   0              ;MESSAGE ADDRESS GOES HERE
4556 016072 000205      RTS      R5            ;RETURN
4557
4558      ;ROUTINE TO TYPE ERRORS
4559      ;CALL
4560      ;
4561      ;
4562      ;
4563      ;
4564 016074 032777 020000 163052  $$DSPLY: BIT      #BIT13,2$WR  ;INHIBIT ERROR TYPEOUT ?
4565 016102 001002      BNE     1$           ;BR IF YES
4566 016104 000137 017722      JMP     $TYPE        ;TYPE THE MESSAGE
4567 016110 062716 000002 1$:  ADD      #2,(SP)      ;INCREMENT THE RETURN
4568 016114 000002      RTI                                ;RETURN
4569
4570      ;THIS ROUTINE IS USED TO CHECK IF AN
4571      ;ASCII CHARACTER IS A DIGIT BETWEEN 0 AND 7.
4572      ;CALL
4573      ;
4574      ;
4575      ;

```

```

4576          :          RETURN2          ; CHARACTER IS IN R2 AS A
4577          :                               ; OCTAL DIGIT
4578
4579 016116 121127 000060 CK.OCT: CMPB (R1),#'0 ; LESS THAN ZERO?
4580 016122 103407          BLO 1$ ; YES -- BRANCH
4581 016124 121127 000067          CMPB (R1),#'7 ; GREATER THAN SEVEN?
4582 016130 101004          BHI 1$ ; YES -- BRANCH
4583 016132 111102          MOVB (R1),R2 ; GET THE CHARACTER
4584 016134 042702 177770          BIC #'C7,R2 ; STRIP AWAY THE ASCII
4585 016140 005725          TST (R5)+ ; ADJUST FOR RETURN
4586 016142 000205          1$: RTS R5 ; RETURN
4587
4588          ; THIS ROUTINE IS USED TO CHECK AN ASCII CHARACTER
4589          ; AND DETERMINE IF IT IS A DIGIT BETWEEN 0 AND 9.
4590          ; CALL
4591          :          MOV #ADR,R1          ; ADDRESS OF ASCII CHARACTER
4592          :          JSR R5,CK.DEC        ; CHECK THE CHARACTER
4593          :          RETURN1             ; NOT BETWEEN 0 AND 9
4594          :          RETURN2             ; BETWEEN 0 AND 9
4595          :                               ; R2 = DIGIT
4596
4597 016144 121127 000060 CK.DEC: CMPB (R1),#'0 ; LESS THAN ZERO?
4598 016150 103407          BLO 1$ ; YES -- BRANCH
4599 016152 121127 000071          CMPB (R1),#'9 ; GREATER THAN NINE?
4600 016156 101004          BHI 1$ ; YES -- BRANCH
4601 016160 111102          MOVB (R1),R2 ; GET THE CHARACTER
4602 016162 042702 000060          BIC #'0,R2 ; STRIP AWAY THE ASCII
4603 016166 005725          TST (R5)+ ; ADJUST FOR RETURN
4604 016170 000205          1$: RTS R5 ; RETURN
4605
4606          ; THIS ROUTINE WILL CHECK AN ASCII CHARACTER TO
4607          ; DETERMINE WHAT IT IS.
4608          ; CALL
4609          :          MOV #ADR,R1          ; ADDRESS OF ASCII CHARACTER
4610          :          JSR R5,CK.CHR        ; CHECK CHARACTER
4611          :          RETURN ADR1         ; UNKNOWN CHARACTER
4612          :          RETURN ADR2         ; CARRIAGE RETURN * (R1)=ADR+1
4613          :          RETURN ADR3         ; COMMA * (R1)=ADR+1
4614          :          RETURN ADR4         ; PERIOD * (R1)=ADR+1
4615          :          RETURN ADR5         ; DIGIT BETWEEN 0 AND 7.
4616          :          RETURN ADR6         ; DIGIT BETWEEN 8 AND 9.
4617          :                               ; R2 = DIGIT * (R1)=ADR+1
4618
4619 016172 105711          CK.CHR: TSTB (R1) ; "CARRIAGE RETURN"?
4620 016174 001417          BEQ 3$ ; YES -- BRANCH
4621 016176 121127 000054          CMPB (R1),#', ; "COMMA"?
4622 016202 001413          BEQ 2$ ; YES -- BRANCH
4623 016204 121127 000056          CMPB (R1),#'. ; "PERIOD"?
4624 016210 001407          BEQ 1$ ; YES -- BRANCH
4625 016212 004537 016144          JSR R5,CK.DEC ; "DIGIT"?
4626 016216 000410          BR 4$ ; NO -- BRANCH
4627 016220 004537 016116          JSR R5,CK.OCT ; OCTAL ?
4628 016224 005725          TST (R5)+ ; DIGIT BETWEEN 8-9
4629 016226 005725          TST (R5)+ ; DIGIT BETWEEN 0-7
4630 016230 005725          1$: TST (R5)+ ; PERIOD
4631 016232 005725          2$: TST (R5)+ ; COMMA
    
```



```

4632 016234 005725      3$:   TST      (R5)+      ;CARRIAGE RETURN
4633 016236 005201      INC      R1              ;MOVE POINTER TO NEXT CHARACTER
4634 016240 011505      4$:   MOV      (R5),R5     ;UNKNOWN CHARACTER
4635 016242 000205      RTS      R5              ;RETURN
4636
4637
4638
4639
4640
4641
4642
4643
4644
4645
4646
4647
4648
4649
4650 016244 010446      CK.DIG: MOV     R4,-(SP)      ;SAVE R4
4651 016246 010346      MOV     R3,-(SP)      ;SAVE R3
4652 016250 010246      MOV     R2,-(SP)      ;SAVE THE MAX. SIZE ON THE STACK
4653 016252 005002      CLR     R2              ;START WITH 0
4654 016254 005003      CLR     R3
4655 016256 005004      CLR     R4
4656 016260 004537 016172      JSR     R5,CK.CHR     ;CHECK ONE CHARACTER
4657 016264 016360      6$:   ;ILLEGAL CHARACTER
4658 016266 016366      9$:   ;CARRIAGE RETURN
4659 016270 016360      6$:   ;"
4660 016272 016362      7$:   ;"
4661 016274 016300      1$:   ;DIGIT 0-7
4662 016276 016300      1$:   ;DIGIT 8-9
4663 016300 062705 000004      1$:   ADD     #4,R5          ;STEP RETURN POINTER PAST "CR" & "PERIOD" RETURNS
4664 016304 006303      2$:   ASL     R3              ;INPUT NUMBER #2
4665 016306 010346      MOV     R3,-(SP)      ;SAVE #2
4666 016310 006303      ASL     R3              ;#4
4667 016312 006303      ASL     R3              ;#8
4668 016314 062603      ADD     (SP)+,R3      ;(#2)+(#8) = #10
4669 016316 060203      ADD     R2,R3          ;UPDATE THE INPUT NUMBER
4670 016320 004537 016172      JSR     R5,CK.CHR     ;CHECK ONE CHARACTER
4671 016324 016364      8$:   ;ILLEGAL CHARACTER
4672 016326 016350      5$:   ;CARRIAGE RETURN
4673 016330 016346      4$:   ;"
4674 016332 016340      3$:   ;"
4675 016334 016304      2$:   ;DIGIT 0-7
4676 016336 016304      2$:   ;DIGIT 8-9
4677 016340 105711      3$:   TSTB   (R1)           ;DOES A "CR" FOLLOW THE "PERIOD"
4678 016342 001010      BNE     8$             ;BR IF NOT
4679 016344 005724      TST     (R4)+          ;INCREMENT THE RETURN
4680 016346 005724      4$:   TST     (R4)+          ;INCREMENT THE RETURN
4681 016350 005724      5$:   TST     (R4)+          ;INCREMENT THE RETURN
4682 016352 020316      CMP     R3,(SP)       ;CHECK THE MAGNITUDE OF THE NUMBER
4683 016354 101004      BHI     9$             ;BR IF ENTERED NUMBER TOO LARGE
4684 016356 000402      BR      8$             ;BYPASS INCREMENT
4685 016360 005725      6$:   TST     (R5)+          ;INCREMENT RETURN PAST INVALID RETURN
4686 016362 005725      7$:   TST     (R5)+          ;INCREMENT RETURN
4687 016364 060405      8$:   ADD     R4,R5          ;SETUP RETURN POINTER
    
```

```

4688 016366 010302          9S:  MOV   R3,R2          ;ENTERED VALUE
4689 016370 005726          TST   (SP)+          ;CLEAN MAX. SIZE OFF OF STACK
4690 016372 012603          MOV   (SP)+,R3       ;RESTORE R3
4691 016374 012604          MOV   (SP)+,R4       ;RESTORE R4
4692 016376 011505          MOV   (R5),R5        ;GET RETURN ADDRESS
4693 016400 000205          RTS    R5            ;RETURN
4694
4695
4696
4697
4698
4699
4700
4701
4702
4703
4704
4705 016402 016637 000002 016426  $SB20: MOV   2(SP),1$      ;SAVE THE BINARY NUMBER
4706 016410 012746 016426          MOV   #1$,-(SP)     ;SET THE POINTER
4707 016414 004737 022046          JSR   PC,$DB20      ;CALL THE DOUBLE LENGTH CONVERT
4708 016420 012666 000002          MOV   (SP)+,2(SP)   ;PICKUP THE POINTER
4709 016424 000207          RTS    PC            ;RETURN
4710 016426 000000 000000          1$:  .WORD 0,0
4711
4712
4713
4714
4715
4716
4717
4718
4719
4720
4721
4722 016432 016637 000002 016456  $SB20: MOV   2(SP),1$      ;SAVE THE BINARY NUMBER
4723 016440 012746 016456          MOV   #1$,-(SP)     ;SET THE POINTER
4724 016444 004737 022242          JSR   PC,$DB20      ;CALL THE DOUBLE LENGTH CONVERT
4725 016450 012666 000002          MOV   (SP)+,2(SP)   ;PICKUP THE POINTER
4726 016454 000207          RTS    PC            ;RETURN
4727 016456 000000 000000          1$:  .WORD 0,0
4728
4729
4730
4731
4732
4733
4734 016462 012737 016512 000060  $TKINT: MOV   #STKSRV,TKVEC ;SETUP VECTOR
4735 016470 012737 000240 000062  MOV   #PR5,TKVEC+2  ;PRIORITY TO 5
4736 016476 005777 162460          TST   #STKB         ;CLEAR THE BUFFER
4737 016502 012777 000100 162450  MOV   #BIT06,#STKS  ;SET INTERRUPT ENABLE
4738 016510 000207          RTS    PC            ;RETURN
4739
4740
4741
4742
4743

```

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

```

;CALL
;      MOV   NUMBER,-(SP)      ;PUT THE NUMBER ON THE STACK
;      JSR   PC,$SB20          ;CALL
;      RETURN                  ;ADDRESS OF THE 1ST ASCIZ CHAR IS ON THE STACK

```

; NOTE: THE PROGRAM REQUIRES THIS FORM OF '\$SB20', NOT THE VERSION ON
 ; THE SYSMAC LIBRARY, REV C AND LATER

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

```

;CALL
;      MOV   NUMBER,-(SP)      ;PUT THE NUMBER ON THE STACK
;      JSR   PC,$SB20          ;CALL
;      RETURN                  ;ADDRESS OF THE 1ST ASCIZ CHAR IS ON THE STACK

```

; NOTE: THE PROGRAM REQUIRES THIS FORM OF '\$SB20', NOT THE VERSION ON
 ; THE SYSMAC LIBRARY, REV C AND LATER

; KEYBOARD INTERRUPT INITIALIZATION ROUTINE

```

;CALL
;      JSR   PC,$TKINT
;      RETURN

```

; KEYBOARD INTERRUPT SERVICE ROUTINE

```

;CALL
;      ENTER VIA INTERRUPT

```



```

4744 016512 104410          STKSRV: RDCHR          ;READ THE KEYBOARD
4745 016514 112637 016642  MOVB      (SP)+,5$      ;GET THE CHARACTER
4746 016520 023727 016642 000003  CMP      5$,#3          ;'CONTROL C' ?
4747 016526 001012          BNE      1$            ;BR IF NOT
4748 016530 104401 001205          TYPE     ,SCRLF         ;CR-LF
4749 016534 104401 017146          TYPE     ,SCNTLC        ;'↑C'
4750 016540 012737 177777 001334  MOV      #-1,CFLAG     ;SET THE 'CONTROL C' FLAG
4751 016546 005077 162406          CLR      @STKS         ;CLEAR THE TTY INTERRUPT
4752 016552 000432          BR       4$           ;EXIT
4753 016554 023727 001154 000176 1$:  CMP      SWR,#SWREG     ;SOFTWARE SWITCH REGISTER IN USE ?
4754 016562 001024          BNE      3$           ;BR IF NOT
4755 016564 023727 016642 000007  CMP      5$,#7          ;'CONTROL G' ?
4756 016572 001020          BNE      3$           ;BR IF NOT
4757 016574 104401 001205          TYPE     ,SCRLF         ;CR-LF
4758 016600 104401 021723          TYPE     ,SCNTLG        ;'↑G'
4759 016604 013746 177776          MOV      PS,-(SP)      ;PUT THE STATUS WORD ON THE STACK
4760 016610 012746 016624          MOV      #2$,-(SP)    ;RETURN ADDRESS
4761 016614 005077 162340          CLR      @STKS         ;CLEAR THE TTY INTERRUPT ENABLE
4762 016620 000137 021364          JMP      $GTSWR        ;GET THE SWITCH REGISTER ENTRY
4763 016624 012777 000100 162326 2$:  MOV      #100,@STKS    ;ENABLE TTY KEYBOARD INTERRUPT
4764 016632 000402          BR       4$           ;EXIT
4765 016634 104401 016642          3$:  TYPE     ,5$         ;ECHO THE CHARACTER
4766 016640 000002          4$:  RTI              ;RETURN
4767
4768 016642 000000          5$:  .WORD  0           ;ENTERED CHARACTER
4769
4770          ; THIS ROUTINE WILL INPUT A STRING FROM THE TTY
4771          ; CALL:
4772          ;
4773          ;       RDLIN          ; INPUT A STRING FROM THE TTY
4774          ;       RETURN HERE   ; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
4775          ;                   ; TERMINATOR WILL BE A BYTE OF ALL 0'S
4776 016644 010346          SRDLIN: MOV      R3,-(SP) ;SAVE R3
4777 016646 005046          CLR      -(SP)        ;CLEAR THE RUBOUT KEY
4778 016650 012703 017122          1$:  MOV      @STTYIN,R3  ;GET ADDRESS
4779 016654 022703 017134          2$:  CMP      @STTYIN+10.,R3 ;BUFFER FULL?
4780 016660 101467          BLOS     4$           ;BR IF YES
4781 016662 104410          RDCHR          ;GO READ ONE CHARACTER FROM THE TTY
4782 016664 112613          MOVB     (SP)+,(R3)    ;GET CHARACTER
4783 016666 122713 000177          CMPB     #177,(R3)     ;IS IT A RUBOUT
4784 016672 001022          BNE      5$           ;BR IF NO
4785 016674 005716          TST      (SP)         ;IS THIS THE FIRST RUBOUT?
4786 016676 001007          BNE      6$           ;BR IF NO
4787 016700 112737 000134 017120  MOVB     #' \ ,9$      ;TYPE A BACK SLASH
4788 016706 104401 017120          TYPE     ,9$
4789 016712 012716 177777          MOV      #-1,(SP)     ;SET THE RUBOUT KEY
4790 016716 005303          6$:  DEC      R3          ;BACKUP BY ONE
4791 016720 020327 017122          CMP      R3,@STTYIN   ;STACK EMPTY?
4792 016724 103445          BLO     4$           ;BR IF YES
4793 016726 111337 017120          MOVB     (R3),9$      ;SETUP TO TYPEOUT THE DELETED CHAR.
4794 016732 104401 017120          TYPE     ,9$
4795 016736 000746          BR       2$           ;GO READ ANOTHER CHAR.
4796 016740 005716          5$:  TST      (SP)         ;RUBOUT KEY SET?
4797 016742 001406          BEQ     7$           ;BR IF NO
4798 016744 112737 000134 017120  MOVB     #' \ ,9$      ;TYPE A BACK SLASH
4799 016752 104401 017120          TYPE     ,9$
    
```

```

4800 016756 005016          CLR      (SP)          ;CLEAR THE RUBOUT KEY
4801 016760 122713 000025  7$:  CMPB    #25,(R3)    ;IS CHARACTER A CTRL U?
4802 016764 001003          BNE     10$           ;BR IF NO
4803 016766 104401 021716  TYPE    ,SCNTLU      ;TYPE A CONTROL "U"
4804 016772 000726          BR      1$           ;GO START OVER
4805 016774 122713 000003  10$:  CMPB    #3,(R3)    ;IS CHARACTER A CTRL C ?
4806 017000 001006          BNE     8$           ;BR IF NOT
4807 017002 012737 177777 001334  MOV     #-1,CFLAG    ;SET CNTRL C FLAG
4808 017010 104401 017146  TYPE    ,SCNTLC      ;ECHO IT
4809 017014 000427          BR      11$          ;EXIT
4810 017016 122713 000012  8$:  CMPB    #12,(R3)   ;IS CHARACTER A "LF"?
4811 017022 001011          BNE     3$           ;BRANCH IF NO
4812 017024 105013          CLRB   (R3)          ;CLEAR THE CHARACTER
4813 017026 104401 001205  TYPE    ,SCRLF      ;TYPE A "CR" & "LF"
4814 017032 104401 017122  TYPE    ,STTYIN     ;TYPE THE INPUT STRING
4815 017036 000706          BR      2$           ;GO PICKUP ANOTHER CHACTER
4816 017040 104401 001204  4$:  TYPE    ,SQUES     ;TYPE A '?'
4817 017044 000701          BR      1$           ;CLEAR THE BUFFER AND LOOP
4818 017046 111337 017120  3$:  MOVB    (R3),9$    ;ECHO THE CHARACTER
4819 017052 104401 017120  TYPE    ,9$         ;
4820 017056 122723 000015  CMPB    #15,(R3)+   ;CHECK FOR RETURN
4821 017062 001274          BNE     2$           ;LOOP IF NOT RETURN
4822 017064 105063 177777  CLRB   -1(R3)       ;CLEAR RETURN (THE 15)
4823 017070 104401 001206  TYPE    ,SLF        ;TYPE A LINE FEED
4824 017074 005726          TST    (SP)+        ;CLEAN RUBOUT KEY FROM THE STACK
4825 017076 012603          MOV     (SP)+,R3    ;RESTORE R3
4826 017100 011646          MOV     (SP)-,(SP)  ;ADJUST THE STACK AND PUT ADDRESS OF THE
4827 017102 016666 000004 000002  MOV     4(SP),2(SP) ;FIRST ASCII CHARACTER ON IT
4828 017110 012766 017122 000004  MOV     #STTYIN,4(SP)
4829 017116 000002          RTI                    ;RETURN
4830 017120 000          9$:  .BYTE   0            ;STORAGE FOR ASCII CHAR. TO TYPE
4831 017121 000          .BYTE   0            ;TERMINATOR
4832 017122 000024          $TTYIN: .BLKB  20.    ;RESERVE 20 BYTES FOR TTY INPUT
4833 017146 041536 005015 000  $CNTLC: .ASCIZ /↑C/<CR><LF> ;CONTROL "C"
4834
4835 017154          .EVEN
4836
4837
4838
4839 017154 005737 001374  FINISH: TST    FAULT    ;COMPATIBLE ?
4840 017160 001006          BNE     1$           ;BRANCH IF NOT
4841 017162 104401 001205  TYPE    ,SCRLF      ;
4842 017166 104401 001205  TYPE    ,SCRLF      ;
4843 017172 104401 035442  TYPE    ,MSG15     ;MESSAGE: ALL DRIVE COMPATIBLE
4844 017176 104401 001205  1$:  TYPE    ,SCRLF      ;
4845 017202 104401 001205  TYPE    ,SCRLF      ;
4846 017206 104401 036375  TYPE    ,MSG20     ;
4847 017212 104401 001205  TYPE    ,SCRLF      ;
4848 017216 000000          HALT                    ;
4849 017220 000137 000200  JMP     #200         ;BRANCH TO 200 START IF DESIRED
4850
4851          .SBTTL  END OF PASS ROUTINE
4852
4853          ;*****
4854          ;*INCREMENT THE PASS NUMBER ($PASS)
4855          ;*IF SW12=1 INHIBIT TRACE TRAP
    
```



```

4856      ;*IF THERES A MONITOR GO TO IT
4857      ;*IF THERE ISN'T JUMP TO FINISH
4858
4859      SEOP:
4860      017224 000240      NOP
4861      017226 005037 001116      CLR      $TSTNM      ;; ZERO THE TEST NUMBER
4862      017232 005037 001174      CLR      $TIMES      ;; ZERO THE NUMBER OF ITERATIONS
4863      017236 005237 001216      INC      $PASS      ;; INCREMENT THE PASS NUMBER
4864      017242 042737 100000 001216      BIC      #100000,$PASS      ;; DON'T ALLOW A NEG. NUMBER
4865      017250 005327      DEC      (PC)+      ;; LOOP?
4866      017252 000001      SEOPCT: .WORD      1
4867      017254 003022      BGT      $DOAGN      ;; YES
4868      017256 012737      MOV      (PC)+,2(PC)+      ;; RESTORE COUNTER
4869      017260 000001      SENDCT: .WORD      1
4870      017262 017252      SEOPCT
4871      017264 013700 000042      $GET42: MOV      2#42,RO      ;; GET MONITOR ADDRESS
4872      017270 001414      BEQ      $DOAGN      ;; BRANCH IF NO MONITOR
4873      017272 005046      CLR      -(SP)      ;; INSURE THE "T" BIT IS CLEAR
4874      017274 012746 017302      MOV      #SCLR.T,-(SP)      ;; SETUP FOR AN RTI OR RTT
4875      017300 000426      BR       $RTRN      ;; GO DO AN RTI OR RTT TO LOAD THE PSW
4876      ;; WITH A CLEARED "T" BIT
4877      017302
4878      017302 013700 000042      SCLR.T: MOV      2#42,RO      ;; INSURE RO CONTAINS THE MONITORS
4879      017306 001405      BEQ      $DOAGN      ;; RETURN ADDRESS
4880      017310 000005      RESET
4881      017312 004710      SENDAD: JSR      PC,(RO)      ;; CLEAR THE WORLD
4882      017314 000240      NOP      ;; GO TO MONITOR
4883      017316 000240      NOP      ;; SAVE ROOM
4884      017320 000240      NOP      ;; FOR
4885      017322      SDOAGN:
4886      017322 104400      TRAP
4887      017324 042716 000020      BIC      #20,(SP)      ;; PUSH OLD PSW AND PC ON STACK
4888      017330 032777 010000 161616      BIT      #BIT12,$SWR      ;; CLEAR THE "T" BIT
4889      017336 001005      BNE      1$      ;; RUN WITH TRACE TRAP?
4890      017340 005137 017364      COM      $TBIT      ;; BR IF NO
4891      017344 100402      BMI      1$      ;; IS IT TIME FOR TRACE TRAP
4892      017346 052716 000020      BIS      #20,(SP)      ;; BR IF NO
4893      017352 012746 017360      1$:      MOV      #SLOOP,-(SP)      ;; SET TRACE TRAP
4894      017356 000002      SRTRN: RTI      ;; JUMP TO START OF TEST
4895      ;; RETURN--THIS IS CHANGED TO
4896      ;; AN "RTT" IF "RTT" IS A LEGAL
4897      ;; INSTRUCTION
4898      017360      SLOOP:
4899      017360 000137      JMP      2(PC)+      ;; RETURN
4900      017362 017154      SRTNAD: .WORD      FINISH
4901      017364 000000      STBIT:  .WORD      0      ;; "T" BIT STATE INDICATOR
4902
4903      .SBTTL  ERROR HANDLER ROUTINE
4904      ;; *****
4905      ;; *THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
4906      ;; *SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
4907      ;; *AND GO TO $ERRTYP ON ERROR
4908      ;; *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
4909      ;; *SW15=1      HALT ON ERROR
4910      ;; *SW13=1      INHIBIT ERROR TYPEOUTS
4911      ;; *SW10=1      BELL ON ERROR
    
```

```

4912 ;#SW09=1      LOOP ON ERROR
4913 ;#CALL
4914 ;*      ERROR  N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER
4915
4916 $ERROR:
4917 017366 104407 001117 7$: CKSWR      ;; TEST FOR CHANGE IN SOFT-SWR
4918 017370 105237 001117 7$: INCB      $ERFLG      ;; SET THE ERROR FLAG
4919 017374 001775 001116 161552 7$: BEQ      7$      ;; DON'T LET THE FLAG GO TO ZERO
4920 017376 013777 001116 161552 7$: MOV      $TSTNM,$DISPLAY ;; DISPLAY TEST NUMBER AND ERROR FLAG
4921 017404 032777 002000 161542 7$: BIT      $BIT10,$SWR ;; BELL ON ERROR?
4922 017412 001402 001200 161542 7$: BEQ      1$      ;; NO - SKIP
4923 017414 104401 001200 161542 7$: TYPE     $BELL      ;; RING BELL
4924 017420 005237 001126 161542 7$: INC      $ERTTL      ;; COUNT THE NUMBER OF ERRORS
4925 017424 011637 001132 161542 7$: MOV      (SP),$ERRPC ;; GET ADDRESS OF ERROR INSTRUCTION
4926 017430 162737 000002 001132 7$: SUB      #2,$ERRPC
4927 017436 117737 161470 001130 7$: MOVB     $ERRPC,$ITEMB ;; STRIP AND SAVE THE ERROR ITEM CODE
4928 017444 032777 020000 161502 7$: BIT      $BIT13,$SWR ;; SKIP TYPEOUT IF SET
4929 017452 001004 001756 161502 7$: BNE      20$      ;; SKIP TYPEOUTS
4930 017454 004737 001756 161502 7$: JSR      PC,$ERRTYP ;; GO TO USER ERROR ROUTINE
4931 017460 104401 001205 161502 7$: TYPE     $SCLF
4932 017464
4933 017464 122737 000001 001230 20$: CMPB     $APTENV,$ENV ;; RUNNING IN APT MODE
4934 017472 001007 001130 017506 20$: BNE      2$      ;; NO SKIP APT ERROR REPORT
4935 017474 113737 001130 017506 20$: MOVB     $ITEMB,21$ ;; SET ITEM NUMBER AS ERROR NUMBER
4936 017502 004737 020222 017506 20$: JSR      PC,$ATY4 ;; REPORT FATAL ERROR TO APT
4937 017506 000
4938 017507 000
4939 017510 000777 161436 21$: .BYTE    0
4940 017512 005777 161436 21$: .BYTE    0
4941 017516 100002 161436 22$: BR      22$      ;; APT ERROR LOOP
4942 017520 000000 161436 22$: TST      $SWR      ;; HALT ON ERROR
4943 017522 104407 161436 22$: BPL      3$      ;; SKIP IF CONTINUE
4944 017524 032777 001000 161422 22$: HALT     $CKSWR    ;; HALT ON ERROR!
4945 017532 001402 001000 161422 22$: CKSWR    ;; TEST FOR CHANGE IN SOFT-SWR
4946 017534 013716 001124 161422 22$: BIT      $BIT09,$SWR ;; LOOP ON ERROR SWITCH SET?
4947 017540 005737 001176 161422 22$: BEQ      4$      ;; BR IF NO
4948 017544 001402 001176 161422 22$: MOV      $LPERR,(SP) ;; FUDGE RETURN FOR LOOPING
4949 017546 013716 001176 161422 22$: TST      $ESCAPE   ;; CHECK FOR AN ESCAPE ADDRESS
4950 017552 022737 017312 000042 22$: BEQ      5$      ;; BR IF NONE
4951 017552 001001 017312 000042 22$: MOV      $ESCAPE,(SP) ;; FUDGE RETURN ADDRESS FOR ESCAPE
4952 017560 000000 017312 000042 22$: CMP      $SENDAD,$#42 ;; ACT-11 AUTO-ACCEPT?
4953 017562 000000 017312 000042 22$: BNE      6$      ;; BRANCH IF NO
4954 017564 000002 017312 000042 22$: HALT
4955 017564 000002 017312 000042 22$: ;;YES
4956
4957 .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
4958
4959 ;;*****
4960 ;;THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
4961 ;;ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
4962 ;;AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
4963
4964 $ERRTYP:
4965 017566 104401 001205 6$: TYPE     $SCLF      ;; "CARRIAGE RETURN" & "LINE FEED"
4966 017572 010046 001205 6$: MOV      RD,-(SP)   ;; SAVE RD
4967 017574 005000 001205 6$: CLR      RD        ;; PICKUP THE ITEM INDEX

```



```

4968 017576 153700 001130      BISB  2(S)ITEMB,RO
4969 017602 001004      BNE   1$
4970
4971 017604 013746 001132      MOV   SERRPC,-(SP)
4972
4973 017610 104402      TYPOC
4974 017612 000426      BR    6$
4975 017614 005300      1$:  DEC  RO
4976 017616 006300      ASL  RO
4977 017620 006300      ASL  RO
4978 017622 006300      ASL  RO
4979 017624 062700 002742      ADD  #SERRTB,RO
4980 017630 012037 017640      MOV  (RO)+,2$
4981 017634 001404      BEQ  3$
4982 017636 104401      TYPE
4983 017640 000000      2$:  .WORD 0
4984 017642 104401 001205      TYPE ,SCLF
4985 017646 012037 017656      3$:  MOV  (RO)+,4$
4986 017652 001404      BEQ  5$
4987 017654 104401      TYPE
4988 017656 000000      4$:  .WORD 0
4989 017660 104401 001205      TYPE ,SCLF
4990 017664 011000      5$:  MOV  (RO),RO
4991 017666 001004      BNE  7$
4992 017670 012600      6$:  MOV  (SP)+,RO
4993 017672 104401 001205      TYPE ,SCLF
4994 017676 000207      RTS  PC
4995 017700      7$:
4996 017700 013046      MOV  2(RO)+,-(SP)
4997 017702 104402      TYPOC
4998 017704 005710      TST  (RO)
4999 017706 001770      BEQ  6$
5000 017710 104401 017716      TYPE ,8$
5001 017714 000771      BR   7$
5002 017716 020040 000      8$:  .ASCIZ / /
5003 017722      .EVEN

```

```

;; IF ITEM NUMBER IS ZERO, JUST
;; TYPE THE PC OF THE ERROR
;; SAVE SERRPC FOR TYPEOUT
;; ERROR ADDRESS
;; GO TYPE--OCTAL ASCII(ALL DIGITS)
;; GET OUT
;; ADJUST THE INDEX SO THAT IT WILL
;; WORK FOR THE ERROR TABLE

;; FORM TABLE POINTER
;; PICKUP "ERROR MESSAGE" POINTER
;; SKIP TYPEOUT IF NO POINTER
;; TYPE THE "ERROR MESSAGE"
;; "ERROR MESSAGE" POINTER GOES HERE
;; "CARRIAGE RETURN" & "LINE FEED"
;; PICKUP "DATA HEADER" POINTER
;; SKIP TYPEOUT IF 0
;; TYPE THE "DATA HEADER"
;; "DATA HEADER" POINTER GOES HERE
;; "CARRIAGE RETURN" & "LINE FEED"
;; PICKUP "DATA TABLE" POINTER
;; GO TYPE THE DATA
;; RESTORE RO
;; "CARRIAGE RETURN" & "LINE FEED"
;; RETURN

;; SAVE 2(RO)+ FOR TYPEOUT
;; GO TYPE--OCTAL ASCII(ALL DIGITS)
;; IS THERE ANOTHER NUMBER?
;; BR IF NO
;; TYPE TWO(2) SPACES
;; LOOP
;; TWO(2) SPACES

```

.SBTTL TYPE ROUTINE

```

5004
5005
5006
5007
5008
5009
5010
5011
5012
5013
5014
5015
5016
5017
5018
5019
5020
5021
5022 017722 105737 001173      $TYPE: TSTB  $TFPLG
5023 017726 100002      BPL    1$

```

```

*****
*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
*
;; IS THERE A TERMINAL?
;; BR IF YES

```

```

5024 017730 000000      HALT
5025 017732 000430      BR 3$
5026 017734 010046      1$: MOV RO,-(SP)
5027 017736 017600 000002 MOV 22(SP),RO
5028 017742 122737 000001 001230 CMPB #APTENV,SENV
5029 017750 001011      BNE 62$
5030 017752 132737 000100 001231 BITB #APTSPool,SENV
5031 017760 001405      BEQ 62$
5032 017762 010037 017772 MOV RO,61$
5033 017766 004737 020212 JSR PC,$ATY3
5034 017772 000000      .WORD 0
5035 017774 132737 000040 001231 61$: BITB #APTC SUP,SENV
5036 020002 001003      BNE 60$
5037 020004 112046      2$: MOV (RO)+,-(SP)
5038 020006 001005      BNE 4$
5039 020010 005726      TST (SP)+
5040 020012 012600      60$: MOV (SP)+,RO
5041 020014 062716 000002 3$: ADD #2,(SP)
5042 020020 000002      RTI
5043 020022 122716 000011 4$: CMPB #HT,(SP)
5044 020026 001430      BEQ 8$
5045 020030 122716 000200      CMPB #CRLF,(SP)
5046 020034 001006      BNE 5$
5047 020036 005726      TST (SP)+
5048 020040 104401      TYPE $CRLF
5049 020042 001205      $CRLF
5050 020044 105037 020200      CLRB $CHARCNT
5051 020050 000755      BR 2$
5052 020052 004737 020134 5$: JSR PC,$TYPEC
5053 020056 123726 001172 6$: CMPB $FILLC,(SP)+
5054 020062 001350      BNE 2$
5055 020064 013746 001170      MOV $NULL,-(SP)
5056
5057 020070 105366 000001 7$: DECB 1(SP)
5058 020074 002770      BLT 6$
5059 020076 004737 020134      JSR PC,$TYPEC
5060 020102 105337 020200      DECB $CHARCNT
5061 020106 000770      BR 7$
5062
5063 ;HORIZONTAL TAB PROCESSOR
5064
5065 020110 112716 000040 8$: MOVB #' ,(SP)
5066 020114 004737 020134 9$: JSR PC,$TYPEC
5067 020120 132737 000007 020200 BITB #7,$CHARCNT
5068 020126 001372      BNE 9$
5069 020130 005726      TST (SP)+
5070 020132 000724      BR 2$
5071 020134 105777 161024 STYPEC: TSTB $STPS
5072 020140 100375      BPL STYPEC
5073 020142 116677 000002 161016 MOVB 2(SP),$STPB
5074 020150 122766 000015 000002 CMPB #CR,2(SP)
5075 020156 001003      BNE 1$
5076 020160 105037 020200      CLRB $CHARCNT
5077 020164 000406      BR STYPEX
5078 020166 122766 000012 000002 1$: CMPB #LF,2(SP)
5079 020174 001402      BEQ STYPEX

```

```

:: HALT HERE IF NO TERMINAL
:: LEAVE
:: SAVE RO
:: GET ADDRESS OF ASCIZ STRING
:: RUNNING IN APT MODE
:: NO GO CHECK FOR APT CONSOLE
:: SPOOL MESSAGE TO APT
:: NO GO CHECK FOR CONSOLE
:: SETUP MESSAGE ADDRESS FOR APT
:: SPOOL MESSAGE TO APT
:: MESSAGE ADDRESS
:: APT CONSOLE SUPPRESSED
:: YES, SKIP TYPE OUT
:: PUSH CHARACTER TO BE TYPED ONTO STACK
:: BR IF IT ISN'T THE TERMINATOR
:: IF TERMINATOR POP IT OFF THE STACK
:: RESTORE RO
:: ADJUST RETURN PC
:: RETURN
:: BRANCH IF <HT>
:: BRANCH IF NOT <CRLF>
:: POP <CR><LF> EQUIV
:: TYPE A CR AND LF
:: CLEAR CHARACTER COUNT
:: GET NEXT CHARACTER
:: GO TYPE THIS CHARACTER
:: IS IT TIME FOR FILLER CHARS.?
:: IF NO GO GET NEXT CHAR.
:: GET # OF FILLER CHARS. NEEDED
:: AND THE NULL CHAR.
:: DOES A NULL NEED TO BE TYPED?
:: BR IF NO--GO POP THE NULL OFF OF STACK
:: GO TYPE A NULL
:: DO NOT COUNT AS A COUNT
:: LOOP

```



```

5080 020176 105227          INCB      (PC)+      ;;COUNT THE CHARACTER
5081 020200 000000          SCHARCNT: .WORD 0      ;;CHARACTER COUNT STORAGE
5082 020202 000207          STYPEX: RTS      PC
5083
5084
5085          .SBTTL  APT COMMUNICATIONS ROUTINE
5086
5087          ;;*****
5088 020204 112737 000001 020450 $ATY1:  MOVB      #1,$FFLG      ;;TO REPORT FATAL ERROR
5089 020212 112737 000001 020446 $ATY3:  MOVB      #1,$MFLG      ;;TO TYPE A MESSAGE
5090 020220 000403          BR          SATYC
5091 020222 112737 000001 020450 $ATY4:  MOVB      #1,$FFLG      ;;TO ONLY REPORT FATAL ERROR
5092 020230          SATYC:
5093 020230 010046          MOV      RO,-(SP)      ;;PUSH RO ON STACK
5094 020232 010146          MOV      R1,-(SP)      ;;PUSH R1 ON STACK
5095 020234 105737 020446          TSTB     $MFLG      ;;SHOULD TYPE A MESSAGE?
5096 020240 001450          BEQ      5$          ;;IF NOT: BR
5097 020242 122737 000001 001230          CMPB     #APTENV,$ENV      ;;OPERATING UNDER APT?
5098 020250 001031          BNE      3$          ;;IF NOT: BR
5099 020252 132737 000100 001231          BITB     #APTPOOL,$ENVM      ;;SHOULD SPOOL MESSAGES?
5100 020260 001425          BEQ      3$          ;;IF NOT: BR
5101 020262 017600 000004          MOV      @4(SP),RO      ;;GET MESSAGE ADDR.
5102 020266 062766 000002 000004          ADD      #2,4(SP)      ;;BUMP RETURN ADDR.
5103 020274 005737 001210          1$:     TST      $MSGTYPE      ;;SEE IF DONE W/ LAST XMISSION?
5104 020300 001375          BNE      1$          ;;IF NOT: WAIT
5105 020302 010037 001224          MOV      RO,$MSGAD      ;;PUT ADDR IN MAILBOX
5106 020306 105720          2$:     TSTB     (RO)+      ;;FIND END OF MESSAGE
5107 020310 001376          BNE      2$
5108 020312 163700 001224          SUB      $MSGAD,RO      ;;SUB START OF MESSAGE
5109 020316 006200          ASR      RO          ;;GET MESSAGE LNTH IN WORDS
5110 020320 010037 001226          MOV      RO,$MSGGLT      ;;PUT LENGTH IN MAILBOX
5111 020324 012737 000004 001210          MOV      #4,$MSGTYPE      ;;TELL APT TO TAKE MSG.
5112 020332 000413          BR          5$
5113 020334 017637 000004 020360 3$:     MOV      @4(SP),4$      ;;PUT MSG ADDR IN JSR LINKAGE
5114 020342 062766 000002 000004          ADD      #2,4(SP)      ;;BUMP RETURN ADDRESS
5115 020350 013746 177776          MOV      177776,-(SP)      ;;PUSH 177776 ON STACK
5116 020354 004737 017722          JSR      PC,$TYPE      ;;CALL TYPE MACRO
5117 020360 000000          4$:     .WORD      0
5118 020362          5$:
5119 020362 105737 020450          10$:    TSTB     $FFLG      ;;SHOULD REPORT FATAL ERROR?
5120 020366 001416          BEQ      12$          ;;IF NOT: BR
5121 020370 005737 001230          TST      $ENV          ;;RUNNING UNDER APT?
5122 020374 001413          BEQ      12$          ;;IF NOT: BR
5123 020376 005737 001210          11$:    TST      $MSGTYPE      ;;FINISHED LAST MESSAGE?
5124 020402 001375          BNE      11$          ;;IF NOT: WAIT
5125 020404 017637 000004 001212          MOV      @4(SP),$FATAL      ;;GET ERROR #
5126 020412 062766 000002 000004          ADD      #2,4(SP)      ;;BUMP RETURN ADDR.
5127 020420 005237 001210          INC      $MSGTYPE      ;;TELL APT TO TAKE ERROR
5128 020424 105037 020450          12$:    CLRB     $FFLG      ;;CLEAR FATAL FLAG
5129 020430 105037 020447          CLRB     $LFLG      ;;CLEAR LOG FLAG
5130 020434 105037 020446          CLRB     $MFLG      ;;CLEAR MESSAGE FLAG
5131 020440 012601          MOV      (SP)+,R1      ;;POP STACK INTO R1
5132 020442 012600          MOV      (SP)+,RO      ;;POP STACK INTO RO
5133 020444 000207          RTS      PC          ;;RETURN
5134 020446          000          $MFLG:  .BYTE      0      ;;MESSG. FLAG
5135 020447          000          $LFLG:  .BYTE      0      ;;LOG FLAG
    
```

```

5136 020450 000 $FFLG: .BYTE 0 ;;FATAL FLAG
5137 020452 .EVEN
5138 000200 APTSIZE=200
5139 000001 APTENV=001
5140 000100 APTSPool=100
5141 000040 APTCSUP=040
5142
5143 .SBTTL POWER DOWN AND UP ROUTINES
5144
5145 ;;*****
5146 :POWER DOWN ROUTINE
5147 020452 012737 020624 000024 $PWRDN: MOV $SILLUP,@#PWRVEC ;;SET FOR FAST UP
5148 020460 012737 000340 000026 MOV #340,@#PWRVEC+2 ;;PRIO:7
5149 020466 010046 MOV RO,-(SP) ;;PUSH RO ON STACK
5150 020470 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
5151 020472 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
5152 020474 010346 MOV R3,-(SP) ;;PUSH R3 ON STACK
5153 020476 010446 MOV R4,-(SP) ;;PUSH R4 ON STACK
5154 020500 010546 MOV R5,-(SP) ;;PUSH R5 ON STACK
5155 020502 017746 160446 MOV @SWR,-(SP) ;;PUSH @SWR ON STACK
5156 020506 010637 020630 MOV SP,$SAVR6 ;;SAVE SP
5157 020512 012737 020524 000024 MOV $PWRUP,@#PWRVEC ;;SET UP VECTOR
5158 020520 000000 HALT
5159 020522 000776 BR .-2 ;;HANG UP
5160
5161 ;;*****
5162 :POWER UP ROUTINE
5163 020524 012737 020624 000024 $PWRUP: MOV $SILLUP,@#PWRVEC ;;SET FOR FAST DOWN
5164 020532 013706 020630 MOV $SAVR6,SP ;;GET SP
5165 020536 005037 020630 CLR $SAVR6 ;;WAIT LOOP FOR THE TTY
5166 020542 005237 020630 15: INC $SAVR6 ;;WAIT FOR THE INC
5167 020546 001375 BNE 15 OF WORD
5168 020550 012677 160400 MOV (SP)+,@SWR ;;POP STACK INTO @SWR
5169 020554 012605 MOV (SP)+,R5 ;;POP STACK INTO R5
5170 020556 012604 MOV (SP)+,R4 ;;POP STACK INTO R4
5171 020560 012603 MOV (SP)+,R3 ;;POP STACK INTO R3
5172 020562 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
5173 020564 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
5174 020566 012600 MOV (SP)+,RO ;;POP STACK INTO RO
5175 020570 012737 020452 000024 MOV $PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
5176 020576 012737 000340 000026 MOV #340,@#PWRVEC+2 ;;PRIO:7
5177 020604 104401 TYPE ;;REPORT THE POWER FAILURE
5178 020606 020632 $PWRMG: .WORD $POWER ;;POWER FAIL MESSAGE POINTER
5179 020610 042766 000020 000002 BIC #20,2(SP) ;;CLEAR "T" BIT
5180 020616 005037 017364 CLR $TBIT ;;CLEAR THE "T" BIT FLAG
5181 020622 000002 RTI
5182 020624 000000 $SILLUP: HALT ;;THE POWER UP SEQUENCE WAS STARTED
5183 020626 000776 BR .-2 ;;BEFORE THE POWER DOWN WAS COMPLETE
5184 020630 000000 $SAVR6: 0 ;;PUT THE SP HERE
5185 020632 005015 047520 042527 $POWER: .ASCIZ <15><12>"POWER"
5186 020640 000122 .EVEN
5187
5188 .SBTTL BINARY TO OCTAL (ASCII) AND TYPE
5189
5190
5191 ;;*****

```



```

5192 ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
5193 ;*OCTAL (ASCII) NUMBER AND TYPE IT.
5194 ;*STYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
5195 ;*CALL:
5196 ;*      MOV      NUM,-(SP)          ;; NUMBER TO BE TYPED
5197 ;*      TYPOS    ;; CALL FOR TYPEOUT
5198 ;*      .BYTE   N                    ;; N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
5199 ;*      .BYTE   M                    ;; M=1 OR 0
5200 ;*                                     ;; 1=TYPE LEADING ZEROS
5201 ;*                                     ;; 0=SUPPRESS LEADING ZEROS
5202 ;*
5203 ;*STYPON-----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
5204 ;*STYPOS OR STYPOC
5205 ;*CALL:
5206 ;*      MOV      NUM,-(SP)          ;; NUMBER TO BE TYPED
5207 ;*      TYPON    ;; CALL FOR TYPEOUT
5208 ;*
5209 ;*STYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
5210 ;*CALL:
5211 ;*      MOV      NUM,-(SP)          ;; NUMBER TO BE TYPED
5212 ;*      TYPOC    ;; CALL FOR TYPEOUT
5213 ;*
5214 020642 017646 000000 21065 STYPOS: MOV      2(SP),-(SP)          ;; PICKUP THE MODE
5215 020646 116637 000001 21065 MOVVB   1(SP),SOFILL      ;; LOAD ZERO FILL SWITCH
5216 020654 112637 021067 21065 MOVVB   (SP)+,SOMODE+1    ;; NUMBER OF DIGITS TO TYPE
5217 020660 062716 000002 21065 ADD     #2,(SP)          ;; ADJUST RETURN ADDRESS
5218 020664 000406 21065 BR      STYPON
5219 020666 112737 000001 021065 STYPOC: MOVVB  #1,SOFILL      ;; SET THE ZERO FILL SWITCH
5220 020674 112737 000006 021067 MOVVB   #6,SOMODE+1      ;; SET FOR SIX(6) DIGITS
5221 020702 112737 000005 021064 STYPON: MOVVB  #5,SOCNT      ;; SET THE ITERATION COUNT
5222 020710 010346 21067 21064 MOV      R3,-(SP)        ;; SAVE R3
5223 020712 010446 21067 21064 MOV      R4,-(SP)        ;; SAVE R4
5224 020714 010546 21067 21064 MOV      R5,-(SP)        ;; SAVE R5
5225 020716 113704 021067 21064 MOVVB   SOMODE+1,R4      ;; GET THE NUMBER OF DIGITS TO TYPE
5226 020722 005404 21067 21064 NEG     R4
5227 020724 062704 000006 21066 ADD     #6,R4           ;; SUBTRACT IT FOR MAX. ALLOWED
5228 020730 110437 021066 21066 MOVVB   R4,SOMODE        ;; SAVE IT FOR USE
5229 020734 113704 021065 21066 MOVVB   SOFILL,R4        ;; GET THE ZERO FILL SWITCH
5230 020740 016605 000012 21066 MOV      12(SP),R5      ;; PICKUP THE INPUT NUMBER
5231 020744 005003 21066 21066 CLR     R3              ;; CLEAR THE OUTPUT WORD
5232 020746 006105 21066 21066 15: ROL    R5              ;; ROTATE MSB INTO "C"
5233 020750 000404 21066 21066 BR      3$             ;; GO DO MSB
5234 020752 006105 21066 21066 25: ROL    R5              ;; FORM THIS DIGIT
5235 020754 006105 21066 21066 ROL    R5
5236 020756 006105 21066 21066 ROL    R5
5237 020760 010503 21066 21066 MOV     R5,R3
5238 020762 006103 21066 21066 35: ROL    R3              ;; GET LSB OF THIS DIGIT
5239 020764 105337 021066 21066 DECB   SOMODE          ;; TYPE THIS DIGIT?
5240 020770 100016 21066 21066 BPL    7$              ;; BR IF NO
5241 020772 042703 177770 21066 BIC    #177770,R3      ;; GET RID OF JUNK
5242 020776 001002 21066 21066 BNE    4$              ;; TEST FOR 0
5243 021000 005704 21066 21066 TST    R4              ;; SUPPRESS THIS 0?
5244 021002 001403 21066 21066 BEQ    5$              ;; BR IF YES
5245 021004 005204 21066 21066 45: INC    R4              ;; DON'T SUPPRESS ANYMORE 0'S
5246 021006 052703 000060 21066 BIS    #'0,R3          ;; MAKE THIS DIGIT ASCII
5247 021012 052703 000040 21066 55: BIS    #' ,R3        ;; MAKE ASCII IF NOT ALREADY
    
```

```

5248 021016 110337 021062          MOVB   R3,B5          ;;SAVE FOR TYPING
5249 021022 104401 021062          TYPE   ,B5           ;;GO TYPE THIS DIGIT
5250 021026 105337 021064          7$:   DECB   $OCNT    ;;COUNT BY 1
5251 021032 003347          BGT    25           ;;BR IF MORE TO DO
5252 021034 002402          BLT    65           ;;BR IF DONE
5253 021036 005204          INC    R4           ;;INSURE LAST DIGIT ISN'T A BLANK
5254 021040 000744          BR     25           ;;GO DO THE LAST DIGIT
5255 021042 012605          6$:   MOV    (SP)+,R5  ;;RESTORE R5
5256 021044 012604          MOV    (SP)+,R4    ;;RESTORE R4
5257 021046 012603          MOV    (SP)+,R3    ;;RESTORE R3
5258 021050 016666 000002 000004  MOV    2(SP),4(SP)  ;;SET THE STACK FOR RETURNING
5259 021056 012616          MOV    (SP)+,(SP)
5260 021060 000002          RTI                    ;;RETURN
5261 021062 000          8$:   .BYTE   0          ;;STORAGE FOR ASCII DIGIT
5262 021063 000          .BYTE   0          ;;TERMINATOR FOR TYPE ROUTINE
5263 021064 000          $OCNT: .BYTE   0          ;;OCTAL DIGIT COUNTER
5264 021065 000          $OFILL: .BYTE  0          ;;ZERO FILL SWITCH
5265 021066 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

$TYPDS:
MOV    R0,-(SP)          ;;PUSH R0 ON STACK
MOV    R1,-(SP)          ;;PUSH R1 ON STACK
MOV    R2,-(SP)          ;;PUSH R2 ON STACK
MOV    R3,-(SP)          ;;PUSH R3 ON STACK
MOV    R5,-(SP)          ;;PUSH R5 ON STACK
MOV    #20200,-(SP)      ;;SET BLANK SWITCH AND SIGN
MOV    20(SP),R5        ;;GET THE INPUT NUMBER
BPL    1$              ;;BR IF INPUT IS POS.
NEG    R5              ;;MAKE THE BINARY NUMBER POS.
MOVB   #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
1$:   CLR    R0          ;;ZERO THE CONSTANTS INDEX
MOV    #SDBLK,R3       ;;SETUP THE OUTPUT POINTER
MOVB   #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
2$:   CLR    R2          ;;CLEAR THE BCD NUMBER
MOV    $DTBL(R0),R1    ;;GET THE CONSTANT
3$:   SUB    R1,R5       ;;FORM THIS BCD DIGIT
BLT    4$              ;;BR IF DONE
INC    R2              ;;INCREASE THE BCD DIGIT BY 1
4$:   ADD    R1,R5       ;;ADD BACK THE CONSTANT
TST    R2              ;;CHECK IF BCD DIGIT=0
BNE    5$              ;;FALL THROUGH IF 0
TSTB   (SP)            ;;STILL DOING LEADING 0'S?
BMI    7$              ;;BR IF YES
    
```



```

5304 021166 106316          55:  ASLB  (SP)          ;;MSD?
5305 021170 103003          BCC  65          ;;BR IF NO
5306 021172 116663 000001 177777  MOVB 1(SP),-1(R3) ;;YES--SET THE SIGN
5307 021200 052702 000060  BIS  #'0,R2      ;;MAKE THE BCD DIGIT ASCII
5308 021204 052702 000040  75:  BIS  #' R2      ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
5309 021210 110223          MOVB R2,(R3)+    ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
5310 021212 005720          TST  (R0)+       ;;JUST INCREMENTING
5311 021214 020027 000010  CMP  R0,#10     ;;CHECK THE TABLE INDEX
5312 021220 002746          BLT  25         ;;GO DO THE NEXT DIGIT
5313 021222 003002          BGT  85         ;;GO TO EXIT
5314 021224 010502          MOV  R5,R2      ;;GET THE LSD
5315 021226 000764          BR   65         ;;GO CHANGE TO ASCII
5316 021230 105726          85:  TSTB (SP)+    ;;WAS THE LSD THE FIRST NON-ZERO?
5317 021232 100003          BPL  95         ;;BR IF NO
5318 021234 116663 177777 177776  MOVB -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
5319 021242 105013          95:  CLRB (R3)      ;;SET THE TERMINATOR
5320 021244 012605          MOV  (SP)+,R5   ;;POP STACK INTO R5
5321 021246 012603          MOV  (SP)+,R3   ;;POP STACK INTO R3
5322 021250 012602          MOV  (SP)+,R2   ;;POP STACK INTO R2
5323 021252 012601          MOV  (SP)+,R1   ;;POP STACK INTO R1
5324 021254 012600          MOV  (SP)+,R0   ;;POP STACK INTO R0
5325 021256 104401 021304  TYPE  $DBLK      ;;NOW TYPE THE NUMBER
5326 021262 016666 000002 000004  MOV  2(SP),4(SP) ;;ADJUST THE STACK
5327 021270 012616          MOV  (SP)+,(SP)
5328 021272 000002          RTI                    ;;RETURN TO USER
5329 021274 023420          SOTBL: 10000.
5330 021276 001750          1000.
5331 021300 000144          100.
5332 021302 000012          10.
5333 021304 000004          SDBLK: .BLKW 4
5334
5335          .SBTTL TTY INPUT ROUTINE
5336
5337          ;:*****
5338          .ENABL LSB
5339
5340          ;:*****
5341          ;:SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
5342          ;:ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
5343          ;:SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
5344          ;:WHEN OPERATING IN TTY FLAG MODE.
5345 021314 022737 000176 001154  $CKSWR: CMP  #SWREG,SWR ;; IS THE SOFT-SWR SELECTED?
5346 021322 001074          BNE  155       ;; BRANCH IF NO
5347 021324 105777 157630  TSTB @STKS     ;; CHAR THERE?
5348 021330 100071          BPL  155       ;; IF NO, DON'T WAIT AROUND
5349 021332 117746 157624  MOVB @STKB,-(SP) ;; SAVE THE CHAR
5350 021336 042716 177600  BIC  #'C177,(SP) ;; STRIP-OFF THE ASCII
5351 021342 022726 000007  CMP  #'7,(SP)+  ;; IS IT A CONTROL G?
5352 021346 001062          BNE  155       ;; NO, RETURN TO USER
5353 021350 123727 001150 000001  CMPB $AUTOB,#1  ;; ARE WE RUNNING IN AUTO-MODE?
5354 021356 001456          BEQ  155       ;; BRANCH IF YES
5355
5356 021360 104401 021723          SGTSWR: TYPE  ,SCNTLG ;; ECHO THE CONTROL-G (↑G)
5357 021364 104401 021730          TYPE  $MSWR    ;; TYPE CURRENT CONTENTS
5358 021370 013746 000176          MOV  $SWREG,-(SP) ;; SAVE SWREG FOR TYPEOUT
5359 021374 104402          TYPOC          ;; GO TYPE--OCTAL ASCII(ALL DIGITS)

```

```

5360 021376 104401 021741          TYPE      ,SMNEW      ;; PROMPT FOR NEW SWR
5361 021402 005046          19$: CLR      -(SP)      ;; CLEAR COUNTER
5362 021404 005046          CLR      -(SP)      ;; THE NEW SWR
5363 021406 105777 157546      7$: TSTB   2$TKS      ;; CHAR THERE?
5364 021412 100375          BPL      7$         ;; IF NOT TRY AGAIN
5365
5366 021414 117746 157542          MOVB   2$TKB, -(SP)  ;; PICK UP CHAR
5367 021420 042716 177600          BIC    #1C177, (SP) ;; MAKE IT 7-BIT ASCII
5368
5369
5370
5371 021424 021627 000025      9$: CMP    (SP), #25   ;; IS IT A CONTROL-U?
5372 021430 001005          BNE    10$         ;; BRANCH IF NOT
5373 021432 104401 021716          TYPE   $CNTLU      ;; YES, ECHO CONTROL-U (↑U)
5374 021436 062706 000006      20$: ADD   #6, SP     ;; IGNORE PREVIOUS INPUT
5375 021442 000757          BR     19$        ;; LET'S TRY IT AGAIN
5376
5377
5378 021444 021627 000015      10$: CMP   (SP), #15   ;; IS IT A <CR>?
5379 021450 001022          BNE   16$         ;; BRANCH IF NO
5380 021452 005766 000004          TST   4(SP)      ;; YES, IS IT THE FIRST CHAR?
5381 021456 001403          BEQ   11$        ;; BRANCH IF YES
5382 021460 016677 000002 157466      MOV    2(SP), 2$SWR ;; SAVE NEW SWR
5383 021466 062706 000006      11$: ADD   #6, SP     ;; CLEAR UP STACK
5384 021472 104401 001205      14$: TYPE  $CR LF    ;; ECHO <CR> AND <LF>
5385 021476 123727 001151 000001      CMPB  $INTAG, #1   ;; RE-ENABLE TTY KBD INTERRUPTS?
5386 021504 001003          BNE   15$        ;; BRANCH IF NOT
5387 021506 012777 000100 157444      MOV   #100, 2$TKS  ;; RE-ENABLE TTY KBD INTERRUPTS
5388 021514 000002          RTI                    ;; RETURN
5389 021516 004737 020134      15$: JSR   PC, $TYPEC  ;; ECHO CHAR
5390 021522 021627 000060      16$: CMP   (SP), #60   ;; CHAR < 0?
5391 021526 002420          BLT   18$        ;; BRANCH IF YES
5392 021530 021627 000067          CMP   (SP), #67   ;; CHAR > 7?
5393 021534 003015          BGT   18$        ;; BRANCH IF YES
5394 021536 042726 000060          BIC   #60, (SP)+  ;; STRIP-OFF ASCII
5395 021542 005766 000002          TST   2(SP)      ;; IS THIS THE FIRST CHAR
5396 021546 001403          BEQ   17$        ;; BRANCH IF YES
5397 021550 006316          ASL   (SP)      ;; NO, SHIFT PRESENT
5398 021552 006316          ASL   (SP)      ;; CHAR OVER TO MAKE
5399 021554 006316          ASL   (SP)      ;; ROOM FOR NEW ONE.
5400 021556 005266 000002      17$: INC   2(SP)      ;; KEEP COUNT OF CHAR
5401 021562 056616 177776          BIS   -2(SP), (SP) ;; SET IN NEW CHAR
5402 021566 000707          BR    7$         ;; GET THE NEXT ONE
5403 021570 104401 001204      18$: TYPE  $QUES      ;; TYPE ?<CR><LF>
5404 021574 000720          BR    20$        ;; SIMULATE CONTROL-U
5405          .DSABL  LSB
5406
5407
5408          ;*****
5409          ;THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
5410          ;CALL:
5411          ; RDCHR          ;; INPUT A SINGLE CHARACTER FROM THE TTY
5412          ; RETURN HERE   ;; CHARACTER IS ON THE STACK
5413          ;               ;; WITH PARITY BIT STRIPPED OFF
5414          ;
5415

```



```

5416 021576 011646          SRDCHR: MOV      (SP), -(SP)      ;; PUSH DOWN THE PC
5417 021600 016666 000004 000002  MOV      4(SP), 2(SP)      ;; SAVE THE PS
5418 021606 105777 157346    1$:  TSTB     2$TKS          ;; WAIT FOR
5419 021612 100375          BPL      1$              ;; A CHARACTER
5420 021614 117766 157342 000004  MOVB     2$TKB, 4(SP)      ;; READ THE TTY
5421 021622 042766 177600 000004  BIC      #1C<177>, 4(SP)   ;; GET RID OF JUNK IF ANY
5422 021630 026627 000004 000023  CMP      4(SP), #23       ;; IS IT A CONTROL-S?
5423 021636 001013          BNE      3$              ;; BRANCH IF NO
5424 021640 105777 157314    2$:  TSTB     2$TKS          ;; WAIT FOR A CHARACTER
5425 021644 100375          BPL      2$              ;; LOOP UNTIL ITS THERE
5426 021646 117746 157310    MOVB     2$TKB, -(SP)      ;; GET CHARACTER
5427 021652 042716 177600    BIC      #1C177, (SP)     ;; MAKE IT 7-BIT ASCII
5428 021656 022627 000021    CMP      (SP)+, #21       ;; IS IT A CONTROL-Q?
5429 021662 001366          BNE      2$              ;; IF NOT DISCARD IT
5430 021664 000750          BR       1$              ;; YES, RESUME
5431 021666 026627 000004 000140  3$:  CMP      4(SP), #140    ;; IS IT UPPER CASE?
5432 021674 002407          BLT      4$              ;; BRANCH IF YES
5433 021676 026627 000004 000175    CMP      4(SP), #175     ;; IS IT A SPECIAL CHAR?
5434 021704 003003          BGT      4$              ;; BRANCH IF YES
5435 021706 042766 000040 000004  BIC      #40, 4(SP)       ;; MAKE IT UPPER CASE
5436 021714 000002          RTI                      ;; GO BACK TO USER
5437 021716 052536 005015 000    SCNTLU: .ASCIZ  /↑U/<15><12>  ;; CONTROL "U"
5438 021723 0136 006507 000012  SCNTLG: .ASCIZ  /↑G/<15><12>  ;; CONTROL "G"
5439 021730 005015 053523 020122  SMSWR:  .ASCIZ  <15><12>/SWR = /
5440 021736 020075 000
5441 021741 040 047040 053505  SMNEW:  .ASCIZ  / NEW = /
5442 021746 036440 000040

```

.SBTTL SAVE AND RESTORE RO-R5 ROUTINES

```

*****
;SAVE RO-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---RO

```

```

$SAVREG:
MOV      RO, -(SP)      ;; PUSH RO 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

```

```

5462 021752          010046
5463 021752 010146
5464 021754 010146
5465 021756 010246
5466 021760 010346
5467 021762 010446
5468 021764 010546
5469 021766 016646 000022
5470 021772 016646 000022
5471 021776 016646 000022

```

```

5472 022002 016646 000022      MOV      22(SP),-(SP)      ;;SAVE PC OF CALL
5473 022006 000002              RTI
5474
5475      ;#RESTORE RO-R5
5476      ;#CALL:
5477      ;# RESREG
5478      $RESREG:
5479 022010 012666 000022      MOV      (SP)+,22(SP)      ;;RESTORE PC OF CALL
5480 022014 012666 000022      MOV      (SP)+,22(SP)      ;;RESTORE PS OF CALL
5481 022020 012666 000022      MOV      (SP)+,22(SP)      ;;RESTORE PC OF MAIN FLOW
5482 022024 012666 000022      MOV      (SP)+,22(SP)      ;;RESTORE PS OF MAIN FLOW
5483 022030 012605              MOV      (SP)+,R5          ;;POP STACK INTO R5
5484 022032 012604              MOV      (SP)+,R4          ;;POP STACK INTO R4
5485 022034 012603              MOV      (SP)+,R3          ;;POP STACK INTO R3
5486 022036 012602              MOV      (SP)+,R2          ;;POP STACK INTO R2
5487 022040 012601              MOV      (SP)+,R1          ;;POP STACK INTO R1
5488 022042 012600              MOV      (SP)+,R0          ;;POP STACK INTO R0
5489 022044 000002              RTI
5490
5491      .SBTTL  DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE
5492
5493      ;*****
5494      ;THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
5495      ;DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
5496      ;POSITIVE.
5497      ;CALL
5498      ;# MOV      #PNTR, -(SP)      ;; POINTER TO LOW WORD OF BINARY NUMBER
5499      ;# JSR      PC, @#$DB2D
5500      ;# RETURN
5501      ;# THE FIRST ADDRESS OF ASCIZ
5502      ;# IS ON THE STACK
5503
5504 022046 104412 000002      $DB2D: SAVREG      ;; SAVE REGISTERS
5505 022050 016602 000002      MOV      2(SP),R2          ;; PICKUP THE DATA POINTER
5506 022054 012700 022226      MOV      #$DECVL,R0        ;; GET ADDRESS OF "SDECVL" STRING
5507 022060 010066 000002      MOV      R0,2(SP)          ;; PUT ADDRESS OF ASCIZ STRING ON STACK
5508 022064 012201              MOV      (R2)+,R1          ;; PICKUP THE BINARY NUMBER
5509 022066 012202              MOV      (R2)+,R2
5510 022070 012737 000012 022144      MOV      #10,4$           ;; SET UP TO DO 10 CONVERSIONS
5511 022076 012704 022156      MOV      #STNPWR,R4        ;; ADDRESS OF TEN POWER
5512 022102 012705 022160      MOV      #STNPWR+2,R5
5513 022106 005003      1$: CLR      R3              ;; CLEAR PARTIAL
5514 022110 161401      2$: SUB      (R4),R1          ;; SUBTRACT TEN POWER
5515 022112 005602              SBC      R2
5516 022114 161502              SUB      (R5),R2
5517 022116 002402              BLT      3$                ;; BR IF TEN POWER TO LARGE
5518 022120 005203              INC      R3                ;; ADD 1 TO PARTIAL
5519 022122 000772              BR       2$                ;; LOOP
5520 022124 062401      3$: ADD      (R4)+,R1          ;; RESTORE SUBTRACTED VALUE
5521 022126 005502              ADC      R2
5522 022130 062402              ADD      (R4)+,R2
5523 022132 022525              CMP      (R5)+,(R5)+
5524 022134 052703 000060      BIS      #'0,R3           ;; MOVE TO NEXT TEN POWER
5525 022140 110320              MOVB    R3,(R0)+          ;; CHANGE PARTIAL TO ASCII
5526 022142 005327              DEC      (PC)+            ;; SAVE IT
5527 022144 000000      4$: .WORD    0            ;; DONE?
    
```



```

5528 022146 001357           BNE      1$           ;; BR IF NO
5529 022150 105020           CLRB    (R0)+        ;; TERMINATOR
5530 022152 104413           RESREG                   ;; RESTORE REGISTERS
5531 022154 000207           RTS     PC           ;; RETURN
5532 022156 145000           STNPR: 145000        ;; 1.0E09
5533 022160 035632           35632
5534 022162 160400           160400                ;; 1.0E08
5535 022164 002765           2765
5536 022166 113200           113200                ;; 1.0E07
5537 022170 000230           230
5538 022172 041100           041100                ;; 1.0E06
5539 022174 000017           17
5540 022176 103240           103240                ;; 1.0E05
5541 022200 000001           1
5542 022202 023420           23420                 ;; 1.0E04
5543 022204 000000           0
5544 022206 001750           1750                  ;; 1.0E03
5545 022210 000000           0
5546 022212 000144           144                   ;; 1.0E02
5547 022214 000000           0
5548 022216 000012           12                    ;; 1.0E01
5549 022220 000000           0
5550 022222 000001           1                      ;; 1.0E00
5551 022224 000000           0
5552 022226 000014           SDECVL: .BLKB 12.    ;; RESERVE STORAGE FOR ASCIZ STRING
5553
5554 .SBTTL DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
5555
5556 ;*****
5557 ;*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
5558 ;*UNSIGNED OCTAL ASCIZ NUMBER.
5559 ;*CALL
5560 ;*   MOV     #PNTR, -(SP)      ;; POINTER TO LOW WORD OF BINARY NUMBER
5561 ;*   JSR    PC, @SDB20       ;; CALL THE ROUTINE
5562 ;*   RETURN                    ;; THE ADDRESS OF THE FIRST ASCIZ CHAR. IS ON THE STACK
5563
5564
5565 022242 104412           SDB20: SAVREG                   ;; SAVE ALL REGISTERS
5566 022244 016601 000002       MOV     2(SP), R1          ;; PICKUP THE POINTER TO LOW WORD
5567 022250 012705 022361       MOV     #SOCTVL+13., R5   ;; POINTER TO DATA TABLE
5568 022254 012704 000014       MOV     #12., R4          ;; DO ELEVEN CHARACTERS
5569 022260 012703 177770       MOV     #1C7, R3         ;; MASK
5570 022264 012100           MOV     (R1)+, R0         ;; LOWER WORD
5571 022266 012101           MOV     (R1)+, R1         ;; HIGH WORD
5572 022270 005002           CLR     R2                ;; TERMINATOR
5573 022272 110245           1$:   MOVB   R2, -(R5)      ;; PUT CHARACTER IN DATA TABLE
5574 022274 010002           MOV     R0, R2           ;; GET THIS DIGIT
5575 022276 005304           DEC     R4               ;; COUNT THIS CHARACTER
5576 022300 003007           BGT    3$                ;; BR IF NOT THE LAST DIGIT
5577 022302 001405           BEQ    2$                ;; BR IF IT IS THE LAST DIGIT
5578 022304 005205           INC     R5               ;; ALL DIGITS DONE-ADJUST POINTER FOR FIRST
5579 022306 010566 000002       MOV     R5, 2(SP)        ;; ASCIZ CHAR. & PUT IT ON THE STACK
5580 022312 104413           RESREG                   ;; RESTORE ALL REGISTERS
5581 022314 000207           RTS     PC               ;; RETURN TO USER
5582 022316 006203           2$:   ASR    R3            ;; POSITION THE MASK FOR THE LAST DIGIT
5583 022320 006001           3$:   ROR    R1            ;; POSITION THE BINARY NUMBER FOR
    
```

```

5584 022322 006000 ROR RO ;; THE NEXT OCTAL DIGIT
5585 022324 006001 ROR R1
5586 022326 006000 ROR RO
5587 022330 006001 ROR R1
5588 022332 006000 ROR RO
5589 022334 040302 BIC R3,R2 ;; MASK OUT ALL JUNK
5590 022336 062702 000060 ADD #'0,R2 ;; MAKE THIS CHAR. ASCII
5591 022342 000753 BR 1$ ;; GO PUT IT IN THE DATA TABLE
5592 022344 000016 $OCTVL: .BLKB 14. ;; RESERVE DATA TABLE
5593
5594 .SBTTL SCOPE HANDLER ROUTINE
5595
5596 *****
5597 *THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
5598 *AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
5599 *AND LOAD THE ERROR FLAG (SERFLG) INTO DISPLAY<15:08>
5600 *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
5601 *SW14=1 LOOP ON TEST
5602 *SW11=1 INHIBIT ITERATIONS
5603 *SW09=1 LOOP ON ERROR
5604 *SW08=1 LOOP ON TEST IN SWR<7:0>
5605 *CALL
5606 * SCOPE ;;SCOPE=IOT
5607
5608 022362 $SCOPE:
5609 022362 104407 CKSWR ;; TEST FOR CHANGE IN SOFT-SWR
5610 022364 032777 040000 156562 1$: BIT #BIT14,$SWR ;; LOOP ON PRESENT TEST?
5611 022372 001114 BNE $OVER ;; YES IF SW14=1
5612 *****START OF CODE FOR THE XOR TESTER*****
5613 022374 000416 $XTSTR: BR 6$ ;; IF RUNNING ON THE "XOR" TESTER CHANGE
5614 ;; THIS INSTRUCTION TO A "NOP" (NOP=240)
5615 022376 013746 000004 MOV @ERRVEC,-(SP) ;; SAVE THE CONTENTS OF THE ERROR VECTOR
5616 022402 012737 022422 000004 MOV #5$,@ERRVEC ;; SET FOR TIMEOUT
5617 022410 005737 177060 TST @#177060 ;; TIME OUT ON XOR?
5618 022414 012637 000004 MOV (SP)+,@ERRVEC ;; RESTORE THE ERROR VECTOR
5619 022420 000463 BR $SVLAD ;; GO TO THE NEXT TEST
5620 022422 022626 5$: CMP (SP)+,(SP)+ ;; CLEAR THE STACK AFTER A TIME OUT
5621 022424 012637 000004 MOV (SP)+,@ERRVEC ;; RESTORE THE ERROR VECTOR
5622 022430 000423 BR 7$ ;; LOOP ON THE PRESENT TEST
5623 022432
5624 022432 032777 000400 156514 6$; *****END OF CODE FOR THE XOR TESTER*****
5625 022440 001404 BIT #BIT08,$SWR ;; LOOP ON SPEC. TEST?
5626 022442 127737 156506 001116 2$: BEQ 2$ ;; BR IF NO
5627 022450 001465 CMPB $SWR,$TSTNM ;; ON THE RIGHT TEST? SWR<7:0>
5628 022452 105737 001117 2$: BEQ $OVER ;; BR IF YES
5629 022456 001421 TSTB SERFLG ;; HAS AN ERROR OCCURRED?
5630 022460 123737 001131 001117 3$: BEQ 3$ ;; BR IF NO
5631 022466 101015 CMPB $ERMAX,SERFLG ;; MAX. ERRORS FOR THIS TEST OCCURRED?
5632 022470 032777 001000 156456 4$: BHI 3$ ;; BR IF NO
5633 022476 001404 BIT #BIT09,$SWR ;; LOOP ON ERROR?
5634 022500 013737 001124 001122 7$: BEQ 4$ ;; BR IF NO
5635 022506 000446 MOV $LPERR,$LPADR ;; SET LOOP ADDRESS TO LAST SCOPE
5636 022510 105037 001117 4$: BR $OVER
5637 022514 005037 001174 5$: CLRB SERFLG ;; ZERO THE ERROR FLAG
5638 022520 000415 CLR $TIMES ;; CLEAR THE NUMBER OF ITERATIONS TO MAKE
5639 022522 032777 004000 156424 3$: BR 1$ ;; ESCAPE TO THE NEXT TEST
BIT #BIT11,$SWR ;; INHIBIT ITERATIONS?

```



```

5640 022530 001011      BNE      1$          ;; BR IF YES
5641 022532 005737 001216  TST      $PASS      ;; IF FIRST PASS OF PROGRAM
5642 022536 001406      BEQ      1$          ;; INHIBIT ITERATIONS
5643 022540 005237 001120  INC      $ICNT      ;; INCREMENT ITERATION COUNT
5644 022544 023737 001174 001120  CMP      $TIMES,$ICNT ;; CHECK THE NUMBER OF ITERATIONS MADE
5645 022552 002024      BGE      $OVER      ;; BR IF MORE ITERATION REQUIRED
5646 022554 012737 000001 001120 1$: MOV     #1,$ICNT    ;; REINITIALIZE THE ITERATION COUNTER
5647 022562 013737 022640 001174  MOV     $SMXCNT,$TIMES ;; SET NUMBER OF ITERATIONS TO DO
5648 022570 105237 001116      INCB     $STNM      ;; COUNT TEST NUMBERS
5649 022574 113737 001116 001214  MOVB   $STNM,$STESTN ;; SET TEST NUMBER IN APT MAILBOX
5650 022602 011637 001122      MOV     (SP),$LPADR ;; SAVE SCOPE LOOP ADDRESS
5651 022606 011637 001124      MOV     (SP),$LPERR ;; SAVE ERROR LOOP ADDRESS
5652 022612 005037 001176      CLR     $ESCAPE    ;; CLEAR THE ESCAPE FROM ERROR ADDRESS
5653 022616 112737 000001 001131  MOVB   #1,$ERMAX   ;; ONLY ALLOW ONE(1) ERROR ON NEXT TEST
5654 022624 013777 001116 156324 $OVER: MOV     $STNM,$DISPLAY ;; DISPLAY TEST NUMBER
5655 022632 013716 001122      MOV     $LPADR,(SP) ;; FUDGE RETURN ADDRESS
5656 022636 000002      RTI                    ;; FIXES PS
5657 022640 003720      SMXCNT: 2000.        ;; MAX. NUMBER OF ITERATIONS

```

.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.

```

```

5667 022642 010046      STRAP: MOV     RO,-(SP) ;; SAVE RO
5668 022644 016600 000002  MOV     2(SP),RO      ;; GET TRAP ADDRESS
5669 022650 005740      TST     -(RO)        ;; BACKUP BY 2
5670 022652 111000      MOVB   (RO),RO      ;; GET RIGHT BYTE OF TRAP
5671 022654 006300      ASL    RO           ;; POSITION FOR INDEXING
5672 022656 016000 022676  MOV     $TRPAD(RO),RO ;; INDEX TO TABLE
5673 022662 000200      RTS     RO          ;; GO TO ROUTINE

```

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

```

5678 022664 011646      STRAP2: MOV    (SP),-(SP) ;; MOVE THE PC DOWN
5679 022666 016666 000004 000002  MOV    4(SP),2(SP)    ;; MOVE THE PSW DOWN
5680 022674 000002      RTI                    ;; RESTORE THE PSW

```

.SBTTL TRAP TABLE

```

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

```

```

5687 : ROUTINE
5688 : -----
5689 022676 022664  $TRPAD: .WORD  $STRAP2
5690 022700 017722      $TYPE  ;; CALL=TYPE      TRAP+1(104401) TTY TYPEOUT ROUTINE
5691 022702 020666      $TYPOC ;; CALL=TYPOC      TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
5692 022704 020642      $TYPOS ;; CALL=TYPOS      TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
5693 022706 020702      $TYPON ;; CALL=TYPON      TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
5694 022710 021070      $TYPDS ;; CALL=TYPDS      TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
5695

```

```

5696 022712 021364          SGTSWR ;;CALL=GTSWR   TRAP+6(104406)  GET SOFT-SWR SETTING
5697
5698 022714 021314          SCKSWR ;;CALL=CKSWR   TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
5699 022716 021576          S'DCHR ;;CALL=RDCHR   TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
5700 022720 016644          SRDLIN ;;CALL=RDLIN   TRAP+11(104411) TTY TYPEIN STRING ROUTINE
5701 022722 021752          S$AVREG ;;CALL=SAVREG TRAP+12(104412) SAVE RO-RS ROUTINE
5702 022724 022010          S$RESREG ;;CALL=RESREG TRAP+13(104413) RESTORE RO-RS ROUTINE
5703 022726 016074          S$DSPLY ;;CALL=DISPLY TRAP+14(104414) ROUTINE TO TYPE ERROR MESSAGES
5704
5705
5706
5707
5708
5709
5710
5711
5712
5713
5714
5715
5716
5717
5718
5719
5720
5721
5722
5723
5724
5725 022730 000000 000000 000000 RMERRS: .WORD 0,0,0,0
5726 022736 000000
5727
5728
5729
5730
5731
5732
5733 022740 000
5734 022741 000
5735 022742 000
5736 022743 000
5737 022744 000
5738 022745 000
5739 022746 000
5740 022747 000
5741
5742
5743
5744
5745
5746
5747 022750 000
5748 022751 000
5749 022752 000
5750 022753 000
5751 022754 000

```

STERM=-STRPAD

;;*****

.SBTTL SINGLE/DUAL PORT RH70/RM03 DRIVER (REV 1.0)

```

;COPYRIGHT (C) 1976
;DIGITAL EQUIPMENT CORP.
;MAYNARD, MA 01754
;AUTHOR(S): JIM LACEY/CHUCK HESS/C. CHEN

```

;;*****

;STORAGE FOR RMDS, RMER1, RMER2, AND RMMR2 ON AN ERROR "2"

```

;RMERRS = RMDS
;RMERRS+2 = RMER1
;RMERRS+4 = RMER2
;RMERRS+6 = RMMR2

```

```

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

```

```

DRVACT: .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

```

```

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

```

```

DRVSTA: .BYTE 0 ;DRIVE 0
        .BYTE 0 ;DRIVE 1
        .BYTE 0 ;DRIVE 2
        .BYTE 0 ;DRIVE 3
        .BYTE 0 ;DRIVE 4

```


5752 022755 000
5753 022756 000
5754 022757 000

.BYTE 0 ;DRIVE 5
.BYTE 0 ;DRIVE 6
.BYTE 0 ;DRIVE 7

5755
5756
5757
5758
5759
5760

;TABLE OF DRIVE TYPES (DRVTYP=8 BYTES)
;DRVTYP=0 IF DRIVE IS NONEXISTENT (DRVSTA=0, ALSO)
;DRVTYP=4 IF DRIVE IS RM03
;DRVTYP=-1 IF NOT RM03

5761 022760 000
5762 022761 000
5763 022762 000
5764 022763 000
5765 022764 000
5766 022765 000
5767 022766 000
5768 022767 000

DRVTYP: .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

5769
5770
5771
5772
5773

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

5774 022770 000
5775 022771 000
5776 022772 000
5777 022773 000
5778 022774 000
5779 022775 000
5780 022776 000
5781 022777 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

5782
5783
5784
5785
5786

;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

5787 023000 000
5788 023001 000
5789 023002 000
5790 023003 000
5791 023004 000
5792 023005 000
5793 023006 000
5794 023007 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

5795
5796
5797
5798
5799

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

5800 023010 000000

TRNSWT: .WORD 0

5801
5802
5803
5804
5805
5806
5807

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

```

5808 023012 000000      SRCHWT: .WORD  0
5809
5810                    ;RMO3 DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)
5811                    ;ACTDRV=0 IF DRIVER IS INACTIVE
5812                    ;ACTDRV>0 IF DRIVER IS ACTIVE
5813
5814 023014      000      ACTDRV: .BYTE  0
5815
5816                    ;SOFTWARE TIMER ROUTINE ACTIVE FLAG (ACTSTR=1 BYTE)
5817                    ;ACTSTR=0 IF SOFTWARE TIMER ROUTINE IS INACTIVE
5818                    ;ACTSTR>0 IF SOFTWARE TIMER ROUTINE IS ACTIVE
5819
5820 023015      000      ACTSTR: .BYTE  0
5821
5822                    ;UNLOAD FLAG (ULDFLG=8 BYTES)
5823                    ;ULDFLG=0 IF NO UNLOAD COMMAND
5824                    ;ULDFLG>0 IF UNLOAD COMMAND IN PROGRESS
5825                    ;ULDFLG<0 IF UNLOAD COMMAND IN WAIT QUEUE
5826
5827 023016      000      ULDFLG: .BYTE  0          ;DRIVE 0
5828 023017      000          .BYTE  0          ;DRIVE 1
5829 023020      000          .BYTE  0          ;DRIVE 2
5830 023021      000          .BYTE  0          ;DRIVE 3
5831 023022      000          .BYTE  0          ;DRIVE 4
5832 023023      000          .BYTE  0          ;DRIVE 5
5833 023024      000          .BYTE  0          ;DRIVE 6
5834 023025      000          .BYTE  0          ;DRIVE 7
5835
5836                    ;LOOK AHEAD COUNT (LACNT=8 BYTES)
5837                    ;LACNT WILL INDICATE THE NUMBER OF LOOK AHEADS PERFORMED
5838
5839 023026      000      LACNT: .BYTE  0          ;DRIVE 0
5840 023027      000          .BYTE  0          ;DRIVE 1
5841 023030      000          .BYTE  0          ;DRIVE 2
5842 023031      000          .BYTE  0          ;DRIVE 3
5843 023032      000          .BYTE  0          ;DRIVE 4
5844 023033      000          .BYTE  0          ;DRIVE 5
5845 023034      000          .BYTE  0          ;DRIVE 6
5846 023035      000          .BYTE  0          ;DRIVE 7
5847
5848                    ;SAVE REGISTERS FLAG (SAVEFG =1 WORD)
5849                    ;SAVEFG <0 IF SAVE THE RH70/RMO3 REGISTERS WHEN THE
5850                    ;OPERATION IS COMPLETED AS PER (DPB+14).
5851                    ;SAVEFG=0 IF SAVE THE RH70/RMO3 REGISTERS, AS PER
5852                    ;(DPB+14), AFTER AN ERROR.
5853
5854 023036 000000      SAVEFG: .WORD  0
5855
5856                    ;SEEK FLAG (SEEKFG=1 WORD)
5857                    ;SEEKFG=0 IF WHEN THE DISK ADDRESS ISN'T IN THE WINDOW
5858                    ;FOR A DATA TRANSFER START A SEARCH COMMAND
5859                    ;SEEKFG<0 IF DATA TRANSFER WILL DO IMPLIED SEEKS,
5860                    ;DISREGARD THE WINDOW
5861
5862 023040 177777      SEEKFG: .WORD -1
5863

```



```

5864 ;TIMEOUT TABLE (TIMER=8 WORDS)
5865 ;THIS TABLE CONTAINS THE TIME ALLOWED FOR AN OPERATION
5866
5867 023042 177777 TIMER: .WORD -1 ;DRIVE 0
5868 023044 177777 .WORD -1 ;DRIVE 1
5869 023046 177777 .WORD -1 ;DRIVE 2
5870 023050 177777 .WORD -1 ;DRIVE 3
5871 023052 177777 .WORD -1 ;DRIVE 4
5872 023054 177777 .WORD -1 ;DRIVE 5
5873 023056 177777 .WORD -1 ;DRIVE 6
5874 023060 177777 .WORD -1 ;DRIVE 7
5875
5876 ;DATA TRANSFER UNDERWAY INDICATOR (DTUW=1 WORD)
5877 ;DTUW<0 IF NO DATA TRANSFER UNDERWAY
5878 ;DTUW=+N (WHERE N=0 TO 7) IMPLIES DATA TRANSFER UNDERWAY ON DRIVE N
5879
5880 023062 177777 DTUW: .WORD -1
5881
5882 ;ATTENTION BITS TABLE (ATABIT=8 BYTES)
5883 ;THIS TABLE CONTAINS THE CORRESPONDING BIT TO EACH DRIVES
5884 ;ATTENTION BIT
5885
5886 023064 001 ATABIT: .BYTE 1 ;DRIVE 0
5887 023065 002 .BYTE 2 ;DRIVE 1
5888 023066 004 .BYTE 4 ;DRIVE 2
5889 023067 010 .BYTE 10 ;DRIVE 3
5890 023070 020 .BYTE 20 ;DRIVE 4
5891 023071 040 .BYTE 40 ;DRIVE 5
5892 023072 100 .BYTE 100 ;DRIVE 6
5893 023073 200 .BYTE 200 ;DRIVE 7
5894
5895 ;FSRMO3 TO RH70 "MASSBUS CONTROL BUS PARITY ERRORS" (MCPE) ALLOWED BEFORE
5896 ;CALLING IT FATAL (MCPEMX=1 WORD)
5897
5898 023074 000003 MCPEMX: .WORD 3
5899
5900 ;STORAGE FOR RMADR (THE FIRST ADDRESS (776700) OF THE RH70/RMO3),
5901 ;RMVEC (THE VECTOR ADDRESS (254)), AND RMVEC+2 (THE BR LEVEL (5)).
5902
5903 023076 176700 RMADR: .WORD 176700
5904 023100 000254 000240 RMVEC: .WORD 254,5*32.
5905
5906 ;MAXIMUM NUMBER OF LOOK AHEADS ALLOWED IS 4 (MXLACT=1 WORD)
5907
5908 023104 000004 MXLACT: .WORD 4
5909 ;MAXIMUM DELTA DELAY IS 8 SECTORS (MXDLTA=1 WORD)
5910
5911 023106 001000 MXDLTA: .WORD 8.*64.
5912 ;MINIMUM DELTA DELAY IS 2 SECTORS (MNDLTA=1 WORD)
5913
5914 023110 000200 MNDLTA: .WORD 2*64.
5915 ;MAXIMUM SEARCH FOR I/O WINDOW IS 5 SECTORS (MXWINDW=1 WORD)
5916
5917 023112 000005 MXWINDW: .WORD 5
5918
5919 ;DEFINITIONS OF THE RH70/RMO3 ADDRESS INDEXES

```

```

5920
5921      000000      RMCS1=0      ;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 00)
5922      000002      RMWC=2      ;WORD COUNT REGISTER (NOT A DRIVE REG)
5923      000004      RMBA=4      ;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)
5924      000006      RMDA=6      ;DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 05)
5925      000010      RMCS2=10     ;CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
5926      000012      RMD5=12     ;DRIVE STATUS REGISTER (DRIVE REG 01)
5927      000014      RMR1=14     ;ERROR REGISTER #1 (DRIVE REG. 02)
5928      000016      RMAS=16     ;ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 04)
5929      000020      RMLA=20     ;LOOK AHEAD REGISTER (DRIVE REG. 07)
5930      000022      RMD8=22     ;DATA BUFFER REGISTER (NOT A DRIVE REG.)
5931      000024      RMMR1=24    ;MAINTAINABILITY REGISTER (DRIVE REG. 03)
5932      000026      RMDT=26     ;DRIVE TYPE REGISTER (DRIVE REG. 06)
5933      000030      RMSN=30     ;SERIAL NUMBER REGISTER (DRIVE REG. 10)
5934      000032      RMOF=32     ;OFFSET REGISTER (DRIVE REG. 11)
5935      000034      RMDC=34     ;DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
5936      000036      RMR2=36     ;DUMMY ADDRESS REGISTER (DRIVE REG. 13)
5937      000040      RMMR2=40    ;MAINTENANCE REGISTER #2
5938      000042      RMR2=42     ;ERROR REGISTER #2 (DRIVE REG. 15)
5939      000044      RMEC1=44    ;ECC POSITION REGISTER (DRIVE REG. 16)
5940      000046      RMEC2=46    ;ECC PATTERN REGISTER (DRIVE REG. 17)
5941
5942      ;RH70/RMO3 DRIVER INITIALIZATION CODE
5943      ;THIS ROUTINE WILL DETERMINE WHICH RMO3 DRIVES ARE
5944      ;AVAILABLE FOR TESTING AND SET THE DRVSTA INDICATOR
5945      ;TO THE PROPER STATE FOR EACH DRIVE.
5946      ;NOTE: THIS ROUTINE CALLS DRVINT
5947
5948      ;CALL
5949
5950      JSR      PC,RMINIT
5951      RETURN
5952
5953      ;NOTE: THE 'P' OR 'L' CLOCK MUST BE STARTED
5954
5955      RMINIT: SAVREG      ;SAVE R0 - R5
5956      MOV      @#PS, -(SP) ;SAVE THE PRESENT PROCESSOR STATUS
5957      MOV      @<5*32.>, @#PS ;CHANGE THE PRIORITY TO 5
5958      JSR      PC, CLRQUE ;CLEAR ALL REQUEST QUEUES
5959      MOV      @RMERRS, R1 ;FIRST ADDRESS TO BE CLEARED
5960      MOV      @SEEKFG, R2 ;LAST ADDRESS TO BE CLEARED
5961      CLR      (R1)+ ;CLEAR
5962      CMP      R1, R2 ;ARE WE DONE?
5963      BLO      1$ ;BRANCH IF NO
5964      MOV      @DTUW, R2 ;LAST ADDRESS
5965      MOV      @-1, (R1)+ ;INITIALIZE
5966      CMP      R1, R2 ;DONE?
5967      BLOS    2$ ;LOOP IF NO
5968      CLR      DRVSTA ;SET ALL DRIVES TO OFFLINE
5969      CLR      DRVSTA+2
5970      CLR      DRVSTA+4
5971      CLR      DRVSTA+6
5972      MOV      @RVEC, R3 ;SETUP THE RH70/RMO3 VECTOR
5973      MOV      @ISR, (R3)+
5974      MOV      @RVEC+2, (R3)
5975      MOV      @RADR, R4 ;FIRST ADDRESS OF RH70/RMO3

```



```

5976 023226 012764 000040 000010      MOV      #BIT05,RMCS2(R4)      ; MASSBUS INIT
5977 023234 005001                    CLR      R1                    ; START WITH DRIVE 0
5978 023236 004037 023326      3S:     JSR      R0,DRVINT        ; INIT THE DRIVE
5979 023242 000401                    BR       4S                    ; 'DVA' NOT SET OR PARITY ERROR
5980 023244 000402                    BR       5S                    ; NORMAL RETURN
5981 023246 105061 022750      4S:     CLRB   DRVSTA(R1)       ; SET DRIVE STATUS TO OFFLINE
5982 023252 005201      5S:     INC      R1                    ; GO TO NEXT DRIVE
5983 023254 042701 177770      BIC     #1C7,R1                ; MASK OUT UNUSED BITS
5984 023260 001366      BNE     3S                    ; BR IF MORE DRIVES TO GO
5985 023262 012701 000007      MOV     #7,R1                  ; START WITH DRIVE 7
5986 023266 005037 177776      CLR     #PS                    ; CLEAR THE PROCESSOR STATUS
5987 023272 105761 022770      6S:     TSTB   DPINT(R1)        ; WAITING FOR DRIVE TO SWITCH PORTS ?
5988 023276 001405      BEQ     8S                    ; BR NOT WAITING
5989 023300 004737 030364      JSR     PC,SET.IE              ; SET INTERRUPT
5990 023304 105761 022770      7S:     TSTB   DPINT(R1)        ; DRIVE SWITCHED PORTS ?
5991 023310 001375      BNE     7S                    ; BR IF NOT
5992 023312 005301      8S:     DEC     R1                    ; GO TO THE NEXT DRIVE
5993 023314 100366      BPL     6S                    ; CHECK NEXT DRIVE
5994 023316 012637 177776      MOV     (SP)+,#PS              ; RESTORE THE PROCESSOR STATUS
5995 023322 104413      RESREG PC                      ; RESTORE R0 - R5
5996 023324 000207      RTS      PC                    ; BYE-BYE
5997
5998      ;DRIVE INITIALIZATION ROUTINE
5999      ;THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
6000      ;AN RMO3. IF IT IS, A "READ-IN PRESET" IS ISSUED AND FMT22
6001      ;IS SET TO A "1". THEN MOL, DPR, DRY, AND VV ARE CHECKED TO
6002      ;INSURE THEY ARE ALL ON A "1". AND DEPENDING ON THEIR STATE,
6003      ;DRVSTA IS SET TO THE PROPER CONDITION.
6004
6005      ;CALL
6006      ;:
6007      ;:
6008      ;:
6009      ;:
6010      ;:
6011      ;:
6012 023326 010546      DRVINT: MOV     R5,-(SP)            ; SAVE R5
6013 023330 105061 022750      DULP:  CLRB   DRVSTA(R1)        ; START DRIVE STATUS AS OFFLINE
6014 023334 105061 022760      CLRB   DRVSTYP(R1)            ; CLEAR THE DRIVE TYPE INDICATOR
6015 023340 105061 023016      CLRB   ULDFLG(R1)            ; CLEAR THE UNLOAD FLAG
6016 023344 010164 000010      MOV     R1,RMCS2(R4)         ; SELECT A DRIVE
6017 023350 112764 000111 000000      MOVB   #111,RMCS1(R4)        ; DO A DRIVE CLEAR COMMAND (& SEIZE DRIVE)
6018 023356 032764 010000 000010      BIT    #BIT12,RMCS2(R4)      ; NONEXISTENT DRIVE?
6019 023364 001403      BEQ     1S                    ; NO---BRANCH
6020 023366 004737 030364      JSR     PC,SET.IE              ; GO SET "IE" WITHOUT A "TRE"
6021 023372 000476      BR      6S                    ; LEAVE THIS ROUTINE
6022 023374 105061 022750      1S:     CLRB   DRVSTA(R1)       ; SET DRIVE STATUS TO OFFLINE
6023 023400 032764 004000 000000      BIT    #BIT11,RMCS1(R4)      ; SEE IF DRIVE AVAILABLE
6024 023406 001750      BEQ     DULP                  ; BR IF DRIVE NOT AVAILABLE
6025 023410 004037 027674      JSR     R0,RD.RM              ; READ THE DRIVE TYPE REG.
6026 023414 000026      RMDT   8S                    ; ERROR RETURN ADDRESS
6027 023416 023614      8S:     MOV     (SP)+,R5        ; PUT DRIVE TYPE IN R5
6028 023420 012605      MOVB   #4,DRVSTYP(R1)        ; SET RMO3 INDICATOR
6029 023422 112761 000004 022760      MOVB   #20024,R5             ; SINGLE PORT RMO3 ?
6030 023430 022705 020024      CMP     #20024,R5             ; BR IF YES
6031 023434 001407      BEQ     2S
    
```

B10

```

6032 023436 022705 024024      CMP      #24024,R5      ;DUAL PORT RMO3 ?
6033 023442 001404      BEQ      2$          ;BR IF YES
6034 023444 112761 177777 022760      MOV      #1,DRVSTP(R1) ;SET INDICATOR TO 'OTHER'
6035 023452 000446      BR       6$          ;EXIT
6036 023454 012746 000121      2$:      MOV      #121,-(SP)   ;DO A "READ-IN PRESET"
6037 023460 004037 030054      JSR      RO,WRT.RM
6038 023464 000000      RMCS1
6039 023466 023614      BS
6040 023470 012746 010000      MOV      #BIT12,-(SP) ;SET FMT22=1
6041 023474 004037 030054      JSR      RO,WRT.RM
6042 023500 000032      RMOF
6043 023502 023614      BS
6044 023504 004037 027674      JSR      RO,RD.RM      ;READ RMO3
6045 023510 000012      RMCS
6046 023512 023614      BS
6047 023514 012605      MOV      (SP)+,R5     ;AND SAVE IT IN R5
6048 023516 100015      BPL      4$          ;BRANCH IF ATA=0
6049 023520 116164 023064 000016      MOV      ATABIT(R1),RMA$ (R4) ;CLEAR ATTENTION BIT
6050 023526 004037 027674      JSR      RO,RD.RM      ;FIND OUT WHY ATA=1
6051 023532 000014      RMER1
6052 023534 023614      BS
6053 023536 006126      ROL      (SP)+       ;IS IT UNSAFE?
6054 023540 100004      BPL      4$          ;BR IF NOT
6055 023542 112761 177777 022750      MOV      #1,DRVSTA(R1) ;SET UNSAFE INDICATOR
6056 023550 000407      BR       6$          ;EXIT
6057 023552 005105      4$:      COM      R5          ;CHECK MOL, DPR, DRY, AND VV
6058 023554 042705 167077      BIC      #1<BIT12:BIT08:BIT07:BIT06>,R5
6059 023560 001003      BNE      6$          ;BRANCH IF MOL, DPR, DRY, OR VV IS CLEAR
6060 023562 112761 000001 022750      MOV      #1,DRVSTA(R1) ;SET DRIVE STATUS TO ONLINE
6061 023570 005720      6$:      TST      (R0)+       ;STEP OVER THE ERROR RETURN
6062 023572 000410      BR       8$          ;EXIT
6063 023574 006301      7$:      ASL      R1          ;CHANGE INDEX TO ADDRESS WORDS
6064 023576 012761 003720 023042      MOV      #2000.,TIMER(R1) ;START 2 SEC TIMER
6065 023604 006201      ASR      R1          ;RESTORE R1
6066 023606 112761 177777 022770      MOV      #1,DPINT(R1) ;SET PORT INITIALIZE INIDICATOR
6067 023614 012605      8$:      MOV      (SP)+,R5     ;RESTORE R5
6068 023616 000200      RTS      R0          ;EXIT
6069
6070      ;REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
6071
6072      ;CALL
6073
6074      ;
6075      ;
6076      ;
6077      ;
6078      ;
6079      ;
6080 023620 013746 177776      RMO3:    MOV      @#PS,-(SP)   ;SAVE THE CALLING STATUS
6081 023624 013737 023102 177776      MOV      RMVEC+2,@#PS ;DON'T ALLOW ANY RMO3 INTERRUPTS
6082 023632 112737 000001 023014      MOV      #1,ACTDRV   ;SET "ACTIVE DRIVER" FLAG
6083 023640 104412      SAVREG   ;SAVE R0 - R5
6084 023642 011002      MOV      (R0),R2     ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
6085 023644 005062 000016      CLR      16(R2)      ;CLEAR THE STATUS/ERROR INDICATOR
6086 023650 111201      MOV      (R2),R1     ;PICKUP THE DRIVE NUMBER
6087 023652 013704 023076      MOV      RMA$R,R4    ;UNIBUS ADDRESS OF RMCS1
  
```



```

6088 023656 105761 022750      TSTB  DRVSTA(R1)      ;CHECK DRIVES STATUS
6089 023662 003014      BGT   1$             ;BRANCH IF ONLINE
6090 023664 105761 023016      TSTB  ULDFLG(R1)     ;UNLOAD COMMAND IN QUEUE?
6091 023670 001036      BNE   3$             ;BRANCH IF YES
6092 023672 105761 022770      TSTB  DPINT(R1)     ;TRYING TO INIT THE DRIVE
6093 023676 001042      BNE   5$             ;BR IF YES
6094 023700 004037 023326      JSR   RD,DRVINT     ;GO INIT. THE DRIVE
6095 023704 000434      BR    4$             ;ERROR RETURN
6096 023706 105761 022750      TSTB  DRVSTA(R1)     ;IS DRIVE STATUS ONLINE?
6097 023712 003445      BLE   6$             ;BR IF NOT
6098 023714 105761 023000      1$:  TSTB  DPRQS(R1)     ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
6099 023720 001031      BNE   5$             ;BR IF YES
6100 023722 010164 000010      MOV   R1,RMCS2(R4)  ;SELECT THE DRIVE
6101 023726 004037 031026      JSR   RD,DRVQUE     ;PUT THIS REQUEST IN QUEUE
6102 023732 000460      BR    9$             ;QUEUE IS FULL
6103 023734 122762 000103 000002      CMPB  #103,2(R2)    ;IS THIS REQ. FOR AN UNLOAD?
6104 023742 001003      BNE   2$             ;BR IF NO
6105 023744 112761 177777 023016      MOVB  #-1,ULDFLG(R1);SET THE "UNLOAD IN QUEUE" FLAG
6106 023752 105761 022740      2$:  TSTB  DRVACT(R1)   ;IS THIS DRIVE ACTIVE?
6107 023756 001043      BNE   8$             ;BR IF YES
6108 023760 004737 024112      JSR   PC,OPT        ;CALL THE OPTIMIZER
6109 023764 000440      BR    8$
6110 023766 012762 120000 000016 3$:  MOV   #BIT15!BIT13,16(R2);SET THE "UNLOAD IN QUEUE" ERROR FLAG
6111 023774 000434      BR    8$             ;EXIT
6112 023776 004737 025172      4$:  JSR   PC,C17        ;GO HANDLE THE PARITY ERROR
6113 024002 000431      BR    8$
6114 024004 004037 031026      5$:  JSR   RD,DRVQUE     ;PUT REQUEST IN QUEUE
6115 024010 000431      BR    9$             ;QUEUE IS FULL
6116 024012 032714 000100      BIT   #BIT06,(R4)   ;IE BIT SET ?
6117 024016 001023      BNE   8$             ;YES
6118 024020 004737 030364      JSR   PC,SET.IE     ;SET THE INTERRUPT
6119 024024 000420      BR    8$             ;RETURN
6120 024026 105761 022750      6$:  TSTB  DRVSTA(R1)   ;SEE IF DRIVE OFFLINE OR UNSAFE
6121 024032 002412      BLT   7$             ;BR IF UNSAFE
6122 024034 012762 140000 000016      MOV   #BIT15!BIT14,16(R2);SET OFFLINE ERROR INDICATOR
6123 024042 105761 022760      TSTB  DRVSTYP(R1)  ;SEE IF OFFLINE OR NONEXISTENT
6124 024046 001007      BNE   8$             ;BR IF OFFLINE
6125 024050 012762 100002 000016      MOV   #BIT15!BIT01,16(R2);REPORT DRIVE NONEXISTENT
6126 024056 000403      BR    8$             ;GO TO EXIT
6127 024060 012762 110000 000016 7$:  MOV   #BIT15!BIT12,16(R2);DRIVE IS UNSAFE
6128 024066 104413      8$:  RESREG ;RESTORE R0 - R5
6129 024070 005720      TST   (R0)+         ;SETUP FOR NORMAL RETURN
6130 024072 000401      BR    10$          ;FINISH UP, THEN EXIT
6131 024074 104413      9$:  RESREG ;RESTORE R0 - R5
6132 024076 005720      10$: TST   (R0)+         ;CORRECT THE RETURN ADDRESS
6133 024100 105037 023014      CLRB  ACTDRV        ;CLEAR "ACTIVE DRIVER" FLAG
6134 024104 012637 177776      MOV   (SP)+,2#PS    ;RETURN "PS" TO USER LEVEL
6135 024110 000200      RTS   RD            ;RETURN TO CALLER
6136
6137 ;OPTIMIZER-CALLED FOR A PARTICULAR DRIVE
6138
6139 ;CALL
6140
6141     MOV   #DRVNUM,R1 ;DRIVE NUMBER TO R1
6142     JSR   PC,OPT     ;SETUP A COMMAND
6143 024112 104412 OPT:  SAVREG ;SAVE R0 - R5

```

D10

```

6144 024114 013746 177776          MOV      2#PS, -(SP)      ;SAVE PROC. STATUS
6145 024120 146137 023064 023012  BICB    ATABIT(R1),SRCHWT ;CLEAR LA SEACH FLAG
6146 024126 105061 023000          CLRB    DPRQS(R1)      ;RESET THE PORT REQ FLAG ****
6147 024132 004737 031102          JSR     PC,GETREQ      ;GET "DPR" POINTER OF REQUEST
6148 024136 005702          TST     R2             ;IS THERE A REQUEST IN QUEUE?
6149 024140 001472          BEQ     7$             ;NO--BRANCH TO EXIT
6150 024142 010164 000010          MOV     R1,RMCS2(R4)   ;LOAD THE DRIVE ADDRESS *****
6151 024146 012764 000111 000000  MOV     #11,RMCS1(R4)  ;CLEAR THE DRIVE
6152 024154 032764 004000 000000  BIT     #BIT11,RMCS1(R4) ;DVA SET ?
6153 024162 001446          BEQ     5$             ;TO PROT REQUEST, IF NOT
6154 024164 105761 022750          TSTB   DRVSTA(R1)     ;IS DRIVE ONLINE?
6155 024170 003014          BGT     1$             ;YES--BRANCH
6156 024172 004737 031124          JSR     PC,POPQUE     ;NO--REMOVE REQUEST FROM QUEUE
6157 024176 012762 140000 000016  MOV     #BIT15!BIT14,16(R2) ;SET OFFLINE STATUS/ERROR INDICATOR
6158 024204 105761 022750          TSTB   DRVSTA(R1)     ;IS DRIVE UNSAFE ?
6159 024210 100053          BPL     8$             ;BR TO EXIT IF NOT
6160 024212 012762 110000 000016  MOV     #BIT15!BIT12,16(R2) ;SET UNSAFE STATUS/ERROR INDICATOR
6161 024220 000447          BR      8$             ;BRANCH TO EXIT
6162 024222          1$:
6163          :
6164          :
6165          :
6166          :
6167          :
6168          :
6169 024222 122762 000150 000002  MOV     #11, -(SP)     ;LOAD COMMAND ONTO THE STACK
6170 024230 002403          JSR     RO,WRT.RM     ;LOAD THE REGISTER
6171 024232 004737 024556          RMCS1  6$             ;REGISTER INCREMENT
6172 024236 000440          B$      ;ERROR RETURN ADDRESS
6173 024240 005737 023062          BIT     #BIT11, (R4)  ;DRIVE AVAILABLE ?
6174 024244 002012          BEQ     9$             ;BR IF NOT
6175 024246 005737 023040          CMPB   #150,2(R2)    ;IS THE REQUEST FOR I/O?
6176 024252 100404          BLT     2$             ;YES--BRANCH
6177 024254 004037 025526          JSR     PC,C14        ;CALL THE COMMAND INITIATOR
6178 024260 000427          BR      8$             ;BRANCH TO EXIT
6179 024262 000403          BR      4$             ;DATA TRANSFER UNDERWAY?
6180 024264 004737 024350          TST     DTUW          ;YES--GO START A SEARCH
6181 024270 000423          TST     SEEKFG        ;DO IMPLIED SEEKS?
6182 024272 004737 024456          BMI     3$             ;YES---BRANCH
6183 024276 000420          JSR     RO,LA         ;NO--DO LOOK AHEAD
6184 024300 112761 177777 023000  BR      8$             ;RETURN HERE ON A PARITY ERROR
6185 024306 010103          BR      4$             ;GO START A SEARCH
6186 024310 006303          JSR     PC,C11        ;START A DATA TRANSFER
6187 024312 012763 023420 023042  BR      8$             ;START A SEARCH
6188 024320 000402          BR      8$             ;GO TO THE EXIT
6189 024322 004737 025172          JSR     PC,C17        ;SET PORT REQUEST INDICATOR
6190 024326 032714 000100          MOV     R1,R3         ;SET UP TO ADDRESS WORDS
6191 024332 001002          ASL     R3             ;CONVERT TO WORD INDEX
6192 024334 004737 030364          MOV     #10000.,TIMER(R3) ;START 10 SEC TIMER
6193 024340 012637 177776          BR      7$             ;EXIT
6194 024344 104413          JSR     PC,C17        ;PROCESS THE PARITY ERROR
6195 024346 000207          BIT     #BIT06, (R4)  ;SEE IF 'IE' ALREADY SET
6196          BNE     8$             ;BR IF SET
6197          JSR     PC,SET.IE  ;SET "IE" WITHOUT A "TRE"
6198          MOV     (SP)+,2#PS ;RESTORE PROC. STATUS
6199          RESREG ;RESTORE RO - R5
          RTS     PC
;COMMAND INITIATOR
;CALL
    
```


E10

MAINDEC-11-DZRM1 AO/00, RM03 DRIVE COMPATIBILITY TEST MACY11 30(1046) 21-JUL-77 16:51 PAGE 121
 DZRM1A.P11 21-JUL-77 15:44 SINGLE/DUAL PORT RH70/RM03 DRIVER (REV 1.0)

SEQ 0120

6200									
6201					MOV	#DRVNUM, R1			: DRIVE NUMBER
6202					MOV	#DPB, R2			: ADDRESS OF DPB
6203					JSR	PC, CI?			: CI? = CI1, CI3, OR CI4
6204									: WHERE:
6205									: CI1=DATA TRANSFER
6206									: CI2=SEARCH REQUESTED BY DATA XFER
6207									: CI4=NOT DATA TRANSFER
6208	024350	004737	031124		CI1:	JSR	PC, POPQUE		: REMOVE REQUEST FROM "DRIVES WAIT" QUEUE
6209	024354	010237	023010			MOV	R2, TRANSW		: PUT REQ. IN TRANSFER WAIT QUEUE
6210	024360	010203				MOV	R2, R3		: DPB ADDRESS TO R3
6211	024362	013704	023076			MOV	RMADR, R4		: RMCS1 ADDRESS
6212	024366	010164	000010			MOV	R1, RMCS2(R4)		: SELECT DRIVE
6213	024372	062703	000004			ADD	#4, R3		: DESIRED WORD COUNT
6214	024376	062704	000002			ADD	#2, R4		: RMCS ADDRESS
6215	024402	012324				MOV	(R3)+, (R4)+		: LOAD WORD COUNT
6216	024404	012324				MOV	(R3)+, (R4)+		: LOAD BUFFER ADDRESS
6217	024406	012346				MOV	(R3)+, -(SP)		: LOAD SECTOR AND TRACK
6218	024410	004037	030054			JSR	RO, WRT.RM		: CALL THE LOAD(WRITE) ROUTINE
6219	024414	000006				RMDA			: INDEX OF REGISTER TO LOAD
6220	024416	025172				CI7			: ERROR RETURN ADDRESS
6221	024420	012346				MOV	(R3)+, -(SP)		: LOAD CYLINDER ADDRESS
6222	024422	004037	030054			JSR	RO, WRT.RM		
6223	024426	000034				RMDC			
6224	024430	025172				CI7			
6225	024432	016246	000002			MOV	2(R2), -(SP)		: LOAD "COMMAND+GO", "A17&A16", AND "PSEL"
6226	024436	004037	030054			JSR	RO, WRT.RM		
6227	024442	000000				RMCS1			
6228	024444	025172				CI7			
6229	024446	010137	023062			MOV	R1, DTUM		: SET "DATA TRANSFER UNDERWAY"
6230	024452	000137	025134			JMP	CI5		
6231	024456	013704	023076		CI3:	MOV	RMADR, R4		: RMCS1 ADDRESS
6232	024462	010164	000010			MOV	R1, RMCS2(R4)		: SELECT DRIVE
6233	024466	016246	000012			MOV	12(R2), -(SP)		: DESIRED CYLINDER ADDRESS
6234	024472	004037	030054			JSR	RO, WRT.RM		
6235	024476	000034				RMDC			
6236	024500	025172				CI7			
6237	024502	116203	000010			MOV	10(R2), R3		: PICKUP SECTOR ADDRESS
6238						SUB	MXWINDOW, R3		: BACKUP BY MAX. SEARCH FOR I/O WINDOW
6239						BGE	15		
6240						ADD	#32, R3		
6241	024506	010346			1s:	MOV	R3, -(SP)		: COMBINE THE ADJUSTED SECTOR WITH
6242	024510	042716	177740			BIC	#177740, (SP)		: CHOP OF ALL EXENTED BITS, IF ANY
6243	024514	116266	000011	000001		MOV	11(R2), 1(SP)		: THE DESIRED TRACK
6244	024522	004037	030054			JSR	RO, WRT.RM		: LOAD DESIRED TRACK & SECTOR
6245	024526	000006				RMDA			
6246	024530	025172				CI7			
6247	024532	012746	000131			MOV	#131, -(SP)		: START A SEARCH
6248	024536	004037	030054			JSR	RO, WRT.RM		
6249	024542	000000				RMCS1			
6250	024544	025172				CI7			
6251	024546	156137	023064	023012		BISB	ATABIT(R1), SRCHWT		: SET "SEARCH WAIT" KEY
6252	024554	000567				BR	CI5		
6253	024556	013704	023076		CI4:	MOV	RMADR, R4		: RMCS1 ADDRESS
6254	024562	010164	000010			MOV	R1, RMCS2(R4)		: SELECT DRIVE
6255	024566	116203	000002			MOV	2(R2), R3		: PICKUP THE REQUESTED COMMAND

F10

MAINFC-11-DZRM1 AD/00, RMO3 DRIVE COMPATIBILITY TEST MACY11 30(1046) 21-JUL-77 16:51 PAGE 122
 DZRM1A.P11 21-JUL-77 15:44 SINGLE/DUAL PORT RH70/RMO3 DRIVER (REV 1.0)

SEQ 0121

6256	024572	122703	000131		CMPB	#131,R3	; IS IT A SEARCH COMMAND?
6257	024576	001007			BNE	1\$; BRANCH IF NO
6258	024600	016246	000010		MOV	10(R2),-(SP)	; LOAD DESIRED TRACK & SECTOR
6259	024604	004037	030054		JSR	RD,WRT.RM	
6260	024610	000006			RMDA		
6261	024612	025172			CI7		
6262	024614	000403			BR	2\$; GO LOAD CYLINDER
6263	024616	122703	000105	1\$:	CMPB	#105,R3	; IS IT A SEEK COMMAND
6264	024622	001007			BNE	3\$; BRANCH IF NO
6265	024624	016246	000012	2\$:	MOV	12(R2),-(SP)	; LOAD DESIRED CYLINDER
6266	024630	004037	030054		JSR	RD,WRT.RM	
6267	024634	000034			RMDC		
6268	024636	025172			CI7		
6269	024640	000546			BR	CI6	
6270	024642	122703	000115	3\$:	CMPB	#115,R3	; IS IT AN "OFFSET" COMMAND?
6271	024646	001013			BNE	4\$; BR IF NO
6272	024650	004037	027674		JSR	RD,RD.RM	; MERGE THE OFFSET VALUE INTO RMOF
6273	024654	000032			RMOF		; BUT DON'T CHANGE THE UPPER
6274	024656	025172			CI7		
6275	024660	116216	000001		MOV	1(R2),(SP)	; BYTE WHEN LOADING THE
6276	024664	004037	030054		JSR	RD,WRT.RM	; REGISTER (RMOF)
6277	024670	000032			RMOF		
6278	024672	025172			CI7		
6279	024674	000530			BR	CI6	; GO START THE COMMAND
6280	024676	122703	000107	4\$:	CMPB	#107,R3	; IS IT A "RECALIBRATE" COMMAND?
6281	024702	001525			BEQ	CI6	; BRANCH IF YES
6282	024704	122703	000117		CMPB	#117,R3	; IS IT A RETURN TO CENTER?
6283	024710	001522			BEQ	CI6	; BRANCH IF YES
6284	024712	122703	000103		CMPB	#103,R3	; IS IT AN "UNLOAD" COMMAND?
6285	024716	001016			BNE	5\$; BRANCH IF NO
6286	024720	112761	000001	022740	MOV	#1,DRVACT(R1)	; SET THE DRIVE ACTIVE INDICATOR
6287	024726	105061	022750		CLRB	DRVSTA(R1)	; PUT DRIVE STATUS TO OFFLINE
6288	024732	112761	000001	023016	MOV	#1,ULDFLG(R1)	; SET "UNLOAD IN PROGRESS" FLAG
6289	024740	010346			MOV	R3,-(SP)	; START THE "UNLOAD" COMMAND
6290	024742	004037	030054		JSR	RD,WRT.RM	
6291	024746	000000			RMCS1		
6292	024750	025172			CI7		
6293	024752	000207			RTS	PC	; RETURN TO USER
6294	024754	122703	000143	5\$:	CMPB	#143,R3	; IS IT A "SET FORMAT" COMMAND?
6295	024760	001014			BNE	6\$; BRANCH IF NO
6296	024762	004037	027674		JSR	RD,RD.RM	; READ THE OFFSET REGISTER
6297	024766	000032			RMOF		
6298	024770	025172			CI7		
6299	024772	116266	000001	000001	MOV	1(R2),1(SP)	; COMBINE "FMT22" "ECI" AND "HCI"
6300	025000	004037	030054		JSR	RD,WRT.RM	; LOAD "FMT22", "ECI", AND/OR "HCI".
6301	025004	000032			RMOF		
6302	025006	025172			CI7		
6303	025010	000436			BR	12\$	
6304	025012	122703	000141	6\$:	CMPB	#141,R3	; IS IT A "GET REGISTER" COMMAND?
6305	025016	001023			BNE	10\$; BRANCH IF NO
6306	025020	016203	000006	7\$:	MOV	6(R2),R3	; POINTS TO 1ST ADDRESS OF WHERE
6307							; TO PUT THE REGISTER(S)
6308	025024	116237	000010	025042	MOV	10(R2),9\$; INIT. THE INDEX FOR THE FIRST REG.
6309	025032	116205	000011		MOV	11(R2),R5	; INDEX OF LAST REG. TO MOVE
6310	025036	004037	027674	8\$:	JSR	RD,RD.RM	; READ RH70/RMO3 REGISTER
6311	025042	000000		9\$:	RMCS1		; INDEX OF REG. TO READ

G10

MAINDEC-11-DZRM1 AO/00, RMO3 DRIVE COMPATIBILITY TEST MACY11 30(1046) 21-JUL-77 16:51 PAGE 123
 DZRM1A.P11 21-JUL-77 15:44 SINGLE/DUAL PORT RH70/RMO3 DRIVER (REV 1.0)

SEQ 0122

6312	025044	025172				CI7		
6313	025046	012623				MOV	(SP)+, (R3)+	; GET THE CONTENTS OF RH70//RMO3 REG.
6314	025050	023705	025042			CMP	95, R5	; LAST REG. BEEN READ?
6315	025054	001414				BEQ	125	; GET OUT IF YES
6316	025056	062737	000002	025042		ADD	#2, 95	; INCREASE THE INDEX BY 2
6317	025064	000764				BR	85	; LOOP--MORE TO READ
6318	025066	122703	000145		105:	CMPB	#145, R3	; IS IT A "SELECT DRIVE" COMMAND?
6319	025072	001405				BEQ	125	; BRANCH IF YES
6320	025074	010346			115:	MOV	R3, -(SP)	; LOAD THE COMMAND
6321	025076	004037	030054			JSR	RO, WRT.RM	
6322	025102	000000				RMCS1		
6323	025104	025172				CI7		
6324	025106	004737	031124		125:	JSR	PC, POPQUE	; REMOVE REQ. FROM QUEUE
6325	025112	052762	000200	000016		BIS	#BIT07, 16(R2)	; SET THE "DONE" BIT
6326	025120	005737	023036			TST	SAVEFG	; SAVE THE RH70/RMO3 REGISTERS?
6327	025124	100002				BPL	135	; BRANCH IF NO
6328	025126	004737	030246			JSR	PC, SVRH70	; YES--GO SAVE THE REGISTERS
6329	025132	000207			135:	RTS	PC	; RETURN TO USER
6330	025134	006301			CI5:	ASL	R1	
6331	025136	012761	001750	023042		MOV	#1000., TIMER(R1)	; SET A ONE SECOND TIMER
6332	025144	006201				ASR	R1	
6333	025146	112761	000001	022740		MOVB	#1, DRVACT(R1)	; SET THE DRIVE ACTIVE
6334	025154	000207				RTS	PC	; RETURN TO THE USER
6335	025156	010346			CI6:	MOV	R3, -(SP)	; LOAD THE COMMAND
6336	025160	004037	030054			JSR	RO, WRT.RM	
6337	025164	000000				RMCS1		
6338	025166	025172				CI7		
6339	025170	000761				BR	CI5	
6340	025172	032764	010000	000010	CI7:	BIT	#BIT12, RMCS2(R4)	; DRIVE NON-EXISTENT ?
6341	025200	001034				BNE	CI8	; BR IF YES
6342	025202	005702			15:	TST	R2	; ANYTHING IN QUEUE ?
6343	025204	001405				BEQ	CI7B	; BR IF NOT
6344	025206	012762	104000	000016		MOV	#BIT15!BIT11, 16(R2)	; SET "PARITY" ERROR INDICATOR
6345	025214	004737	030246			JSR	PC, SVRH70	; GO SAVE THE RH70/RMO3 REGISTERS
6346	025220	012746	000111		CI7B:	MOV	#111, -(SP)	; DO A "DRIVE CLEAR"
6347	025224	004037	030054			JSR	RO, WRT.RM	
6348	025230	000000				RMCS1		
6349	025232	025272				CI8		
6350	025234	004737	031006			JSR	PC, EMPTYQ	; EMPTY THE QUEUE
6351	025240	105061	023016			CLRB	ULDFLG(R1)	; CLEAR THE UNLOAD IN QUEUE FLAG
6352	025244	105061	022740			CLRB	DRVACT(R1)	; DRIVE IS IDLE
6353	025250	020137	023062			CMP	R1, DTUW	; IF THIS DRIVE HAD AN I/O REQUEST
6354	025254	001005				BNE	15	; IN PROGRESS CLEAR ALL OF THE FLAGS
6355	025256	005037	023010			CLR	TRNSWT	
6356	025262	012737	177777	023062		MOV	#-1, DTUW	
6357	025270	000207			15:	RTS	PC	
6358	025272	104412			CI8:	SAVREG		; SAVE RO - R5
6359	025274	032764	010000	000010		BIT	#BIT12, RMCS2(R4)	; IS 'NED' SET ?
6360	025302	001002				BNE	15	; BR IF YES
6361	025304	005001				CLR	R1	
6362	025306	005003				CLR	R3	
6363	025310	105761	022740		15:	TSTB	DRVACT(R1)	; DRIVE ACTIVE?
6364	025314	001443				BEQ	55	; BRANCH IF NO
6365	025316	013702	023010			MOV	TRNSWT, R2	; GET THE "TRANSFER WAIT" QUEUE
6366	025322	020137	023062			CMP	R1, DTUW	; DID THIS DRIVE HAVE AN I/O IN PROGRESS?
6367	025326	001402				BEQ	25	; BRANCH IF YES

H10

```

6368 025330 004737 031102          JSR    PC,GETREQ      ;GET THE DPB POINTER
6369 025334 005702          2$:   IST    R2        ;QUEUE ENTRY FOR DRIVE ?
6370 025336 001415          BEQ    4$            ;BR IF NOT
6371 025340 032764 010000 000010    BIT    #BIT12,RMCS2(R4) ;'NED' SET ?
6372 025346 001404          BEQ    3$            ;BR IF NOT
6373 025350 012762 100002 000016    MOV    #BIT15!BIT01,16(R2) ;SET 'DRIVE NON-EXISTENT' INDICATOR
6374 025356 000405          BR     4$            ;CONTINUE
6375 025360 012762 102000 000016    3$:   MOV    #BIT15!BIT10,16(R2) ;SET "NON-CLEARABLE PARITY" ERROR INDICATOR
6376 025366 004737 030246          JSR    PC,SVRH70     ;SAVE RH70/RMD3 REGISTERS
6377 025372 012763 177777 023042    4$:   MOV    #-1,TIMER(R3) ;STOP THE TIMER
6378 025400 105061 022740          CLRB  DRVACT(R1)    ;SET "DRIVE ACTIVE" TO IDLE
6379 025404 020137 023062          CMP    R1,DTUM      ;IS THIS DRIVE SETUP FOR A TRANSFER
6380 025410 001005          BNE   5$            ;BR IF NOT
6381 025412 012737 177777 023062    MOV    #-1,DTUM     ;RESET THE INDICATOR
6382 025420 005037 023010          CLR  TRNSWT        ;CLEAR THE TRANSFER QUEUE
6383 025424 105061 023016          5$:   CLRB  ULDFLG(R1)   ;CLEAR UNLOAD FLAG
6384 025430 032764 010000 000010    BIT    #BIT12,RMCS2(R4) ;'NED' SET ?
6385 025436 001021          BNE   6$            ;BR IF YES
6386 025440 005201          INC    R1           ;MOVE TO THE NEXT DRIVE
6387 025442 062703 000002          ADD    #2,R3
6388 025446 042701 177770          BIC    #1<7,R1
6389 025452 001316          BNE   1$            ;BRANCH IF MORE DRIVES
6390 025454 012737 177777 023062    MOV    #-1,DTUM     ;NO DATA TRANSFERS UNDERWAY
6391 025462 005037 023010          CLR  TRNSWT        ;CLEAR THE 'TRANSFER WAIT' QUEUE
6392 025466 004737 030730          JSR    PC,CLRQUE    ;CLEAR ALL OF THE REQUEST QUEUES
6393 025472 012764 000040 000010    MOV    #BIT05,RMCS2(R4) ;DO A MASSBUS INIT.
6394 025500 000406          BR     7$            ;CONTINUE
6395 025502 004737 031006          6$:   JSR    PC,EMPTYQ   ;CLEAR THE DRIVE'S QUEUE
6396 025506 105061 022750          CLRB  DRVSTA(R1)   ;SET DRIVE TO OFFLINE
6397 025512 105061 022760          CLRB  DRVTYP(R1)  ;CLEAR THE DRIVE TYPE INDICATOR
6398 025516 004737 030364          7$:   JSR    PC,SET.IE   ;SET "IE" WITHOUT "TRE"
6399 025522 104413          RESREG ;RESTORE R0 - R5
6400 025524 000207          RTS    PC           ;RETURN
6401
6402          ;LOOK AHEAD ROUTINE
6403
6404          ;CALL
6405          MOV    #DRVNUM,R1 ;DRIVE NUMBER
6406          MOV    #DPB,R2    ;POINT TO DPB
6407          JSR    RO,LA     ;GO CHECK THE WINDOW
6408          RETURN1 ;ERROR RETURN
6409          RETURN2 ;START A SEARCH
6410          RETURN3 ;START A DATA TRANSFER
6411
6412 025526 013704 023076          LA:   MOV    RMADR,R4  ;GET RMCS1'S ADDRESS
6413 025532 010164 000010          MOV    R1,RMCS2(R4) ;SELECT DRIVE
6414 025536 004037 027674          JSR    RO,RO.RM     ;READ DRIVE STATUS
6415 025542 000012          RMD5
6416 025544 025674          4$   ;ERROR RETURN ADDRESS
6417 025546 042716 157577          BIC    #1<020200,(SP) ;ON CYLINDER ?
6418 025552 022726 000200          CMP    #200,(SP)+  ;PIP=0,DRY=1?
6419 025556 001044          BNE   3$            ;NO
6420 025560 105261 023026          INCB  LACNT(R1)    ;INCREMENT THE LOOK AHEAD COUNT
6421 025564 126137 023026 023104    CMPB  LACNT(R1),MXLACT ;EXCEED MAX?
6422 025572 003033          BGT   2$            ;BRANCH IF YES
6423 025574 116203 000010          MOVB  10(R2),R3    ;GET DESIRED SECTOR ADDRESS AND
  
```



```

6424 025600 000303          SWAB      R3          ;MULT. BY 64--ALIGN WITH
6425 025602 006203          ASR      R3          ;LOOK AHEAD REGISTER
6426 025604 006203          ASR      R3
6427 025606 012737 000340 177776 6S:  MOV      #340,2#PS  ;PRIORITY LEVEL "7"
6428 025614 004037 027674          JSR      RO,RO.RM  ;READ LOOK AHEAD REGISTER
6429 025620 000020          RMLA
6430 025622 025674          4S
6431 025624 021664 000020          CMP      (SP),RMLA(R4) ;CORRECT LA NUMBER ?
6432 025630 001402          BEQ      7S          ;YES
6433 025632 005726          TST      (SP)+       ;NO,CLEAR STACK
6434 025634 000415          BR       3S
6435 025636 162603          7S:  SUB      (SP)+,R3  ;CALCULATE THE DELTA
6436 025640 002002          BGE      1S
6437 025642 062703 004000          ADD      #(<32.#64.>),R3 ;MAKE THE DELTA POSITIVE
6438 025646 023703 023106 1S:  CMP      MXDLTA,R3  ;CHECK THE DELTA TO SEE
6439 025652 002406          BLT      3S          ;IF IT IS WITHIN THE
6440 025654 023703 023110          CMP      MNDLTA,R3  ;WINDOW---IF YES, ZERO
6441 025660 002003          BGE      3S          ;THE LOOK AHEAD COUNT
6442 025662 105061 023026 2S:  CLRB    LACNT(R1)  ;AND TAKE THE I/O EXIT
6443 025666 005720          TST      (RO)+
6444 025670 005720 3S:  TST      (RO)+       ;ADJUST THE RETURN ADDRESS
6445 025672 000402          BR       5S          ;EXIT
6446 025674 004737 025172 4S:  JSR      PC,C17    ;PROCESS THE ERROR
6447 025700 000200 5S:  RTS      RO        ;RETURN
6448
6449          ;INTERRUPT SERVICE ROUTINE
6450
6451 025702 112737 000001 023014 ISR:  MOVVB   #1,ACTDRV  ;SET "ACTIVE DRIVER" FLAG
6452 025710 104412          SAVREG  ;SAVE RO - R5
6453 025712 013704 023076          MOV     RMADR,R4  ;ADDRESS OF RHSCSI
6454 025716 013701 023062          MOV     DTUW,R1  ;GET "DATA TRANSFER UNDERWAY" INDICATOR
6455 025722 002403          BLT     1S        ;BRANCH IF NO DATA TRANSFER UNDERWAY
6456 025724 004737 025746          JSR     PC,TD     ;CALL TRANSFER DONE
6457 025730 000402          BR     2S        ;EXIT
6458 025732 004737 026116 1S:  JSR     PC,SC     ;CALL SPECIAL CONDITIONS
6459 025736 104413 2S:  RESREG  ;RESTORE RO - R5
6460 025740 105037 023014          CLRB   ACTDRV   ;CLEAR "ACTIVE DRIVER" FLAG
6461 025744 000002          RTI          ;RETURN
6462
6463          ;TRANSFER DONE ROUTINE
6464
6465 025746 105061 022740 023062 TD:  CLRB   DRVACT(R1) ;SET DRIVE ACTIVE INDICATOR TO IDLE
6466 025752 012737 177777          MOV     #-1,DTUW ;NO DATA TRANSFERS UNDERWAY
6467 025760 006301          ASL     R1
6468 025762 012761 177777 023042          MOV     #-1,TIMER(R1) ;CANCEL TIMEOUT
6469 025770 006201          ASR     R1
6470 025772 013702 023010          MOV     TRNSWT,R2 ;GET "DPB" ADDRESS FROM THE
6471 025776 005037 023010          CLR     TRNSWT   ;TRANSFER WAIT QUEUE--CLEAR QUEUE
6472 026002 052762 000200 000016          BIS     #BIT07,16(R2) ;SET DONE
6473 026010 010164 000010          MOV     R1,RMC52(R4) ;SELECT THE DRIVE
6474 026014 004037 027674          JSR     RO,RO.RM  ;TRANSFER ERROR(TRE=1)?
6475 026020 000000          RMCS1
6476 026022 025172          C17
6477 026024 006126          ROL     (SP)+
6478 026026 100417          BMI     3S
6479 026030 005737 023036          TST     SAVEFG   ;BR IF YES
                   ;SAVE THE RH70/RMO3 REGISTERS?

```

```

6480 026034 100002          BPL      1$          ;BRANCH IF NO
6481 026036 004737 030246    JSR      PC,SVRH70  ;YES--SAVE THE REGISTERS
6482 026042          1$:          ;
6483 026042 004737 031102    JSR      PC,GETREQ  ;GET DPB POINTER
6484 026046 005702          TST      R2          ;ENTRY FOR DRIVE ?
6485 026050 001403          BEQ      2$          ;BR IF NOT
6486 026052 004737 024112    JSR      PC,OPT     ;CALL OPTIMIZER
6487 026056 000417          BR       SC         ;CHECK OTHER DRIVES
6488          ;THE RELEASE DRIVE COMMAND IS FORECD TO ENTER FOR DUAL PORT OPERATION
6489 026060 012714 000113    2$:      MOV      #113,(R4) ;RELEASE THE DRIVE
6490 026064 000414          BR       SC         ;CHECK FOR OTHER DRIVES
6491 026066 052762 100100 000016 3$:      BIS      #BIT15!BIT06,16(R2) ;SET DATA ERROR FLAG
6492 026074 004737 031006    JSR      PC,EMPTYQ  ;EMPTY THE "DRIVE'S WAIT" QUEUE
6493 026100 004737 030246    JSR      PC,SVRH70  ;SAVE THE RH70/RMO3 REGISTERS
6494 026104 012714 040111    MOV      #40111,(R4) ;ISSUE A "DRIVE CLEAR"
6495 026110 012714 000113    MOV      #113,(R4)  ;ISSUE A RELEASE TO THE DRIVE
6496 026114 000400          BR       SC         ;CHECK FOR OTHER DRIVES
6497
6498
6499
6500          ;SPECIAL CONDITION ROUTINE
6501
6502 026116 116403 000016    SC:      MOVB     RMA5(R4),R3 ;READ "RMA5"
6503 026122 001014          BNE      2$          ;BRANCH IF ANY 'ATA' BITS SET
6504 026124 004037 027674    JSR      RD,RD.RM   ;READ CONTROL AND STATUS REGISTER
6505 026130 000000          RMCS1
6506 026132 025272          CIB
6507 026134 106126          ROLB     (SP)+      ;IS "IE"=1?
6508 026136 100405          BMI     1$          ;YES, NO DRIVES TO CHECK
6509 026140 004037 031172    JSR      RD,ES.SAV  ;SAVE THE ADDRESS IN 'ESCAPE'
6510 026144 104001          ERROR    1          ;REPORT AN ILLEGAL INTERRUPT
6511 026146 004737 030364    JSR      PC,SET.IE  ;SET INTERRUPT ENABLE
6512 026152 000207          1$:      RTS      PC          ;RETURN
6513 026154 005046          2$:      CLR      -(SP)      ;PROCESS ALL DRIVES THAT HAVE
6514 026156 110316          MOVB     R3,(SP)    ;AN "ATA"=1
6515 026160 012703 000001    MOV      #1,R3
6516 026164 005001          CLR      R1
6517 026166 030316          SC3:     BIT      R3,(SP) ;ATA=1?
6518 026170 001005          BNE      SC5        ;YES--BRANCH
6519 026172 005201          SC4:     INC      R1     ;MOVE TO THE NEXT DRIVE
6520 026174 106303          ASLB     R3
6521 026176 001373          BNE      SC3        ;BRANCH IF MORE TO CHECK?
6522 026200 005726          TST      (SP)+      ;CLEAN OFF THE STACK
6523 026202 000207          RTS      PC          ;RETURN TO USER
6524 026204 105761 022770          SC5:     TSTB     DPINT(R1) ;INITIALIZING THE DRIVE ?
6525 026210 001402          BEQ      1$          ;BR IF NOT
6526 026212 000137 027130          JMP      SC13       ;PROCESS THE DRIVE
6527 026216 105761 023000          1$:      TSTB     DPRQS(R1) ;PORT REQUEST OUTSTANDING ?
6528 026222 001402          BEQ      2$          ;BR IF NOT
6529 026224 000137 027130          JMP      SC13       ;START THE OUTSTANDING COMMAND
6530 026230 105761 022750          2$:      TSTB     DRVSTA(R1) ;CHECK THE DRIVE STATUS
6531 026234 003025          BGT      5$          ;BRANCH IF ONLINE
6532 026236 105761 023016          TSTB     ULDFLG(R1) ;UNLOAD IN PROGRESS?
6533 026242 003422          BLE      5$          ;BRANCH IF NOT
6534 026244 004737 031102    JSR      PC,GETREQ  ;GET DPB POINTER
6535 026250 004737 030246    JSR      PC,SVRH70  ;SAVE THE RH70/RMO3 REGISTERS
    
```


K10

MAINDEC-11-DZRM1 AD/DO, RMD3 DRIVE COMPATIBILITY TEST MACY11 30(1046) 21-JUL-77 16:51 PAGE 127
 DZRMIA.P11 21-JUL-77 15:44 SINGLE/DUAL PORT RH70/RMD3 DRIVER (REV 1.0)

SEQ 0126

6536	026254	004737	027060		JSR	PC,SC12	;SAVE RMD5, RMER1, RMER2, AND RMMR2
6537							;ALSO DO A DRIVE INIT (DRVINT)
6538	026260	105761	022750		TSTB	DRVSTA(R1)	;DID DRIVE COME ONLINE?
6539	026264	003416			BLE	6\$;NO---BRANCH
6540	026266	032737	040000	022730	BIT	#BIT14,RMERRS	;WAS THERE AN ERROR?
6541	026274	001002			BNE	3\$;BR IF ERROR
6542	026276	000137	026750		JMP	SC11	;NO ERROR
6543	026302	013705	022732	3\$:	MOV	RMERRS+2,R5	;YES -- PICKUP RMER1 AND
6544	026306	000502			BR	SC6A	;GO PROCESS THE ERROR
6545	026310	105761	022740	5\$:	TSTB	DRVACT(R1)	;DRIVE ACTIVE WITH COMMAND OR ERROR RECOVERY ?
6546	026314	001033			BNE	SC6	;BR IF EITHER
6547	026316	004737	027060		JSR	PC,SC12	;SAVE RMD5, RMER1, RMER2, AND RMMR2
6548							;ALSO DO A DRVINT
6549	026322	105761	022770	6\$:	TSTB	DPINT(R1)	;TRYING TO INIT THE DRIVE ?
6550	026326	001321			BNE	SC4	;BR IF YES, CHECK ON MORE DRIVES
6551	026330	105761	022750		TSTB	DRVSTA(R1)	;CHECK ON DRIVE'S STATUS
6552	026334	100412			BMI	7\$;BR IF UNSAFE
6553	026336	032737	020000	022734	BIT	#BIT13,RMERRS+4	;ADDRESS PLUG CHANGED ?
6554	026344	001013			BNE	8\$;BR IF YES
6555	026346	012746	000113		MOV	#113,-(SP)	;RELEASE COMMAND
6556	026352	004037	030054		JSR	RO,WRT.RM	;WRITE THE COMMAND INTO RMCS1
6557	026356	000000			RMCS1		;REGISTER INDEX
6558	026360	026720			SC8		;PARITY EXIT ADDRESS
6559	026362	011605		7\$:	MOV	(SP),R5	;PICKUP (RMAS) BEFORE THE ERROR CALL
6560	026364	004037	031172		JSR	RO,ES.SAV	;SAVE THE ADDRESS IN 'SESCAPE'
6561	026370	104002			ERROR	2	;REPORT THE UNEXPECTED ATTENTION
6562	026372	000677			BR	SC4	;GO CHECK FOR MORE ATA'S
6563	026374			8\$:			
6564	026374	004037	031172		JSR	RO,ES.SAV	;SAVE THE ADDRESS IN 'SESCAPE'
6565	026400	104005			ERROR	5	;REPORT THE ADDRESS PLUG CHANGE
6566	026402	000673			BR	SC4	;CHECK FOR MORE DRIVES
6567	026404	006301		SC6:	ASL	R1	;SETUP TO ADDRESS WORDS
6568	026406	012761	177777	023042	MOV	#-1,TIMER(R1)	;STOP THE TIMER
6569	026414	006201			ASR	R1	;RESTORE THE DRIVE ADDRESS
6570	026416	004737	031102		JSR	PC,GETREQ	;GET THE DPB POINTER FROM THE QUEUE
6571	026422	010164	000010		MOV	R1,RMCS2(R4)	;SELECT DRIVE
6572	026426	004037	027674		JSR	RO,RO.RM	;READ THE RMD3'S STATUS REG.
6573	026432	000012			RMD5		
6574	026434	026720			SC8		
6575	026436	011605			MOV	(SP),R5	;AND PUT IT IN R5
6576	026440	006126			ROL	(SP)+	;WAS THERE AN ERROR?
6577	026442	100407			BMI	1\$;BR IF ERROR
6578	026444	105761	022740		TSTB	DRVACT(R1)	;CHECK DRIVE'S STATE
6579	026450	003137			BGT	SC11	;BR IF DRIVE ACTIVE WITH ORDER
6580	026452	052762	100210	000016	BIS	#BIT15!BIT07!BIT03,16(R2)	;INFORM USER OF ERROR RECOVER COMPLETION
6581	026460	000470			BR	SC7	
6582	026462	004037	027674	1\$:	JSR	RO,RO.RM	;READ ERROR REGISTER #1
6583	026466	000014			RMER1		
6584	026470	026720			SC8		
6585	026472	012605			MOV	(SP)+,R5	;AND SAVE IT IN R5
6586	026474	004737	030246		JSR	PC,SVRH70	;SAVE RH70/RMD3 REGISTERS
6587	026500	012746	000111		MOV	#111,-(SP)	;ISSUE A DRIVE CLEAR
6588	026504	004037	030054		JSR	RO,WRT.RM	
6589	026510	000000			RMCS1		
6590	026512	026720			SC8		
6591	026514	006105		SC6A:	ROL	R5	;WAS "UNSAFE" CONDITION =1?

6592	026516	100406			BMI	1\$;BRANCH IF YES
6593	026520	005702			TST	R2	;ANYTHING IN QUEUE ?
6594	026522	001447			BEQ	SC7	;BR IF NOT
6595	026524	052762	100240	000016	BIS	#BIT15!BIT07!BIT05,16(R2)	;INFORM USER OF ERROR
6596	026532	000443			BR	SC7	
6597	026534	004037	027674		1\$: JSR	RD,RD.RM	;READ DRIVE STATUS REG. #1
6598	026540	000012			RMD5		
6599	026542	026720			SC8		
6600	026544	011605			MOV	(SP),R5	;SAVE RMD5 IN R5
6601	026546	006126			ROL	(SP)↓	; "ERR"=1?
6602	026550	100011			BPL	2\$;BR IF NO--UNSAFE CLEARED
6603	026552	112761	177777	022750	MOVB	#-1,DRVSTA(R1)	;DRIVE IS UNSAFE
6604	026560	004737	030246		JSR	PC,SVRH70	;SAVE RH70/RMO3 REGISTERS
6605	026564	052762	110000	000016	BIS	#BIT15!BIT12,16(R2)	;INFORM USER OF UNSAFE ERROR
6606	026572	000423			BR	SC7	
6607	026574	032705	010000		2\$: BIT	#BIT12,R5	; "MOL" = 1 ?
6608	026600	001015			BNE	3\$;BR IF YES
6609	026602	112761	177777	022740	MOVB	#-1,DRVACT(R1)	;ACTIVE ERROR RECOVER
6610	026610	112761	000001	022750	MOVB	#1,DRVSTA(R1)	;ONLINE
6611	026616	006301			ASL	R1	
6612	026620	012761	072460	023042	MOV	#30000.,TIMER(R1)	;START 30 SECOND TIMER
6613	026626	006201			ASR	R1	
6614	026630	000137	026172		JMP	SC4	
6615	026634	052762	100220	000016	3\$: BIS	#BIT15!BIT07!BIT04,16(R2)	;INFORM USER OF ERROR
6616	026642	105061	022740		SC7: CLRB	DRVACT(R1)	;DRIVE IS IDLE
6617	026646	004737	031006		JSR	PC,EMPTYQ	;DUMP THE QUEUE
6618	026652	105761	023016		TSTB	ULDFLG(R1)	;UNLOAD IN PROGRESS OR QUEUE?
6619	026656	003002			BGT	1\$;BR IF NOT
6620	026660	105061	023016		CLRB	ULDFLG(R1)	;CLEAR UNLOAD FLAG
6621	026664	116164	023064	000016	1\$: MOVB	ATABIT(R1),RMA5(R4)	;CLEAR ATTENTION BIT
6622	026672	105761	022750		TSTB	DRVSTA(R1)	;IS THE DRIVE UNSAFE ?
6623	026676	100406			BMI	2\$;BR IF IT IS
6624	026700	012746	000113		MOV	#113,-(SP)	;RELEASE COMMAND
6625	026704	004037	030054		JSR	RD,WRT.RM	;WRITE THE COMMAND INTO RPCS1
6626	026710	000000			RMCS1		;REGISTER INDEX
6627	026712	026720			SC8		;PARITY EXIT ADDRESS
6628	026714	000137	026172		2\$: JMP	SC4	;CHECK FOR MORE DRIVES
6629	026720	105761	022740		SC8: TSTB	DRVACT(R1)	;IS DRIVE IDLE?
6630	026724	001405			BEQ	1\$;YES--BRANCH
6631	026726	004737	031102		JSR	PC,GETREQ	;GET DPB POINTER
6632	026732	004737	025172		JSR	PC,CI7	;PROCESS THE PARITY ERROR
6633	026736	000402			BR	2\$;CONTINUE
6634	026740	004737	025220		1\$: JSR	PC,CI7B	;PROCESS THE UNCORRECTABLE PARITY ERROR
6635	026744	000137	026172		2\$: JMP	SC4	;CHECK MORE DRIVES
6636	026750	105761	023016		SC11: TSTB	ULDFLG(R1)	; "UNLOAD IN PROGRESS" ?
6637	026754	003402			BLE	1\$;BRANCH IF NO
6638	026756	105061	023016		CLRB	ULDFLG(R1)	;CLEAR UNLOAD FLAG
6639	026762	105061	022740		1\$: CLRB	DRVACT(R1)	;SET DRIVE IDLE
6640	026766	136137	023064	023012	BITB	ATABIT(R1),SRCHWT	;DOING A SEARCH OPERATION FOR ;AN I/O COMMAND?
6641							
6642	026774	001012			BNE	2\$;BRANCH IF YES
6643	026776	004737	031124		JSR	PC,POPQUE	;REMOVE REQUEST FROM QUEUE
6644	027002	052762	000200	000016	BIS	#BIT07,16(R2)	;SET "DONE" BIT
6645	027010	005737	023036		TST	SAVEFG	;SAVE THE REGISTERS?
6646	027014	100002			BPL	2\$;BRANCH IF NO
6647	027016	004737	030246		JSR	PC,SVRH70	;YES--SAVE ALL OF THE RH70/RMO3 REG'S

M10

MAINDEC-11-DZRM1 AO/00, RMO3 DRIVE COMPATIBILITY TEST MACY11 30(1046) 21-JUL-77 16:51 PAGE 129
 DZRMIA.P11 21-JUL-77 15:44 SINGLE/DUAL PORT RM70/RMO3 DRIVER (REV 1.0)

SEQ 0128

```

6648 027022 116164 023064 000016 2S:  MOVB  ATABIT(R1),RMAS(R4) ;CLEAR ATTENTION BIT
6649 027030 146137 023064 023012      BICB  ATABIT(R1),SRCHMT ;CLEAR IMPLIED SEEK SET
6650 027036 006301      ASL   R1 ;WORD INDEX
6651 027040 012761 177777 023042      MOV   #-1,TIMER(R1) ;STOP CLOCK
6652 027046 006201      ASR   R1 ;RESTORE R1
6653 027050 004737 024112      JSR   PC,OPT ;START A REQUEST
6654 027054 000137 026172      JMP   SC4 ;CHECK FOR MORE DRIVES
6655 027060 010164 000010      SC12: MOV   R1,RMCS2(R4) ;SELECT DRIVE
6656 027064 016437 000012 022730      MOV   RMD5(R4),RMERRS ;SAVE THE FOUR REGISTERS THAT
6657 027072 016437 000014 022732      MOV   RMER1(R4),RMERRS+2 ;WILL TELL US SOMETHING
6658 027100 016437 000042 022734      MOV   RMER2(R4),RMERRS+4
6659 027106 016437 000040 022736      MOV   RMER3(R4),RMERRS+6
6660 027114 004037 023326      JSR   RD,DRVINT ;INIT. THE STATE OF THE DRIVE
6661 027120 000401      BR    1$ ;TAKE ERROR EXIT
6662 027122 000207      RTS   PC ;RETURN
6663 027124 005726      1$:  TST  (SP)+ ;POP PC OFF OF THE STACK
6664 027126 000674      BR    SC8 ;PROCESS THE PARITY ERROR
6665 027130 006301      SC13: ASL   R1 ;SETUP TO ADDRESS WORDS
6666 027132 012761 177777 023042      MOV   #-1,TIMER(R1) ;STOP THE TIMER
6667 027140 006201      ASR   R1
6668 027142 010164 000010      MOV   R1,RMCS2(R4) ;SELECT THE DRIVE
6669 027146 116164 023064 000016      MOVB  ATABIT(R1),RMAS(R4) ;CLEAR THE ATTENTION BIT
6670 027154 105761 022770      1$:  TSTB DPINT(R1) ;INITIALIZING THE DRIVE ?
6671 027160 001424      BEQ   2$ ;BR IF NOT
6672 027162 105061 022770      CLRB  DPINT(R1) ;CLEAR THE INIT INDICATOR
6673 027166 004037 023326      JSR   RD,DRVINT ;GO INIT THE DRIVE
6674 027172 000240      NOP   ;DUMMY PARITY ERROR RETURN
6675 027174 105761 022750      TSTB  DRVSTA(R1) ;DRIVE ONLINE ?
6676 027200 003014      BGT   2$ ;BR IF YES -- START ORDER
6677 027202 005702      TST  R2 ;QUEUE ENTRY FOR THE DRIVE
6678 027204 001426      BEQ   3$ ;BR IF NOT
6679 027206 004737 031102      JSR   PC,GETREQ ;GET DPB ADDRESS
6680 027212 052762 140000 000016      BIS   #BIT15:BIT14,16(R2) ;INFORM USER THAT DRIVE OFFLINE
6681 027220 004737 030246      JSR   PC,SVRH70 ;SAVE THE REGISTERS
6682 027224 004737 031006      JSR   PC,EMPTYQ ;EMPTY THE REQUEST QUEUE
6683 027230 000414      BR    3$
6684 027232 032764 004000 000000 2$:  BIT   #BIT11,RMCS1(R4) ;DVA SET ?
6685 027240 001006      BNE   4$ ;SET THEN CALL OPT
6686 027242 006301      ASL   R1
6687 027244 012761 023420 023042      MOV   #10000.,TIMER(R1)
6688 027252 006201      ASR   R1
6689 027254 000402      BR    3$
6690 027256 004737 024112      4$:  JSR   PC,OPT ;START THE PENDING REQUEST
6691 027262 000137 026172      3$:  JMP   SC4 ;PROCESS OTHER DRIVES
6692
6693 ;/RMO3 TIMER ROUTINE
6694 ;CALL
6695 ;
6696 ;
6697 ;
6698 027266 005737 023014      RPTMR: TST   ACTDRV ;CHECK "ACTDRV & ACTSTR"
6699 027272 001027      BNE   4$ ;IF NON ZERO EXIT
6700 027274 112737 000001 023015      MOVB  #1,ACTSTR ;SET "ACTSTR"
6701 027302 104412      SAVREG ;SAVE R0 - R5
6702 027304 005001      CLR   R1 ;START WITH DRIVE 0
6703 027306 005003      CLR   R3
    
```

N10

MAINDEC-11-DZRM1 AO/00, RMO3 DRIVE COMPATIBILITY TEST MACY11 30(1046) 21-JUL-77 16:51 PAGE 130
 DZRM1A.P11 21-JUL-77 15:44 SINGLE/DUAL PORT RH70/RMO3 DRIVER (REV 1.0)

SEQ 0129

```

6704 027310 005763 023042      1$:   TST   TIMER(R3)      ; IS THE TIMER RUNNING?
6705 027314 002406                BLT   2$                ; BRANCH IF NO
6706 027316 166663 000002 023042      SUB   2(SP),TIMER(R3)  ; COUNT THE INTERVAL
6707 027324 003002                BGT   2$                ; BR IF NO SOFTWARE TIMEOUT
6708 027326 004737 027356      JSR   PC,STO          ; CALL SOFTWARE TIMEOUT ROUTINE
6709 027332 005201                2$:   INC   R1           ; MOVE TO NEXT DRIVE
6710 027334 005723                TST   (R3)+
6711 027336 022701 000010      CMP   #8.,R1          ; OUT OF DRIVES?
6712 027342 003362                BGT   1$              ; BRANCH IF NO
6713 027344 104413                RESREG                ; RESTORE R0 - R5
6714 027346 105037 023015      CLRB  ACTSTR         ; ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
6715 027352 012616                4$:   MOV   (SP)+,(SP)  ; ADJUST THE STACK
6716 027354 000207                RTS   PC              ; RETURN

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

6728 027356 010146      STO:  MOV   R1,-(SP)    ; SAVE R1
6729 027360 010246      MOV   R2,-(SP)    ; SAVE R2
6730 027362 010346      MOV   R3,-(SP)    ; SAVE R3
6731 027364 010446      MOV   R4,-(SP)    ; SAVE R4
6732 027366 013704 023076      MOV   RMADR,R4    ; GET ADDRESS OF "RMCS1"
6733 027372 010164 000010      MOV   R1,RMCS2(R4) ; SELECT THE DRIVE
6734 027376 004037 027674      JSR   RD,RD.RM    ; READ "DRIVE STATUS REG"
6735 027402 000012      RMOVS
6736 027404 027556      STOS
6737 027406 105726      TSTB  (SP)+
6738 027410 100434                BMI  ST02          ; IS "DRY"=1?
6739 027412 105761 022770      ST01: TSTB  DPINT(R1)  ; BR IF YES
6740 027416 001031                BNE  ST02          ; TRYING TO INITIALIZE THE DRIVE ?
6741 027420 105761 023000      TSTB  DPRQS(R1)   ; BR IF YES
6742 027424 001026                BNE  ST02          ; OUTSTANDING PORT REQUEST FOR THE DRIVE ?
6743 027426 013702 023010      MOV   TRANSW,R2   ; BR IF YES
6744 027432 020137 023062      CMP   R1,DTUM     ; PICKUP TRANSFER WAIT QUEUE
6745 027436 001404                BEQ  1$            ; TRANSFER UNDERWAY ON THIS DRIVE?
6746 027440 000137 027662      JMP   ST09        ; BRANCH IF YES
6747 027444 004737 031102      JSR   PC,GETREQ   ; IF NOT DON'T BOTHER DRIVES
6748 027450 052762 101000 000016 1$:   BIS   #BIT15:BIT09,16(R2) ; GET DPB ADDRESS
6749 027456 004737 030246                JSR   PC,SVRH70    ; SET THE ERROR FLAGS
6750 027462 012764 000040 000010      MOV   #BIT05,RMCS2(R4) ; SAVE RH70/RMO3 REGISTERS
6751 027470 105061 022740                CLRB  DRVACT(R1)  ; "INIT" THE MASS BUS
6752 027474 105061 023016                CLRB  ULDFLG(R1) ; DRIVE IS IDLE
6753 027500 000470                BR    ST09        ; CLEAR THE UNLOAD FLAG
6754 027502 116405 000016      ST02: MOVB  RMA5(R4),R5  ; DON'T BOTHER OTHER DRIVES
6755 027506 136105 023064      BITB  ATABIT(R1),R5 ; READ ATTENTION REG
6756 027512 001007                BNE  ST03          ; IS ATTENTION FOR THIS DRIVE UP ?
6757 027514 105761 022770      TSTB  DPINT(R1)  ; YES--BRANCH
6758 027520 001021                BNE  ST06          ; TRYING TO INITIALIZE THE DRIVE ?
6759 027522 105761 023000      TSTB  DPRQS(R1)  ; BR IF YES - DRIVE NOT ONLINE
; OUTSTANDING PORT REQUEST FOR THE DRIVE ?

```



```

6760 027526 001035      BNE      ST07      ;BR IF YES - NO RESPONSE TO REQUEST
6761 027530 000454      BR       ST09      ;OTHER WISE EXIT
6762 027532 105761 022770  ST03:  TSTB   DPINT(R1) ;INITIALIZING THE DRIVE ?
6763 027536 001003      BNE      1$        ;BR IF INIT PENDING
6764 027540 105761 023000  TSTB   DPRQS(R1) ;PORT REQUEST PENDING ?
6765 027544 001446      BEQ      ST09      ;BR IF NOT
6766 027546 012763 177777 023042  1$:    MOV    #-1,TIMER(R3) ;STOP THE TIMER
6767 027554 000442      BR       ST09      ;EXIT
6768 027556 004737 025272  ST05:  JSR    PC,C18  ;GO HANDLE THE PARITY ERROR
6769 027562 000437      BR       ST09
6770 027564 105061 022770  ST06:  CLRB   DPINT(R1) ;CLEAR THE INITIALIZE INDICATOR
6771 027570 105061 022750      CLRB   DRVSTA(R1) ;SET UNIT OFFLINE
6772 027574 012763 177777 023042  MOV    #-1,TIMER(R3) ;STOP THE TIMER
6773 027602 004737 031102      JSR    PC,GETREQ ;GET THE DPB ADDRESS
6774 027606 005702      TST    R2         ;REQUEST IN QUEUE ?
6775 027610 001424      BEQ      ST09      ;BR IF NOT
6776 027612 052762 140000 000016  BIS    #BIT15:BIT14,16(R2) ;INFORM THE USER DRIVE NOT AVAILABLE
6777 027620 000414      BR       ST08
6778 027622 012763 177777 023042  ST07:  MOV    #-1,TIMER(R3) ;STOP THE TIMER
6779 027630 105061 023000      CLRB   DPRQS(R1) ;CLEAR PORT REQUEST INDICATOR
6780 027634 004737 031102      JSR    PC,GETREQ ;GET DPB ADDRESS
6781 027640 005702      TST    R2         ;QUEUE ENTRY FOR DRIVE ?
6782 027642 001407      BEQ      ST09      ;BR IF NONE
6783 027644 012762 100004 000016  MOV    #BIT15:BIT2,16(R2) ;INFORM USER OF PORT REQUEST ERROR
6784 027652 004737 031006  ST08:  JSR    PC,EMPTYQ ;CLEAR THE QUEUE FOR THE DRIVE
6785 027656 004737 030246      JSR    PC,SVRH70 ;SAVE THE REGISTERS
6786 027662 012604  ST09:  MOV    (SP)+,R4    ;RESTORE R4
6787 027664 012603      MOV    (SP)+,R3    ;RESTORE R3
6788 027666 012602      MOV    (SP)+,R2    ;RESTORE R2
6789 027670 012601      MOV    (SP)+,R1    ;RESTORE R1
6790 027672 000207      RTS     PC         ;RETURN
6791
6792 ;ROUTINE TO READ A RH70/RMO3 REGISTER
6793 ;CALL
6794 ;
6795 ;
6796 ;
6797 ;
6798 ;
6799 ;
6800 ;
6801 027674 013737 023074 030042  RD.RM: MOV    MCPEMX,RD.RM2 ;MAX. RETRYs ALLOWED
6802 027702 011646      MOV    (SP)-,(SP) ;SAVE RD FOR RETURN
6803 027704 013737 023076 027720  MOV    RMADR,RD.ADR ;FORM THE DESIRED ADDRESS
6804 027712 062037 027720      ADD    (RD)+,RD.ADR ;USING THE BASE AND THE INDEX
6805 027716 013727      RD.RM1: MOV   @((PC)+),(PC)+ ;READ THE DESIRED REGISTER OF THE RMO3
6806 027720 000000      RD.ADR: .WORD 0 ;ADDRESS IS FORMED HERE
6807 027722 000000      RD.WRD: .WORD 0 ;REG. CONTENTS PUT HERE
6808 027724 013766 027722 000002  MOV    RD.WRD,2(SP) ;RETURN IT TO THE USER
6809 027732 013746 023076      MOV    RMADR,-(SP) ;PUT THE ADDRESS ON THE STACK
6810 027736 062716 000010      ADD    #RMCS2,(SP) ;FORM THE ADDRESS OF RMCS2
6811 027742 032736 010000      BIT    #BIT12,@(SP)+ ;CHECK THE 'MED' BIT
6812 027746 001037      BNE    RD.RM3     ;BR IF DRIVE NON-EXISTENT
6813 027750 017746 173122      MOV    @RMADR,-(SP) ;READ RMCS1
6814 027754 032716 020000      BIT    #BIT13,(SP) ;DID MCPE SET?
6815 027760 001002      BNE    1$        ;BRANCH IF YES
    
```

```

6816 027762 022620          CMP      (SP)+,(RO)+      ;ADJUST FOR RETURN
6817 027764 000432          BR       RD.RM4          ;EXIT
6818 027766                1$:          JSR      RO,ES.SAV      ;SAVE THE ADDRESS IN 'SESCAPE'
6819 027766 004037 031172          ERROR   3              ;REPORT "MCPE" ERROR
6820 027772 104003          TST     DTUW            ;DATA TRANSFER UNDERWAY?
6821 027774 005737 023062          BMI     2$             ;NO--BRANCH
6822 030000 100405          BIT     #BIT14,(SP)    ;NO--"TRE"=1?
6823 030002 032716 040000          BEQ     2$             ;NO--BRANCH
6824 030006 001402          TST     (SP)+          ;YES--CLEAN OFF THE STACK AND
6825 030010 005726          BR       RD.RM3        ;TAKE THE FATAL ERROR EXIT
6826 030012 000415          2$:      BIS     #BIT14,(SP) ;CLEAR "MCPE" BY SENDING A "1" TO "TRE"
6827 030014 052716 040000          SWAB    (SP)           ;POSITION BEFORE WRITING
6828 030020 000316          MOV     RMADR,3$      ;FORM ADDRESS OF HIGH BYTE
6829 030022 013737 023076 030036          INC     3$
6830 030030 005237 030036          MOV     (SP)+,2(PC)+  ;WRITE THE HIGH BYTE OF RMCS1
6831 030034 112637          .WORD  0              ;ADDRESS STORAGE
6832 030036 000000          3$:      DEC     (PC)+    ;EXCEEDED MAX. RETRYS
6833 030040 005327          RD.RM2: .WORD  3
6834 030042 000003          BGE     RD.RM1        ;BRANCH IF NO
6835 030044 002324          RD.RM3: MOV     (RO),RO ;FATAL ERROR EXIT
6836 030046 011000          MOV     (SP)+,(SP)
6837 030050 012616          RD.RM4: RTS     RO
6838 030052 000200
6839
6840          ;ROUTINE TO WRITE A REGISTER
6841          ;CALL
6842          ;
6843          ;CALL
6844          ;
6845          ;
6846          ;
6847          ;
6848          ;
6849          ;
6850          ;
6851          ;
6852          ;
6853          ;
6854          ;
6855          ;
6856          ;
6857          ;
6858          ;
6859          ;
6860          ;
6861          ;
6862          ;
6863          ;
6864          ;
6865          ;
6866          ;
6867          ;
6868          ;
6869          ;
6870          ;
6871          ;

```



```

6872 030172 030236          WRT.R3
6873 030174 032726 000010  BIT      #BIT03,(SP)+
6874 030200 001420          BEQ      WRT.R4          ;BRANCH IF "PAR=0"
6875 030202 016037 177776 030214  MOV      -2(R0),1$      ;PICKUP THE INDEX
6876 030210 004037 027674          JSR      R0,RD.RM      ;READ THE REG.
6877 030214 000000          1$:     .WORD 0          ;REG. INDEX
6878 030216 030236          WRT.R3          ;RETURN TO THIS ADDRESS ON ERROR
6879 030220 004037 031172  JSR      R0,ES.SAV     ;SAVE THE ADDRESS IN 'SESCAPE'
6880 030224 104004          ERROR 4          ;REPORT THE PARITY ON WRITE ERROR
6881 030226 005726          TST      (SP)+        ;CLEAR OFF THE STACK
6882 030230 005327          DEC      (PC)+        ;DECREMENT THE ERROR COUNT
6883 030232 000003          WRT.R2: .WORD 3       ;RETRY COUNTER
6884 030234 002341          BGE      WRT.R1       ;TRY AGAIN IF NOT FINISHED
6885 030236 011000          WRT.R3: MOV      (R0),R0 ;TAKE THE "PARITY ON WRITE" ERROR EXIT
6886 030240 000401          BR       WRT.R5
6887 030242 005720          WRT.R4: TST      (R0)+ ;EXIT
6888 030244 000200          WRT.R5: RTS      R0    ;ADJUST FOR ERROR FREE EXIT
6889
6890          ;ROUTINE TO SAVE THE RH70/RP04/5/RMO3 REGISTERS AS PER DPB+14
6891          ;CALL
6892          ;
6893          MOV      #DPBNUM,R2          ;DPB POINTER TO R2
6894          JSR      PC,SVRH70          ;SAVE THE DRIVES REG'S
6895
6896          SVRH70: SAVREG          ;SAVE R0 - R5
6897          TST      R2          ;QUEUE ENTRY FOR THE DRIVE ?
6898          BEQ      6$          ;BR IF NONE
6899          MOV      RMADR,R4
6900          MOV      (R2),RMCS2(R4) ;SELECT DRIVE
6901          MOV      14(R2),R3      ;GET THE ERROR TABLE POINTER
6902          BEQ      6$          ;EXIT IF NO ADDRESS
6903          CLR      3$          ;COUNTER & POINTER
6904          CMP      3$,#RMDB      ;REACHED THE BUFFER REGISTER ?
6905          BNE      2$          ;BR IF NOT
6906          BIT      #BIT07,RMCS2(R4) ;'OR' SET ?
6907          BNE      2$          ;BR IF SET
6908          CLR      (R3)+        ;STORE RMDB AS ZEROES
6909          BR       4$          ;CONTINUE
6910          JSR      R0,RD.RM      ;READ THE SELECTED REGISTER
6911          .WORD 0          ;REGISTER INDEX
6912          5$:     .WORD 0          ;ERROR RETURN ADDRESS
6913          MOV      (SP)+,(R3)+    ;STORE THE REGISTER CONTENTS
6914          CMP      3$,#RMEC2     ;REACHED THE END ?
6915          BEQ      6$          ;BR IF YES
6916          ADD      #2,3$        ;INCREMENT THE REGISTER INDEX
6917          BR       1$          ;CONTINUE READING THE REGISTERS
6918          JSR      PC,C17        ;PROCESS THE UNCORRECTABLE PARITY ERROR
6919          RESREG
6920          RTS      PC          ;RESTORE R0 - R5
6921          ;RETURN
6922          ;ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A "TRE"
6923          ;CALL
6924          MOV      #DRVNUM,R1      ;DRIVE NUMBER TO R1
6925          JSR      PC,SET.IE      ;SET "IE"
6926          RETURN
6927

```

```

6928 030364 010446          SET.IE: MOV      R4, -(SP)          ;SAVE R4
6929 030366 013704 023076    MOV      RMADR, R4          ;PICKUP ADDRESS OF RMCS1
6930 030372 010164 000010    MOV      R1, RMCS2(R4)     ;SELECT DRIVE
6931 030376 011446          MOV      (R4), -(SP)       ;READ RMCS1
6932 030400 052716 040000    BIS      #BIT14, (SP)      ;SET THE "TRE" BIT OF THE WORD READ
6933 030404 000316          SWAB     (SP)              ;ADJUST FOR JATO
6934 030406 112714 000100    MOVB     #BIT06, (R4)      ;SET "IE"
6935 030412 032764 010000 000010 BIT      #BIT12, RMCS2(R4) ;IS "NEED"=1?
6936 030420 001002          BNE     1$                ;YES CLEAR "TRE"
6937 030422 005726          TST     (SP)+             ;CLEAN OFF THE STACK
6938 030424 000402          BR      2$
6939 030426 112664 000001    1$:  MOVB  (SP)+, 1(R4)      ;CLEAR "TRE"
6940 030432 012604          2$:  MOV  (SP)+, R4         ;RESTORE R4
6941 030434 000207          RTS     PC                ;RETURN TO CALLER

;QUEUE COUNT
6944 030436          000    QCNT: .BYTE 0          ;DRIVE 0
6945 030437          000    .BYTE 0          ;DRIVE 1
6946 030440          000    .BYTE 0          ;DRIVE 2
6947 030441          000    .BYTE 0          ;DRIVE 3
6948 030442          000    .BYTE 0          ;DRIVE 4
6949 030443          000    .BYTE 0          ;DRIVE 5
6950 030444          000    .BYTE 0          ;DRIVE 6
6951 030445          000    .BYTE 0          ;DRIVE 7

;QUEUE INPUT POINTERS
6955 030446 030530    QINPT: .WORD  QDRV0        ;DRIVE 0
6956 030450 030550    .WORD  QDRV1        ;DRIVE 1
6957 030452 030570    .WORD  QDRV2        ;DRIVE 2
6958 030454 030610    .WORD  QDRV3        ;DRIVE 3
6959 030456 030630    .WORD  QDRV4        ;DRIVE 4
6960 030460 030650    .WORD  QDRV5        ;DRIVE 5
6961 030462 030670    .WORD  QDRV6        ;DRIVE 6
6962 030464 030710    .WORD  QDRV7        ;DRIVE 7

;QUEUE OUTPUT POINTERS
6966 030466 030530    QOUTPT: .WORD  QDRV0        ;DRIVE 0
6967 030470 030550    .WORD  QDRV1        ;DRIVE 1
6968 030472 030570    .WORD  QDRV2        ;DRIVE 2
6969 030474 030610    .WORD  QDRV3        ;DRIVE 3
6970 030476 030630    .WORD  QDRV4        ;DRIVE 4
6971 030500 030650    .WORD  QDRV5        ;DRIVE 5
6972 030502 030670    .WORD  QDRV6        ;DRIVE 6
6973 030504 030710    .WORD  QDRV7        ;DRIVE 7

;DRIVE 0 START ADDRESS
6975 030506 030530    QSTART: .WORD  QDRV0        ;DRIVE 0 START ADDRESS
6976 030510 030550    QSTOP:  .WORD  QDRV1        ;DRIVE 0 STOP ADDRESS & DRIVE 1 START ADDRESS
6977 030512 030570    .WORD  QDRV2        ;STOP DRIVE 1--START DRIVE 2
6978 030514 030610    .WORD  QDRV3        ;STOP DRIVE 2--START DRIVE 3
6979 030516 030630    .WORD  QDRV4        ;STOP DRIVE 3--START DRIVE 4
6980 030520 030650    .WORD  QDRV5        ;STOP DRIVE 4--START DRIVE 5
6981 030522 030670    .WORD  QDRV6        ;STOP DRIVE 5--START DRIVE 6
6982 030524 030710    .WORD  QDRV7        ;STOP DRIVE 6--START DRIVE 7
6983 030526 030730    .WORD  QTERM        ;STOP DRIVE 7
    
```



```

6984
6985
6986
6987 030530 000010
6988 030550 000010
6989 030570 000010
6990 030610 000010
6991 030630 000010
6992 030650 000010
6993 030670 000010
6994 030710 000010
6995 030730
6996
6997
6998
6999
7000
7001
7002 030730 104412
7003 030732 012702 030436
7004 030736 005022
7005 030740 005022
7006 030742 005022
7007 030744 005022
7008 030746 012703 000010
7009 030752 012701 030506
7010 030756 012122
7011 030760 005303
7012 030762 001375
7013 030764 012703 000010
7014 030770 012701 030506
7015 030774 012122
7016 030776 005303
7017 031000 001375
7018 031002 104413
7019 031004 000207
7020
7021
7022
7023
7024
7025
7026
7027 031006 105061 030436
7028 031012 006301
7029 031014 016161 030446 030466
7030 031022 006201
7031 031024 000207
7032
7033
7034
7035
7036
7037
7038
7039

;DRIVE REQUEST QUEUES
QDRV0: .BLKW 10
QDRV1: .BLKW 10
QDRV2: .BLKW 10
QDRV3: .BLKW 10
QDRV4: .BLKW 10
QDRV5: .BLKW 10
QDRV6: .BLKW 10
QDRV7: .BLKW 10
QTERM=.

;ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
:CALL
: JSR PC,CLRQUE
CLRQUE: SAVREG ;SAVE R0 - R5
MOV #QCNT,R2 ;ZERO THE QUEUE COUNTS
CLR (R2)+ ;DRIVES 0 & 1
CLR (R2)+ ;DRIVES 2 & 3
CLR (R2)+ ;DRIVES 4 & 5
CLR (R2)+ ;DRIVES 6 & 7
MOV #8,R3 ;MOVE THE STARTING
MOV #QSTART,R1 ;ADDRESS OF THE QUEUE INTO
1$: MOV (R1)+,(R2)+ ;THE QUEUE INPUT POINTER
DEC R3
BNE 1$
MOV #8,R3 ;MOVE THE STARTING ADDRESS
MOV #QSTART,R1 ;OF THE QUEUE INTO THE
2$: MOV (R1)+,(R2)+ ;QUEUE OUTPUT POINTER
DEC R3
BNE 2$
RESREG ;RESTORE R0 - R5
RTS PC

;EMPTY THE QUEUE SPECIFIED BY R1
:CALL
: MOV DRVNUM,R1 ;DRIVE NUMBER TO R1
: JSR PC,EMPTYQ
EMPTYQ: CLRB QCNT(R1) ;CLEAR NUMBER OF ITEMS IN QUEUE
ASL R1
MOV QINPT(R1),QOUTPT(R1) ;SET OUTPUT QUEUE POINTER=INPUT POINTER
ASR R1
RTS PC

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

G11

MAINDEC-11-DZRM1 AO/00, RMO3 DRIVE COMPATIBILITY TEST MACY11 30(1046) 21-JUL-77 16:51 PAGE 136
 DZRM1A.P11 21-JUL-77 15:44 SINGLE/DUAL PORT RH70/RMO3 DRIVER (REV 1.0)

SEQ 0135

```

7040 ; RETURN2 ;RETURN HERE IF REQUEST IS IN QUEUE
7041
7042 031026 122761 000010 030436 DRVQUE: CMPB #10,QCNT(R1) ;IS QUEUE FULL?
7043 031034 001421 BEQ 2$ ;BR IF YES-TAKE RETURN1
7044 031036 105261 030436 INCB QCNT(R1) ;INCREMENT QUEUE COUNT
7045 031042 006301 ASL R1
7046 031044 010271 030446 MOV R2,@QINPT(R1) ;PUT THIS REQUEST IN QUEUE
7047 031050 062761 000002 030446 ADD #2,QINPT(R1) ;UPDATE THE QUEUE POINTER
7048 031056 026161 030446 030510 CMP QINPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER
7049 031064 001003 BNE 1$ ;BRANCH IF NO
7050 031066 016161 030506 030446 MOV QSTART(R1),QINPT(R1) ;YES--RESET POINTER
7051 031074 006201 1$: ASR R1
7052 031076 005720 TST (R0)+ ;TAKE RETURN 2
7053 031100 000200 2$: RTS R0 ;RETURN TO USER
7054
7055 ;ROUTINE TO GET THE "DPB" ADDRESS OF NEXT REQUEST IN QUEUE
7056
7057 ;CALL
7058 ; MOV #DRVNUM,R1 ;DRIVE NUMBER TO R1
7059 ; JSR PC,GETREQ ;GO GET THE REQUEST
7060 ; RETURN ;R2="DPB" ADDRESS OF THE REQUEST
7061 ; ;R2=0 IF NO REQUEST IN QUEUE
7062
7063 031102 005002 GETREQ: CLR R2
7064 031104 105761 030436 TSTB QCNT(R1) ;IS THERE ANY REQUEST IN QUEUE?
7065 031110 001404 BEQ 2$ ;NO---BRANCH
7066 031112 006301 1$: ASL R1
7067 031114 017102 030466 MOV @QOUTPT(R1),R2 ;PICKUP "DPB" POINTER FOR THIS DRIVE
7068 031120 006201 ASR R1
7069 031122 000207 2$: RTS PC ;RETURN TO USER
7070
7071 ;ROUTINE TO "POP" THE REQUEST FROM QUEUE
7072
7073 ;CALL
7074 ; MOV #DRVNUM,R1 ;DRIVE NUMBER TO R1
7075 ; JSR PC,POPQUE ;CALL TO REMOVE REQUEST
7076 ; RETURN ;R2=ADDRESS OF DPB REMOVED
7077
7078 031124 105361 030436 POPQUE: DECB QCNT(R1) ;DECREMENT QUEUE COUNT
7079 031130 006301 ASL R1
7080 031132 017102 030466 MOV @QOUTPT(R1),R2 ;GET THE "DPB" POINTER
7081 031136 005071 030466 CLR @QOUTPT(R1) ;REMOVE DPB ADDRESS FROM THE QUEUE
7082 031142 062761 000002 030466 ADD #2,QOUTPT(R1) ;UPDATE THE QUEUE POINTER
7083 031150 026161 030466 030510 CMP QOUTPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER?
7084 031156 001003 BNE 1$ ;NO--BRANCH TO EXIT
7085 031160 016161 030506 030466 MOV QSTART(R1),QOUTPT(R1) ;YES--RESET THE POINTER
7086 031166 006201 1$: ASR R1
7087 031170 000207 RTS PC ;RETURN TO USER
7088
7089 ;ROUTINE TO SAVE THE CONTENTS OF 'SESCAPE' WHEN THE DRIVER
7090 ;REPORTS AN ERROR DIRECTLY.
7091
7092 ;CALL
7093 ; JSR ERROR,R0,ES.SAV
7094 ; RETURN N ;: THE ERROR CALL
7095 ; ;THE RETURN IS PAST THE ERROR CALL

```


H11

```

7096
7097 031172 012037 031206 ES.SAV: MOV (R0)+,1$ ;GET THE ERROR CALL
7098 031176 013746 001176 MOV $ESCAPE,-(SP) ;SAVE THE ADDRESS IN '$ESCAPE'
7099 031202 005037 001176 CLR $ESCAPE ;CLEAR THE ESCAPE RETURN
7100 031206 000000 1$: .WORD 0 ;THE ERROR CALL IS MOVED HERE
7101 031210 012637 001176 MOV (SP)+,$ESCAPE ;RESTORE THE ESCAPE ADDRESS
7102 031214 000200 RTS R0 ;RETURN
7103
7104 ;*****
7105 .SBTTL DATA, CONTROL, & STATUS BLOCKS
7106
7107 ;*****
7108
7109 .SBTTL ROUTINE TO SIZE MEMORY
7110
7111 ;*****
7112 ;*CALL:
7113 ;* JSR PC,$SIZE
7114 ;* RETURN
7115 ;*$LSTAD WILL CONTAIN THE LAST AVAILABLE MEMORY LOCATION
7116
7117 $SIZE: MOV R0,-(SP) ;;SAVE R0 ON THE STACK
7118 031216 010046 MOV R1,-(SP) ;;SAVE R1 ON THE STACK
7119 031220 010146 MOV @#ERRVEC,-(SP) ;;SAVE PRESENT ERROR VECTOR PS & PC
7120 031222 013746 000004 MOV @#ERRVEC+2,-(SP)
7121 031226 013746 000006 MOV SP,R0 ;;SAVE THE STACK POINTER
7122 031232 010600 ;;SET THE ERRVEC PS TO THE PRESENT PS
7123 TRAP ;;PUSH OLD PSW AND PC ON STACK
7124 031234 104400 MOV (SP)+,@#ERRVEC+2 ;;SAVE THE PSW IN @#ERRVEC+2
7125 031236 012637 000006 000004 MOV #2,@#ERRVEC ;;SET FOR TIMEOUT
7126 031242 012737 031262 MOV #2000,R1 ;;FIRST ADDRESS
7127 031250 012701 020000 1$: TST (R1) ;;TEST THIS ADDRESS
7128 031254 005711 TST (R1)+ ;;STEP TO NEXT ADDRESS
7129 031256 005721 BR 1$ ;;TRY ANOTHER
7130 031260 000775 2$: SUB #2,R1 ;;DROP BACK
7131 031262 162701 000002 MOV R0,SP ;;RESTORE THE STACK
7132 031266 010006 MOV (SP)+,@#ERRVEC+2 ;;RESTORE ERROR VECTOR
7133 031270 012637 000006 MOV (SP)+,@#ERRVEC
7134 031274 012637 000004 MOV R1,$LSTAD ;;LAST ADDRESS
7135 031300 010137 031312 MOV (SP)+,R1 ;;RESTORE R1
7136 031304 012601 MOV (SP)+,R0 ;;RESTORE R0
7137 031306 012600 RTS PC
7138 031310 000207 $LSTAD: .WORD 0 ;;CONTAINS THE LAST ADDRESS
7139 031312 000000
7140
7141 .SBTTL BUSADR - GET BUS ADDRESS AND VECTOR ADDRESS FOR RH11
7142 ;THIS ROUTINE IS USED TO INSURE THE BUS ADDRESS
7143 ;OF THE RH11 IS SETUP FOR THE PROPER ADDRESS.
7144 ;IT WILL ALSO READ THE ADDRESS FROM THE TTY IF
7145 ;REQUIRED.
7146 ;NOTE: THIS ROUTINE DESTROYS R0-R4
7147 ;CALL
7148
7149 ; JSR PC,BUSADR
7150 ; RETURN
7151
  
```

```

7152 031314 104412          BUSADR: SAVREG          :SAVE ALL REG
7153 031316 012700 001274 1$:  MOV      #SRMADR,R0      ;FIRST ADDRESS
7154 031322 104401 031464   TYPE      MRMCS1        ;"RMCS1="
7155 031326 011046          MOV      (R0),-(SP)     ;PRESENT RMCS1 ADDRESS
7156 031330 104402          TYPOC           ;TYPE IT
7157 031332 104401 036121   TYPE      ,LINS        ;2 SPACES
7158 031336 104411          RDLIN          ;GET THE ENTRY
7159 031340 012601          MOV      (SP)+,R1      ;ADDRESS OF ASCII TEXT
7160 031342 004537 031506   JSR      R5,CK.NUM     ;ENTER AND STORE THE NEW ADDRESS
7161 031346 000763          BR       1$           ;ERROR EXIT
7162 031350 012700 001276 2$:  MOV      #SRMVEC,R0     ;VECTOR ADDRESS
7163 031354 104401 031475   TYPE      MRHVEC       ;"RHVEC="
7164 031360 011046          MOV      (R0),-(SP)   ;PRESENT RH11 VECTOR ADDRESS ON THE STACK
7165 031362 104402          TYPOC           ;TYPE IT
7166 031364 104401 036121   TYPE      ,LINS        ;2 SPACES
7167 031370 104411          RDLIN          ;READ THE ENTRY
7168 031372 012601          MOV      (SP)+,R1      ;ASCII TEXT ADDRESS
7169 031374 004537 031506   JSR      R5,CK.NUM     ;ENTER AND STORE NEW ADDRESS
7170 031400 000763          BR       2$           ;ERROR EXIT
7171 031402 012700 001274 3$:  MOV      #SRMADR,R0     ;FIRST ADDRESS OF NEW PARAMETERS
7172 031406 012701 023076   MOV      #RMADR,R1     ;FIRST ADDRESS OF WHERE TO PUT THEM
7173 031412 013705 000004   MOV      @#ERRVEC,R5   ;SAVE ERROR VECTOR
7174 031416 012737 031442 000004  MOV      #45,@#ERRVEC  ;LOAD NEW VECTOR ADDRESS
7175 031424 005777 147644   TST     @SRMADR        ;LEGAL I/O ADDRESS ?
7176 031430 010537 000004   MOV      R5,@#ERRVEC  ;YES, IF NOT TRAP TO TIMEOUT
7177 031434 012021          MOV      (R0)+,(R1)+  ;LOAD THE BUS ADDRESS
7178 031436 012021          MOV      (R0)+,(R1)+  ;LOAD VECTOR ADDRESS
7179 031440 000407          BR       6$           ;COMMON EXIT
7180 031442 012716 031450 4$:  MOV      #5$, (SP)     ;SET RETURN ADDRESS
7181 031446 000002          RTI                ;RETRUN FROM TIME OUT TRAP
7182 031450 104006          ERROR      6          ;NO RESPONSE FROM RMADR
7183 031452 010537 000004   MOV      R5,@#ERRVEC  ;RESTORE THE TIME OUT TRAP
7184 031456 000717          BR       1$           ;TRY AGAIN
7185 031460 104413          6$:  RESREG          ;RESTORE ALL REG
7186 031462 000207          RTS      PC
7187
7188 031464 046522 051503 020061 MRMCS1: .ASCIZ @RMCS1 = @
7189 031472 020075 000          000
7190 031475 122 053110 041505 MRHVEC: .ASCIZ @RHVEC = @
7191 031502 036440 000040
7192
7193          .SBTTL CK.NUM - CHECK NUMBER (OCTAL)
7194          ;THIS ROUTINE CHECKS AN ASCIZ STRING FOR LEGAL CHARACTERS
7195          ;AND FORMS AN OCTAL NUMBER IN R2
7196          ;CALL:
7197          ;
7198          ;
7199          ;
7200          ;
7201          CK.NUM: MOV      #ADR,R1          ;ADDRESS OF ASCIZ STRING
7202          JSR      R5,CK.NUM          ;R5 CHANGED
7203          RET
7204          RET          ;ERROR EXIT
7205          RET          ;NORMAL EXIT
7206          MOV      R2,-(SP)          ;SAVE R2
7207          MOV      R3,-(SP)          ;SAVE R3
7208          MOV      R4,-(SP)          ;SAVE R4
7209          MOV      #6,R3            ;MAX OCTAL DIGITS IN THE NUMBER
7210          CLR     R2                ;FINAL OCTAL VALUE
7211          1$:  MOV      (R1)+,R4          ;GET CURRENT POINTED BYTE
7212          BEQ     3$                ;BRANCH, IF TERMINATOR DETECTED
    
```


7208	031526	120427	000060		CMPB	R4,#'0	;	SMALLER THAN ASCII-0 ?
7209	031532	103425			BLO	5\$;	YES, ERROR EXIT
7210	031534	120427	000067		CMPB	R4,#'7	;	LARGER THAN ASCII-7 ?
7211	031540	101022			BHI	5\$;	YES, ERROR EXIT
7212	031542	006302			ASL	R2	;	SHIFT LEFT
7213	031544	103420			BCS	5\$;	
7214	031546	006302			ASL	R2	;	ONE
7215	031550	103416			BCS	5\$;	
7216	031552	006302			ASL	R2	;	OCTAL DIGIT
7217	031554	103414			BCS	5\$;	ERROR IF CARRY BIT SET
7218	031556	042704	177770		BIC	#177770,R4	;	CHOP OFF HIGHER BITS
7219	031562	060402			ADD	R4,R2	;	APPENDING CURRENT DIGIT TO NUMBER
7220	031564	005303			DEC	R3	;	DECREMENT BYTE COUNT
7221	031566	001401			BEQ	2\$;	BRANCH, IF LAST BYTE
7222	031570	000754			BR	1\$;	LOOPING BACK
7223	031572	112104		2\$:	MOVB	(R1)+,R4	;	CHECK TERMINATOR
7224	031574	001004			BNE	5\$;	ERROR EXIT
7225	031576	005702		3\$:	TST	R2	;	FINAL VALUE = 0
7226	031600	001402			BEQ	5\$;	YES, TAKE ERROR EXIT
7227	031602	010210			MOV	R2,(R0)	;	REPLACE THE ORIGINAL VALUE
7228	031604	005725		4\$:	TST	(R5)+	;	ADJUST FOR NORMAL RETURN
7229	031606	012604		5\$:	MOV	(SP)+,R4	;	RESTORE R4
7230	031610	012603			MOV	(SP)+,R3	;	RESTORE R3
7231	031612	012602			MOV	(SP)+,R2	;	RESTORE R2
7232	031614	000205			RTS	R5	;	EXIT

;;*****

.SBTTL ERROR MESSAGES

;;*****

.NLIST BEX

031616	044122	030067	030450	EM1:	.ASCIZ	/RH70(11) INTERRUPT OCCURRED (RMAS = 0)/
031665	125	042516	050130	EM2:	.ASCIZ	/UNEXPECTED ATTENTION OCCURRED/
031723	115	051501	041123	EM3:	.ASCIZ	/MASSBUS PARITY ERROR (MCPE=1)/
031761	115	051501	041123	EM4:	.ASCIZ	/MASSBUS PARITY ERROR (PAR=1)/
032016	042101	051104	051505	EM5:	.ASCIZ	/ADDRESS PLUG CHANGE BIT SET/
032052	044122	030461	033450	EM6:	.ASCIZ	/RH11(70) DIDN'T RESPOND TO ADDRESSING/
032120	047125	047503	051122	EM10:	.ASCIZ	/UNCORRECTABLE MASSBUS PARITY ERROR/
032163	106	052101	046101	EM11:	.ASCIZ	/FATAL MASSBUS PARITY ERROR/
032216	042520	051522	051511	EM12:	.ASCIZ	/PERSISTENT DEVICE UNSAFE/
032247	117	042520	040522	EM13:	.ASCIZ	/OPERATION NOT COMPLETED WITHIN TIME LIMIT/
032321	104	044522	042526	EM14:	.ASCIZ	/DRIVE WENT OFFLINE/

K11

 MAINDEC-11-DZRM1 AD/00, RMO3 DRIVE COMPATIBILITY TEST
 DZRMIA.P11 21-JUL-77 15:44 ERROR MESSAGES

MACY11 30(1046) 21-JUL-77 16:51 PAGE 140

SEQ 0139

032344	047516	051040	051505	EM15:	.ASCIZ	/NO RESPONSE TO PORT REQUEST/
032400	042510	042101	051105	EM20:	.ASCIZ	/HEADER CRC ERROR/
032421	104	052101	020101	EM21:	.ASCIZ	/DATA CHECK ('DCK') ERROR/
032452	051127	052111	020105	EM22:	.ASCIZ	/WRITE CHECK ERROR - DATA CHECK ('DCK') SET/
032525	127	044522	042524	EM23:	.ASCIZ	/WRITE CHECK ERROR - DATA CHECK ('DCK') NOT SET/
032604	042510	042101	051105	EM24:	.ASCIZ	/HEADER READ ERROR - 'FMT' BIT DROPPED/
032652	042510	042101	051105	EM25:	.ASCIZ	/HEADER READ ERROR - HEADER COMPARE ('HCE') ERROR/
032733	106	051117	040515	EM26:	.ASCIZ	/FORMAT ERROR ('FER')/
032760	042510	042101	051105	EM27:	.ASCIZ	/HEADER COMPARE ('HCE') ERROR/
033015	115	051511	042503	EM30:	.ASCIZ	/MISCELLANEOUS DRIVE ERROR/
033047	117	042520	040522	EM31:	.ASCIZ	/OPERATION INCOMPLETE ('OPI') ERROR/
033112	051104	053111	020105	EM32:	.ASCIZ	/DRIVE TIMING ('DTE') ERROR/
033145	120	051101	052111	EM33:	.ASCIZ	/PARITY ('PAR') ERROR AFTER OPERATION STARTED/
033222	051127	052111	020105	EM34:	.ASCIZ	/WRITE CLOCK FAILURE ('WCF') ERROR/
033264	047111	040526	044514	EM35:	.ASCIZ	/INVALID ADDRESS ('IAE') ERROR/
033322	051127	052111	020105	EM36:	.ASCIZ	/WRITE LOCK ('WLE') ERROR/
033353	104	052101	020101	EM37:	.ASCIZ	/DATA CHECK ('DCK') SET DURING WRITE CHECK COMMAND/
033435	122	030510	024061	EM40:	.ASCIZ	/RH11(70) OR UNIBUS TRANSFER ERROR/
033477	102	051525	040440	EM41:	.ASCIZ	/BUS ADDRESS OR WORD COUNT INCORRECT/
033543	104	052101	020101	EM42:	.ASCIZ	/DATA COMPARE ERRORS - NO OTHER ERROR(S) DETECTED/
033624	040503	023516	020124	EM43:	.ASCIZ	/CAN'T MATCH DATA READ WITH A PATTERN/
033671	105	051122	051117	EM44:	.ASCIZ	/ERROR BIT(S) SET, BUT NO ERROR SIGNALLED BY THE RH11/
033755	105	041503	046040	EM45:	.ASCIZ	/ECC LOGIC FAILURE - POSITION REGISTER VALUE NOT VALID/
034043	102	051525	040440	EM46:	.ASCIZ	/BUS ADDRESS AND WORD COUNT NOT CONSISTENT/
034115	123	042505	020113	EM50:	.ASCIZ	/SEEK INCOMPLETE ('SKI') ERROR/
034153	120	047522	051107	EM51:	.ASCIZ	/PROGRAM DETECTED POSITIONING ERROR/
034216	051104	053111	020105	EM60:	.ASCIZ	/DRIVE UNSAFE ERROR/
034241	040	000040		EM61:	.ASCIZ	/ /
				.EVEN		


```

035503 123 047503 042522 MSG16: .ASCIZ /SCORES FOR DRIVE /
035526 051124 041501 020113 MSG17: .ASCIZ /TRACK DRIVE OVRWRT OVRWRT READ READ/
035622 047516 020056 020040 MSG18: .ASCIZ /NO. READ OFST- OFST+ OFST- OFST+/
035717 040 051440 046105 SELFX: .ASCII / SELF/
035725 011 000 TAB: .BYTE 11,0 ;THE SEQUENCE OF SELFX,TAB MUST BE NOT REVERSED.
035727 040 020052 060 MARKX: .ASCII / & 0/
035733 011 000 .BYTE 11,0
035735 054 000 COMMA: .ASCIZ /,
035737 127 046111 020114 MSG5: .ASCIZ /WILL TEST /
035752 047117 000040 MSG6: .ASCIZ /ON /
035756 051511 052040 042510 MSG7: .ASCIZ /IS THERE ANOTHER SUB-SYSTEM (Y OR N<CR>) ?/
036031 116 052117 040440 NOTRM: .ASCIZ /NOT AN RMD3/
036045 040 047516 020124 NOTSAF: .ASCIZ / NOT SAFE/
036057 040 047516 020124 NOTPRS: .ASCIZ / NOT PRESENT/
036074 047440 043106 044514 UNTOFF: .ASCIZ / OFFLINE/
036105 040 047117 044514 UNTON: .ASCIZ / ONLINE/
036115 040 020040 040 LIN4SP: .ASCII / /
036121 040 000 LINSPO: .ASCIZ / /
036123 122 030115 000063 RMD3A: .ASCIZ /RMD3/
036130 005015 023440 023514 NEDCLK: .ASCIZ <CR><LF>/ 'L' OR 'P' CLOCK REQUIRED ON SYSTEM/<CR><LF>
036201 072 000 COLON: .ASCIZ /:/
036203 105 052116 051105 ENTDRV: .ASCIZ /ENTER I.D. FOR DRIVE #/
036232 020077 047111 040526 BADENT: .ASCIZ /? INVALID ENTRY/
036252 047105 042524 020122 ENTADR: .ASCIZ /ENTER BAD TRK/SEC ADDRESSES FOR DRV #2
036316 027440 000040 SLASH: .ASCIZ / /
036322 000077 QUES: .ASCIZ /?/
036324 051104 053111 000105 UNTMSG: .ASCIZ /DRIVE/
036332 000040 LINSPO: .ASCIZ / /
036334 051104 053111 020105 STATUS: .ASCIZ /DRIVE STATUS/
036351 040 020040 041040 MSG19: .ASCIZ / BAD SPOT FILE /
036375 040 025052 020040 MSG20: .ASCIZ / ** END OF TEST **/
036424 051104 053111 020105 MSG21: .ASCIZ /DRIVE NOT ON-LINE OR NOT ASSIGNED/<CR><LF>
036467 120 047522 051107 .ASCIZ /PROGRAM HALT/<CR><LF>

```

.EVEN

.LIST BEX

;HISTORY FILE FOR 15 LOGICAL DRIVES:(0-17)

```

SFMT = 1 ;COMMAND CODE
$COMND = SFMT+1 ;PROT SELECT AND A16,A17
$PSEL = SFMT+2 ;WORD COUNT
$WRDM = SFMT+3 ;BUFFER ADDRESS
$BUF = SFMT+5 ;SECTOR ADDRESS
$SEC = SFMT+7 ;TRACK ADDRESS
$TRK = SFMT+10 ;CYLINDER ADDRESS
$CYL = SFMT+11 ;SUB SYSTEM A-H
$SYSNM = SFMT+13 ;PHYSICAL DRIVE CODE (ASCII )
$PHYDR = SFMT+14 ;LEFT TWO NULL BYTES
$GAP = SFMT+15 ;END OF HISTORY TABLE
$ENTAB = $GAP+4

```

```

7271
7272
7273
7274 000001
7275 000002
7276 000003
7277 000004
7278 000006
7279 000010
7280 000011
7281 000012
7282 000014
7283 000015
7284 000016
7285 000022
7286
7287 036506 000 000 DRIVO: .BYTE 7&0,0 ;HISTORY BLOCK OF LOGICAL DRIVE 0
7288 036510 000020 .BLKB $ENTAB-$COMND

```


7289	036530	001	000	DRIV1:	.BYTE	7&1,0	:HISTORY BLOCK OF LOGICAL DRIVE 1
7290	036532	000020			.BLKB	SEMTAB-SCOMND	
7291	036552	002	000	DRIV2:	.BYTE	7&2,0	:HISTORY BLOCK OF LOGICAL DRIVE 2
7292	036554	000020			.BLKB	SEMTAB-SCOMND	
7293	036574	003	000	DRIV3:	.BYTE	7&3,0	:HISTORY BLOCK OF LOGICAL DRIVE 3
7294	036576	000020			.BLKB	SEMTAB-SCOMND	
7295	036516	004	000	DRIV4:	.BYTE	7&4,0	:HISTORY BLOCK OF LOGICAL DRIVE 4
7296	036620	000020			.BLKB	SEMTAB-SCOMND	
7297	036640	005	000	DRIV5:	.BYTE	7&5,0	:HISTORY BLOCK OF LOGICAL DRIVE 5
7298	036642	000020			.BLKB	SEMTAB-SCOMND	
7299	036662	006	000	DRIV6:	.BYTE	7&6,0	:HISTORY BLOCK OF LOGICAL DRIVE 6
7300	036664	000020			.BLKB	SEMTAB-SCOMND	
7301	036704	007	000	DRIV7:	.BYTE	7&7,0	:HISTORY BLOCK OF LOGICAL DRIVE 7
7302	036706	000020			.BLKB	SEMTAB-SCOMND	
7303	036726	000	000	DRIV10:	.BYTE	7&10,0	:HISTORY BLOCK OF LOGICAL DRIVE 10
7304	036730	000020			.BLKB	SEMTAB-SCOMND	
7305	036750	001	000	DRIV11:	.BYTE	7&11,0	:HISTORY BLOCK OF LOGICAL DRIVE 11
7306	036752	000020			.BLKB	SEMTAB-SCOMND	
7307	036772	002	000	DRIV12:	.BYTE	7&12,0	:HISTORY BLOCK OF LOGICAL DRIVE 12
7308	036774	000020			.BLKB	SEMTAB-SCOMND	
7309	037014	003	000	DRIV13:	.BYTE	7&13,0	:HISTORY BLOCK OF LOGICAL DRIVE 13
7310	037016	000020			.BLKB	SEMTAB-SCOMND	
7311	037036	004	000	DRIV14:	.BYTE	7&14,0	:HISTORY BLOCK OF LOGICAL DRIVE 14
7312	037040	000020			.BLKB	SEMTAB-SCOMND	
7313	037060	005	000	DRIV15:	.BYTE	7&15,0	:HISTORY BLOCK OF LOGICAL DRIVE 15
7314	037062	000020			.BLKB	SEMTAB-SCOMND	
7315	037102	006	000	DRIV16:	.BYTE	7&16,0	:HISTORY BLOCK OF LOGICAL DRIVE 16
7316	037104	000020			.BLKB	SEMTAB-SCOMND	
7317	037124	007	000	DRIV17:	.BYTE	7&17,0	:HISTORY BLOCK OF LOGICAL DRIVE 17
7318	037126	000020			.BLKB	SEMTAB-SCOMND	
7319							
7320							
7321							
7322	037146	000		FMTDPB:	.BYTE	0	:DRIVER PARAMETER BLOCK, DRIVE #
7323	037147	000			.BYTE	0	:OFFSET FMT16, HCI, ECI
7324	037150	000			.BYTE	0	:COMMAND CODE
7325	037151	000			.BYTE	0	:PSEL, A16 AND A17
7326	037152	000000			.WORD	0	:WORD COUNT (NEG)
7327	037154	037262			.WORD	#ENDPGM	:BUFFER ADDRESS
7328	037156	000			.BYTE	0	:SECTOR ADDRESS
7329	037157	000			.BYTE	0	:TRACK ADDRESS
7330	037160	000000			.WORD	0	:CYLINDER ADDRESS
7331	037162	037166			.WORD	RM.REG	:ADDRESS TO SAVE ALL RH11/RMO3 REG'S
7332	037164	000000			.WORD	0	:STATUS WORD
7333							
7334	037166	000000		RM.REG:	.WORD	0	:RMCS1
7335	037170	000000			.WORD	0	:RMIC
7336	037172	000000			.WORD	0	:RMBR
7337	037174	000000			.WORD	0	:RMDR
7338	037176	000000			.WORD	0	:RMCS2
7339	037200	000000			.WORD	0	:RMDR
7340	037202	000000			.WORD	0	:RMR1
7341	037204	000000			.WORD	0	:RMR1
7342	037206	000000			.WORD	0	:RMR1
7343	037210	000000			.WORD	0	:RMR1
7344	037212	000000			.WORD	0	:RMR1

```

7345 037214 000000 .WORD 0 ;RMDT
7346 037216 000000 .WORD 0 ;RMSN
7347 037220 000000 .WORD 0 ;RMOF
7348 037222 000000 .WORD 0 ;RMCA
7349 037224 000000 .WORD 0 ;RMDC
7350 037226 000000 .WORD 0 ;RMER2
7351 037230 000000 .WORD 0 ;RMR2
7352 037232 000000 .WORD 0 ;RMEC1
7353 037234 000000 .WORD 0 ;RMEC2

```

; INDEX OF STATUS AND REGISTER WORDS RELATIVE TO FMTDPB

```

7354
7355
7356
7357
7358 000016 STATUS = 16
7359 000020 SRMCS1 = 20
7360 000022 SRMWC = SRMCS1+2
7361 000024 SRMBA = SRMWC+2
7362 000026 SRMDA = SRMBA+2
7363 000030 SRMCS2 = SRMDA+2
7364 000032 SRMDS = SRMCS2+2
7365 000034 SRMER1 = SRMDS+2
7366 000036 SRMAS = SRMER1+2
7367 000040 SRMLA = SRMAS+2
7368 000042 SRMDB = SRMLA+2
7369 000044 SRMR1 = SRMDB+2
7370 000046 SRMDT = SRMR1+2
7371 000050 SRMSN = SRMDT+2
7372 000052 SRMOF = SRMSN+2
7373 000054 SRMDC = SRMOF+2
7374 000056 SRMR2 = SRMDC+2
7375 000060 SRMR2 = SRMR2+2
7376 000062 SRMER2 = SRMR2+2
7377 000064 SRMEC1 = SRMER2+2
7378 000066 SRMEC2 = SRMEC1+2

```

```

7380
7381 037236 000000 000000 177776 GENDPB: .WORD 0,0,-2,CYLDER
7382 037244 037256
7383 037246 000000 000000 037166 .WORD 0,0,RM.REG,0
7384 037254 000000
7385 037256 000002 CYLDER: .BLKW 2
7386 037262 037262 .ENDPGM = ;LAST LOCATION OF PROGRAM+2
7387 .NLIST BEX

```

```

037262 005015 040515 047111 TITLE: .ASCII <CR><LF>/MAINDEC-11-DZRM1-A/<CR><LF>
037310 046522 031460 042040 .ASCIZ RM03 DRIVE COMPATIBILITY TEST<CR><LF><LF>
037351 015 052012 020117 LOADRV: .ASCII <CR><LF>/TO TEST DRIVE 0, REPLACE THE 'XXDP' PACK ON DRIVE 0/<CR><LF>
037440 044527 044124 040440 .ASCII /WITH ANOTHER PACK, CLEAR MEMORY LOCATION 40,/<CR><LF>
037516 047101 020104 042522 .ASCIZ /AND RESTART THE PROGRAM/<CR><LF>

```

```

7388 .LIST BEX
7389 000001 .END

```


SW07 = 000200	1600#	1610												
SW08 = 000400	1599#	1609												
SW09 = 001000	1598#	1608												
SW1 = 000002	1616#													
SW10 = 002000	1597#	4264												
SW11 = 004000	1596#													
SW12 = 010000	1595#													
SW13 = 020000	1594#	4267												
SW14 = 040000	1593#													
SW15 = 100000	1592#													
SW2 = 000004	1615#													
SW3 = 000010	1614#													
SW4 = 000020	1613#													
SW5 = 000040	1612#													
SW6 = 000100	1611#													
SW7 = 000200	1610#													
SW8 = 000400	1609#													
SW9 = 001000	1608#													
SYSADR 002004	1995#	2503	2519*	2520*	2642	2643	2866	2867	3080	3081	3116	3117		
TAB 035725	3901	3905	3915	3922	7270#									
TABLEX 002044	1997#	3444*	3463											
TAB.XY= 001114	1740#	1747												
TBITVE= 000014	1650#	2348*	2349*											
TD 025746	6456	6465#												
TIMER 023042	3396#	5867#	6064*	6187*	6331*	6377*	6468*	6568*	6612*	6651*	6666*	6687*	6704	
	6706#	6766*	6772*	6778*										
TITLE 037262	2387	7387#												
TKVEC = 000060	1657#	4734*	4735*											
TPVEC = 000064	1658#													
TRAPVE= 000034	1656#	2338*	2339*	2384*										
TRFER 014224	4021	4145#												
TRK.LT 001370	1873#	2688												
TRNSWT 023010	3391#	5800#	6209*	6355*	6365	6382*	6391*	6470	6471*	6743				
TRTVEC= 000014	1651#													
TSTNM 001336	1860#	2826*	2892*	2945*	3024*	3103*	3185*	3386	3425	3523*	3616*	3710*		
TST1 005450	2727	2824#	2878	3061										
TST10 011476	3614#													
TST11 012150	3708#													
TST2 005762	2876	2890#												
TST3 006200	2943#													
TST4 006574	3022#													
TST5 007056	3086	3101#	3151	3843										
TST6 007340	3148	3183#												
TST7 011030	3306	3521#												
TYPDS = 104405	5694#													
TYPE = 104401	2387	2476	2477	2478	2479	2480	2481	2579	2580	2597	2598	2599	2600	
	2605	2606	2647	2648	2649	2650	2651	2657	2658	2659	2660	2661	2702	
	2728	2729	2794	2795	2836	2837	2838	2839	2840	2841	2842	2844	2845	
	2846	2851	2852	2853	2854	2877	3028	3029	3030	3035	3036	3083	3120	
	3121	3122	3123	3124	3130	3131	3132	3133	3149	3150	3811	3813	3816	
	3817	3822	3879	3880	3881	3886	3887	3888	3889	3890	3895	3896	3901	
	3904	3905	3911	3912	3913	3915	3922	3925	4204	4205	4206	4266	4335	
	4348	4353	4358	4523	4748	4749	4757	4758	4765	4788	4794	4799	4803	
	4808	4813	4814	4816	4819	4823	4841	4842	4843	4844	4845	4846	4847	
	4923	4931	4965	4982	4984	4987	4989	4993	5000	5048	5177	5249	5325	
	5356	5357	5360	5373	5384	5403	5690#	7154	7157	7163	7166			

TYPOC = 104402	4973	4997	5359	5691#	7156	7165									
TYPON = 104404	5693#														
TYPOS = 104403	2589	2654	2848	3032	3127	3819	3883	3898	5692#						
TYPRI4 016046	4551#														
UCPAR 013446	3965	3984#													
ULDFLG 023016	5827#	6015#	6090	6105#	6288#	6351#	6383#	6532	6618	6620*	6636	6638*	6752*		
UNIT 001326	1855#	3941#													
UNSAF 014254	4015	4162#													
UNTHSG 036324	7270#														
UNTOFF 036074	7270#														
UNTON 036105	7270#														
WCFER 014244	4039	4156#													
WCKD = 000151	1684#	2912	3240	3385	3557										
WCKER 014034	4018	4073#													
WCKMD = 000153	1685#														
WLEER 014174	4045	4130#													
WRTDAT = 000161	1686#	2903	2920	2957	3197	3531									
WRTHD = 000163	1687#														
WRT.AO 030144	6852#	6862#	6865#	7248											
WRT.RM 030054	6037	6041	6218	6222	6226	6234	6244	6248	6259	6266	6276	6290	6300		
	6321	6336	6347	6556	6588	6625	6849#								
WRT.R1 030140	6863#	6884													
WRT.R2 030232	6849#	6883#													
WRT.R3 030236	6858	6869	6872	6878	6885#										
WRT.R4 030242	6874	6887#													
WRT.R5 030244	6886	6888#													
WRT.WD 030142	6850#	6854	6861#	6864#	7248										
XEND2 012762	3076	3838	3846#												
XPASS1 004564	2614	2634#	2734												
XPASS2 006772	2616	3060	3074#												
\$APTHD 001100	1727	1733#													
\$ASTAT = ##### U	5119	5134													
\$ATYC 020230	5090	5092#													
\$ATY1 020204	5088#														
\$ATY3 020212	5033	5089#													
\$ATY4 020222	4936	5091#													
\$AUTO8 001150	1765#	2398#	5353	5443											
\$BASE 001264	1833#														
\$BADR 001136	1760#														
\$BDDAT 001142	1762#														
\$BELL 001200	1780#	4266	4923	4956											
\$BUF = 000006	2684#	2900#	2958#	2964	3199#	3205	3241*	3271*	3336	3537*	3627*	4196	4239		
	7278#														
\$CDW1 001270	1835#	2504#	2509	2638#	2644	2828	2830	2895	3049#	3078#	3082	3107	3193		
	3842#														
\$CDW2 001272	1836#	2472#	2478	2498	2552	2586	2600	2602*	2603	2639#	2651	2831	2833#		
	2841	2846	2861	3030	3079#	3108	3110*	3111	3124	3817	3881				
\$CHARC 020200	5050#	5060#	5067	5076#	5081#										
\$CKSWR 021314	5345#	5698													
\$CLR.T 017302	4874	4877#													
\$CHTAG 001114	1748#	2327	2328	2336	2342	2343	2344								
\$CM3 = 000000	1778#														
\$CNTLC 017146	4749	4808	4833#												
\$CNTLG 021723	4758	5356	5438#												
\$CNTLU 021716	4803	5373	5437#												
\$COMND = 000002	2690#	2903#	2912#	2920*	2957*	3197*	3240*	3531*	3557*	3626*	7275#	7288	7290		

SLKCSB	001302	1844#	4322#							
SLKCSR	001300	1843#	4316	4323#						
SLKS	001306	1846#	4327	4332#						
SLLVEC	001310	1847#	4329							
SLOOP	017360	4893	4897#							
SLPADR	001122	1753#	2360#	5634#	5650#	5655	5657			
SLPERR	001124	1754#	2361#	4946	5634	5651#	5657			
SLPVEC	001304	1845#	4319							
SLSTAD	031312	2419	7135#	7139#						
SMADR1	001242	1820#								
SMADR2	001246	1824#								
SMADR3	001252	1827#								
SMADR4	001256	1830#								
SMAIL	001210	1735	1739	1793#	2378	2392	4933	5028	5649	
SMAMS1	001240	1814#								
SMAMS2	001244	1822#								
SMAMS3	001250	1825#								
SMAMS4	001254	1828#								
SMBADR	001102	1735#								
SMFLG	020446	5089#	5095	5130#	5134#					
SMHEM	021741	5360	5441#							
SMSGAD	001224	1800#	5105#	5108						
SMSGLG	001226	1801#	5110#							
SMSGTY	001210	1794#	5103	5111#	5123	5127#				
SMSWR	021730	5357	5439#							
SMTYP1	001241	1815#								
SMTYP2	001245	1823#								
SMTYP3	001251	1826#								
SMTYP4	001255	1829#								
SMXCNT	022640	5647	5657#							
SMULL	001170	1774#	5055	5084						
SMNTST=	000000	2823#	2889#	2942#	3021#	3100#	3182#	3520#	3613#	3707#
SOCNT	021064	5221#	5250#	5263#						
SOCTVL	022344	5567	5592#							
SOMODE	021066	5216#	5220#	5225	5228#	5239#	5265#			
SOVER	022624	5611	5627	5635	5645	5654#				
SPASS	001216	1797#	2378#	4863#	4864#	4900	5641	5658		
SPASTM	001106	1737#								
SPHYDR=	000015	2568#	7283#							
SPOWER	020632	5178	5185#							
SPSEL =	000003	7276#								
SPWRDN	020452	2340	5147#	5175						
SPWRMG	020606	5178#								
SPWRUP	020524	5157	5163#							
SQUES	001204	1781#	4816	4956	5084	5403				
SROCHR	021576	5416#	5699							
SRODEC=	***** U	5701								
SROLIN	016644	4776#	5700							
SROCT=	***** U	5701								
SROSZ =	000001	5437#								
SRESRE	022010	5478#	5702							
SRMADR	001274	1841#	2400	2510#	2513	2519	7153	7171	7175	
SRMAS =	000036	7366#	7367							
SRMBA =	000024	4197	4207	4224	7361#	7362				
SRMCS1=	000020	3944	3946	4177	7359#	7360				
SRMCS2=	000030	4016	4179	7363#	7364					

SRMDA = 000026	3259	3290	3492	3649	3671	7362#	7363										
SRMDB = 000042	7368#	7369															
SRMDC = 000054	7373#	7374															
SRMDS = 000032	3948	4019	4048	7364#	7365												
SRMDT = 000046	7370#	7371															
SRMEC1 = 000064	7377#	7378															
SRMEC2 = 000066	7378#																
SRMER1 = 000034	4022	4025	4028	4031	4034	4037	4040	4043	4046	4051	4054	4074	4181				
	7365#	7366															
SRMER2 = 000062	4057	4059	4183	7376#	7377												
SRMR = 000056	7374#	7375															
SRMLA = 000040	7367#	7368															
SRMR1 = 000044	7369#	7370															
SRMR2 = 000060	7375#	7376															
SRMDF = 000052	7372#	7373															
SRMSN = 000050	7371#	7372															
SRMVEC = 001276	1842#	2401	2511*	2514	2520	7162											
SRMVC = 000022	3257	3260	3288	3291	3499	3501	3647	3650	3669	3672	4191	4210	7360#				
	7361																
SRTNAD 017362	4899#																
SRTN 017356	2348	2350*	2355*	4875	4894#												
SR2A = ***** U	5703																
SSAVRE 021752	5462#	5701															
SSAVR6 020630	5156*	5164	5165*	5166*	5184#												
SSB20 016402	4298	4350	4355	4360	4705#												
SSB20 016432	4282	4722#															
SSCOPE 022362	2334	5608#															
SSEC = 000010	2689*	2897*	2982*	2990*	2991	2993*	3196*	3237*	3259*	3270*	3290*	3334	3493				
	3503	3532*	3550*	3595*	3625*	3649*	3654*	3671*	7279#								
SSETUP = 000137	2311#	2333	2334	2336	2338	2340	2342	2343	2344	2346	2360	2389	4861				
	4917	4943	4951	5180	5340	5443	5609										
SSIZE 031216	2418	7118#															
SSTUP = 177777	2311#																
SSUPRS 016006	4299	4535#															
SSVLAD 022570	5619	5648#															
SSVPC = 000224	1711#	1716															
SSWR = 177600	1526#	1537	1541	1542	1543	1544	1545	1546	1547	1548	1778	1779	1780				
	2343	2344	2346	2360	2361	2825	2891	2944	3023	3102	3184	3522	3615				
	3709	4855	4862	4873	4886	4900	4908	4909	4910	4911	4912	4921	4928				
	4940	4944	4956	5179	5600	5601	5602	5603	5604	5610	5622	5624	5625				
	5628	5629	5630	5637	5638	5639	5651	5654	5657								
SSWREG 001232	1805#	2381															
SSWRM = 000000	1548	1549	5604	5605	5626												
SSYSM = 000014	2552*	2586	2831	2833	3108	3110	3910	7282#									
STATUS = 000016	3942	3956	3962	3964	3966	3968	3970	3972	4013	7358#							
STBIT 017364	2359*	4890*	4900#	5180*													
STERM = 000032	5704#																
STESTN 001214	1796#	5649*															
STIME 015136	4271	4346#	4397														
STIMES 001174	1778#	2343*	2825*	2891*	2944*	3023*	3102*	3184*	3522*	3615*	3709*	4862*	5637*				
	5644	5647*	5657														
STKB 001162	1771#	4398	4405	4736	5338	5349	5366	5420	5426								
STKINT 016462	2388	4734#															
STKS 001160	1770#	4406*	4737*	4751*	4761*	4763*	5338	5347	5363	5387*	5418	5424					
STKSRV 016512	4734	4744#															
STN = 000012	1526#	1537	2823	2825#	2889	2891#	2942	2944#	3021	3023#	3100	3102#	3182				

G13

MAINDEC-11-DZRM1 AO/00, RMD3 DRIVE COMPATIBILITY TEST MACY11 30(1046) 21-JUL-77 16:51 PAGE 164
DZRMIA.P11 21-JUL-77 15:44 CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0161

.SDB2D	1526#	5491
.SDB20	1526#	5554
.SEOP	1526#	4851
.SERRO	1526#	4902
.SERRT	1526#	4957
.SPOME	1526#	5143
.SREAD	1526#	5335
.SSAVE	1526#	5445
.SSCOP	1526#	5594
.SSIZE	1526#	7110
.STRAP	1526#	5659
.STYPD	1526#	5267
.STYPE	1526#	5005
.STYPO	1526#	5189

. ABS. 037550 000

ERRORS DETECTED: 0

DSKZ:DZRMIA, DSKZ:DZRMIA, SEQ/NL:MD:MC:CND:TOC/LI:ME/DOC/SOL/CRF=DSKM:RMDRV4.P11, DSKM:DZRMIA.P11
RUN-TIME: 29 20 2 SECONDS
RUN-TIME RATIO: 717/52=13.7
CORE USED: 40K (79 PAGES)

DOCUMENT PAGES: 161