

RM05/3/2

RM05/3/2 DRV CMPT TSTAH-F943A-MC
CZRMTA0 FICHE 1 OF 1

JUN 1980
COPYRIGHT © 1980
MADE IN USA



The main body of the document is a large grid of approximately 15 columns and 25 rows. Each cell in the grid contains a small, dense block of text, likely representing a data point or a small table. The text is too small to be legible in this image. The grid is organized into several distinct sections, with some cells containing larger blocks of text or diagrams. The overall layout is highly structured and repetitive.

.REM @

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

IDENTIFICATION

PRODUCT CODE: AC-F942A-MC
PRODUCT NAME: CZRMTAO RM05/3/2 DRV CMPT TST
DATE CREATED: APRIL 1980
MAINTAINER: CX DIAGNOSTIC ENGINEERING
AUTHOR: MIKE LEAVITT

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

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

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

COPYRIGHT (C) 1980 DIGITAL EQUIPMENT CORPORATION

CONTENTS

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

- 1.0 ABSTRACT
- 2.0 HARDWARE REQUIREMENTS
- 3.0 PRELIMINARY PROGRAM REQUIREMENTS
- 4.0 GENERAL PROGRAM CONSIDERATIONS
 - 4.1 SYSMAC
 - 4.2 XXDP
 - 4.3 ACT
 - 4.4 APT
 - 4.5 DUAL-ACCESS
 - 4.6 MEMORY MANAGEMENT
 - 4.7 MEMORY PARITY OPTION
 - 4.8 BAD SECTORS
 - 4.9 EXECUTION TIME
- 5.0 PROGRAM LOAD MEDIA
- 6.0 PROGRAM OPTIONS
 - 6.1 STARTING ADDRESSES
 - 6.2 SWITCH REGISTER OPTIONS USED
- 7.0 RUNNING THE PROGRAM
- 8.0 OPERATIONAL DIALOGUE
 - 8.1 DIALOGUE FOR ADDRESS 200 START
 - 8.2 DIALOGUE FOR ADDRESS 204 START
 - 8.3 DIALOGUE FOR ADDRESS 210 START
 - 8.4 PASS 1 DIALOGUE
 - 8.5 PASS 2 DIALOGUE
- 9.0 DESCRIPTION OF TESTS
 - 9.1 DESCRIPTION OF PASS 1 TESTS
 - 9.2 DESCRIPTION OF PASS 2 TESTS
- 10.0 PRINTOUT OF TEST RESULTS
 - 10.1 OVERWRITE AND DRIVE COMPATIBILITY DATA TEST RESULTS
- 11.0 ERROR REPORTING
 - 11.1 COMMON ERRORS
 - 11.2 ERROR HANDLING
 - 11.3 ERROR PRINTOUT EXAMPLE
- 12.0 TABLE DESCRIPTIONS
 - 12.1 TABLE A - BASIC READ/WRITE TEST SECTORS
 - 12.2 TABLE B - WORSE CASE DATA PATTERN

58
59
60
61
62
63
64

12.3 TABLE C - CYLINDER BLOCK ASSIGNMENT FOR A GIVEN SURFACE
12.4 TABLE D - BASIC CYLINDER BLOCK LAYOUT EXAMPLE
12.5 TABLE E - OVERWRITE CYLINDERS
12.6 TABLE F - SELF-TEST CYLINDERS
12.7 TABLE G - PSEUDO-RANDOM DATA PATTERN

13.0 RM SOFTWARE DRIVER DOCUMENT

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

1.0 ABSTRACT

THE PURPOSE OF THIS PROGRAM IS TO VERIFY THE COMPATIBILITY OF UP TO 16 RM05/3/2 DRIVES WHICH MAY RESIDE ON 1 OR MORE RH/RM 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.

2.0 HARDWARE REQUIREMENTS

THE FOLLOWING HARDWARE IS REQUIRED TO RUN THE RM05/3/2 DRIVE COMPATIBILITY PROGRAM.

PDP-11/04, (05,10 MFG. ONLY), 20,30,34,35,40,45,50,70
16K MEMORY

58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114

KW11-L OR KW11-P CLOCK
PROGRAM LOADING DEVICE
TERMINAL
RH11 OR RH70 CONTROLLER
1 TO 8 DISK DRIVES PER CONTROLLER (ANY COMBINATION OF RM05'S, RM03'S OR RM02'S)

ANY COMBINATION OF DRIVE TYPES MAY BE MIXED TOGETHER ON A CONTROLLER. BUT, DO TO THE PHYSICAL SIZE OF THE DISK PACKS, THE RM03/2'S AND THE RM05'S CANNOT BE SELECTED FOR COMPATIBILITY TOGETHER. IF FOR SOME REASON AN RM03/2 AND AN RM05 ARE SELECTED FOR COMPATIBILITY TOGETHER, THE PROGRAM WILL RECOGNIZE THIS UPON THE DIFFERENT DRIVE TYPE AND TYPE THE FOLLOWING MESSAGE:

?CANNOT SELECT RM03/2'S AND RM05'S TOGETHER (NOT COMPATIBLE)

IN ADDITION, A SINGLE RM03/2 OR RM05 DISK CARTRIDGE IS REQUIRED WHICH MUST BE FORMATTED IN 32 SECTOR FORMAT, ON A RELIABLE WELL-ALIGNED(REFERENCE PACK) RM03/2 OR RM05 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 RM05/3/2 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.

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

115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171

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. AFTER READING THE BAD SECTOR FILE, THE PROGRAM SEARCHES THE LIST OF BAD SECTORS TO DETERMINE IF ANY BAD SPOTS EXIST IN ANY TEST AREAS ON THE DISK PACK. IF A BAD SPOT IS FOUND TO BE PRESENT IN ANY OF THE TEST AREAS, THE FOLLOWING MESSAGE WILL BE TYPED:

PACK IS NOT ACCEPTABLE, CHANGE PACK AND TRY AGAIN.

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 RM03/2 DRIVE AND ABOUT 5 MINUTES PER RM05 DRIVE, NOT INCLUDING OPERATOR INTERVENTION.

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.

172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228

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 RH/RM BASE ADDRESS, INTERRUPT VECTOR AND PRIORITY. THE PROGRAM WILL ASSUME ALL DRIVES TO BE TESTED RESIDE ON ONE RH/RM SUBSYSTEM ONLY.
- 204 - THIS IS THE STARTING ADDRESS TO RUN PASS 1 ON ALL RH/RM SUBSYSTEMS WHICH RESIDE ON THIS PDP-11 SYSTEM. THE PROGRAM WILL ASK FOR THE RH/RM 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.
- 210 - THIS IS THE STARTING ADDRESS TO RUN PASS 2 ON ALL RH/RM SUBSYSTEMS WHICH RESIDE ON THIS PDP-11 SYSTEM. THE PROGRAM WILL ASK FOR THE RH/RM 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
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:

229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285

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.

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:

CZRMTAO - RM05/3/2 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

286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342

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:

CZRMTAO - RM05/3/2 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:

IS THERE ANOTHER SUBSYS (Y OR N)?

THE OPERATOR TYPES 'Y' OR 'N'. (IF JUST <CR> IS TYPED, THE PROGRAM ASSUMES THAT 'N' 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)?

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

343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399

THE RH/RM 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: RH/RM 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 TO PERFORM PASS 1 ON THE NEXT SUBSYSTEM.

8.3 DIALOGUE FOR ADDRESS 210 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:

CZRMTAO - RM05/3/2 DRIVE COMPATIBILITY TEST

THE DIALOGUE FOR 210 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 ON SUBSYS A

NOTE: THAT SUB-SYSTEM 'A' IS ALWAYS THE FIRST SUB-SYSTEM TO BE TESTED REGARDLESS OF HOW MANY SUB-SYSTEMS ARE TO BE TESTED.

400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456

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 <CR> WHEN DRIVE READY.

THE OPERATOR PERFORMS THIS TASK AND TYPES <CR> WHEN THE DRIVE IS READY. 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 <CR> WHEN DONE.

THE OPERATOR PERFORMS THESE FUNCTIONS AND TYPES <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

NOTE: THAT SUB-SYSTEM 'A' IS ALWAYS THE FIRST SUB-SYSTEM TO BE TESTED REGARDLESS OF HOW MANY SUB-SYSTEMS ARE TO BE TESTED.

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 210 START (SEE SECTION 8.3). IN EITHER CASE, THE PROGRAM INDICATES THE START OF PASS 2 BY TYPING:

** STARTING PASS 2 ON SUBSYS A

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.

457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513

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 210 START), THE ENTIRE TESTING AND REPORTING HAVE BEEN COMPLETED, AND THE PROGRAM TYPES:

TEST COMPLETE

(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

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:

A0,A1, A2,...B0,B1,B2,...C0,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

514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570

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 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.

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

571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627

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. 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

628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684

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 8.5), TO PROCEED WITH THE ABOVE STEPS ON THE NEXT DRIVE.

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.

11.0 ERROR REPORTING

685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741

11.1 COMMON ERRORS

THE FOLLOWING IS A LIST OF COMMON ERROR MESSAGES WHICH ACCOMPANY ERROR TYPEOUTS FROM THE RM05/3/2 DRIVE COMPATIBILITY PROGRAM. THE ERRORS ARE SELF-EXPLANATORY.

- ADDRESS PLUG CHANGE BIT SET
- RH 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
- WRITE CLOCK FAILURE 'WCF' ERROR
- INVALID ADDRESS 'IAE' ERROR
- WRITE LOCK 'WLE' ERROR
- DATA CHECK 'DCK' SET DURING WRITE CHECK COMMAND
- RH OR UNIBUS TRANSFER ERROR

742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798

BUS ADDRESS OR WORD COUNT INCORRECT
 DATA COMPARE ERRORS - NO OTHER ERROR(S) DETECTED
 CAN'T MATCH DATA READ WITH A PATTERN
 ERROR BIT(S) SET, BUT NO ERROR SIGNALLED BY THE RH
 ECC LOGIC FAILURE - POSITION REGISTER VALUE NOT VALID
 BUS ADDRESS AND WORD COUNT NOT CONSISTENT
 SEEK INCOMPLETE 'SKI' ERROR
 PROGRAM DETECTED POSITIONING ERROR
 DRIVE UNSAFE ERROR

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, RH/RM REGISTERS) ARE REPORTED IN A GIVEN ERROR. OTHER ERROR REPORTS MAY CONSIST OF A SINGLE DESCRIPTIVE LINE.

11.3 ERROR PRINTOUT EXAMPLE

RH OR UNIBUS TRANSFER ERROR				
DRIVE	RMCS1	RMWC	RMBA	RMDA
000001	144250	174400	0055030	000431
RMCS2	RMDS	RMER1	RMAS	RMDB
000100	010700	000000	000000	000000
RMMR1	RMDT	RMOF	RMDC	RMMR2
000050	024024	010000	000716	011777
RMER2	RMEC1	RMEC2		
000000	004066	000000		

12.0 TABLE DESCRIPTIONS

12.1 TABLE A - BASIC READ/WRITE TEST SECTORS

ADDRESS OF SECTOR ON EACH SURFACE

799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855

DRIVE NO.	CYLINDER	SECTORS
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

12.2 TABLE B - WORST CASE DATA PATTERN (REPEATS EVERY 16 WORDS)

WORD NO.	DATA (OCTAL)
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

12.3 TABLE C - CYLINDER BLOCK ASSIGNMENT FOR A GIVEN SURFACE

CURRENT ZONE - RANGE	OVERWRITE CYL BLOCK RANGE	COMPATIBILITY CYL BLOCK 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

856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912

6 - 640-767 640-655 752-767
7 - 768-822 768-783 ---

12.4 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	
112	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	->
113	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	->
114	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1	2	->
115	6	7	8	9	10	11	12	13	14	15	0	1	2	3	4	5	->
116	10	11	12	13	14	15	0	1	2	3	4	5	6	7	8	9	->
117	15	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	->
118	5	6	7	8	9	10	11	12	13	14	15	0	1	2	3	4	->
119	12	13	14	15	0	1	2	3	4	5	6	7	8	9	10	11	->
120	4	5	6	7	8	9	10	11	12	13	14	15	0	1	2	3	->
121	13	14	15	0	1	2	3	4	5	6	7	8	9	10	11	12	->
122	7	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	->
123	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1	->
124	14	15	0	1	2	3	4	5	6	7	8	9	10	11	12	13	->
125	11	12	13	14	15	0	1	2	3	4	5	6	7	8	9	10	->
126	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7	8	->
127	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7	->
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
->	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
->	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	
->	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1	2	
->	6	7	8	9	10	11	12	13	14	15	0	1	2	3	4	5	
->	10	11	12	13	14	15	0	1	2	3	4	5	6	7	8	9	
->	15	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
->	5	6	7	8	9	10	11	12	13	14	15	0	1	2	3	4	
->	12	13	14	15	0	1	2	3	4	5	6	7	8	9	10	11	
->	4	5	6	7	8	9	10	11	12	13	14	15	0	1	2	3	
->	13	14	15	0	1	2	3	4	5	6	7	8	9	10	11	12	
->	7	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	
->	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1	
->	14	15	0	1	2	3	4	5	6	7	8	9	10	11	12	13	
->	11	12	13	14	15	0	1	2	3	4	5	6	7	8	9	10	
->	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7	8	
->	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7	

12.5 TABLE E - OVERWRITE CYLINDERS

DRIVE #	CYLINDERS OVERWRITTEN
0	0,128,256,384,512,640,768

913	1	1,129,257,385,513,641,769
914	2	2,130,258,386,514,642,770
915	3	3,131,259,387,515,643,771
916	4	4,132,260,388,516,644,772
917	5	5,133,261,389,517,645,773
918	6	6,134,262,390,518,646,774
919	7	7,135,263,391,519,647,775
920	8	8,136,264,392,520,648,776
921	9	9,137,265,393,521,649,777
922	10	10,138,266,394,522,650,778
923	11	11,139,267,395,523,651,779
924	12	12,140,268,396,524,652,780
925	13	13,141,269,397,525,653,781
926	14	14,142,270,398,526,654,782
927	15	15,143,271,399,527,655,783

12.6 TABLE F - SELF-TEST CYLINDERS

	<u>DRIVE #</u>	<u>CYLINDERS</u>
933	0	17,145,273,401,529,657,785
934	1	18,146,274,402,530,658,786
935	2	19,147,275,403,531,659,787
936	3	20,148,276,404,532,660,788
937	4	21,149,277,405,533,661,789
938	5	22,150,278,406,534,662,790
939	6	23,151,279,407,535,663,791
940	7	24,152,280,408,536,664,792
941	8	25,153,281,409,537,665,793
942	9	26,154,282,410,538,666,794
943	10	27,155,283,411,539,667,795
944	11	28,156,284,412,540,668,796
945	12	29,157,285,413,541,669,797
946	13	30,158,286,414,542,670,798
947	14	31,159,287,415,543,671,799
948	15	32,160,288,416,544,672,800

12.8 TABLE G - PSEUDO-RANDOM DATA PATTERN

	<u>WORD #</u>	<u>DATA (OCTAL)</u>
953	0	040135
954	1	177070
955	2	070414
956	3	064531
957	4	174473
958	5	062422
959	6	114352
960	7	036620
961	8	010031
962	9	052336
963	10	017310
964	11	011347

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

12 102367
13 152567
14 001246
15 160073

13.0 RM SOFTWARE DRIVER DOCUMENT

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

13.1 TO INITIALIZE THE DRIVER:

JSR PC,RMINIT
RETURN

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

DRVSTA	DRIVE STATE
>0	ONLINE
=0	OFFLINE, DRIVE IS NOT AN RM05/3/2, OR NONEXISTENT DRIVE
<0	UNSAFE

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

DRVTYP	CONDITION
0	NONEXISTENT DRIVE
5	RM02
4	RM03
7	RM05
-1	NOT AN RM05/3/2

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

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

CALL:

JSR RO,RM05	: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)

```

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
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

```

```

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

```

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

```

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

```

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

13.4 EXAMPLE - WRITE 1000. WORDS

```

1$:      JSR      R0,RM05      ;CALL THE DRIVER
         WRTDPB      ;DPB ADDRESS
         BR        1$         ;WAIT FOR QUEUE IF FULL
2$:      TST      WRTDPB+16    ;WAIT FOR COMMAND TO COMPLETE
         BEQ      2$
         BMI      ERROR1      ;ERROR OCCURRED
         .
         .
         .
WRTDPB:  .BYTE    5           ;DRIVE #5
         .BYTE    0           ;
         .BYTE   161         ;WRITE COMMAND
         .BYTE    0           ;
         .WORD   -1000.      ;WORD COUNT

```

1084	.WORD	WRTBUF	:BUFFER ADDRESS
1085	.BYTE	3	:SECTOR
1086	.BYTE	5	:TRACK
1087	.WORD	400	:CYLINDER
1088	.WORD	ERRTB5	:ERROR TABLE
1089	.WORD	0	:STATUS/ERROR INDICATOR

ALTERNATE DPB SETUP

1091	WRTDPB: .WORD	5	:THIS SETUP ACHIEVED
1092	.WORD	WRITE	:EVERYTHING THE
1093	.WORD	-1000.	:ABOVE TABLE DID, BUT
1094	.WORD	WRTBUF	:IN A CLEANER FORMAT
1095	.BYTE	3,5	
1096	.WORD	400,ERRTB5,0	

13.5 RH/RM REGISTERS

	<u>MNEMONIC</u>	<u>INDEX</u>
1100		
1101		
1102		
1103		
1104		
1105	RMCS1	0
1106	RMWC	2
1107	RMBA	4
1108	RMDA	6
1109	RMCS2	10
1110	RMDS	12
1111	RMER1	14
1112	RMAS	16
1113	RMLA	20
1114	RMDB	22
1115	RMMR1	24
1116	RMDT	26
1117	RMSN	30
1118	RMOF	32
1119	RMDC	34
1120	RMHR	36
1121	RMMR2	40
1122	RMER2	42
1123	RMEC1	44
1124	RMEC2	46

13.6 COMMANDS PERFORMED BY THE DRIVER

	<u>COMMAND</u>	<u>CODE</u>	<u>COMMAND TYPE</u>
1125			
1126			
1127			
1128			
1129			
1130			
1131	NO OPERATION	101	N
1132	UNLOAD	103	N
1133	SEEK	105	P
1134	RECALIRATE	107	P
1135	DRIVE CLEAR	111	P
1136	RELEASE	113	N
1137	OFFSET	115	P
1138	RETURN TO CENTER	117	P
1139	READIN PRESET	121	P
1140	PACK ACKNOWLEDGE	123	N

1141	SEARCH	131	P
1142	GET REGISTER(S)	141	S
1143	SET FORMAT	143	S
1144	SELECT DRIVE	145	S
1145	WRITE CHECK DATA	151	D
1146	WRITE CHK HEADER & DATA	153	D
1147	WRITE DATA	161	D
1148	WRITE HEADER & DATA	163	D
1149	READ DATA	171	D
1150	READ HEADER & DATA	173	D

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

13.7 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>
1164	15	ERROR OCCURRED DONE (BIT07=0); BITS 14-10 SPECIFIES TYPE DONE (BIT07=1); BITS 06-03 SPECIFIES TYPE
1165	14(1)	USER MADE A REQUEST FOR A FUNCTION TO BE PERFORMED ON AN OFFLINE OR UNSAFE DRIVE
1166	13(1)	USER MADE A REQUEST FOR A FUNCTION TO BE PERFORMED ON A DRIVE THAT HAS AN UNLOAD REQUEST IN QUEUE.
1167	12(2)	PERSISTENT UNSAFE CONDITION EXIST.
1168	11(2)	UNCORRECTABLE PARITY ERROR OCCURRED
1169	10(2)(4)	FATAL PARITY ERROR. A MASSBUS CLEAR WAS PERFORMED, ALL QUEUES WERE EMPTIED, AND ALL DRVACT'S SET TO THE IDLE STATE
1170	9(3)(4)	SOFTWARE TIMEOUT OCCURRED ON THIS DRIVE
1171	8(4)	SOFTWARE TIMEOUT OCCURRED ON ANOTHER DRIVE
1172	7	DONE
1173	6(2)	ERROR OCCURRED DURING AN I/O OPERATION
1174	5(2)	ERROR OCCURRED DURING AN OPERATION OTHER THAN I/O.
1175	4(2)	CORRECTABLE UNSAFE CONDITION OCCURRED

1197

1198		
1199		
1200	3(2)	DRIVE ERROR OCCURRED THAT CAUSED AN AUTOMATIC 'RECALIBRATE' SEQUENCE
1201		
1202	2	PORT REQUEST TIMEOUT. THE DRIVER REQUESTED THE DRIVE BUT THE OPPOSITE PORT DID NOT RELEASE THE DRIVE WITHIN 20 SECONDS.
1203		
1204		
1205		
1206	1	NON-EXISTENT DRIVE REQUESTED. USER MADE A REQUEST FOR A NON-EXISTENT DRIVE.
1207		
1208		
1209	(1) =>	REQUEST WASN'T PUT IN QUEUE. (RH/RM REGISTERS WERE NOT SAVED)
1210		
1211		
1212	(2) =>	REQUEST QUEUE HAS BEEN EMPTIED. THE DRIVER ISSUED A 'DRIVE CLEAR' TO THE DRIVE. NOTE: ALL RH/RM REGISTERS ARE SAVED AS PER DPB+14 BEFORE THE 'DRIVE CLEAR'.
1213		
1214		
1215		
1216		
1217	(3) =>	REQUEST QUEUE HAS BEEN EMPTIED. THE DRIVER ISSUED A MASSBUS INIT. ALL RH/RM REGISTERS FOR THE DRIVE WERE SAVED AS PER DPB+14 BEFORE THE INIT.
1218		
1219		
1220		
1221		
1222	(4) =>	A 'RECALIBRATE' SHOULD BE ISSUED BEFORE ANY OTHER COMMAND.
1223		
1224		

13.8 ERROR CALLS MADE BY THE DRIVER.

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

N	TYPE	DATA AVAILABLE
-	----	-----
1	RH70 INTERRUPT OCCURRED (RHAS=0)	*R4= RMCS1'S ADDRESS
2	UNEXPECTED ATTENTION OCCURRED	R1= DRIVE NUMBER R3= ATA BIT *R4= RMCS1'S ADDRESS R5= (RMAS) RMERRS =RMDS RMERRS+2=RMER1 RMERRS+4=RMER2 RMERRS+6=RMMR2
3	MASSBUS PARITY ERROR (MCPE=1)	RD.ADR= ADDRESS OF REG. READ RD.WRD= WORD READ
4	MASSBUS PARITY ERROR (PAR=1)	WRT.AD= ADDRESS OF REG. WRITTEN WRT.WD= WORD WRITTEN RD.WRD= WORD READ BACK

1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267

5 ADDRESS PLUG CHANGE R1= DRIVE NUMBER
 BIT SET ('OPE' ERROR) R3= ATA BIT
 *R4= RMCS1'S ADDRESS
 R5= (RMAS)
 RMERRS =RMDS
 RMERRS+2=RMER1
 RMERRS+4=RMER2
 RMERRS+6=RMMR2

* THIS IS THE ACTUAL UNIBUS ADDRESS (176700)

@

1
62
63

```

:PROGRAM REVISION #001

.TITLE CZRMTAO RM05/3/2 DR CMPT TST
:*COPYRIGHT (C) 1980
:*DIGITAL EQUIPMENT CORP.
:*MAYNARD, MASS. 01754
:*
:*PROGRAM BY MIKE LEAVITT
:*
:*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
:*PACKAGE (MAINDEC-11-DZQAC-C4), 1980.

```

64

```

.SBTTL OPERATIONAL SWITCH SETTINGS
:*
:*      SWITCH          USE
:*      -----          -
:*      15             HALT ON ERROR
:*      14             LOOP ON TEST
:*      13             INHIBIT ERROR TYPEOUTS
:*      12             INHIBIT TRACE TRAP
:*      11             INHIBIT ITERATIONS
:*      10             BELL ON ERROR
:*      9              LOOP ON ERROR
:*      8              LOOP ON TEST IN SWR<7:0>
:*      7              TYPE THE BAD SECTOR FILE

```

65
66
67

```

.SBTTL BASIC DEFINITIONS

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

```

001100
104000
000004

```

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

```

000011
000012
000015
000200
177776
177776
177774
177772
177570
177570

```

:*GENERAL PURPOSE REGISTER DEFINITIONS
R0 = %0                ::GENERAL REGISTER
R1 = %1                ::GENERAL REGISTER
R2 = %2                ::GENERAL REGISTER
R3 = %3                ::GENERAL REGISTER
R4 = %4                ::GENERAL REGISTER
R5 = %5                ::GENERAL REGISTER
R6 = %6                ::GENERAL REGISTER
R7 = %7                ::GENERAL REGISTER
SP = %6                ::STACK POINTER
PC = %7                ::PROGRAM COUNTER

```

000000
000001
000002
000003
000004
000005
000006
000007
000006
000007

```
000000      : *PRIORITY LEVEL DEFINITIONS
000040      PR0      = 0          :: PRIORITY LEVEL 0
000100      PR1      = 40         :: PRIORITY LEVEL 1
000140      PR2      = 100        :: PRIORITY LEVEL 2
000200      PR3      = 140        :: PRIORITY LEVEL 3
000240      PR4      = 200        :: PRIORITY LEVEL 4
000300      PR5      = 240        :: PRIORITY LEVEL 5
000340      PR6      = 300        :: PRIORITY LEVEL 6
          PR7      = 340        :: PRIORITY LEVEL 7
```

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

```
100000      : *DATA BIT DEFINITIONS (BIT00 TO BIT15)
040000      BIT15    = 100000
020000      BIT14    = 40000
010000      BIT13    = 20000
004000      BIT12    = 10000
002000      BIT11    = 4000
001000      BIT10    = 2000
000400      BIT09    = 1000
000200      BIT08    = 400
000100      BIT07    = 200
000040      BIT06    = 100
000020      BIT05    = 40
000010      BIT04    = 20
000004      BIT03    = 10
000002      BIT02    = 4
000001      BIT01    = 2
          BIT00    = 1
001000      BIT9=BIT09
000400      BIT8=BIT08
```

000200 BIT7=BIT07
000100 BIT6=BIT06
000040 BIT5=BIT05
000020 BIT4=BIT04
000010 BIT3=BIT03
000004 BIT2=BIT02
000002 BIT1=BIT01
000001 BIT0=BIT00

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

68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102

.SBTTL RM REGISTERS

;INDEX OF STATUS AND REGISTER WORDS RELATIVE TO FMTDPB

000016 \$STATUS = 16
000020 \$RMCS1 = \$STATUS+2
000022 \$RMWC = \$RMCS1+2
000024 \$RMBA = \$RMWC+2
000026 \$RMDA = \$RMBA+2
000030 \$RMCS2 = \$RMDA+2
000032 \$RMDS = \$RMCS2+2
000034 \$RMER1 = \$RMDS+2
000036 \$RMAS = \$RMER1+2
000040 \$RMLA = \$RMAS+2
000042 \$RMDB = \$RMLA+2
000044 \$RMMR1 = \$RMDB+2
000046 \$RMDT = \$RMMR1+2
000050 \$RMSN = \$RMDT+2
000052 \$RMOF = \$RMSN+2
000054 \$RMDC = \$RMOF+2
000056 \$RMHR = \$RMDC+2
000060 \$RMMR2 = \$RMHR+2
000062 \$RMER2 = \$RMMR2+2
000064 \$RMEC1 = \$RMER2+2
000066 \$RMEC2 = \$RMEC1+2

.SBTTL RM DRIVER COMMANDS

000101 RNOP = 101 ;NO OPERATION
000105 SEEK = 105 ;SEEK
000107 RECAL = 107 ;RECALIBRATE
000111 DRVCLR = 111 ;DRIVE CLEAR
000113 RELSE = 113 ;RELEASE
000115 OFFSET = 115 ;OFFSET

103	000117	RTC	= 117	:RETURN TO CENTER LINE
104	000121	READIN	= 121	:READ IN PRESET
105	000123	ACK	= 123	:PACK ACKNOWLEDGE
106	000131	SEARCH	= 131	:SEARCH
107	000141	GETREG	= 141	:GET REGISTERS
108	000143	SETFMT	= 143	:SET FORMAT (& ECI OR HCI)
109	000145	SELDRV	= 145	:SELECT DRIVE
110	000151	WCKD	= 151	:WRITE CHECK DATA
111	000153	WCKHD	= 153	:WRITE CHECK HEADER & DATA
112	000161	WRTDAT	= 161	:WRITE DATA
113	000163	WRTHD	= 163	:WRITE HEADER & DATA
114	000171	RDDAT	= 171	:READ DATA
115	000173	RDHD	= 173	:READ HEADER & DATA

117	176700	ABASE	= 176700
118	000254	AVECT1	= 254

121
122 .SBTTL TRAP CATCHER

000000

. = 0
:*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
:*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
:*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS

000174	000174	000000
000176	000000	000000

. = 174
DISPREG: .WORD 0 ;:SOFTWARE DISPLAY REGISTER
SWREG: .WORD 0 ;:SOFTWARE SWITCH REGISTER

.SBTTL STARTING ADDRESS(ES)

123	000200	000137	005502	JMP @#START	:;JUMP TO STARTING ADDRESS OF PROGRAM
124	000204	000137	005520	JMP @#START1	:CHANGE THE RH/RM UNIBUS ADDRESS
125					
126	000210	000137	005536	JMP @#START2	:SECOND PASS STARTING ADDRESS

127
128
129

.SBTTL ACT11 HOOKS

:;*****
:;HOOKS REQUIRED BY ACT11
\$SVPC=. ;SAVE PC
. = 46
\$ENDAD ;:1)SET LOC.46 TO ADDRESS OF \$ENDAD IN .SEOP
. = 52
.WORD 40000 ;:2)SET LOC.52 TO 40000
.\$SVPC ;: RESTORE PC

130		000214
131		001100
132		

. = 1100
.SBTTL APT PARAMETER BLOCK

:;*****
:;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
:;*****
.\$X=. ;:SAVE CURRENT LOCATION
. = 24 ;:SET POWER FAIL TO POINT TO START OF PROGRAM
200 ;:FOR APT START UP

000024	001100	000024	000200
--------	--------	--------	--------

000044 000044
001100 001100
001100 001100

. = 44 :: POINT TO APT INDIRECT ADDRESS PNTR.
\$APTHDR :: POINT TO APT HEADER BLOCK
. = \$X :: RESET LOCATION COUNTER

::*****
: SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
: INTERFACE SPEC.

001100
001100 000000
001102 001212
001104 000764
001106 000764
001110 000764
001112 000032
001114

\$APTHD:
\$HIBTS: .WORD 0 :: TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
\$MBADR: .WORD \$MAIL :: ADDRESS OF APT MAILBOX (BITS 0-15)
\$STMT: .WORD 500. :: RUN TIME OF LONGEST TEST
\$PASTM: .WORD 500. :: RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
\$UNITM: .WORD 500. :: ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDED UNIT
\$ETEND-\$MAIL/2 :: LENGTH MAILBOX-ETABLE (WORDS)
TAB.XY=.

133
134

0

.SBTTL COMMON TAGS

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

001114	001114			\$CMTAG: . =TAB.XY	:::START OF COMMON TAGS
001114	000000			.WORD 0	
001116	000			\$TSTNM: .BYTE 0	:::CONTAINS THE TEST NUMBER
001117	000			\$ERFLG: .BYTE 0	:::CONTAINS ERROR FLAG
001120	000000			\$ICNT: .WORD 0	:::CONTAINS SUBTEST ITERATION COUNT
001122	000000			\$LPADR: .WORD 0	:::CONTAINS SCOPE LOOP ADDRESS
001124	000000			\$LPERR: .WORD 0	:::CONTAINS SCOPE RETURN FOR ERRORS
001126	000000			\$ERTTL: .WORD 0	:::CONTAINS TOTAL ERRORS DETECTED
001130	000			\$ITEMB: .BYTE 0	:::CONTAINS ITEM CONTROL BYTE
001131	001			\$ERMAX: .BYTE 1	:::CONTAINS MAX. ERRORS PER TEST
001132	000000			\$ERRPC: .WORD 0	:::CONTAINS PC OF LAST ERROR INSTRUCTION
001134	000000			\$GDADR: .WORD 0	:::CONTAINS ADDRESS OF 'GOOD' DATA
001136	000000			\$BDADR: .WORD 0	:::CONTAINS ADDRESS OF 'BAD' DATA
001140	000000			\$GDDAT: .WORD 0	:::CONTAINS 'GOOD' DATA
001142	000000			\$BDDAT: .WORD 0	:::CONTAINS 'BAD' DATA
001144	000000			.WORD 0	:::RESERVED--NOT TO BE USED
001146	000000			.WORD 0	
001150	000			\$AUTOB: .BYTE 0	:::AUTOMATIC MODE INDICATOR
001151	000			\$INTAG: .BYTE 0	:::INTERRUPT MODE INDICATOR
001152	000000			.WORD 0	
001154	177570			\$SWR: .WORD DSWR	:::ADDRESS OF SWITCH REGISTER
001156	177570			\$DISPLAY: .WORD DDISP	:::ADDRESS OF DISPLAY REGISTER
001160	177560			\$TKS: 177560	:::TTY KBD STATUS
001162	177562			\$TKB: 177562	:::TTY KBD BUFFER
001164	177564			\$TPS: 177564	:::TTY PRINTER STATUS REG. ADDRESS
001166	177566			\$TPB: 177566	:::TTY PRINTER BUFFER REG. ADDRESS
001170	000			\$NULL: .BYTE 0	:::CONTAINS NULL CHARACTER FOR FILLS
001171	002			\$FILLS: .BYTE 2	:::CONTAINS # OF FILLER CHARACTERS REQUIRED
001172	012			\$FILLC: .BYTE 12	:::INSERT FILL CHARS. AFTER A 'LINE FEED'
001173	000			\$TPFLG: .BYTE 0	:::'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
001174	000000			\$TMP0: .WORD 0	:::USER DEFINED
001176	000000			\$TIMES: 0	:::MAX. NUMBER OF ITERATIONS
001200	000000			\$ESCAPE: 0	:::ESCAPE ON ERROR ADDRESS
001202	207	377	377	\$BELL: .ASCIZ <207><377><377>	:::CODE FOR BELL
001206	077			\$QUES: .ASCII /?/	:::QUESTION MARK
001207	015			\$CRLF: .ASCII <15>	:::CARRIAGE RETURN
001210	012	000		\$LF: .ASCIZ <12>	:::LINE FEED

::*****
.SBTTL APT MAILBOX-ETABLE

001212				.EVEN	
001212	000000			\$MAIL: .	:::APT MAILBOX
001214	000000			\$MSGTY: .WORD AMSGTY	:::MESSAGE TYPE CODE
001216	000000			\$FATAL: .WORD AFATAL	:::FATAL ERROR NUMBER
001220	000000			\$TESTN: .WORD ATESTN	:::TEST NUMBER
001222	000000			\$PASS: .WORD APASS	:::PASS COUNT
001224	000000			\$DEVCT: .WORD ADEVCT	:::DEVICE COUNT
001226	000000			\$UNIT: .WORD AUNIT	:::I/O UNIT NUMBER
				\$MSGAD: .WORD AMSGAD	:::MESSAGE ADDRESS

001230	000000	\$MSGLG: .WORD	AMSGLG	::MESSAGE LENGTH
001232		\$ETABLE:		::APT ENVIRONMENT TABLE
001232	000	\$ENV: .BYTE	AENV	::ENVIRONMENT BYTE
001233	000	\$ENVM: .BYTE	AENVM	::ENVIRONMENT MODE BITS
001234	000000	\$SWREG: .WORD	ASWREG	::APT SWITCH REGISTER
001236	000000	\$USWR: .WORD	AUSWR	::USER SWITCHES
001240	000000	\$CPUOP: .WORD	ACPUOP	::CPU TYPE,OPTIONS
		:*		BITS 15-11=CPU TYPE
		:*		11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
		:*		11/70=06,PDQ=07,Q=10
		:*		BIT 10=REAL TIME CLOCK
		:*		BIT 9=FLOATING POINT PROCESSOR
		:*		BIT 8=MEMORY MANAGEMENT
001242	000	\$MAMS1: .BYTE	AMAMS1	::HIGH ADDRESS,M.S. BYTE
001243	000	\$MTYP1: .BYTE	AMTYP1	::MEM. TYPE,BLK#1
		:*		MEM.TYPE BYTE -- (HIGH BYTE)
		:*		900 NSEC CORE=001
		:*		300 NSEC BIPOLAR=002
		:*		500 NSEC MOS=003
001244	000000	\$MADR1: .WORD	AMADR1	::HIGH ADDRESS,BLK#1
		:*		MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF 'TYPE' ABOVE
001246	000	\$MAMS2: .BYTE	AMAMS2	::HIGH ADDRESS,M.S. BYTE
001247	000	\$MTYP2: .BYTE	AMTYP2	::MEM. TYPE,BLK#2
001250	000000	\$MADR2: .WORD	AMADR2	::MEM.LAST ADDRESS,BLK#2
001252	000	\$MAMS3: .BYTE	AMAMS3	::HIGH ADDRESS,M.S.BYTE
001253	000	\$MTYP3: .BYTE	AMTYP3	::MEM. TYPE,BLK#3
001254	000000	\$MADR3: .WORD	AMADR3	::MEM.LAST ADDRESS,BLK#3
001256	000	\$MAMS4: .BYTE	AMAMS4	::HIGH ADDRESS,M.S.BYTE
001257	000	\$MTYP4: .BYTE	AMTYP4	::MEM. TYPE,BLK#4
001260	000000	\$MADR4: .WORD	AMADR4	::MEM.LAST ADDRESS,BLK#4
001262	000254	\$VECT1: .WORD	AVECT1	::INTERRUPT VECTOR#1,BUS PRIORITY#1
001264	000000	\$VECT2: .WORD	AVECT2	::INTERRUPT VECTOR#2BUS PRIORITY#2
001266	176700	\$BASE: .WORD	ABASE	::BASE ADDRESS OF EQUIPMENT UNDER TEST
001270	000000	\$DEVM: .WORD	ADEVN	::DEVICE MAP
001272	000000	\$CDW1: .WORD	ACDW1	::CONTROLLER DESCRIPTION WORD#1
001274	000000	\$CDW2: .WORD	ACDW2	::CONTROLLER DESCRIPTION WORD#2
001276		\$ETEND:		
		.MEXIT		

0

.SBTTL USER DEFINED TAGS

001276	176700	\$RMADR:	.WORD	176700	:FIRST ADDRESS OF RH/RM REGISTERS
001300	000254	\$RMVEC:	.WORD	254	:VECTOR ADDRESS
001302	172540	\$LKCSR:	.WORD	172540	:ADDR OF KW11-P STATUS REGISTER
001304	172542	\$LKCSB:	.WORD	172542	:ADDR OF KW11-P COUNTER BUFFER
001306	000104	\$LPVEC:	.WORD	104	:ADDR OF KW11-P VECTOR
001310	177546	\$LKS:	.WORD	177546	:ADDR OF KW11-L STATUS REGISTER
001312	000100	\$LLVEC:	.WORD	100	:ADDR OF KW11-L VECTOR
001314	177777	PCLOCK:	.WORD	-1	: '0' IF KW11-P IS ON SYSTEM
001316	177777	CLKFLG:	.WORD	-1	: '0' IF A CLOCK IS AVAILABLE
001320	000074	HZ:	.WORD	74	:74 (8) IF 60 HZ SYSTEM; 62 (8) IF 50 HZ SYSTEM
001322	000000	STATIN:	.WORD	0	: 'TYPE STATISTICS' INDICATOR
001324	000000	PACK:	.WORD	0	:ENTRY TO THE TABLE D
	001224	DRIVE	= \$UNIT		:DRIVE # STORAGE:
001326	000000	ATTN:	.WORD	0	:ATTN REG STORAGE:
001330	000000	UNIT:	.WORD	0	:DRIVE # STORAGE FOR PRINTOUT
					:RETRY COUNT IN THE UPPER BYTE
001332	000000	LSTAD:	.WORD	0	:STORE LAST MEMORY ADDRESS HERE
001334	000000	CHGADR:	.WORD	0	:CHANGE RH/RM UNIBUS ADDRESS FLAG
001336	000000	CFLAG:	.WORD	0	: 'CONTROL C' FLAG
001340	000000	TSTNM:	.WORD	0	:TEST NUMBER FOR PRINT AND SCORE RT.
001342	000000	BADSEC:	.WORD	0	:BAD SECTOR/TRACK FLAG
001344	000000	HOUR:	.WORD	0	:HOUR COUNT STORED HERE (MAXIMUM - 999.)
001346	000000	MINUTE:	.WORD	0	:MINUTE'S COUNT STORED HERE
001350	000000	SECOND:	.WORD	0	:SECOND'S COUNT STORED HERE
001352	000000	SIXTEE:	.WORD	0	:TIMER ROUTINE COUNTER (FOR ONE SECOND)
001354	000000	CMCNT:	.WORD	0	:ZONE COUNT
001356	000000	CMCYL:	.WORD	0	:CYLINDER ADDRESS
001360	000000	STARSC:	.WORD	0	:STARTING SECTOR (FOR TEST 6,8)
001362	000000	CMSEC:	.WORD	0	:DALTA CYLINDER COUNT
001364	000000	CMTRK:	.WORD	0	:TRACK ADDRESS
001366	000000	NULINE:	.WORD	0	:NEW LINE FLAG AND COLUMN CTR
001370	000037	SECLMT:	.WORD	31.	:SECTOR ADDRESS LIMIT
001372	000000	TRKLMT:	.WORD	0	:TRACK ADDRESS LIMIT, RM03/2 = 4. AND RM05 = 18.
001374	001466	CYLIMT:	.WORD	822.	:CYLINDER ADDRESS LIMIT
001376	000000	FAULT:	.WORD	0	: =1, IF ALL DRIVES NOT COMPATIBLE
001400	000000	RSTART:	.WORD	0	:CONTAINS PROGRAM RESTARTING ADDRESS
001402	000000	DTYP:	.WORD	0	:CONTAINS DRIVE TYPE CODE OF DRIVE BEING TESTED
001404	000000	XXDP:	.WORD	0	:THE LOW BYTE CONTAINS THE DRIVE NUMBER FROM WHICH :THE PROGRAM WAS LOADED. THE HIGH BYTE CONTAINS THE : 'XXDP' DEVICE CODE THE RM05/3/2.

.SBTTL TABLES, CONSTANTS, AND VARIABLE LOCATIONS

:TABLE D
 :TABLE LISTED BELOW SPECIFIES THE SECTORS TO BE WRITTEN
 :BY A LOGICAL DRIVE. EACH LOGICAL DRIVE WRITES TWO SECTORS ON ONE
 :CYLINDER, 16 CYLINDERS IN ONE BLOCK, 2 BLOCKS IN ONE WRITE-CURRENT
 :ZONE AND 7 CURRENT ZONES IN A PACK.

001406	000	017	015	LOG0:	.BYTE	0,15,13,10,6,1,11,4,12,13,9,14,2,5,7,8.
001426	001	000	016	LOG1:	.BYTE	1,0,14,11,7,2,12,5,13,4,10,15,3,6,8,9.
001446	002	001	017	LOG2:	.BYTE	2,1,15,12,8,3,13,6,14,5,11,0,4,7,9,10.
001466	003	002	000	LOG3:	.BYTE	3,2,0,13,9,4,14,7,15,6,12,1,5,8,10,11.
001506	004	003	001	LOG4:	.BYTE	4,3,1,14,10,5,15,8,0,7,13,2,6,9,11,12.
001526	005	004	002	LOG5:	.BYTE	5,4,2,15,11,6,0,9,1,8,14,3,7,10,12,13.


```
003116 000 000 000 OVWP15: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 15
003136 000 000 000 OVWP16: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 16
003156 000 000 000 OVWP17: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 17
003176 000 000 000 OVWP18: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 18
```

;TABLE OF READ SCORE,NEGATIVE OFFSET SCORE

```
003216 000 000 000 RDN0: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 0
003236 000 000 000 RDN1: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 1
003256 000 000 000 RDN2: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 2
003276 000 000 000 RDN3: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 3
003316 000 000 000 RDN4: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 4
003336 000 000 000 RDN5: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 5
003356 000 000 000 RDN6: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 6
003376 000 000 000 RDN7: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 7
003416 000 000 000 RDN8: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 8
003436 000 000 000 RDN9: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 9
003456 000 000 000 RDN10: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 10
003476 000 000 000 RDN11: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 11
003516 000 000 000 RDN12: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 12
003536 000 000 000 RDN13: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 13
003556 000 000 000 RDN14: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 14
003576 000 000 000 RDN15: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 15
003616 000 000 000 RDN16: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 16
003636 000 000 000 RDN17: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 17
003656 000 000 000 RDN18: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 18
```

;TABLE OF READ SCORE,POSITIVE OFFSET SCORE

```
003676 000 000 000 RDP0: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 0
003716 000 000 000 RDP1: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 1
003736 000 000 000 RDP2: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 2
003756 000 000 000 RDP3: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 3
003776 000 000 000 RDP4: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 4
004016 000 000 000 RDP5: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 5
004036 000 000 000 RDP6: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 6
004056 000 000 000 RDP7: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 7
004076 000 000 000 RDP8: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 8
004116 000 000 000 RDP9: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 9
004136 000 000 000 RDP10: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 10
004156 000 000 000 RDP11: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 11
004176 000 000 000 RDP12: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 12
004216 000 000 000 RDP13: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 13
004236 000 000 000 RDP14: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 14
004256 000 000 000 RDP15: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 15
004276 000 000 000 RDP16: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 16
004316 000 000 000 RDP17: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 17
004336 000 000 000 RDP18: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :TRACK 18
```

;TABLE OF SELF TEST SCORE

```
004356 000 000 SELF0: .BYTE 0,0 :TRACK 0
004360 000 000 SELF1: .BYTE 0,0 :TRACK 1
004362 000 000 SELF2: .BYTE 0,0 :TRACK 2
004364 000 000 SELF3: .BYTE 0,0 :TRACK 3
004366 000 000 SELF4: .BYTE 0,0 :TRACK 4
004370 000 000 SELF5: .BYTE 0,0 :TRACK 5
```

```

004372 000 000 SELF6: .BYTE 0,0 ;TRACK 6
004374 000 000 SELF7: .BYTE 0,0 ;TRACK 7
004376 000 000 SELF8: .BYTE 0,0 ;TRACK 8
004400 000 000 SELF9: .BYTE 0,0 ;TRACK 9
004402 000 000 SELF10: .BYTE 0,0 ;TRACK 10
004404 000 000 SELF11: .BYTE 0,0 ;TRACK 11
004406 000 000 SELF12: .BYTE 0,0 ;TRACK 12
004410 000 000 SELF13: .BYTE 0,0 ;TRACK 13
004412 000 000 SELF14: .BYTE 0,0 ;TRACK 14
004414 000 000 SELF15: .BYTE 0,0 ;TRACK 15
004416 000 000 SELF16: .BYTE 0,0 ;TRACK 16
004420 000 000 SELF17: .BYTE 0,0 ;TRACK 17
004422 000 000 SELF18: .BYTE 0,0 ;TRACK 18
    
```

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

```

004424 000 001 003 INDST: .BYTE 0,1,3,6,10.,15.,5,12.,4,13.,7,2,14.,11.,9.,8.
    
```

```

004444 BLKADR:
004444 004622 .WORD DRIV0 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 0
004446 004644 .WORD DRIV1 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 1
004450 004666 .WORD DRIV2 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 2
004452 004710 .WORD DRIV3 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 3
004454 004732 .WORD DRIV4 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 4
004456 004754 .WORD DRIV5 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 5
004460 004776 .WORD DRIV6 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 6
004462 005020 .WORD DRIV7 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 7
004464 005042 .WORD DRIV10 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 10
004466 005064 .WORD DRIV11 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 11
004470 005106 .WORD DRIV12 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 12
004472 005130 .WORD DRIV13 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 13
004474 005152 .WORD DRIV14 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 14
004476 005174 .WORD DRIV15 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 15
004500 005216 .WORD DRIV16 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 16
004502 005240 .WORD DRIV17 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 17
    
```

```

004504 000000 OFFCOD: .WORD 0 ;OFFSET CODE TABLE
;NUMBER FOR NEGATIVE OFFSET (DIR = OUT)
;NUMBER FOR POSITIVE OFFSET (DIR = IN)
    
```

;DATA/PARAMETER BLOCK

```

004506 000 FMTDPB: .BYTE 0 ;DRIVER PARAMETER BLOCK, DRIVE #
004507 000 .BYTE 0 ;OFFSET VALUE OR FMT16, HCI OR ECI
004510 000 .BYTE 0 ;COMMAND CODE
004511 000 .BYTE 0 ;PSEL, A16 AND A17
004512 000000 .WORD 0 ;WORD COUNT (NEG)
004514 043032 .WORD BUFFER ;BUFFER ADDRESS
004516 000 .BYTE 0 ;SECTOR ADDRESS
004517 000 .BYTE 0 ;TRACK ADDRESS
004520 000000 .WORD 0 ;CYLINDER ADDRESS
004522 004526 .WORD RM.REG ;ADDRESS TO SAVE ALL RH/RM REG'S
004524 000000 .WORD 0 ;STATUS WORD
    
```

```

004526 000000 RM.REG: .WORD 0 ;RMCS1
004530 000000 .WORD 0 ;RMWC
004532 000000 .WORD 0 ;RMBA
004534 000000 .WORD 0 ;RMDA
004536 000000 .WORD 0 ;RMCS2
004540 000000 .WORD 0 ;RMDS
004542 000000 .WORD 0 ;RMER1
004544 000000 .WORD 0 ;RMAS
004546 000000 .WORD 0 ;RMLA
004550 000000 .WORD 0 ;RMDB
004552 000000 .WORD 0 ;RMMR1
004554 000000 .WORD 0 ;RMDT
004556 000000 .WORD 0 ;RMSN
004560 000000 .WORD 0 ;RMOF
004562 000000 .WORD 0 ;RMCA
004564 000000 .WORD 0 ;RMDC
004566 000000 .WORD 0 ;RMER2
004570 000000 .WORD 0 ;RMMR2
004572 000000 .WORD 0 ;RMEC1
004574 000000 .WORD 0 ;RMEC2
    
```

:GENERAL PURPOSE PARAMETER BLOCK

```

004576 000 GENDPB: .BYTE 0 ;DRIVER PARAMETER BLOCK, DRIVE #
004577 000 .BYTE 0 ;OFFSET VALUE OR FMT16, HCI OR ECI
004600 000 .BYTE 0 ;COMMAND CODE
004601 000 .BYTE 0 ;PSEL, A16 AND A17
004602 177776 .WORD -2 ;WORD COUNT (NEG)
004604 004616 .WORD CYLNDR ;BUFFER ADDRESS
004606 000 .BYTE 0 ;SECTOR ADDRESS
004607 000 .BYTE 0 ;TRACK ADDRESS
004610 000000 .WORD 0 ;CYLINDER ADDRESS
004612 004526 .WORD RM.REG ;ADDRESS TO SAVE ALL RH/RM REG'S
004614 000000 .WORD 0 ;STATUS WORD
    
```

```

004616 CYLNDR: .BLKW 2 ;BUFFER
    
```

:HISTORY FILE FOR 16. LOGICAL DRIVES:(0-17 OCTAL)

```

000001 $FMT = 1
000002 $COMND = $FMT+1 ;COMMAND CODE
000003 $PSEL = $FMT+2 ;PROT SELECT AND A16,A17
000004 $WRDM = $FMT+3 ;WORD COUNT
000006 $BUF = $FMT+5 ;BUFFER ADDRESS
000010 $SEC = $FMT+7 ;SECTOR ADDRESS
000011 $TRK = $FMT+10 ;TRACK ADDRESS
000012 $CYL = $FMT+11 ;CYLINDER ADDRESS
000014 $$SYSNM = $FMT+13 ;SUB SYSTEM A-H
000015 $PHYDR = $FMT+14 ;PHYSICAL DRIVE CODE (ASCII )
000016 $GAP = $FMT+15 ;LEFT TWO NULL BYTES
000022 $EMTAB = $GAP+4 ;END OF HISTORY TABLE
    
```

```

004622 000 000 DRIV0: .BYTE 7&0,0 ;HISTORY BLOCK OF LOGICAL DRIVE 0
004644 001 000 .BLKB $EMTAB-$COMND
004666 002 000 DRIV1: .BYTE 7&1,0 ;HISTORY BLOCK OF LOGICAL DRIVE 1
        .BLKB $EMTAB-$COMND
        DRIV2: .BYTE 7&2,0 ;HISTORY BLOCK OF LOGICAL DRIVE 2
    
```

004710	003	000	.BLKB \$EMTAB-\$COMND DRIV3: .BYTE 7&3,0	;HISTORY BLOCK OF LOGICAL DRIVE 3
004732	004	000	.BLKB \$EMTAB-\$COMND DRIV4: .BYTE 7&4,0	;HISTORY BLOCK OF LOGICAL DRIVE 4
004754	005	000	.BLKB \$EMTAB-\$COMND DRIV5: .BYTE 7&5,0	;HISTORY BLOCK OF LOGICAL DRIVE 5
004776	006	000	.BLKB \$EMTAB-\$COMND DRIV6: .BYTE 7&6,0	;HISTORY BLOCK OF LOGICAL DRIVE 6
005020	007	000	.BLKB \$EMTAB-\$COMND DRIV7: .BYTE 7&7,0	;HISTORY BLOCK OF LOGICAL DRIVE 7
005042	000	000	.BLKB \$EMTAB-\$COMND DRIV10: .BYTE 7&10,0	;HISTORY BLOCK OF LOGICAL DRIVE 10
005064	001	000	.BLKB \$EMTAB-\$COMND DRIV11: .BYTE 7&11,0	;HISTORY BLOCK OF LOGICAL DRIVE 11
005106	002	000	.BLKB \$EMTAB-\$COMND DRIV12: .BYTE 7&12,0	;HISTORY BLOCK OF LOGICAL DRIVE 12
005130	003	000	.BLKB \$EMTAB-\$COMND DRIV13: .BYTE 7&13,0	;HISTORY BLOCK OF LOGICAL DRIVE 13
005152	004	000	.BLKB \$EMTAB-\$COMND DRIV14: .BYTE 7&14,0	;HISTORY BLOCK OF LOGICAL DRIVE 14
005174	005	000	.BLKB \$EMTAB-\$COMND DRIV15: .BYTE 7&15,0	;HISTORY BLOCK OF LOGICAL DRIVE 15
005216	006	000	.BLKB \$EMTAB-\$COMND DRIV16: .BYTE 7&16,0	;HISTORY BLOCK OF LOGICAL DRIVE 16
005240	007	000	.BLKB \$EMTAB-\$COMND DRIV17: .BYTE 7&17,0	;HISTORY BLOCK OF LOGICAL DRIVE 17

;STANDARD DATA PATTERN

005262	066666	STNDAT: .WORD	066666
005264	155554	.WORD	155554
005266	133331	.WORD	133331
005270	066663	.WORD	066663
005272	155546	.WORD	155546
005274	133315	.WORD	133315
005276	066633	.WORD	066633
005300	155466	.WORD	155466
005302	133155	.WORD	133155
005304	066333	.WORD	066333
005306	154666	.WORD	154666
005310	131555	.WORD	131555
005312	063333	.WORD	063333
005314	146666	.WORD	146666
005316	115555	.WORD	115555
005320	033333	.WORD	033333

005322	040135	PSEUDO: .WORD	040135
005324	177070	.WORD	177070
005326	070414	.WORD	070414
005330	064531	.WORD	064531
005332	174473	.WORD	174473
005334	062422	.WORD	062422
005336	114352	.WORD	114352
005340	036620	.WORD	036620
005342	010031	.WORD	010031
005344	052336	.WORD	052336

005346	017310	.WORD	017310
005350	011347	.WORD	011347
005352	102367	.WORD	102367
005354	152567	.WORD	152567
005356	001246	.WORD	001246
005360	160073	.WORD	160073

0

.SBTTL ERROR POINTER TABLE

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

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

\$ERRTB:

;ERROR 1

005362
 1
 2
 3
 4 005362 036166
 5 005364 040570
 6 005366 041212
 7 005370 041336
 8
 9

EM1
 DH1
 DT1
 DF1

;RH70 INTERRUPT OCCURRED (RMAS = 0)

;ERROR 2

10
 11 005372 036227
 12 005374 040577
 13 005376 041216
 14 005400 041337
 15
 16

EM2
 DH2
 DT2
 DF2

;UNEXPECTED ATTENTION OCCURRED

;ERROR 3

17
 18 005402 036265
 19 005404 040661
 20 005406 041234
 21 005410 041345
 22
 23

EM3
 DH3
 DT3
 DF3

;MASSBUS PARITY ERROR (MCPE=1)

;ERROR 4

24
 25 005412 036323
 26 005414 040712
 27 005416 041244
 28 005420 041350
 29
 30

EM4
 DH4
 DT4
 DF4

;MASSBUS PARITY ERROR (PAR=1)

;ERROR 5

31
 32 005422 036360
 33 005424 040577
 34 005426 041216
 35 005430 041337
 36
 37

EM5
 DH2
 DT2
 DF2

;ADDRESS PLUG BIT CHANGED

;ERROR 6

38
 39 005432 036414
 40 005434 040754
 41 005436 041256
 42 005440 041336

EM6
 DH6
 DT6
 DF1

;RH DIDN'T RESPOND TO ADDRESSING

```
43
44
45      ;* ERRORS 7 - 12 ARE PART OF THE 'DUMP' SUBROUTINE
46      :ERROR 7
47
48 005442 000000
49      0
49 005444 040765      DH7
50 005446 041262      DT7
51 005450 000000      0
52
53      :ERROR 10
54
55 005452 000000
56      0
56 005454 041036      DH10
57 005456 041276      DT10
58 005460 000000      0
59
60      :ERROR 11
61
62 005462 000000
63      0
63 005464 041107      DH11
64 005466 041312      DT11
65 005470 000000      0
66
67      :ERROR 12
68
69 005472 000000
70      0
70 005474 041160      DH12
71 005476 041326      DT12
72 005500 000000      0
73
```

```

1          .SBTTL  START OF PROGRAM
2
3 005502  012737  005502  001400  START:  MOV    #.,RSTART      ;SETUP RESTART ADDRESS
4 005510  012737  000400  001334  MOV    #400,CHGADR    ;200 START ADDRESS FLAG
5 005516  000414                      BR     START3
6
7 005520  012737  005520  001400  START1: MOV   #.,RSTART      ;SETUP RESTART ADDRESS
8 005526  012737  177777  001334  MOV   #-1,CHGADR     ;204 START ADDRESS FLAG
9 005534  000405                      BR     START3
10
11 005536  012737  005536  001400  START2: MOV   #.,RSTART      ;SETUP RESTART ADDRESS
12 005544  005037  001334                      CLR   CHGADR         ;210 START ADDRESS FLAG
13
14 005550  000240                      START3: NOP
15 005552  005227  000000                      INC   #0              ;TTY LOOP, WAIT FOR INCREMENT
16 005556  001375                      BNE   #-4             ;OF WORD
17 005560  000005                      RESET                    ;CLEAR THE WORLD
18
19          .SBTTL  INITIALIZE THE COMMON TAGS
          ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
          MOV   #CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
          CLR   (R6)+          ;;CLEAR MEMORY LOCATION
          CMP   #SWR,R6        ;;DONE?
          BNE   #-6            ;;LOOP BACK IF NO
          MOV   #STACK,SP      ;;SETUP THE STACK POINTER
          ;;INITIALIZE A FEW VECTORS
          MOV   #SCOPE,@IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
          MOV   #340,@IOTVEC+2 ;;LEVEL 7
          MOV   #ERROR,@EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
          MOV   #340,@EMTVEC+2 ;;LEVEL 7
          MOV   #STRAP,@TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
          MOV   #340,@TRAPVEC+2;LEVEL 7
          MOV   #SPWRDN,@PWRVEC ;;POWER FAILURE VECTOR
          MOV   #340,@PWRVEC+2 ;;LEVEL 7
          MOV   SENDCT,$EOPCT  ;;SETUP END-OF-PROGRAM COUNTER
          CLR   $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
          CLR   $ESCAPE        ;;CLEAR THE ESCAPE ON ERROR ADDRESS
          MOVB  #1,$SERMAX     ;;ALLOW ONE ERROR PER TEST
          ;;INITIALIZE THE 'T-BIT' TRAP VECTOR. THEN LOAD LOCATION '$RTRN', IN
          ;;THE 'END-OF-PASS' ($EOP) ROUTINE, WITH A 'RTI' OR 'RTT'.
          MOV   #RTRN,@TBITVEC ;;SET 'T' BIT VECTOR TO $RTRN
          MOV   #340,@TBITVEC+2 ;;LEVEL 7
          MOV   #RTI,$RTRN     ;;SET $RTRN TO A RTI
          MOV   #65$,@RESVEC   ;;TRY TO DO A RTT
          CLR   -(SP)          ;;DUMMY PS
          MOV   #64$,-(SP)     ;;AND PC
          RTT                    ;;TRY THE RTT
          MOV   #RTT,$RTRN     ;;RTT IS LEGAL--SET $RTRN TO A RTT
          BR    66$
          MOV   #10,SP         ;;RTT ILLEGAL--CLEAN OFF THE STACK
          MOV   #RESVEC+2,@RESVEC ;;RESTORE TRAP CATCHER
          CLR   $TBIT          ;;CLEAR 'T' BIT SWITCH
          MOV   #.,$LPADR      ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
          MOV   #.,$LPERR      ;;SETUP THE ERROR LOOP ADDRESS
          ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
          ;;EQUAL TO A '-1', SETUP FOR A SOFTWARE SWITCH REGISTER.
          MOV   @ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
006010  013746  000004

```

```

006014 012737 006050 000004      MOV    #67$,@#ERRVEC    ;;SET UP ERROR VECTOR
006022 012737 177570 001154      MOV    #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
006030 012737 177570 001156      MOV    #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
006036 022777 177777 173110      CMP    #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
006044 001012                                BNE    69$              ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                ;;AND THE HARDWARE SWR IS NOT = -1
006046 000403                                BR     68$              ;;BRANCH IF NO TIMEOUT
006050 012716 006056 67$:          MOV    #68$, (SP)      ;;SET UP FOR TRAP RETURN
006054 000002                                RTI
006056 012737 000176 001154 68$:          MOV    #SWREG,SWR      ;;POINT TO SOFTWARE SWR
006064 012737 000174 001156      MOV    #DISPREG,DISPLAY
006072 012637 000004 69$:          MOV    (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR

006076 005037 001220                                CLR    $PASS           ;;CLEAR PASS COUNT
006102 132737 000200 001233      BITB  #APTSIZE,$ENVM   ;;TEST USER SIZE UNDER APT
006110 001403                                BEQ    70$              ;;YES,USE NON-APT SWITCH
006112 012737 001234 001154      MOV    #$$SWREG,SWR    ;;NO,USE APT SWITCH REGISTER
006120                                70$:

20  .SBTTL  TYPE PROGRAM NAME
    ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
006120 005227 177777      INC    #-1              ;;FIRST TIME?
006124 001056                                BNE    71$              ;;BRANCH IF NO
006126 022737 023506 000042      CMP    #SENDAD,@#42    ;;ACT-11?
006134 001452                                BEQ    71$              ;;BRANCH IF YES
006136 104401 006204      TYPE    ,72$           ;;TYPE ASCIZ STRING
    .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
006142 005737 000042      TST    @#42             ;;ARE WE RUNNING UNDER XXDP/ACT?
006146 001012                                BNE    73$              ;;BRANCH IF YES
006150 123727 001232 000001      CMPB  $ENV,#1           ;;ARE WE RUNNING UNDER APT?
006156 001406                                BEQ    73$              ;;BRANCH IF YES
006160 023727 001154 000176      CMP    SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
006166 001005                                BNE    74$              ;;BRANCH IF NO
006170 104406                                GTSWR                    ;;GET SOFT-SWR SETTINGS
006172 000403                                BR     74$
006174 112737 000001 001150 73$:          MOVB  #1,$AUTOB        ;;SET AUTO-MODE INDICATOR
006202                                74$:
006202 000427      BR     71$              ;;GET OVER THE ASCIZ
    ;;72$: .ASCIZ <CRLF>@CZRMTAO - RM05/3/2 DRIVE COMPATIBILITY TST@<CRLF>
006262 71$:

21  ;THE FOLLOWING FINDS OUT THE PROGRAM CONTROL MODE:
22  ;PAPER TAPE (MANUAL), ACT11, XXDP CHAIN OR DUMP
23
24
25 006262 005037 001404      CLR    XXDP             ;;CLEAR 'XXDP' LOAD DEVICE STORAGE
26 006266 122737 000016 000041      CMPB  #16,@#41         ;;LOADED FROM AN RM05/3/2 ?
27 006274 001160                                BNE    3$               ;;BR IF NOT
28 006276 013737 000040 001404      MOV    @#40,XXDP       ;;GET DEVICE INDICATOR AND NUMBER
29 006304 122737 000007 001404      CMPB  #7,XXDP          ;;IS IT A VALID NUMBER ?
30 006312 103002                                BHIS  1$               ;;YES
31 006314 105037 001404      CLRB  XXDP             ;;NO, DEFAULT TO DRIVE 0
32 006320 005737 000042 1$:          TST    @#42             ;;CHAIN MODE OR ACT11 AUTO ACCEPT ?
33 006324 001425                                BEQ    2$               ;;BR IF NEITHER
34 006326 104401 006334      TYPE    ,76$           ;;TYPE ASCIZ STRING
    006332 000412                                BR     75$              ;;GET OVER THE ASCIZ
    ;;76$: .ASCIZ <CRLF>/NOT TESTING DRIVE /
    75$:
35 006360                                CLR    -(SP)           ;;CLEAR WORD ON STACK
006360 005046
  
```

```

36 006362 113716 001404      MOVB    XXDP,(SP)      ;GET DRIVE ADDRESS
37 006366 104403              TYPOS                ;TYPE THE ADDRESS
38 006370      001              .BYTE    1            ;ONLY 1 CHARACTER
39 006371      000              .BYTE    0            ;SUPPRESS LEADING ZEROS
40 006372 104401 001207      TYPE    $CRLF         ;CR-LF
41 006376 000517              BR      3$           ;GET NUMBER OF DRIVES
42
43 006400 005227 177777      2$:    INC    #-1      ;FIRST TIME THRU HERE ?
44 006404 001114              BNE    3$           ;NO
45 006406 104401 006414      TYPE    78$         ;:TYPE ASCIZ STRING
      006412 000410              BR      77$         ;:GET OVER THE ASCIZ
      ;:78$: .ASCIZ <CRLF>/TO TEST DRIVE /
      ;:77$:
46 006434 005046              CLR    -(SP)        ;CLEAR WORD ON STACK
47 006436 113716 001404      MOVB    XXDP,(SP)   ;GET DRIVE ADDRESS
48 006442 104403              TYPOS                ;TYPE DRIVE ADDRESS
49 006444      001              .BYTE    1            ;ONLY 1 CHARACTER
50 006445      000              .BYTE    0            ;SUPPRESS LEADING ZEROS
51 006446 104401 006454      TYPE    80$         ;:TYPE ASCIZ STRING
      006452 000431              BR      79$         ;:GET OVER THE ASCIZ
      ;:80$: .ASCIZ /, HALT PROGRAM, REMOVE RRD P PACK AND REPLACE IT/<CRLF>
      ;:79$:
52 006536 104401 006544      TYPE    81$         ;:TYPE ASCIZ STRING
      006542 000435              BR      3$           ;:GET OVER THE ASCIZ
      ;:81$: .ASCIZ /WITH A WORK PACK, CLEAR LOCATION 40 AND RESTART PROGRAM./<CRLF>
      ;:3$:
56 006636 004737 022654      JSR    PC,$TKINT    ;TURN ON THE KEYBOARD INTERRUPT
57 006642 013737 001276 027476  MOV    $RMADR,RMADR ;RH/RM ADDRESS
58 006650 013737 001300 027500  MOV    $RMVEC,RMVEC ;VECTOR ADDRESS
59 006656 012705 002006              MOV    #ASNLST,R5   ;START OF AREA TO CLEAR
60 006662 005025              CLR    (R5)+         ;
61 006664 022705 004424      CMP    #INDST,R5    ;LOOK FOR END OF CLEAR AREA
62 006670 001374              BNE    4$           ;BR IF NOT FINISHED
63 006672 012706 001100      MOV    #STACK,SP    ;SETUP THE STACK POINTER
64 006676 005037 177776      CLR    PS           ;CLEAR THE PROCESSOR STATUS WORD
65 006702 013737 001320 001352  MOV    HZ,SIXTEE     ;1/60 TH OR 1/50 TH SECOND COUNTER VALUE
66 006710 005037 001344      CLR    HOUR         ;CLEAR THE HOUR'S COUNTER
67 006714 005037 001346      CLR    MINUTE       ;CLEAR THE MINUTE'S COUNTER
68 006720 005037 001350      CLR    SECOND       ;CLEAR THE SECOND'S COUNTER
69 006724 005037 001336      CLR    CFLAG        ;CLEAR THE 'CONTROL C' FLAG
70
71
72      ;ROUTINE TO DETERMINE BUFFER AREA SIZE
73 006730 005227 177777      SIZMEM: INC    #-1    ;SEE IF TIME TO SIZE MEMORY
74 006734 001002              BNE    1$           ;BR IF NOT
75 006736 004737 027110      JSR    PC,$SIZE     ;SEE HOW MUCH MEMORY ON SYSTEM
76 006742 013737 027240 001332  1$:    MOV    $LSTAD,LSTAD ;SAVE THE LAST ADDRESS
77 006750 023727 001332 160000  CMP    LSTAD,#160000 ;OVER 28K ?
78 006756 101403              BLOS   2$           ;NO, THEN DON'T SET THE NEW LIMIT
79 006760 012737 160000 001332  MOV    #160000,LSTAD ;SET NEW LIMIT
80 006766 162737 005670 001332  2$:    SUB    #1500.*2,LSTAD ;SAVE XXDP LOADER AND ABSOLUTE LOADER
81
82      ;SET UP THE OTHER SYSTEM DEVICES THAT THE PROGRAM WILL USE
83
84 006774 004737 021134      SETVEC: JSR    PC,CKCLK ;START THE CLOCK
85 007000 012737 177777 027436  MOV    #-1,SAVEFG   ;SET THE SAVE REGISTERS FLAG
86
    
```

```

87      ;SETUP IF 'XXDP' OR 'ACT11' OPERATION
88
89 007006 005001      MONTR:  CLR      R1      ;DRIVE #
90 007010 005002      CLR      R2      ;AVAIL TABLE INDEX
91 007012 005003      CLR      R3      ;DRIVE# X 2
92 007014 016300 004444 1$:  MOV      BLKADR(R3),R0 ;LOAD DPB ADDRESS
93 007020 004737 021620 JSR      PC,CLRDPB ;CLEAR DPB BLOCK
94 007024 022322      2$:  CMP      (R3)+,(R2)+ ;INCREMENT INDEX
95 007026 005201      INC      R1      ;NEXT DRIVE
96 007030 022701 000007 CMP      #7,R1 ;ALL DRIVE ASSIGN ?
97 007034 002367      BGE      1$      ;NO
98
99
100     ;ASSIGN LOGICAL DRIVES TO BE TEST IN THE PASS1 AND PASS 2
101     ;THREE WORDS ARE USED IN THE BIT MAPS:
102     ;      ASNLST = SPECIFIES THE LOGICAL DRIVES ASSIGNED
103     ;      ASSGN1 = SPECIFIES THE LOGICAL DRIVES WILL BE TESTED IN PASS1.
104     ;      ASSGN2 = SPECIFIES THE LOGICAL DRIVES WILL BE TESTED IN PASS2.
105
106     ;EACH LOGICAL DRIVE HAS A HISTORY FILE LABELED DIRV'Z (Z=0 TO 17)
107     ;THE LOCATIONS LABELED $SYSNM AND $PHYRD IN THE HISTORY FILE
108     ;STORE THE SYSTEM NAME (A TO H) AND PHYSICAL DRIVE NUMBER
109     ;(0 TO 7).
110     ;THE SUB-SYSTEM ADDRESS AND INTERRUPT VECTOR ARE STORED
111     ;IN THE TABLE LABELED 'SYSADR:'.
112
113     ;THE LOCATIONS SYSADR AND SYSADR+2 FOR SUB-SYSTEM A,SYSADR+4
114     ;AND SYSADR+6 FOR SUBSYSTEM B , ETC,THE FIRST WORD
115     ;IS THE SUB SYSTEM ADDRESS WHILE THE SECOND WORD IS THE VECTOR
116
117     ;THE LOGICAL DRIVES ARE ASSIGNED FROM CONSOLE KEYBOARD.
118
119 007036 012700 002006 MOD00: MOV      #ASNLST,R0 ;ADDRESS OF 1ST BIT MAPS IN R0
120 007042 012701 004424 MOV      #INDST,R1 ;LAST ADDRESS TO CLEAR
121 007046 005020      1$:  CLR      (R0)+ ;CLEAR CURRENT POINTED ADDRESS
122 007050 020100      CMP      R1,R0 ;ALL DONE ?
123 007052 101375      BHI      1$      ;NO,THEN BRANCH BACK
124 007054 012737 000101 001274 MOV      #'A,$CDW2 ;TEMP STORAGE OF SYS NAME
125
126
127 007062 005037 001224 MOD21: CLR      DRIVE ;TEMP STORAGE OF PHYSICAL DRIVE BIT MAP
128 007066 104401 001207 TYPE      ,SCLF ;CR-LF
129 007072 104401 041354 TYPE      ,MSG1 ;SUB-SYSTEM
130 007076 104401 042222 TYPE      ,QUOTM ;TYPE "" QUOTATION MARK
131 007102 104401 001274 TYPE      ,SCDW2 ;SYS-NAME(A TO H)
132 007106 104401 042222 TYPE      ,QUOTM ;TYPE "" QUOTATION MARK
133 007112 104401 042330 TYPE      ,BLNKS1 ;TYPE 1 BLANK
134 007116 104401 041364 TYPE      ,MSG2 ;DRIVE(S)
135
136 007122 104411      RDLIN ;READ IN THE DRIVE NUMBERS
137 007124 012601      MOV      (SP)+,R1 ;GET THE INPUT LINE ADDRESS
138 007126 105711      1$:  TSTB   (R1) ;END OF STRING <CR> ?
139 007130 001437      BEQ      3$      ;YES
140 007132 004537 022310 JSR      R5,CK.OCT ;CHECK THE DIGIT MUST 0 TO 7,RETURN VALUE IN R2
141 007136 000751      BR       MOD21 ;INCORRECT DRIVE NUMBER,ENTER AGAIN
142 007140 156237 027464 001224 BISB   ATABIT(R2),DRIVE ;SET THE PHYS. DRIVE BIT, R2 = DRIVE NUMBER
143 007146 022737 176700 001276 CMP      #176700,$RMADR ;IS IT STANDARD RH/RM ADDRESS ?
    
```

```

144 007154 001016          BNE      2$          ;BR IF NO
145 007156 005737 001404   TST      XXDP        ;IS THIS LOAD DEVICE ?
146 007162 001413          BEQ      2$          ;BR IF NO
147 007164 123702 001404   CMPB    XXDP,R2     ;IS THIS THE DRIVE ?
148 007170 001010          BNE      2$          ;BR IF NO
149 007172 104401 042704   TYPE    ,QDRIV     ;TYPE ' ?DRIVE'
150 007176 010246          MOV      R2,-(SP)   ;SAVE R2 FOR TYPEOUT
      007200 104403          TYPOS    ;GO TYPE--OCTAL ASCII
      007202 002           .BYTE    2         ;TYPE 2 DIGIT(S)
      007203 000           .BYTE    0         ;SUPPRESS LEADING ZEROS
151 007204 104401 042714   TYPE    ,LODEV     ;TYPE 'IS LOAD DEVICE'
152 007210 000724          BR       MOD21      ;TRY AGAIN
153 007212 005201          2$:     INC      R1          ;
154 007214 105711          TSTB    (R1)       ;END OF STRING <CR> ?
155 007216 001404          BEQ     3$          ;YES
156 007220 122721 000054   CMPB    #'',(R1)+  ;MUST BE A COMMA
157 007224 001316          BNE     MOD21      ;ENTER AGAIN ,IF NOT
158 007226 000737          BR      1$         ;LOCATE NEXT DRIVE
159
160 007230 005737 001334   3$:     TST      CHGADR   ;START AT 200 ?
161 007234 003035          BGT     MOD22      ;BRANCH IF SO
162 007236 013701 001274   MOV     $CDW2,R1   ;SYS NAME ASCII FROM A TO H
163 007242 042701 177760   BIC     #177760,R1 ;LEFT ONLY 4 BITS
164 007246 005301          DEC     R1         ;ADJUST INDEX VALUE
165 007250 006301          ASL     R1         ;2 WORD INDEX VALUE
166 007252 006301          ASL     R1         ;
167 007254 062701 002014   ADD     #SYSADR,R1 ;SYS ADDRESS TABLE ADDRESS
168 007260 010137 001272   MOV     R1,$CDW1  ; SYS ADDRESS TABLE'S ENTRY TO CDW1
169
170 007264 005737 001224   MOD23: TST      DRIVE   ;CHECK IF ANY PHYSICAL DRIVE(S) ASSIGNED
171 007270 001416          BEQ     2$         ;BRANCH IF NONE
172 007272 013700 001272   1$:     MOV     $CDW1,R0 ;SYS ADDRESS TABLE ENTRY
173 007276 011037 001276   MOV     (R0), $RMADR ; SYS ADDRESS
174 007302 016037 000002 001300   MOV     2(R0), $RMVEC ; SYS VECTOR
175 007310 004737 035670   JSR    PC,BUSADR  ;CHECK THE ADDRESS WITH THE OPERATOR
176 007314 013710 001276   MOV     $RMADR,(R0) ;NEW RH/RM ADDRESS INTO TABLE
177 007320 013760 001300 000002   MOV     $RMVEC,2(R0) ;NEW VECTOR OF RH/RM INTO TABLE
178 007326 000406          2$:     BR      MOD11    ;BRANCH TO NEXT MODULE
179
180 007330 013737 001276 002014 MOD22: MOV     $RMADR,SYSADR ;LOAD THE SYSTEM ADDRESS TABLE
181 007336 013737 001300 002016   MOV     $RMVEC,SYSADR+2 ;LOAD THE VECTOR
182
183           ;$CDW2 = ASCII NAME OF SUB SYSTEM (A TO H)
184           ;$CDW1 = ENTRY TO THE SYS ADDRESS TABLE
185           ;DRIVE = PHYSICAL DRIVES TO BE ASSIGNED
186           ;
187           ;THIS SECTION OF CODING USES THE ABOVE PARAMETERS
188           ;TO SET UP THE BIT MAP OF ASNLST,ASSGN1,ASSGN2
189
190 007344 005737 001224   MOD11: TST      DRIVE   ;ANY DRIVE ASSIGN ?
191 007350 001002          BNE     MOD30      ;BR IF YES
192 007352 000137 007524   JMP     MOD12      ;BRANCH,IF NONE
193
194 007356 022737 177777 002006 MOD30: CMP     #-1,ASNLST ;HAVE ALL 16 LOGICAL DRIVES BEEN ASSIGNED ?
195 007364 001457          BEQ     5$         ;BR IF YES
196 007366 013700 002006   MOV     ASNLST,R0  ;FOUND THE AVAILABLE LOGICAL DRIVE LOCATION
197 007372 012701 000001   MOV     #1,R1     ;START FROM LOGICAL DRIVE 0

```



```

198 007376 005002          CLR      R2          ;INDEX VALUE
199 007400 030100          BIT      R1,R0       ;IS THE LOGICAL DRIVE AVAILABLE ?
200 007402 001406          BEQ     2$           ;YES
201 007404 000241          CLC
202 007406 006101          ROL     R1           ;NEXT LOGICAL DRIVE
203 007410 103445          BCS     5$           ;BRANCH IF NONE IS AVAILABLE
204 007412 062702 000002    ADD     #2,R2        ;INCREMENT INDEX VALUE
205 007416 000770          BR      1$           ;LOCATE NEXT LOGICAL DRIVE
206 007420 016204 004444          MOV     BLKADR(R2),R4 ;GET THE LOGICAL DRIVE'S HISTORY FILE
207 007424 113764 001274 000014  MOVB   $CDW2,$SYSNM(R4) ;LOAD THE ASCII SYS NAME
208 007432 050137 002006          BIS    R1,ASNLST    ;SET LOGICAL DRIVE ASSIGN BIT
209 007436 050137 002010          BIS    R1,ASSGN1    ;SET PASS 1 BIT
210 007442 050137 002012          BIS    R1,ASSGN2    ;SET PASS 2 BIT
211 007446 005002          CLR     R2           ;DRIVE #
212 007450 136237 027464 001224 3$:    BITB   ATABIT(R2),DRIVE ;IS THIS DRIVE ASSIGNED ?
213 007456 001005          BNE     4$           ;BR IF YES
214 007460 005202          INC     R2           ;PHYSICAL DRIVE NUMBER
215 007462 020227 000007    CMP     R2,#7        ;ALL DRIVES DONE YET ?
216 007466 003770          BLE     3$           ;BR IF NO
217 007470 000415          BR      5$           ;YES, EXIT
218 007472 110214          MOVB   R2,(R4)       ;LOAD THE PHYSICAL DRIVE # INTO HISTORY FILE
219 007474 111464 000015          MOVB   (R4),$PHYDR(R4) ;GET PHYSICAL DRIVE NUMBER AND
220 007500 152764 000060 000015  BISB   #'0,$PHYDR(R4) ;MAKE IT ASCII
221 007506 112764 000011 000016  MOVB   #HT,$GAP(R4)   ;MAKE UP FOR SCORE TYPE
222 007514 146237 027464 001224  BICB   ATABIT(R2),DRIVE ;DEASSIGN DRIVE BIT FROM LIST
223 007522 001315          BNE     MOD30        ;BRANCH IF NOT ALL DONE
224 007524
225
226 007524 005737 001334          MOD12: TST     CHGADR      ;200 START ?
227 007530 003100          BGT     7$           ;YES, THEN EXIT
228 007532 022737 177777 002006 1$:    CMP     #-1,ASNLST   ;FULL HOUSE
229 007540 001474          BEQ     7$           ;YES, THEN EXIT
230 007542 104401 042227          2$:    TYPE   ,MSG5       ;WILL TEST
231 007546 104401 041364          TYPE   ,MSG2       ;DRIVE(S)
232 007552 005003          CLR     R3           ;INDEX TO LOGICAL DRIVE HISTORY FILE
233 007554 012704 000001          MOV     #1,R4        ;BIT MAP OF ASNLST
234 007560 030437 002006          3$:    BIT     R4,ASNLST ;ASSIGNED LOGICAL DRIVE ?
235 007564 001421          BEQ     5$           ;NO
236 007566 016305 004444          MOV     BLKADR(R3),R5 ;LOAD THE HISTORY FILE ADDRESS
237 007572 126537 000014 001274  CMPB   $SYSNM(R5),$CDW2 ;ON THE SAME SYSTEM ?
238 007600 001005          BNE     4$           ;NO
239 007602 111546          MOVB   (R5),-(SP)    ;TYPE THE PHYSICAL DRIVE #
240 007604 104403          TYPOS
241 007606 001          .BYTE  1
242 007607 000          .BYTE  0
243 007610 104401 042330          TYPE   ,BLNKS1     ;TYPE 1 BLANK
244 007614 062703 000002          4$:    ADD     #2,R3        ;INCREMENT TO NEXT LOGICAL DRIVE
245 007620 000241          CLC
246 007622 006104          ROL     R4           ;BIT MAP OF THE NEXT LOGICAL DRIVE
247 007624 103401          BCS     5$           ;BRANCH IF ALL LOGICAL DRIVE'S CHECKED
248 007626 000754          BR      3$           ;BRANCH BACK
249 007630 104401 042242          5$:    TYPE   ,MSG6       ;ON
250 007634 104401 041354          TYPE   ,MSG1       ;SUB SYSTEM
251 007640 104401 042222          TYPE   ,QUOTM      ;TYPE "" QUOTATION MARK
252 007644 104401 001274          TYPE   , $CDW2     ; A TO H
253 007650 104401 042222          TYPE   ,QUOTM      ;TYPE "" QUOTATION MARK
254
    
```

```

255 007654 005237 001274      INC      $CDW2      ;CHECK NEXT SUB SYSTEM
256 007660 122737 000110 001274    CMPB    #'H,$CDW2  ;ALL EIGHT SUB SYSTEMS CHECKED?
257 007666 103421          BLO     7$         ;YES
258 007670 104401 001207          TYPE   ,$CRLF     ;CR-LF
259 007674 104401 042246          TYPE   ,MSG7      ;ASK FOR OTHER SUB SYSTEM
260 007700 104410          RDCHR          ;READ IN A LINE
261 007702 012637 001174          MOV     (SP)+,$TMPO ;GET INPUT CHARACTER
262 007706 023727 001174 000131    CMP     $TMPO,#'Y  ;IS IT INPUT 'YES' ?
263 007714 001004          BNE     6$         ;BR IF NO
264 007716 104401 042317          TYPE   ,Y         ;TYPE 'Y' CHARACTER
265 007722 000137 007062          JMP     MOD21      ;SET UP OTHER SUB SYSTEM
266 007726 104401 042322          6$:    TYPE   ,N         ;TYPE 'N' CHARACTER
267 007732 005737 001334          7$:    TST    CHGADR ;START AT 210
268 007736 001402          BEQ     8$         ;YES,EXECUTE PASS2 ONLY
269 007740 000137 007754          JMP     XPASS1     ;TO PASS1
270 007744 005037 002010          8$:    CLR     ASSGN1 ;CLEAR THE PASS1 BIT MAP
271 007750 000137 012360          JMP     XPASS2     ;BRANCH TO PASS2
272
273      ;PASS ONE, THE VARIABLES ARE ASSIGNED AS FOLLOWS:
274      :      $CDW1 = ADDRESS OF THE CURRENT LOGICAL DRIVE HISTORY FILE
275      :      $CDW2 = SYSTEM NAME A THROUGH H
276      :      $DEV# = CURRENT LOGICAL DRIVE #
277      :      ASNLST = ASSIGNED LOGICAL DRIVES
278      :      ASSGN1 = ASSIGNED LOGICAL DRIVES IN THIS PASS
279      :      R0 = ADDRESS OF DPB BLOCK FEED INTO DRIVER-HANDLER
280      :      R1 = TEMP STORAGE OF ADDRESS OF THE LOGICAL BLOCK
281
282 007754 005737 002010          XPASS1: TST    ASSGN1 ;ANY DRIVES ASSIGNED FOR PASS1
283 007760 001002          BNE     1$         ;BR IF YES, ELSE GO TO
284 007762 000137 010704          JMP     ENDX1      ;END OF PASS 1
285
286 007766 005037 001270          1$:    CLR     $DEV#  ;INDEX OF LOGICAL BLOCK
287 007772 013737 004444 001272    MOV     BLKADR,$CDW1 ;ADDRESS OF LOGICAL BLOCK 0
288 010000 112737 000101 001274    MOVB   #'A,$CDW2    ;SYS NAME STARTS FROM A
289 010006 013737 002014 027476    MOV     SYSADR,RMADR ;LOAD SYS-TEM A INTO
290 010014 013737 002016 027500    MOV     SYSADR+2,RMVEC ;DIRVER-HANDLER
291 010022 013701 001272          MOV     $CDW1,R1   ;R1=ADDRESS OF LOGICAL BLOCK 1
292 010026 012777 017344 017444    MOV     #IDLEX,@RMVEC ;RESET ALL INTERRUPT VECTOR
293 010034 005077 017442          CLR     @RMVEC+2   ;CLEAR THE INTERRUPT LEVEL
294 010040 104401 001207          TYPE   ,$CRLF     ;CR-LF
295 010044 104401 041377          TYPE   ,MSG3      ;** STARTING PASS 1
296 010050 104401 042242          TYPE   ,MSG6      ;ON
297 010054 104401 041354          TYPE   ,MSG1      ;SUB-SYSTEM
298 010060 104401 042222          TYPE   ,QUOTM     ;TYPE "" QUOTATION MARK
299 010064 104401 001274          TYPE   ,$CDW2     ;SYS-NAME(A TO H)
300 010070 104401 042222          TYPE   ,QUOTM     ;TYPE "" QUOTATION MARK
301 010074 111137 001224          MOVB   (R1),DRIVE  ;LOAD THE PHYSICAL DRIVE #
302 010100 104401 001207          TYPE   ,$CRLF     ;CR-LF
303 010104 104401 041423          TYPE   ,MSG4      ;MOUNT PACK ON THE DRIVE
304 010110 104401 001274          TYPE   ,$CDW2     ;SYS-NAME(A - H)
305 010114 113746 001224          MOVB   DRIVE,-(SP) ;TYPE THE DRIVE #
306 010120 104403          TYPOS
307 010122          .BYTE 1
308 010123          .BYTE 0
309 010124 104401 042330          TYPE   ,BLNKS1   ;TYPE 1 BLANK
310 010130 104401 041451          TYPE   ,MSG8
311 010134 104401 001207          TYPE   ,$CRLF     ;CR-LF
    
```

```

312 010140 104401 041464          TYPE ,MSG9
313 010144 104401 042330          TYPE ,BLNKS1          ;TYPE 1 BLANK
314
315 010150 104411          2$: RDLIN          ;CHECK IF O.P. READY
316 010152 012602          MOV (SP)+,R2          ;LOCATE THE INPUT LINE
317 010154 105712          TSTB (R2)          ;FOLLOW BY <CR> ?
318 010156 001374          BNE 2$          ;BRANCH IF NOT
319 010160 004737 027514          JSR PC,RMINIT          ;INITIALIZE THE SUB SYSTEM
320 010164 012737 177777 027436          MOV #-1,SAVEFG          ;SAVE ALL RH/RM REGISTERS
321 010172 012737 177777 027440          MOV #-1,SEEKFG          ;DON'T DO IMPLY SEEK
322 010200 013700 001224          MOV DRIVE,R0          ;R0=PHYSICAL DRIVE # OF THE SUB SYSTEM
323 010204 105760 027350          TSTB DRVSTA(R0)          ;DRIVE EXIST AND ON LINE ?
324 010210 003467          BLE 6$          ;BRANCH IF NOT
325 010212 105760 027360          TSTB DRVSTYP(R0)          ;DRIVE IS AN RM05/3/2 ?
326 010216 003464          BLE 6$          ;BRANCH IF NOT
327 010220 116037 027360 001402          MOVB DRVSTYP(R0),DTYP          ;GET DRIVE TYPE TO BE TESTED
328 010226 012737 000022 001372          MOV #18.,TRKLMT          ;GET LAST TRACK FOR AN RM05
329 010234 122760 000007 027360          CMPB #7,DRVSTYP(R0)          ;IS DRIVE AN RM05 ?
330 010242 001403          BEQ 3$          ;BR IF YES
331 010244 012737 000004 001372          MOV #4.,TRKLMT          ;GET LAST TRACK FOR AN RM03/2
332
333 010252 012700 004506          3$: MOV #FMTDPB,R0          ;DPB ADDRESS
334 010256 113710 001224          MOVB DRIVE,(R0)          ;LOAD THE DRIVE NUMBER
335 010262 012760 043032 000006          MOV #BUFFER,$BUF(R0)          ;LOAD BUFFER ADDRESS
336 010270 012760 004526 000014          MOV #RM.REG,14(R0)          ;AREA TO SAVE ALL RH/RM REG'S
337 010276 012760 177400 000004          MOV #-256.,$WRDM(R0)          ;WORD COUNT (NEG)
338 010304 013760 001374 000012          MOV CYLIMT,$CYL(R0)          ;CYLINDER 822.
339 010312 113760 001372 000011          MOVB TRKLMT,$TRK(R0)          ;GET TRACK ADDRESS
340 010320 105060 000010          CLRB $SEC(R0)          ;SEC 0
341 010324 112760 000171 000002          MOVB #RDDAT,$COMND(R0)          ;READ DATA COMMAND
342
343 010332 004037 030264          4$: JSR R0,RM05          ;CALL THE DRIVER-HANDLER
344 010336 004506          FMTDPB          ;PARAMETER ADDRESS
345 010340 000774          BR 4$          ;LOOPING IF QUEUE IS NOT SUCCESSFUL
346 010342 005737 004524          5$: TST FMTDPB+16          ;COMMAND DONE ?
347 010346 001775          BEQ 5$          ;BRANCH IF NOT
348 010350 100015          BPL 7$          ;BRANCH IF DONE, WITHOUT ERROR
349 010352 062737 000002 004516          ADD #2,FMTDPB+10          ;TRY NEXT SECTOR (0,2,4,6,8)
350 010360 122737 000010 004516          CMPB #8.,FMTDPB+10          ;ALL FIVE SECTORS CHECKED ?
351 010366 101361          BHI 4$          ;NO,THEN TRY AGAIN
352
353 010370 104401 001207          6$: TYPE ,$CRLF          ;CR-LF
354 010374 104401 041520          TYPE ,MSG10          ;DRIVE IS NOT READY
355 010400 000137 010704          JMP ENDX1          ;STOP THE TEST
356
357          ;BAD SPOT FILE IS RETRIVED, IS STORED FROM BUFFER+4 TO BUFFER+256. FIRST
358          ;WORD CYLINDER #, SECOND WORD TRK AND SEC NUMBERS, FILE IS TERMINATED BY
359          ;A -1 IN THE CYLINDER NUMBER.
360
361 010404 012704 043042          7$: MOV #BUFFER+10,R4          ;R4 ADDRESS OF BAD SPOT FILE
362 010410 022714 177777          8$: CMP #-1,(R4)          ;END OF BAD SPOT FILE ?
363 010414 001411          BEQ 9$          ;YES
364 010416 022704 044032          CMP #BUFFER+1000,R4          ;END OF BAD SPOT FILE ?
365 010422 101406          BLOS 9$          ;BRANCH IF IT IS
366 010424 004537 010536          JSR R5,SPOTX          ;CHECK THE CYLINDER POINTED BY (R4)
367 010430 000422          BR 11$          ;BRANCH IF BAD SPOT IN THE TEST ZONES
368 010432 062704 000004          ADD #4,R4          ;NEXT BAD SPOT ADDRESS
    
```

```

369 010436 000764          BR      8$          ;LOOPING BACK
370
371 010440 032777 000200 170506 9$:  BIT      #BIT7,@SWR      ;SWITCH 7 SET ?
372 010446 001404          BEQ      10$          ;BRANCH IF NOT SET
373 010450 012700 004506          MOV      #FMTDPB,R0      ;DPB ADDRESS
374 010454 004737 020602          JSR      PC,PRTBAD       ;PRINT THE BAD SPOT FILE
375 010460 105760 000010          10$:  TSTB     $SEC(R0)      ;SECTOR 10 HAS BEEN READ ?
376 010464 001022          BNE      13$          ;BRANCH IF SO
377 010466 112760 000012 000010  MOVB     #10.,$SEC(R0)   ;READ SECTOR 10
378 010474 000716          BR      4$          ;LOOPING BACK
379
380 010476 104401 041565          11$:  TYPE     ,MMSG11      ;PACK NOT ACCEPTABLE
381 010502 104401 001207          TYPE     ,$CRLF         ;CR-LF
382 010506 032777 000200 170440  BIT      #BIT7,@SWR      ;SWITCH 7 SET ?
383 010514 001404          BEQ      12$          ;BRANCH IF NOT SET
384 010516 012700 004506          MOV      #FMTDPB,R0      ;DPB ADDRESS
385 010522 004737 020602          JSR      PC,PRTBAD       ;TYPE THE BAD SPOT FILE
386 010526 000137 007754          12$:  JMP      XPASS1       ;RESTART PASS 1
387 010532 000137 010720          13$:  JMP      TST1         ;PROCEED TO TEST 1
388
389
390          ;SUBROUTINE SPOTX
391          ;SEE IF THE CYLINDER POINTED BY (R4) IS IN THE TESTING ZONES
392          ;BELOW:
393          ;
394          ;(0-15, 128-143, 256-271, 384-399, 512-527, 640-655, 768-783,
395          ;112-127, 240-255, 368-383, 496-511, 624-639, 752-767, 17-32,
396          ;145-160, 273-288, 401-416, 529-544, 657-672, 785-800 AND 620)
397          ;
398          ;CALL
399          ;
400          ;      JSR      R5,SPOTX      ;R4=POINT TO CYLINDER NUMBER
401          ;      RET1     ;ERROR RET
402          ;      RET2     ;NORMAL RET1
403
403 010536 010146          SPOTX:  MOV      R1,-(SP)      ;SAVE R1 THROUGH R3
404 010540 010246-        MOV      R2,-(SP)
405 010542 010346        MOV      R3,-(SP)
406
407 010544 005003          CLR      R3            ;ERROR FLAG
408 010546 005046          CLR      -(SP)         ;DUMMY PAIR
409 010550 005046          CLR      -(SP)         ;DUMMY PAIR
410 010552 005046          CLR      -(SP)         ;DUMMY PAIR
411 010554 012746 000007        MOV      #7,-(SP)      ;ZONE STARTING ADDRESS
412 010560 012746 000021        MOV      #17.,-(SP)    ;SEGMENT NUMBER
413 010564 012746 000007        MOV      #7,-(SP)      ;ZONE STARTING ADDRESS
414 010570 012701 000160        MOV      #112.,R1      ;SEGMENT NUMBER
415 010574 012702 000006        MOV      #112.,R1      ;R1=ZONE STARTING ADDRESS
416 010600 021401          1$:  MOV      #6,R2         ;R2=SEGMENT NUMBER
417 010602 103410          CMP      (R4),R1       ;CYL IN THE ZONE ?
418 010604 062701 000017        BLO      3$            ;BRANCH IF NOT
419 010610 021401          ADD      #15.,R1       ;CHECK WITH THE UPPER BOND
420 010612 101002          CMP      (R4),R1       ;CYL IN THE ZONE ?
421 010614 052703 000002        BHI      2$            ;BRANCH IF NOT
422          BIS      #BIT1,R3      ;SET THE ERROR FLAG
423 010620 162701 000017        2$:  SUB      #15.,R1      ;RESTORE TO THE LOWER BOND
424 010624 005302          3$:  DEC      R2           ;DECREMENT THE SEGMENT COUNT
425 010626 001403          BEQ      4$           ;ALL SEGMENT CHECKED ?

```

```

426 010630 062701 000200      ADD    #128.,R1      ;ADJUST ZONE STARTING ADDRESS
427 010634 000761              BR     1$           ;LOOPING BACK UNTIL ALL SEGMENTS ARE CHECKED
428
429 010636 012602              4$:   MOV    (SP)+,R2      ;POP THE NEXT SET OF ZONE PARAMETERS
430 010640 012601              MOV    (SP)+,R1
431 010642 005702              TST   R2           ;DUMMY PAIR
432 010644 001355              BNE   1$           ;BRANCH IF NOT
433 010646 005701              TST   R1           ;DUMMY PAIR
434 010650 001353              BNE   1$           ;BRANCH IF NOT
435
436 010652 021427 001154      5$:   CMP    (R4),#620.   ;ON CYLINDER 620 ?
437 010656 001002              BNE   6$           ;NO
438 010660 052703 000002      BIS   #BIT1,R3     ;SET ERROR FLAG
439 010664 005703              6$:   TST   R3           ;ANY ERROR ?
440 010666 001002              BNE   7$           ;YES
441 010670 062705 000002      ADD   #2,R5        ;ADJUST FOR NORMAL RETURN
442 010674 012603              7$:   MOV    (SP)+,R3     ;RESTORE R3 THROUGH R1
443 010676 012602              MOV    (SP)+,R2
444 010700 012601              MOV    (SP)+,R1
445 010702 000205              RTS   R5           ;EXIT
446
447 010704 104401 001207      ENDX1: TYPE  , $CRLF      ;CR-LF
448 010710 104401 042422      TYPE  , $MSG21     ;DRIVE NOT ONLINE OR NOT ASSIGNED
449 010714 000177 170460      JMP   @RSTART     ;JUMP TO RESTART
450

```

```

451 ;THE FOLLOWING CODING FOR TEST 1 THROUGH TEST 4
452 ;PARAMETER IN TST 1
453 ; $CDW1 = ADDRESS OF LOGICAL DRIVE BLOCK
454 ; DRIVE = PHYSICAL DRIVE #
455 ; $DEVN = LOGICAL DRIVE # 0-17
456 ; ASSGN1 = ASSIGN LOGICAL DRIVE BIT MAP
457 ; ASNLST = ASSIGNED LOGICAL DRIVE MAP INDICATOR
458 ; $CDW2 = SYS-NAME
459

```

```

460 ;$CDW2,DRIVE ARE ONLY CHANGED IN TST1 DURING PASS 1
461 ;$DEVN,ASSGN1,$CDW1 ARE CHANGED BY THE TEST 4
462

```

```

463 ;TST 1:DIRECT OPERATOR TO MOUNT AND LOAD PACKS
464

```

```

465 ;*****
TST1: SCOPE
466 010720 000004              MOV    #1,$TIMES    ;;DO 1 ITERATION
467 010722 012737 000001 001176  MOV    #1,TSTNM     ;LOAD THE TEST NUMBER
468 010730 012737 000001 001340  MOV    #STACK,SP    ;INITIALIZE THE STACK POINT
469 010736 012706 001100              CMP    $CDW1,BLKADR ;LOGICAL DRIVE 0
470 010742 023737 001272 004444  BEQ   6$           ;THEN EXIT
471 010750 001551              MOV    $CDW1,R1     ;R1=LOGICAL DRIVE BLOCK ADDRESS
472 010752 013701 001272              CMPB  $$SYSNM(R1),$CDW2 ;STILL ON THE SAME SYSTEM ?
473 010756 126137 000014 001274  BEQ   1$           ;THEN ,EXIT
474 010764 001426              MOV    $$SYSNM(R1),$CDW2 ;LOAD THE NEW SUB SYSTEM NAME
475 010766 116137 000014 001274  MOV    #IDLEX,@RMVEC ;RESET THE INTERRUPT VECTOR
476 010774 012777 017344 016476  CLR   @RMVEC+2     ;CLEAR THE INTERRUPT LEVEL
477 011002 005077 016474              TYPE  , $CRLF      ;CR-LF
478 011006 104401 001207              TYPE  , $MSG3     ;** STARTING PASS 1
479 011012 104401 041377              TYPE  , $MSG6     ;ON
480 011016 104401 042242              TYPE  , $MSG1     ;SUB-SYSTEM
481 011022 104401 041354              TYPE  , $QUOTM    ;TYPE "" QUOTATION MARK
482 011026 104401 042222

```

```

481 011032 104401 001274          TYPE      ,SCDW2          ;SYS-NAME(A TO H)
482 011036 104401 042222          TYPE      ,QUOTM          ;TYPE "" QUOTATION MARK
483 011042 111137 001224          1$:      MOVVB      (R1),DRIVE ;LOAD THE PHYSICAL DRIVE #
484 011046 104401 001207          TYPE      ,SCRLF          ;CR-LF
485 011052 104401 041423          TYPE      ,MSG4           ;MOUNT PACK ON THE DRIVE
486 011056 104401 001274          TYPE      ,SCDW2          ;SYS-NAME(A TO H)
487 011062 113746 001224          MOVVB     DRIVE,-(SP)    ;THE PHYSICAL DRIVE #
488 011066 104403          TYPOS
489 011070      001          .BYTE     1
490 011071      000          .BYTE     0
491 011072 104401 042330          TYPE      ,BLNKS1        ;TYPE 1 BLANK
492 011076 104401 041451          TYPE      ,MSG8           ;AND LOAD
493 011102 104401 001207          TYPE      ,SCRLF          ;CR-LF
494 011106 104401 041464          TYPE      ,MSG9
495
496 011112 104411          2$:      RDLIN
497 011114 012605          MOV      (SP)+,R5      ;LOCATE THE READIN LINE
498 011116 105715          TSTB     (R5)          ;NOT CORRECT INPUT LINE FORMAT
499 011120 001374          BNE      2$           ;BRANCH IF NOT
500 011122 113701 001274          MOVVB     $CDW2,R1     ;LOCATE THE SYSTEM ADDRESS TABLE
501 011126 042701 177760          BIC      #177760,R1    ;LEFT ON 4 BITS
502 011132 005301          DEC      R1           ;ADJUST THE INDEX VALUE
503 011134 006301          ASL      R1           ;FOUR WORD INDEX VALUE
504 011136 006301          ASL      R1
505 011140 016137 002014 027476          MOV      SYSADR(R1),RMADR ;LOAD THE SYSTEM ADDRESS
506 011146 016137 002016 027500          MOV      SYSADR+2(R1),RMVEC ;LOAD THE SYSTEM INTERRUPT VECTOR
507 011154 004737 027514          JSR      PC,RMIN!T     ;INITIALIZE THE SYSTEM
508 011160 012737 177777 027436          MOV      #-1,SAVEFG    ;SAVE ALL RH/RM REGISTER
509 011166 012737 177777 027440          MOV      #-1,SEEKFG    ;DON'T DO ANY IMPLY SEEK
510 011174 013700 001224          MOV      DRIVE,R0     ;R0=PHYSICAL DRIVE #
511 011200 105760 027350          TSTB     DRVSTA(R0)    ;ON-LINE ?
512 011204 003426          BLE     5$           ;BRANCH IF NOT
513 011206 105760 027360          TSTB     DRVSTYP(R0)   ;CHECK DRIVE TYPE
514 011212 003423          BLE     5$           ;BR IF NOT AN RM05/3/2
515 011214 122737 000007 001402          CMPB     #7,DTYP       ;WHAT WAS FIRST DRIVE TESTED ?
516 011222 001005          BNE     3$           ;BRANCH IF AN RM02 OR RM03, ELSE
517 011224 122760 000007 027360          CMPB     #7,DRVSTYP(R0) ;SEE IF DRIVE IS STILL AN RM05.
518 011232 001420          BEQ     6$           ;BR IF YES
519 011234 000404          BR      4$           ;ERROR ENCOUNTERED
520 011236 122760 000007 027360 3$:      CMPB     #7,DRVSTYP(R0) ;SEE IF DRIVE IS STILL AN RM02 OR RM03.
521 011244 001013          BNE     6$           ;BR IF YES
522 011246 104401 001207          4$:      TYPE      ,SCRLF          ;CR-LF
523 011252 104401 042734          TYPE      ,NOTST        ;CANNOT SELECT RM03/2'S AND RM05'S TOGETHER
524 011256 000177 170116          JMP      @RSTART       ;JUMP TO RESTART
525
526 011262 104401 001207          5$:      TYPE      ,SCRLF          ;CR-LF
527 011266 104401 041520          TYPE      ,MSG10        ;DRIVE NOT READY
528 011272 000612          BR      TST1          ;TRY AGAIN
529 011274
530
531          ;TEST 2
532          ;BASIC READ AND WRITE TEST
533          ;ALL LOGICAL DRIVE ACCESS CYLINDER 620
534          ;AND SECTOR ADDRESS IS CORRESPONDING TO THE LOGICAL DRIVE #
535          ;EACH LOGICAL DRIVE PERFORM WRITE AND WRITE CHECK ON ALL TRACKS,
536          ;(TRK0 - TRK4 ON AN RM03/2 AND TRK0 - TRK18 ON AN RM05)
537
    
```

```

538 011274 000004
    011276 012737 000001 001176
539 011304 012737 000002 001340
540 011312 012706 001100
541 011316 012700 004506
542 011322 013701 001272
543 011326 113710 001224
544 011332 013760 001270 000010
545 011340 012760 001154 000012
546 011346 012760 177400 000004
547 011354 012760 043032 000006
548 011362 012760 004526 000014
549 011370 105060 000011
550 011374 112760 000161 000002
551 011402 004737 020742
552 011406 004037 030264 2$:
553 011412 004506
554 011414 000774
555 011416 005737 004524 3$:
556 011422 001775
557 011424 012700 004506
558 011430 004737 017352
559 011434 005760 000016
560 011440 100433
561 011442 112760 000151 000002
562 011450 004037 030264 4$:
563 011454 004506
564 011456 000774
565 011460 005737 004524 5$:
566 011464 001775
567 011466 012700 004506
568 011472 004737 017352
569 011476 005760 000016
570 011502 100412
571 011504 112760 000161 000002
572 011512 105260 000011
573 011516 126037 000011 001372
574 011524 003730
575 011526 000410
576 011530 104401 001207 6$:
577 011534 104401 042503
578 011540 104401 042464
579 011544 000177 167630
580
581
582
583
584
585
586
587
588
589
590
591
592

:*****
TST2: SCOPE
MOV #1,$TIMES ;;DO 1 ITERATION
MOV #2,TSTNM ;:LOAD TEST NUMBER
MOV #STACK,SP ;:INITIAL THE STACK POINTER
MOV #FMTDPB,R0 ;:DPB BLOCK ADDRESS
MOV $CDW1,R1 ;:ADDRESS OF THE LOGICAL DRIVE BLOCK
MOVB DRIVE,(R0) ;:PHYSICAL DRIVE #
MOV $DEV,$SEC(R0) ;:LOAD THE SECTOR #,FROM THE LOGICAL DRIVE NUMBER.
MOV #620,$CYL(R0) ;:LOAD CYLINDER NUMBER
MOV #-256,$WRDM(R0) ;:LOAD NEG WORD COUNT
MOV #BUFFER,$BUF(R0) ;:LOAD BUFFER ADDRESS
MOV #RM.REG,14(R0) ;:ADDRESS TO SAVE ALL RH/RM REG'S
CLRB $TRK(R0) ;:START FROM TRACK 0
MOVB #WRDAT,$COMND(R0) ;:WRITE DATA COMMAND
JSR PC,FILBUF ;:FILL THE BUFFER WITH STANDARD PATTERN
2$: JSR R0,RM05 ;:CALL THE DRIVER
BR 2$ ;:BRANCH IF NOT QUEUE SUCCESSFULLY
3$: TST FMTDPB+16
BEQ 3$ ;:BRANCH IF NOT DONE
MOV #FMTDPB,R0 ;:R0=FMTDPB ADDRESS
JSR PC,PROCESS ;:CHECK THE TERMINATION
TST 16(R0) ;:ERROR FLAG SET ?
BMI 6$ ;:BRANCH IF SO
4$: MOVB #WCKD,$COMND(R0) ;:CHANGE TO THE WRITE CHECK DATA COMMAND
JSR R0,RM05 ;:CALL THE DRIVER
BR 4$ ;:BRANCH IF NOT QUEU SUCCESSFULLY
5$: TST FMTDPB+16
BEQ 5$ ;:DONE ?
MOV #FMTDPB,R0 ;:BRANCH IF NOT DONE
JSR PC,PROCS ;:PROCESS IF ANY ERROR HAPPENS ?
TST 16(R0) ;:ERROR FLAG SET ?
BMI 6$ ;:BRANCH IF SO
6$: MOVB #WRDAT,$COMND(R0) ;:RESET TO WRITE DATA COMMAND
INCB $TRK(R0) ;:INCREMENT TO THE NEXT TRACK
CMPB $TRK(R0),TRKLMT ;:ALL TRACKS DONE ?
BLE 2$ ;:NO
BR TST3 ;:BRANCH TO NEXT TEST
6$: TYPE $CRLF ;:CR-LF
TYPE $HALT1
TYPE $HALTX
JMP @RSTART ;:JUMP TO RESTART

:TEST 3
:WRITE 7 ZONES FOR WRITE TEST IN PASS 2
:WRITE 6 ZONES FOR READ TEST IN PASS2
:
$DEV = LOGICAL DRIVE #
$CDW1 = ADDRESS OF LOGICAL DRIVE HISTORY BLOCK FILE
$CDW2 = SYS NAME
DRIVE = PHYSICAL DRIVE # OF THIS LOGICAL DRIVE
PACK = ENTRY OF TABLE-D
CMCNT = ZONE COUNT
CMSEC = DELTA CYLINDER COUNT
R4 = ENTRY POINTER OF TABLE-D, CANNOT BE DESTORYED.
R0 = ADDRESS OF FMTDPB
    
```

```

593
594
011550 000004
011552 012737 000001 001176
595 011560 012737 000003 001340
596 011566 012706 001100
597 011572 012704 001406
598 011576 013700 001270
599 011602 001404
600 011604 062704 000020
601 011610 005300
602 011612 001374
603 011614 010437 001324
604
605 011620 012700 004506
606 011624 113710 001224
607 011630 112760 000161 000002
608 011636 012760 043032 000006
609 011644 012760 004526 000014
610 011652 012760 177400 000004
611 011660 013701 001270
612 011664 006301
613 011666 016104 005322
614 011672 016002 000006
615 011676 012703 000400
616 011702 010422
617 011704 005303
618 011706 001375
619
620
621 011710 005046
622 011712 005046
623 011714 012746 000006
624 011720 012746 000160
625 011724 012737 000007 001354
626 011732 005037 001356
627 011736 012737 000020 001362
628 011744 012700 004506
629 011750 013704 001324
630 011754 013760 001356 000012
631 011762 105060 000011
632 011766 111460 000010
633 011772 004037 030264
634 011776 004506
635 012000 000774
636 012002 005737 004524
637 012006 001775
638 012010 012700 004506
639 012014 004737 017352
640 012020 005760 000016
641 012024 100453
642 012026 062760 000020 000010
643 012034 122760 000037 000010
644 012042 103353
645 012044 111460 000010
646 012050 105260 000011
647 012054 126037 000011 001372

*****
TST3: SCOPE
MOV #1,$TIMES ;:DO 1 ITERATION
MOV #3,TSTNM ;:LOAD THE TEST NUMBER
MOV #STACK,SP ;:INITIAL THE STACK POINTER
MOV #LOG0,R4 ;:ADDRESS OF TABLE-D
MOV $DEV0,R0 ;:LOGICAL DRIVE #
BEQ 2$ ;:BRANCH IF LOGICAL DRIVE 0
1$: ADD #16.,R4 ;:EACH LOGICAL DRIVE TAKES 16 BYTES IN THE TABLE
DEC R0 ;:DECREMENT THE DRIVE # COUNT
BNE 1$ ;:BRANCH ,UNTIL THE ENTRY IS LOCATED
2$: MOV R4,PACK ;:SAVE THE TABLE-D ENTRY IN PACK
;:SET UP THE FMTDPB BLOCK
MOV #FMTDPB,R0 ;:ADDRESS OF FMTDPB
MOVB DRIVE,(R0) ;:PHYSICAL DRIVE NUMBER
MOVB #WRDAT,$COMND(R0) ;:WRITE DATA COMMAND
MOV #BUFFER,$BUF(R0) ;:BUFFER ADDRESS
MOV #RM.REG.14(R0) ;:ADDRESS TO SAVE ALL RH/RM REG'S
MOV #-256.,$WRDM(R0) ;:NEG WORD COUNT
MOV $DEV0,R1 ;:LOGICAL DRIVE #
ASL R1 ;:WORD INDEX
MOV PSEUDO(R1),R4 ;:LOAD THE DATA PATTERN
MOV $BUF(R0),R2 ;:BUFFER ADDRESS IN R2
MOV #256.,R3 ;:POS WORD COUNT
3$: MOV R4,(R2)+ ;:FULL THE BUFFER WITH SIGLE WORD PATTERN
DEC R3 ;:DECREMENT THE WORD COUNT
BNE 3$ ;:BRANCH,UNTIL IT IS FULL

CLR -(SP) ;:DUMY PAIR OF ZONE COUNT
CLR -(SP) ;:DUMY STARTING CYLINDER NUMBER
MOV #6,-(SP) ;:ZONE COUNT
MOV #112,-(SP) ;:STARTING CYLINDER NUMBER
MOV #7,CMCNT ;:ZONE COUNT
CLR CMCYL ;:STARTING CYLINDER NUMBER
4$: MOV #16.,CMSEC ;:DELTA CYLINDER NUMBER
MOV #FMTDPB,R0 ;:R0 DPB ADDRESS
MOV PACK,R4 ;:R4 ENTRY TO TABLE-D
5$: MOV CMCYL,$CYL(R0) ;:STARTING CYLINDER
CLR $TRK(R0) ;:STARTS FROM TRACK 0
6$: MOVB (R4),$SEC(R0) ;:LOAD SECTOR NUMBER FROM TABLE-D
7$: JSR R0,RM05 ;:CALL THE DRIVER
FMTDPB
BR 7$ ;:BRANCH IF QUEU IS NOTSUCCESSFUL
8$: TST FMTDPB+16 ;:DONE ?
BEQ 8$ ;:BRANCH IF NOT
MOV #FMTDPB,R0
JSR PC,PROCES ;:PROCESS TO CHECK IF ANY ERROR
TST 16(R0) ;:ERROR FLAG SET ?
BMI 9$ ;:BRANCH IF SO
ADD #16.,$SEC(R0) ;:WRITE TWO SECTORS ON ONE CYLINDER
CMPB #31.,$SEC(R0) ;:ALL DONE-TWO SECTORS
BHS 7$ ;:BRANCH IF NOT
MOVB (R4),$SEC(R0) ;:RESTORE SECTOR #
INCB $TRK(R0) ;:INCREMENT TO NEXT TRACK
CMPB $TRK(R0),TRKLMT ;:LAST TRACK ?
    
```



```

648 012062 003743          BLE      7$          ;NO, THEN BRANCH
649 012064 105060 000011   CLRB    $TRK(R0)    ;RESTORE TO TRACK-0
650 012070 005260 000012   INC     $CYL(R0)    ;INCREMENT CYLINDER NUMBER
651 012074 005204          INC     R4          ;INCREMENT TABLE-D ENTRY
652 012076 005337 001362   DEC     CMSEC       ;DECREMENT THE DELTA CYLINDER COUNT
653 012102 001331          BNE     6$          ;BRANCH IF NOT END OF THIS BLOCK
654 012104 062760 000160 000012 ADD     #112,$CYL(R0);INCREMENT THE CYLINDER NUMBER TO NEXT ZONE
655 012112 016037 000012 001356 MOV     $CYL(R0),CMCYL ;INITAIL THE STARTING CYLINDER IN THE BLOCK
656 012120 005337 001354   DEC     CMCNT       ;DECREMENT THE ZONE COUNT
657 012124 001304          BNE     4$          ;LOOPING IF NOT END OF ZONE
658 012126 012637 001356   MOV     (SP)+,CMCYL ;LOAD NEW PAIR OF STARTING CYLINDER
659 012132 012637 001354   MOV     (SP)+,CMCNT ;AND ZONE COUNT
660 012136 005737 001356   TST     CMCYL       ;NOT END YET ?
661 012142 001275          BNE     4$          ;BRANCH IF NOT
662 012144 005737 001354   TST     CMCNT       ;BRANCH IF NOT END
663 012150 001272          BNE     4$          ;LOOPING BACK
664
665 012152 000410          BR      TST4        ;BRANCH TO THE NEXT TEST
666 012154 104401 001207 9$:   TYPE   ,SCLF       ;CR-LF
667 012160 104401 042554   TYPE   ,HALT2
668 012164 104401 042464   TYPE   ,HALTX
669 012170 000177 167204   JMP     @RSTART     ;JUMP TO RESTART
670
671
672 :TEST 4
673 :UPDATE THE PARAMETERS,$CDW1,$DEVM,ASSGN1
674 :DIRECT THE OPERATOR TO DISMOUNT PACK AND LOAD TO OTHER DRIVE
675 :
676 :$CDW2,DRIVE ARE CHANGED BY TEST ONE ONLY AFTER THE TEST LOOPING TO TEST1
677
:*****
TST4: SCOPE
678 012174 000004          MOV     #1,$TIMES   ;;DO 1 ITERATION
679 012176 012737 000001 001176 MOV     #4,TSTNM    ;LOAD THE TEST NUMBER
680 012204 012737 000004 001340 MOV     #STACK,SP   ;LOAD THE STACK POINTER
681 012212 012706 001100          MOV     #IDLEX,@RMVEC ;RESET THE INTERRUPT VECTOR
682 012216 012777 017344 015254 CLR     @RMVEC+2    ;CLEAR THE INTERRUPT LEVEL
683 012224 005077 015252          TYPE   ,SCLF       ;CR-LF
684 012230 104401 001207          TYPE   ,MESG12    ;UNLOAD DRIVE
685 012234 104401 041650          TYPE   ,SCDW2     ;SYS-NAME(A - H)
686 012240 013746 001274          MOV     DRIVE,-(SP);PHYSICAL DRIVE #
687 012244 104403          TYPOS
688 012250 104403          .BYTE  1
689 012252 001          .BYTE  0
690 012253 000          TYPE   ,MESG13
691 012254 104401 041666          MOV     #1,R1
692 012260 012701 000001          CLR     R2
693 012264 005002          1$:   CMP     R2,$DEVM   ;LOCATE THE CORESPONDING BIT MAP
694 012266 020237 001270          BEQ    2$          ;BRANCH IF LOCATED
695 012272 001404          CLC
696 012274 000241          ROL    R1          ;LOCATE NEXT DRIVE
697 012276 006101          INC    R2
698 012300 005202          BR     1$          ;NEXT DRIVE #
699 012302 000771          BIC    R1,ASSGN1  ;LOCATE THE BIT MAP
700 012310 001410          BEQ    4$          ;DEASSIGN THE LOGICAL DRIVE FOR PASS 1
701 012312 005202          3$:   INC    R2          ;NO MORE DRIVES
702 012314 006302          ASL    R2          ;GET NEXT LOGICAL DRIVE #
702 012316 016237 004444 001272 MOV     BLKADR(R2),$CDW1 ;WORD INDEX
;LOAD THE NEW DPB ADDRESS
    
```

```

703 012324 006202          ASR      R2          ;RESTORE R2
704 012326 010237 001270  MOV      R2,$DEV  ;LOAD THE NEW LOGICAL DRIVE #
705
706 012332 104411          4$:    RDLIN          ;WAIT UNTIL IT IS DONE
707 012334 012605          MOV      (SP)+,R5  ;LOCATE THE INPUT LINE
708 012336 105715          TSTB    (R5)       ;TERMINATOR ?
709 012340 001374          BNE     4$         ;BRANCH IF NOT
710 012342 005737 002010  TST     ASSGN1     ;OTHER DRIVES ?
711 012346 001002          BNE     5$         ;BRANCH IF MORE DRIVES IN TEST
712 012350 000137 012360  JMP     XPASS2     ;BRANCH TO PASS 2
713 012354 000137 010720  5$:    JMP     TST1     ;JUMP TO TEST 1
714
715
716          ;XPASS2      INITILIZE FOR PASS 2 TEST
717          :
718          :          $CDW1    = ADDRESS OF THE CURRENT LOGICAL DRIVE HISTORY FILE
719          :          $CDW2    = SYSTEM NAME A THROUGH H
720          :          $DEV     = CURRENT LOGICAL DRIVE # 0 TO 15.
721          :          ASSGN2   = ASSIGNED LOGICAL DRIVE FOR PASS 2
722          :          ASNLST   = ASSIGNED LOGICAL DRIVE
723          :          DRIVE    = PHYSICAL DRIVE # OF CURRENT RH/RM SYSTEM
724
725 012360 005737 002012  XPASS2: TST     ASSGN2 ;ANYTHING IN TEST FOR PASS 2
726 012364 001002          BNE     1$         ;YES, THEN GO ON
727 012366 000137 017046  JMP     XEND2     ;JUMP TO END OF PASS 2
728
729 012372 005037 001270          1$:    CLR     $DEV     ;START FROM LOGICAL DRIVE 0
730 012376 013737 004444 001272  MOV     BLKADR,$CDW1 ;ADDRESS OF LOGICAL BLOCK DRIVE 0
731 012404 112737 000101 001274  MOVB   #'A,$CDW2    ;LOAD SYSTEM NAME 'A'
732 012412 013737 002014 027476  MOV     SYSADR, RMADR ;LOAD SYSTEM-A ADDRESS TO DRIVER
733 012420 013737 002016 027500  MOV     SYSADR+2, RMVEC ;LOAD SYSTEM-A VECTOR TO DRIVER
734 012426 013701 001272          MOV     $CDW1,R1    ;R1=ADDRESS OF LOGICAL BLOCK
735 012432 012777 017344 015040  MOV     #IDLEX,@RMVEC ;RESET THE INTERRUPT VECTOR
736 012440 005077 015036          CLR     @RMVEC+2    ;CLEAR THE INTERRUPT LEVEL
737 012444 005037 001376          CLR     FAULT      ;RESET THE NOT COMPATIBLE FLAG
738 012450 104401 001207          TYPE   , $CRLF     ;CR-LF
739 012454 104401 041734          TYPE   , MSG14     ;START THE PASS 2
740 012460 104401 042242          TYPE   , MSG6      ;ON
741 012464 104401 041354          TYPE   , MSG1      ;SUB-SYSTEM
742 012470 104401 042222          TYPE   , QUOTM     ;TYPE "" QUOTATION MARK
743 012474 104401 001274          TYPE   , $CDW2     ;SYS-NAME(A TO H)
744 012500 104401 042222          TYPE   , QUOTM     ;TYPE "" QUOTATION MARK
745 012504 111137 001224          MOVB   (R1),DRIVE  ;LOCATE THE PHYSICAL DRIVE #
746 012510 104401 001207          TYPE   , $CRLF     ;CR-LF
747 012514 104401 041423          TYPE   , MSG4      ;MOUNT PACK ON DRIVE
748 012520 104401 001274          TYPE   , $CDW2     ;SYSTEM NAME
749 012524 111146          MOVB   (R1),-(SP)
750 012526 104403          TYPOS
751 012530          .BYTE  1
752 012531          .BYTE  0
753 012532 104401 042330          TYPE   , BLNKS1    ;TYPE 1 BLANK
754 012536 104401 041451          TYPE   , MSG8      ;AND LOAD
755 012542 104401 001207          TYPE   , $CRLF     ;CR-LF
756 012546 104401 041464          TYPE   , MSG9      ;TYPE <CR> WHEN DRIVE IS READY
757
758 012552 104411          2$:    RDLIN
759 012554 012602          MOV     (SP)+,R2  ;LOCATE THE READ IN LINE
  
```

```

760 012556 105712          TSTB   (R2)          ;CARRIAGE RETURN ?
761 012560 001374          BNE    2$           ;BR IF NO
762 012562 004737 027514    JSR    PC,RMINIT
763 012566 012737 177777 027436  MOV    #-1,SAVEFG   ;SAVE ALL RH/RM REGISTERS
764 012574 012737 177777 027440  MOV    #-1,SEEKFG   ;DON'T DO IMPLY SEEK
765 012602 013700 001224          MOV    DRIVE,RO     ;LOAD THE PHYSICAL DRIVE NUMBER
766 012606 105760 027350          TSTB   DRVSTA(RO)   ;DRIVE EXISTS AND ON-LINE ?
767 012612 003421          BLE    4$           ;BRANCH IF NOT
768 012614 105760 027360          TSTB   DRVTyp(RO)   ;CHECK DRIVE TYPE
769 012620 003416          BLE    4$           ;BR IF NOT AN RM05/3/2
770 012622 116037 027360 001402 3$:  MOVB   DRVTyp(RO),DTYP ;GET DRIVE TYPE TO BE TESTED
771 012630 012737 000022 001372    MOV    #18.,TRKLMT  ;GET LAST TRACK FOR AN RM05
772 012636 122760 000007 027360    CMPB   #7,DRVTyp(RO) ;IS DRIVE AN RM05 ?
773 012644 001412          BEQ    5$           ;BR IF YES
774 012646 012737 000004 001372    MOV    #4.,TRKLMT   ;GET LAST TRACK FOR AN RM03/2
775 012654 000406          BR     5$
776
777 012656 104401 001207          4$:   TYPE   , $CRLF      ;CR-LF
778 012662 104401 041520          TYPE   , MSG10      ;DRIVE IS NOT READY
779 012666 000137 010704          JMP    ENDX1
780 012672
781
782
783
784
785
786
787
788
789
790
791
:
:   $CDW1 = ADDRESS OF CURRENT LOGICAL DRIVE BLOCK, ONLY CHANGED BY TST9
:   $CDW2 = SYSTEM-NAME, ONLY CHANGED BY TEST 5
:   $DEV# = CURRENT LOGICAL DRIVE #
:   ASSGN2 = ASSIGNED LOGICAL DRIVE'S BIT MAP FOR PASS 2
:   DRIVE  = PHYSICAL DRIVE #
:   ASNLST = ASSIGNED LOGICAL DRIVE IN THE TEST
:
: IN TEST 5 DIRECT OPERATOR TO CHANGE, LOAD THE PACK
:
: *****
TST5: SCOPE
MOV    #1,$TIMES      ;;DO 1 ITERATION
MOV    #5,TSTNM       ;LOAD THE TEST NUMBER
MOV    #STACK,SP      ;LOAD THE STACK POINTER
CMP    $CDW1,BLKADR   ;LOGICAL DRIVE 0
BEQ    6$             ;THEN EXIT
MOV    $CDW1,R1       ;ADDRESS OF THE HISTORY FILE
CMPB   $$SYSNM(R1),$CDW2 ;ON THE SAME SUB-SYSTEM
BEQ    1$             ;THEN DON'T UPDATE SYSTEM ADDRESS
MOVB   $$SYSNM(R1),$CDW2
MOV    #IDLEX,@RMVEC  ;RESET THE INTERRUPT VECTOR
CLR    @RMVEC+2       ;CLEAR THE INTERRUPT LEVEL
MOV    $CDW2,RO       ;LOCATE SYSTEM ADDRESS TABLE
DEC    RO             ;ADJUST FOR INDEX FORM 0
BIC    #177760,RO     ;LEFT ON FOUR BITS
ASL    RO
ASL    RO             ;INDEX FOR TWO WORD
MOV    SYSADR(RO),RMADR ;SYSTEM ADDRESS
MOV    SYSADR+2(RO),RMVEC ;SYSTEM INTERRUPT VECTOR
TYPE   , $CRLF      ;CR-LF
TYPE   , MSG14      ;START THE PASS 2
TYPE   , MSG6       ;ON
TYPE   , MSG1       ;SUB-SYSTEM
TYPE   , QUOTM      ;TYPE "" QUOTATION MARK
TYPE   , $CDW2      ;SYS-NAME(A TO H)
    
```

```

815 013042 104401 042222
816 013046 111137 001224
817 013052 104401 001207
818 013056 104401 041423
819 013062 104401 001274
820 013066 111146
821 013070 104403
822 013072 001
823 013073 000
824 013074 104401 042330
825 013100 104401 041451
826 013104 104401 001207
827 013110 104401 041464
828
829 013114 104411
830 013116 012602
831 013120 105712
832 013122 001374
833 013124 004737 027514
834 013130 012737 177777 027436
835 013136 012737 177777 027440
836 013144 013700 001224
837 013150 105760 027350
838 013154 003426
839 013156 105760 027360
840 013162 003423
841 013164 122737 000007 001402
842 013172 001005
843 013174 122760 000007 027360
844 013202 001420
845 013204 000404
846 013206 122760 000007 027360
847 013214 001013
848 013216 104401 001207
849 013222 104401 042734
850 013226 000177 166146
851
852 013232 104401 001207
853 013236 104401 041520
854 013242 000613
855 013244
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871

```

1\$: TYPE ,QUOTM ;TYPE "" QUOTATION MARK
 MOV (R1),DRIVE ;LOCATE THE PHYSICAL DRIVE #
 TYPE ,SCRLF ;CR-LF
 TYPE ,MESG4 ;MOUNT PACK ON DRIVE
 TYPE ,SCDW2 ;SYSTEM NAME
 MOV (R1),-(SP)
 TYPOS
 .BYTE 1
 .BYTE 0
 TYPE ,BLNKS1 ;TYPE 1 BLANK
 TYPE ,MESG8 ;AND LOAD
 TYPE ,SCRLF ;CR-LF
 TYPE ,MESG9 ;TYPE <CR> WHEN DRIVE IS READY

2\$: RDLIN
 MOV (SP)+,R2 ;LOCATE THE READ IN LINE
 TSTB (R2) ;CARRIAGE RETURN ?
 BNE 2\$;BR IF NO
 JSR PC,RMINIT
 MOV #-1,SAVEFG ;SAVE ALL RH/RM REGISTERS
 MOV #-1,SEEKFG ;DON'T DO IMPLY SEEK
 MOV DRIVE,R0 ;LOAD THE PHYSICAL DRIVE NUMBER
 TSTB DRVSTA(R0) ;DRIVE EXISTS AND ON-LINE ?
 BLE 5\$;BRANCH IF NOT
 TSTB DRVTYP(R0) ;CHECK DRIVE TYPE
 BLE 5\$;BR IF NOT AN RM05/3/2
 CMPB #7,DTYP ;WHAT WAS FIRST DRIVE TESTED ?
 BNE 3\$;BRANCH IF AN RM02 OR RM03, ELSE
 CMPB #7,DRVTYP(R0) ;SEE IF DRIVE IS STILL AN RM05.
 BEQ 6\$;BR IF YES
 BR 4\$;ERROR ENCOUNTERED
 CMPB #7,DRVTYP(R0) ;SEE IF DRIVE IS STILL AN RM02 OR RM03.
 BNE 6\$;BR IF YES

3\$: CMPB #7,DRVTYP(R0)
 BNE 6\$;BR IF YES

4\$: TYPE ,SCRLF ;CR-LF
 TYPE ,NOTST ;CANNOT SELECT RM03/2'S AND RM05'S TOGETHER
 JMP @RSTART ;JUMP TO RESTART

5\$: TYPE ,SCRLF ;CR-LF
 TYPE ,MESG10 ;DRIVE IS NOT READY
 BR TST5 ;TRY AGAIN

6\$:

: DRIVE = PHYSICAL DRIVE NUMBER
 : \$CDW1 = DPB BLOCK OF THIS LOGICAL DRIVE
 : \$DEV# = LOGICAL DRIVE #
 : \$CDW2 = 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:
 : R0 = ADDRESS OF DPB FIELD INTO DRIVER HANDLER=FMTDPB
 : R1 = ADDRESS OF THE HISTORY FILE BLOCK OF THE LOGICAL DRIVE=\$CDW1

:IN TEST 6, EACH LOGICAL DRIVE WRITES 7 CYLINDERS ON EACH TRACK WITH
 :A UNIQUE DATA PATTERN. THESE 7 CYLINDERS HAVE BEEN WRITTEN BY OTHER
 :DRIVES IN PASS 1.

```

872                                     ;THEN, THIS LOGICAL DRIVE EXECUTES 'WRITE CHECK DATA' TO SEE IF THIS
873                                     ;LOGICAL DRIVE CAN OVER-WRITE ALL DATA WRITTEN BY OTHER DRIVES.
874
875                                     ;THESE 7 CYLINDERS ARE SPECIFIED AS $DEVN, $DEVN+128, $DEVN+256, $DEVN+384,
876                                     ;$DEVN+512, $DEVN+640 AND $DEVN+768.
877
878                                     ;THE OVER-WRITE TEST IS PERFORMED WITH OFFSETS IN BOTH DIRECTIONS.
879
880                                     ;:*****
013244 000004 TST6: SCOPE
013246 012737 000001 001176      MOV #1,$TIMES      ;;DO 1 ITERATION
881 013254 012737 000006 001340      MOV #6,TSTNM      ;LOAD TEST NUMBER
882 013262 012706 001100              MOV #STACK,SP     ;INITIAL THE STACK POINT
883 013266 012703 002056              MOV #OVWNO,R3     ;1 ST ADDRESS TO CLEAR
884 013272 012704 003216              MOV #RDNO,R4      ;LAST ADDRESS+2
885 013276 005023 1$: CLR (R3)+      ;RESET ALL SCORE BOARD
886 013300 020403              CMP R4,R3         ;ALL LOCATIONS ARE CLEARED ?
887 013302 101375              BHI 1$           ;BRANCH IF NOT
888
889 013304 012700 004506              MOV #FMTDPB,R0    ;SET UP STARTING ADDRESS OF DPB
890 013310 013701 001272              MOV $CDW1,R1      ;HISTORY FILE BLOCK
891 013314 113710 001224              MOV#B DRIVE,(R0)  ;LOAD THE PHYSICAL DRIVE #
892 013320 012760 160000 000004      MOV #-8192,$WRDM(R0) ;LOAD THE WORD COUNT (ONE TRACK)
893 013326 005060 000010              CLR $SEC(R0)      ;START FROM TRACK = 0, SECTOR = 0
894 013332 112760 000161 000002      MOV#B #WRDAT,$COMND(R0) ;WRITE DATA COMMAND
895 013340 013760 001270 000012      MOV $DEVN,$CYL(R0) ;LOAD THE STARTING CYLINDER
896 013346 012760 043032 000006      MOV #BUFFER,$BUF(R0) ;LOAD THE BUFFER ADDRESS
897 013354 012760 004526 000014      MOV #RM.REG,14(R0) ;REG'S SAVE ADDRESS
898 013362 013702 001270              MOV $DEVN,R2      ;LOCATE THE DATA PATTERN
899 013366 006302              ASL R2            ;WORD INDEX
900 013370 016202 005322              MOV PSEUDO(R2),R2 ;LOAD R2 WITH THE DATA PATTERN POINTER
901 013374 016003 000006              MOV $BUF(R0),R3   ;BUFFER ADDRESS
902 013400 012704 020000              MOV #8192,R4      ;WORD COUNT
903 013404 010223 2$: MOV R2,(R3)+      ;FILL THE BUFFER
904 013406 005304              DEC R4            ;DECREMENT WORD CTR
905 013410 001375              BNE 2$           ;BRANCH IF NOT DONE
906
907                                     ;START TO WRITE ONE CYLINDER ON EACH TRACK OF EACH WRITE CURRENT ZONE
908                                     ;:(7 ZONES ON EACH PACK)
909
910 013412 004037 030264 3$: JSR R0,RM05      ;CALL THE DRIVER
911 013416 004506              FMTDPB           ;PARAMETER BLOCK
912 013420 000774              BR 3$            ;BRANCH IF QUEUE FAIL
913 013422 005737 004524 4$: TST FMTDPB+16    ;WRITE COMMAND DONE ?
914 013426 001775              BEQ 4$          ;NO, THEN WAIT
915 013430 012700 004506              MOV #FMTDPB,R0    ;PROCESS IF ANY ERROR
916 013434 004737 017352              JSR PC,PROCES
917 013440 005760 000016              TST 16(R0)       ;ERROR FLAG SET ?
918 013444 100010              BPL 22$         ;BRANCH IF NOT
919 013446 004737 027514              JSR PC,RMINIT    ;INITIAL THE DRIVE
920 013452 012737 177777 027436      MOV #-1,SAVEFG
921 013460 012737 177777 027440      MOV #-1,SEEKFG
922 013466 105260 000011 22$: INCB $TRK(R0)    ;NEXT TRACK
923 013472 126037 000011 001372      CMPB $TRK(R0),TRKLMT ;LAST TRACK IS DONE ?
924 013500 003744              BLE 3$          ;NO, THEN BRANCH
925 013502 105060 000011              CLRB $TRK(R0)    ;RESET TRACK NUMBER
926 013506 062760 000200 000012      ADD #128,$CYL(R0) ;MOVE TO NEXT ZONE
  
```

```

927 013514 022760 001417 000012      CMP      #783.,$CYL(R0)  ;LAST ZONE IS DONE ?
928 013522 103333      BHIS     3$              ;BRANCH IF NOT
929
930      ;RESET THE FMTDPB BLOCK AND EXECUTE WRITE-CHECK COMMAND TO DETECT ANY
931      ;COMPATIBLE PROBLEM. THE FOLLOWING SUBROUTINE ARE CALLED SCORE, OFFST
932      ;AND MAKEUP.
933
934 013524 005037 004504      CLR      OFFCOD          ;SET NEGATIVE OFFSET DIRECTION FLAG
935 013530 012700 004506      LOOP1:  MOV      #FMTDPB,R0  ;RO=FMTDPB ADDRESS
936 013534 005060 000010      CLR      $SEC(R0)        ;START FROM SECTOR 0,TRACK 0
937 013540 013760 001270 000012      MOV      $DEVN,$CYL(R0)  ;STARTING CYLINDER NUMBER
938      ;TOTAL 7 CYLINDERS ON ONE TRACK
939 013546 112760 000151 000002      LOOP2:  MOVVB   #WCKD,$COMND(R0) ;WRITE-CHECK-DATA COMMAND
940 013554 012760 043032 000006      MOV      #BUFFER,$BUF(R0) ;RESET BUFFER ADDRESS
941 013562 012760 004526 000014      MOV      #RM.REG,14(R0)  ;ADDRESS TO SAVE RH/RM REG'S
942 013570 004037 030264      1$:     JSR      R0,RM05     ;CALL THE DRIVER
943 013574 004506      FMTDPB  ;PARAMETER BLOCK ADDRESS
944 013576 000774      BR      1$              ;BRANCH IF QUEUE FAILURE
945 013600 005737 004524      2$:     TST      FMTDPB+16   ;TEST IF COMMAND IS DONE ?
946 013604 001775      BEQ     2$              ;BRANCH , IF NOT
947 013606 012700 004506      MOV      #FMTDPB,R0     ;LOAD THE PARAMETER BLOCK ADDRESS
948 013612 004737 017352      JSR     PC,PROCES       ;REPORT IF ANY ERROR
949 013616 004737 014734      JSR     PC,LABAD        ;LOCATE STARTING AND ENDING SECTORS
950 013622 005760 000016      TST     16(R0)         ;ANY ERROR ?
951 013626 100011      BPL     3$              ;BRANCH IF NONE
952 013630 004737 027514      JSR     PC,RMINIT       ;INITIAL THE SYSTEM
953 013634 012737 177777 027436      MOV     #-1,SAVEFG
954 013642 012737 177777 027440      MOV     #-1,SEEKFG
955 013650 000414      BR      5$              ;NOT UPDATE THE SCORE
956 013652 004737 014514      3$:     JSR     PC,SCORE     ;INCREMENT SCORE
957 013656 005760 000022      4$:     TST     $RMWC(R0)  ;WORD COUNT = 0 /
958 013662 001407      BEQ     5$              ;BRANCH, IF WORD COUNT IS 0
959 013664 116060 000026 000010      MOVVB   $RMDA(R0),$SEC(R0) ;UPDATE STARTING SECTOR
960 013672 016060 000022 000004      MOV     $RMWC(R0),$WRDM(R0) ;UPDATE WORD COUNT
961 013700 000733      BR      1$              ;TO READ THE REST SECTORS
962
963      ;THE FOLLOWING CODING TEST THE COMPATIBLE PROBLEM IN OFFSET MODE OFFCOD = 0
964      ;(NEGATIVE) AND OFFCOD = 1(POSITIVE).
965
966 013702 012700 004506      5$:     MOV     #FMTDPB,R0  ;RESET THE DPB BLOCK
967 013706 012760 160000 000004      MOV     #-8192,$WRDM(R0) ;FULL TRACK WORD COUNT
968 013714 105060 000010      CLRB   $SEC(R0)        ;RESET TO SECTOR 0,TRACK NOT CHANGED
969 013720 012760 043032 000006      MOV     #BUFFER,$BUF(R0) ;BUFFER ADDRESS
970 013726 012760 004526 000014      MOV     #RM.REG,14(R0)  ;ADDRESS TO SAVE ALL RH/RM REG'S
971 013734 004537 014144      6$:     JSR     R5,OFFST   ;CALL OFFSET
972 013740 000443      BR      10$            ;BRANCH TO NEXT CYLINDER,IF OFFSET FAILS
973 013742 004737 014376      JSR     PC,MAKEUP       ;CALL WRITE CHECK IN OFFSET MODE
974 013746 005737 004524      7$:     TST     FMTDPB+16  ; OFFSET WRITE CHECK IS DONE ?
975 013752 001775      BEQ     7$              ;BRANCH IF NOT
976
977 013754 012700 004506      MOV     #FMTDPB,R0     ;LOAD THE DPB ADDRESS
978 013760 004737 017352      JSR     PC,PROCES       ;REPORT , IF ANY ERROR
979 013764 004737 014734      JSR     PC,LABAD        ;LOCATE STARTINF AND ENDING SECTORS
980 013770 005760 000016      TST     16(R0)         ;ANY ERROR ?
981 013774 100011      BPL     8$              ;BRANCH, IF NONE
982 013776 004737 027514      JSR     PC,RMINIT       ;INITIAL THE SYSTEM
983 014002 012737 177777 027436      MOV     #-1,SAVEFG
    
```

```

984 014010 012737 177777 027440      MOV    #-1,SEEKFG
985 014016 000414                      BR     10$      ;NOT UPDATE THE SCORE
991 014020 004737 014514      8$:   JSR    PC,SCORE      ;UPDATE THE TEST SCORE
992 014024 005760 000022      9$:   TST    $RMWC(R0)     ;WORD CTR = 0 ?
993 014030 001407                      BEQ    10$      ;IF WORD CTR = 0 ,BRANCH TO NEXT OP
994 014032 116060 000026 000010    MOVVB  $RMDA(R0),$SEC(R0) ;UPDATE THE NEW STARTING ADDRESS
995 014040 016060 000022 000004    MOV    $RMWC(R0),$WRDM(R0) ;UPDATE THE NEW WORD COUNT
996 014046 000732                      BR     6$
997 014050 062760 000200 000012    10$:  ADD    #128.,$CYL(R0)   ;ADJUST CYLINDER ADDRESS TO NEXT ZONE
998 014056 022760 001417 000012    CMP    #783.,$CYL(R0)   ;ALL 7 ZONES HAVE BEEN TESTED ?
999 014064 103402                      BLO   11$      ;BRANCH, IF ALL DONE
1000 014066 000137 013546          JMP    LOOP2         ;TO NEXT WRITE CURRENT ZONE
1001 014072 013760 001270 000012    11$:  MOV    $DEVN,$CYL(R0)   ;RESET CYLINDER ADDRESS
1002 014100 105260 000011          INCB  $TRK(R0)        ;INCREMENT TO NEXT TRACK
1003 014104 126037 000011 001372    CMPB  $TRK(R0),TRKLMT  ;ALL TRACKS ARE TESTED ?
1004 014112 003002                      BGT   12$      ;BRANCH IF ALL DONE
1005 014114 000137 013546          JMP    LOOP2         ;TO NEXT TRACK
1006 014120 005737 004504          12$:  TST    OFFCOD        ;FINISHING TEST THE POSITIVE OFFSET ?
1007 014124 001005                      BNE   13$      ;BRANCH IF ALL DONE
1008 014126 012737 000001 004504    MOV    #1,OFFCOD      ;SET POSITIVE OFFSET DIRECTION FLAG
1009 014134 000137 013530          JMP    LOOP1        ;RESTART LOCATION
1010 014140 000137 015040          13$:  JMP    TST7         ;BRANCH TO NEXT TEST
1011
1012      ;OFFST ROUTINE
1013      ;OFFSET THE HEAD IN THE DIRECTION TOWARD SPINDLE OR AWAY FROM THE SPINDLE
1014      :      OFFCOD = 1, TOWARD SPINDLE (POSITIVE)
1015      :      OFFCOD = 0, AWAY FROM SPINDLE (NEGATIVE)
1016      :      DRIVE  = PHYSICAL DRIVE NUMBER
1017      :      R0     = DPB ADDRESS
1018      :      RETRY  = 3 TIMES
1019
1020      ;CALL
1021      :      JSR    R5,OFFST
1022      :      RET1  OFFSET FAIL RETURN ADDRESS
1023      :      RET2  OFFSET SUCCESSFUL RETURN
1024
1025 014144 010146      OFFST: MOV    R1,-(SP)      ;SAVE ALL REGISTERS
1026 014146 010246      MOV    R2,-(SP)
1027 014150 010346      MOV    R3,-(SP)
1028 014152 010446      MOV    R4,-(SP)
1029 014154 012700 004506      MOV    #FMTDPB,R0    ;R0 DPB ADDRESS
1030 014160 116046 000002      MOVVB  $COMND(R0),-(SP) ;SAVE THE I/O COMMAND
1031 014164 113710 001224      MOVVB  DRIVE,(R0)    ;LOAD THE DRIVE NUMBER
1032 014170 112760 000117 000002    MOVVB  #117,$COMND(R0) ;RETURN TO CENTER COMMAND
1033 014176 004037 030264      JSR    R0,RM05      ;CALL THE DRIVE HANDLE
1034 014202 004506      FMTDPB
1035 014204 000465      BR     6$          ;BRANCH IF QUEUE FAILS
1036 014206 005737 004524          1$:   TST    FMTDPB+16   ;COMMAND DONE ?
1037 014212 001775      BEQ    1$          ;BRANCH IF NOT
1038 014214 100461      BMI   6$          ;BRANCH IF ERROR EXIST
1039 014216 013746 177776      MOV    @#PS,-(SP)   ;SAVE THE PSW
1040 014222 012737 000240 177776    MOV    #<5*32.>,@#PS ;LOAD PS 5
1041 014230 012700 004506      MOV    #FMTDPB,R0  ;DPB ADDRESS
1042 014234 013704 027476      MOV    RMADR,R4     ;MUSS BUS ADDRESS
1043 014240 013701 001224      MOV    DRIVE,R1    ;DRIVE NUMBER
1044 014244 010164 000010      MOV    R1,RMCS2(R4) ;LOAD THE DRIVE NUMBER INTO CONTROLLER
1045 014250 016064 000012 000034    MOV    $CYL(R0),RMDC(R4) ;CYLINDER NUMBER

```

```

1046 014256 016064 000010 000006      MOV      $SEC(R0),RMDA(R4) ;SECTOR AND TRACK NUMBER
1047 014264 016064 000004 000002      MOV      $WRDM(R0),RMWC(R4) ;WORD COUNT
1048 014272 016064 000006 000004      MOV      $BUF(R0),RMBA(R4) ;BUFFER ADDRESS
1049 014300 012637 177776      MOV      (SP)+,@#PS      ;LOAD THE PSW BACK
1050 014304 112760 000200 000001      MOVVB   #BIT7,$FMT(R0)  ;LOAD THE OFFSET DIRECTION
1051 014312 005737 004504      TST     OFFCOD          ;NEG ?
1052 014316 001003      BNE     2$             ;BRANCH IF NOT
1053 014320 112760 000000 000001      MOVVB   #0,$FMT(R0)    ;CHANGE TO OTHER DIRECTION
1054 014326 112760 000115 000002 2$:      MOVVB   #115,$COMND(R0);LOAD THE OFFSET COMMAND
1055 014334 004037 030264      JSR     R0,RM05        ;CALL THE DRIVE HANDLE
1056 014340 004506      FMTDPB
1057 014342 000406      BR      6$             ;BRANCH IF QUEUE FAILS
1058 014344 005737 004524      3$:     TST     FMTDPB+16 ;OFFSET DONE ?
1059 014350 001775      BEQ     3$             ;BRANCH IF SO
1060 014352 100402      BMI     6$             ;BRANCH IF ERROR
1061 014354 062705 000002      ADD     #2,R5          ;ADJUST RETURN ADDRESS
1062 014360      6$:
1063 014360 112637 004510      MOVVB   (SP)+,FMTDPB+$COMND ;RESTORE THE I/O COMMAND
1064 014364 012604      MOV     (SP)+,R4      ;RESTORE REG
1065 014366 012603      MOV     (SP)+,R3
1066 014370 012602      MOV     (SP)+,R2
1067 014372 012601      MOV     (SP)+,R1
1068 014374 000205      RTS     R5             ;EXIT
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
    
```

```

:MAKEUP ROUTINE
:THIS ROUTINE ISSUES A WRITE CHECK OR READ COMMAND TO THE SELECTED DRIVE
:IN OFFSET MODE.
:AND SET UP THE FOLLOWING PARAMETERS
:
:      DTUW   = PHYSICAL DRIVE NUMBER
:      TRNSWT = FMTDPB
:      DRVACT = 1
:      TIMER  = 1 SECOND
:CALL
:      JSR    PC,MAKEUP
:      RET
:
:MAIN PURPOSE OF THIS ROUTINE,TO EXECUTE A COMMAND WHILE ASSURE THAT
:THIS READ OR WRITE-CHECK COMMAND BEING EXECUTED IN OFFSET MODE.
:ROUTINES USED TD,SC,STO,( IN DRIVE HANDLER )
    
```

```

1089 014376 010146      MAKEUP: MOV     R1,-(SP)
1090 014400 010246      MOV     R2,-(SP)
1091 014402 010446      MOV     R4,-(SP)
1092 014404 012702 000151      MOV     #WCKD,R2      ;WRITE CHECK DATA IN TEST 6
1093 014410 022737 000010 001340      CMP     #10,TSTNM     ;ON TEST 8 ?
1094 014416 001002      BNE     1$             ;BRANCH IF NOT
1095 014420 012702 000171      MOV     #RDDAT,R2     ;READ DATA COMMAND IN TEST 8
1096 014424 005037 004524      1$:     CLR     FMTDPB+16    ;CLEAR THE STATUS WORD
1097 014430 110237 004510      MOVVB   R2,FMTDPB+$COMND ;LOAD THE COMMAND INTO PDB
1098 014434 013737 001224 027462      MOV     DRIVE,DTUW    ;ACTIVE DRIVE NUMBER
1099 014442 012737 004506 027410      MOV     #FMTDPB,TRNSWT ;TRANSFER UNDERWAY FLAG
1100 014450 013701 001224      MOV     DRIVE,R1      ;DRIVE NUMBER
1101 014454 013704 027476      MOV     RMADR,R4      ;RH/RM BASE ADDRESS
1102 014460 112761 000001 027340      MOVVB   #1,DRVACT(R1) ;ACTIVE DRIVE FLAG
    
```



```

1103 014466 006301          ASL      R1          ;WORD INDEX
1104 014470 012761 060000 027442  MOV     #60000,TIMER(R1) ; ONE SECOND TIMER
1105 014476 006201          ASR      R1
1106 014500 010264 000000 .    MOV     R2,RMCS1(R4)    ;ISSURE WRITE CHECK OR READ COMMAND
1107 014504 012604          MOV     (SP)+,R4        ;RESTORE REG 4, 1
1108 014506 012602          MOV     (SP)+,R2
1109 014510 012601          MOV     (SP)+,R1
1110 014512 000207          RTS      PC            ;EXIT
1111
1112          ;SCORE ROUTINE
1113          ;ROUTINE TO UPDATE THE TEST SCORE
1114          ;TABLEX = ADDRESS OF CURRENT SCORE BOARD
1115          ;$DEV# = LOGICAL DRIVE #
1116          ;DRIVE = PHYSICAL DRIVE NUMBER
1117          ;CMSEC = END SECTOR ADDRESS
1118          ;STARSC = START SECTOR ADDRESS
1119
1120          ;CALL
1121          ;JSR      PC,SCORE
1122          ;RET
1123
1124          SCORE:
1124 014514 010146          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
1124 014514 010146          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
1124 014516 010246          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
1124 014520 010346          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
1124 014522 010446          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
1125 014524 023737 001362 001360  CMP     CMSEC,STARSC   ;CORRECT START AND STOP ADDRESSES
1126 014532 003473          BLE     9$             ;BRANCH IF NOT
1127 014534 022737 000010 001340  CMP     #10,TSTNM      ;ON TEST 8
1128 014542 001011          BNE     2$             ;BRANCH IF NOT (MUST BE TEST 6)
1129 014544 005737 004504          TST     OFFCOD         ;NEGATIVE OFFSET ?
1130 014550 001403          BEQ     1$             ;BRANCH IF NEGATIVE OFFSET
1131 014552 012703 003676          MOV     #RDPO,R3       ;SCORE BOARD ADDRESS
1132 014556 000413          BR      4$
1133 014560 012703 003216          1$:    MOV     #RDNO,R3     ;SCORE BOARD ADDRESS
1134 014564 000410          BR      4$
1135 014566 005737 004504          2$:    TST     OFFCOD         ;NEGATIVE OFFSET
1136 014572 001403          BEQ     3$             ;BRANCH,IF NEGATIVE OFFSET
1137 014574 012703 002536          MOV     #OVWPO,R3     ;SCORE BOARD ADDRESS
1138 014600 000402          BR      4$
1139 014602 012703 002056          3$:    MOV     #OVWNO,R3     ;SCORE BOARD ADDRESS
1140 014606 113702 004517          4$:    MOV     FMTDPB+$TRK,R2 ;LOAD THE TRACK NUMBER
1141 014612 005702          TST     R2             ;ON TRACK 0
1142 014614 001404          BEQ     6$             ;BRANCH IF IT IS
1143 014616 062703 000020          5$:    ADD     #16.,R3       ;EACH SCORE BOARD TAKES 16 BYTES
1144 014622 005302          DEC     R2             ;LOCATED ?
1145 014624 001374          BNE     5$             ;BRANCH IF NOT
1146 014626 010337 002054          6$:    MOV     R3,TABLEX     ;STORE THE TABLE STARTING ADDRESS
1147 014632 010301          MOV     R3,R1         ;RE ASSIGN REGISTERS
1148 014634 013702 001270          MOV     $DEV# ,R2     ;LOGICAL DRIVE #
1149 014640 013703 001360          MOV     STARSC,R3     ;START SECTOR
1150 014644 116204 004424          MOV     INDST(R2),R4  ;LOCATE THE STARTING POINT FOR SCORE BOARD
1151 014650 060304          MOV     R3,R4         ;UPDATE POINTER
1152 014652 022704 000017          11$:   CMP     #15.,R4       ;SHOULD POINTER BE ADJUSTED ?
1153 014656 003003          BGT     7$             ;BR IF NO
1154 014660 162704 000020          SUB     #16.,R4       ;ENTRY POINT AT THE SCORE BOARD
1155 014664 000772          BR      11$

```

```

1156 014666 060401
1157 014670 023703 001362
1158 014674 002412
1159 014676 105221
1160 014700 005204
1161 014702 005203
1162 014704 022704 000017
1163 014710 002367
1164 014712 013701 002054
1165 014716 005004
1166 014720 000763
1167 014722
    014722 012604
    014724 012603
    014726 012602
    014730 012601
1168 014732 000207
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183 014734
    014734 010046
    014736 010246
1184 014740 012700 004506
1185 014744 116002 000026
1186 014750 042702 177740
1187 014754 116037 000010 001360
1188 014762 005702
1189 014764 001404
1190 014766 005302
1191 014770 010237 001362
1192 014774 000416
1193 014776 005760 000022
1194 015002 001410
1195 015004 026060 000022 000004
1196 015012 001004
1197 015014 116037 000010 001362
1198 015022 000403
1199 015024 112737 000037 001362
1200 015032
    015032 012602
    015034 012600
1201 015036 000207
1202
1203
1204

7$: ADD R4,R1
8$: CMP CMSEC,R3
   BLT 9$
   INCB (R1)+
   INC R4
   INC R3
   CMP #15.,R4
   BGE 8$
   MOV TABLEX,R1
   CLR R4
   BR 8$

9$: MOV (SP)+,R4
   MOV (SP)+,R3
   MOV (SP)+,R2
   MOV (SP)+,R1
   RTS PC

:LABAD ROUTINE
:LOCATE THE START SECTOR AND THE TERMINATE SECTOR OF THE PREVIOUS
:OPERATION.
:
:   STARSC = STARTING SECTOR
:   CMSEC  = ENDING SECTOR
:
:INFORMATION FROM $RMDA(FMTDPB), $RMWC(FMTDPB), $SEC(FMTDPB)
:                $SEC(FMTDPB), $WRDM(FMTDPB)
:CALL
:   JSR PC,LABAD
:   RET

LABAD:
   MOV R0,-(SP)
   MOV R2,-(SP)
   MOV #FMTDPB,R0
   MOVB $RMDA(R0),R2
   BIC #177740,R2
   MOVB $SEC(R0),STARSC
   TST R2
   BEQ 1$
   DEC R2
   MOV R2,CMSEC
   BR 3$

1$: TST $RMWC(R0)
   BEQ 2$
   CMP $RMWC(R0),$WRDM(R0)
   BNE 2$
   MOVB $SEC(R0),CMSEC
   BR 3$

2$: MOVB #31.,CMSEC
3$: MOV (SP)+,R2
   MOV (SP)+,R0
   RTS PC

:TEST 7
:SELF TEST:WRITE CYLINDERS $DEV+17,$DEV+17+128,$DEV+17+128X2,ETC.

```

1205
 1206
 1207
 1208
 1209
 1210

: THEN EXECUTE WRITE CHECK TO DETECT ANY DATA OR HEADER PROBLEM
 : FOR THE SELECTED LOGICAL DRIVE
 : \$DEV#M: LOGICAL DRIVE #
 : DRIVE: PHYSICAL DRIVE #

015040 000004
 015042 012737 000001 001176
 1211 015050 004737 027514
 1212 015054 012737 177777 027436
 1213 015062 012737 177777 027440
 1214 015070 012737 000007 001340
 1215 015076 012706 001100
 1216 015102 012702 004356
 1217 015106 005022
 1218 015110 022702 004422
 1219 015114 103374
 1220 015116 012700 004506
 1221 015122 012760 160000 000004
 1222 015130 112760 000161 000002
 1223 015136 005060 000010
 1224 015142 113710 001224
 1225 015146 013702 001270
 1226 015152 062702 000021
 1227 015156 010260 000012
 1228 015162 012760 043032 000006
 1229 015170 012760 004526 000014
 1230 015176 004737 020742
 1231 015202 004037 030264
 1232 015206 004506
 1233 015210 000774
 1234 015212 005737 004524
 1235 015216 001775
 1236 015220 012700 004506
 1237 015224 004737 017352
 1238 015230 005760 000016
 1239 015234 100010
 1240 015236 004737 027514
 1241 015242 012737 177777 027436
 1242 015250 012737 177777 027440
 1243 015256 105260 000011
 1244 015262 126037 000011 001372
 1245 015270 003744
 1246 015272 005060 000010
 1247 015276 062760 000200 000012
 1248 015304 022760 001440 000012
 1249 015312 103333
 1250
 1251
 1252
 1253 015314 112760 000151 000002
 1254 015322 012702 004356
 1255
 1256 015326 013703 001270
 1257 015332 062703 000021
 1258 015336 010360 000012
 1259 015342 005037 004504

```

:*****
TST7:  SCOPE
      MOV #1,$TIMES           ;;DO 1 ITERATION
      JSR PC,RMINIT          ;:INITIAL THE DRIVE
      MOV #-1,SAVEFG        ;:SAVE THE RH REG'S
      MOV #-1,SEEKFG
      MOV #7,TSTNM          ;:LOAD TEST NUMBER
      MOV #STACK,SP         ;:INITIAL THE STACK
      MOV #SELF0,R2         ;:CLEAR THE SELF TEST SCORE BOARDS
1$:   CLR (R2)+
      CMP #SELF18,R2        ;:ALL DONE ?
      BHS 1$                 ;:BRANCH IF NOT
      MOV #FMTDPB,R0         ;:SET UP DPB
      MOV #-8192,,$WRDM(R0) ;:LOAD FULL TRACK WORD COUNT
      MOV #WRTDAT,$COMND(R0) ;:WRITE DATA COMMAND
      CLR $SEC(R0)          ;:SECTOR 0,TRACK 0
      MOVB DRIVE,(R0)       ;:LOAD PHY. DRIVE
      ADD #17,R2            ;:LOCATE THE STARTING CYLINDER
      MOV R2,$CYL(R0)       ;:CYLINDER ADDRESS
      MOV #BUFFER,$BUF(R0)  ;:RESET BUFFER ADDRESS
      MOV #RM.REG,14(R0)    ;:ADDRESS TO SAVE ALL RH/RM REG'S
2$:   JSR PC,FILBUF         ;:FILL THE BUFFER WITH WORSE CASE PATTERN
      JSR R0,RM05           ;:CALL DRIVER HANDLER
      BR 2$                  ;:BRANCH IF QUEUE FAILS
3$:   TST FMTDPB+16         ;:ALL DONE ?
      BEQ 3$                 ;:BRANCH IF NOT
      MOV #FMTDPB,R0        ;:REPORT IF ANY ERROR
      JSR PC,PROCES
      TST 16(R0)             ;:ERROR FLAG SET ?
      BPL 22$               ;:BRANCH IF NOT
      JSR PC,RMINIT          ;:INITIAL THE DRIVE
      MOV #-1,SAVEFG
      MOV #-1,SEEKFG
22$:  INCB $TRK(R0)          ;:NEXT TRACK
      CMPB $TRK(R0),TRKLMT ;:ALL TRACK ARE DONE ?
      BLE 2$                 ;:BRANCH IF NOT
      CLR $SEC(R0)          ;:RESET SECTOR AND TRACK
      ADD #128,,$CYL(R0)    ;:ADVANCE TO NEXT ZONE
      CMP #800,,$CYL(R0)    ;: ALL 7 ZONES ARE DONE ?
      BHS 2$                 ;:BRANCH IF NOT

;EXECUTE WRITE CHECK TO DETECT ANY DATA OR HEADER PROBLEM
      MOV #WCKD,$COMND(R0)  ;:CHANGE TO WRITE CHECK COMMAND
      MOV #SELF0,R2         ;:R2 POINTS TO SCORE BOARD
      MOV $DEV#M,R3         ;:CAN NOT BE DESTORIED
      ADD #17,R3            ;:LOACATE STARTING ADDRESS
      MOV R3,$CYL(R0)       ;:STARTING CYLINDER
      CLR OFFCOD            ;:SET NEGATIVE OFFSET FLAG
    
```

```

1260 015346 004037 030264      4$:   JSR    R0,RM05      :CALL DRIVE HANDLER
1261 015352 004506                FMTDPB
1262 015354 000774                BR      4$             :BRANCH IF QUEUE FAILS
1263 015356 005737 004524      5$:   TST    FMTDPB+16    :ALL DONE ?
1264 015362 001775                BEQ    5$             :BRANCH IF NOT
1265 015364 012700 004506                MOV    #FMTDPB,R0
1266 015370 004737 017352                JSR    PC,PROCES
1267 015374 005760 000016                TST    16(R0)         :ANY ERROR
1268 015400 100401                BMI    6$             :YES,THEN DON'T INCREMENT SCORE
1269 015402 105212                INCB   (R2)           :INCREMENT SCORE
1270 015404 004537 014144      6$:   JSR    R5,OFFST      :OFFSET
1271 015410 000415                BR     8$             :BRANCH IF OFFSET FAILS
1272 015412 004737 014376                JSR    PC,MAKEUP      :EXECUTE WRITE CHECK IN OFFSET MODE
1273 015416 005737 004524      7$:   TST    FMTDPB+16    :ALL DONE /
1274 015422 001775                BEQ    7$             :BRANCH IF NOT
1275 015424 012700 004506                MOV    #FMTDPB,R0
1276 015430 004737 017352                JSR    PC,PROCES      :REPORT IF ANY ERROR
1277 015434 005760 000016                TST    16(R0)         :ERROR BIT SET ?
1278 015440 100401                BMI    8$             :DON'T INCREMENT SCORE
1279 015442 105212                INCB   (R2)           :INCREMENT SCORE
1280 015444 005737 004504      8$:   TST    OFFCOD        :POSITIVE OFFSET IS DONE ?
1281 015450 001006                BNE    9$             :BRANCH ,IF DONE
1282 015452 005202                INC    R2             :INCREMENT SCORE BOARD POINTER
1283 015454 012737 000001 004504      MOV    #1,OFFCOD      :SET POSITIVE OFFSET FLAG
1284 015462 000137 015346                JMP    4$             :POSITIVE OFFSET TEST
1285 015466 105260 000011      9$:   INCB   $TRK(R0)      :INCREMENT TO NEXT TRACK
1286 015472 005202                INC    R2             :ADVANCE SCORE BOARD POINTER
1287 015474 126037 000011 001372      CMPB   $TRK(R0),TRKMT :ALL TRACKS ARE DONE ?
1288 015502 003004                BGT    10$            :BRANCH , IF ALL DONE
1289 015504 005037 004504                CLR    OFFCOD         :RESET OFFSET DIRECTION
1290 015510 000137 015346                JMP    4$             :TRY THE NEXT TRACK
1291 015514 005060 000010      10$:  CLR    $SEC(R0)       :TRACK 0, SECTOR 0
1292 015520 012702 004356                MOV    #SELF0,R2     :RESET SCORE BOARD
1293 015524 062760 000200 000012      ADD    #128.,$CYL(R0) :ADVANCE TO NEXT ZONE
1294 015532 022760 001440 000012      CMP    #800.,$CYL(R0) :ALL 7 ZONES ARE DONE ?
1295 015540 103404                BLO    11$            :BRANCH ,IF ALL DONE
1296 015542 005037 004504                CLR    OFFCOD         :RESET OFFSET FLAG
1297 015546 000137 015346                JMP    4$             :TO NEXT ZONE
1298 015552 000240      11$:  NOP                    :TO NEXT TEST
1299
1300      :TEST10 TEST 8      READ COMPATIBLE TEST
1301      :CYLINDERS TESTED : $DEVN+112,$DEVN+112+128XN  N=1 TO 5
1302      :THIS TEST SELECT ONE CYLINDER FOR EACH LOGICAL DRIVE FROM
1303      :ZONE 1 TO ZONE 6
1304      :
1305      :
1306      :      $DEVN  = LOGICAL DRIVE #
1307      :      DRIVE  = PHYSICAL DRIVE #
1308      :*****
1309 015554 000004      TST10: SCOPE
1310 015556 012737 000001 001176      MOV    #1,$TIMES     ;;DO 1 ITERATION
1311 015564 012737 000010 001340      MOV    #10,TSTNM     :LOAD TEST NUMBER
1312 015572 012706 001100                MOV    #STACK,SP     :INITAIL THE STACK POINTER
1313 015576 012702 003216                MOV    #RDNO,R2      :CLEAR THE SCORE BOARD OF READ TEST
1314 015602 005022      1$:   CLR    (R2)+
1315 015604 022702 004356                CMP    #RDP18+16.,R2 :ALL DONE
1316 015610 101374                BHI    1$            :BRANCH IF NOT
    
```



```

1382 016160 000137 015676          JMP      1$
1383
1384 016164 013703 001270          11$:    MOV      $DEV#R3      ;LOCATE STARTING CYLINDER
1385 016170 062703 000160          ADD      #112.,R3      ;
1386 016174 010360 000012          MOV      R3,$CYL(R0)   ;STARTING CYLINDER
1387 016200 105260 000011          INCB    $TRK(R0)       ;TO NEXT TRACK
1388 016204 126037 000011 001372  CMPB    $TRK(R0),TRKLT ;ALL SRUFACE ARE DONE ?
1389 016212 003002          BGT     12$           ;BRANCH IF ALL DONE
1390 016214 000137 015676          JMP     1$           ;LOOPING UNTIL CURRENT TRACK IS DONE
1391
1392 016220 005737 004504          12$:    TST     OFFCOD    ;OFFSET CODE = POSITIVE ?
1393 016224 001005          BNE     13$           ;IF EQUAL,THEN ALL DONE
1394 016226 012737 000001 004504  MOV     #1,OFFCOD     ;SET POSITIVE OFFSET FLAG
1395 016234 000137 015616          JMP     LOOP3
1396 016240 000240          13$:    NOP            ;TO NEXT TEST
1397
1398          ;TEST11 TEST 9
1399          ;REPORT THE TEST SCORES
1400          ;DIRECT THE OPERATOR TO CHANGE PACK AND MOUNT TO OTHER DIRVE
1401          ;
1402          ; $DEV# = LOGICAL DRIVE #, SHOULD NOT BE UPDATED BEFORE THE
1403          ; REPORT IS COMPLETED.
1404          ;
1405          ; BADSEC = 0, (DRIVES ARE COMPATIBLE)
1406          ; = -1, (DRIVES NOT COMPATIBLE)
1407          ;
1408          ; CMTRK = TRACK NUMBER FOR CONTROLLING THE SCORE TYPING.
1409          ;
1410          ;*****
1411          ;TST11: SCOPE
1412          MOV     #1,$TIMES      ;;DO 1 ITERATION
1413          MOV     #11,TSTNM     ;LOAD THE TEST NUMBER
1414          MOV     #STACK,SP     ;LOAD THE STACK POINTER
1415          CLR     -(SP)         ;DUMMY SCORE BOARD ADDRESSES
1416          CLR     -(SP)         ;
1417          MOV     #SELF0+1,-(SP) ;POSITIVE OFFSET READ TEST SCORE ADDRESS
1418          MOV     #RDPO,-(SP)  ;POSITIVE OFFSET TEST SCORE
1419          MOV     #SELF0,R2     ;NEGATIVE OFFSET READ TEST SCORE
1420          MOV     #RDNO,R3     ;NEGATIVE OFFSET READ COMPATIBLE TEST SCORE
1421          1$:    MOV     TRKLT,R1 ;R1= TRACK NUMBER
1422          2$:    CMPB    #7,(R2) ;SELF READ TEST SCORE IS TOO LOW
1423          BLOS   4$           ;BRANCH IF NOT
1424          MOV     #16.,R5     ;INCREMENT READ COMPATIBLE SCORE FOR ALL 16 DIRVES
1425          3$:    MOVB   (R3),R4 ;ADJUST SCORE BY ADDING 6
1426          ADD    #6,R4
1427          MOVB  R4,(R3)+     ;UPDATE SCORE AND POINTS TO NEXT DRIVE
1428          DEC   R5           ;ALL DRIVES ARE UPDATED ?
1429          BNE   3$           ;BRANCH IF NOT
1430          BR    5$           ;
1431          4$:    ADD    #16.,R3 ;ADJUST POINTER OF SCORE BOARD ADDRESS
1432          5$:    ADD    #2,R2  ;UPDATE THE SELF TEST SCORE ADDRESS
1433          DEC   R1           ;ALL TRACKS DONE ?
1434          BGE   2$           ;BRANCH IF NOT
1435          MOV   (SP)+,R3     ;GET NEXT PAIR
1436          MOV   (SP)+,R2
1437          BNE   1$

```

```

1437                                     ;SET ACCEPTANCE FLAG, INITIALIZE THE TABLE POINTERS AND TRACK NUMBER
1438
1439 016366 005037 001342                CLR    BADSEC      ;SET ACCEPTANCE FLAG
1440 016372 005037 001364                CLR    CMTRK      ;START FROM TRACK 0
1441 016376 005001                       CLR    R1         ;START FROM LOG DRV 0
1442 016400 012702 002056                MOV    #OVWNO,R2  ;SCORE BOARD BASE ADDRESS
1443 016404 012703 000001                MOV    #BIT0,R3   ;BIT POSITION FOR LOG DRV 0
1444 016410 030337 002006                LOOP4: BIT    R3,ASNLST ;IS THE LOG DRV UNDER TEST ?
1445 016414 001502                       BEQ    LOOP5      ;BRANCH IF NOT
1446 016416 005037 001366                CLR    NULINE     ;NEW LINE INDICATOR
1447 016422 123701 001270                CMPB   $DEVM,R1   ;SELF SCORE ?
1448 016426 001434                       BEQ    SPATH      ;BRANCH IF IT IS
1449 016430 122712 000016                KPATH: CMPB   #14.,(R2) ;OVERWRITE NEG OFFSET < 14 ?
1450 016434 101403                       BLOS   1$        ;BRANCH IF NOT
1451 016436 004537 017052                JSR    R5,PRINT   ;REPORT EXCEPTIONS
1452 016442 000001                       000001          ;COLUMN POSITION ON REPORT
1453 016444 122762 000016 000460 1$:    CMPB   #14.,OVWPO-OVWNO(R2) ;OVERWRITE POS OFFSET < 14 ?
1454 016452 101403                       BLOS   2$        ;BRANCH IF NOT
1455 016454 004537 017052                JSR    R5,PRINT   ;REPORT EXECPTIONS
1456 016460 000002                       000002          ;COLUMN POSITION ON REPORT
1457 016462 122762 000014 001140 2$:    CMPB   #12.,RDNO-OVWNO(R2) ;READ NEG OFFSET < 6
1458 016470 101403                       BLOS   3$        ;BRANCH IF NOT
1459 016472 004537 017052                JSR    R5,PRINT   ;REPORT EXCEPTIONS
1460 016476 000003                       000003          ;COLUMN POSITION ON REPORT
1461 016500 122762 000014 001620 3$:    CMPB   #12.,RDPO-OVWNO(R2) ;READ POS OFFSET < 12 ?
1462 016506 101403                       BLOS   4$        ;BRANCH IF NOT
1463 016510 004537 017052                JSR    R5,PRINT   ;REPORT EXECPTIONS
1464 016514 000004                       000004          ;COLUMN POSITION ON REPORT
1465 016516 000441                       4$:    BR     LOOP5   ;EXIT
1466
1467 016520 122712 000016                SPATH: CMPB   #14.,(R2)   ;OVERWRITE NEG OFF < 14
1468 016524 101403                       BLOS   1$        ;BRANCH IF NOT
1469 016526 004537 017052                JSR    R5,PRINT   ;REPORT EXECPTION
1470 016532 000001                       000001          ;COLUMN POSITION ON REPORT
1471 016534 122762 000016 000460 1$:    CMPB   #14.,OVWPO-OVWNO(R2) ;OVERWRITE POS OFFSET < 14
1472 016542 101403                       BLOS   2$        ;BRANCH IF NOT
1473 016544 004537 017052                JSR    R5,PRINT   ;REPORT EXCEPTION
1474 016550 000002                       000002          ;COLUMN POSITION ON REPORT
1475 016552 013704 001364                2$:    MOV    CMTRK,R4   ;LOCATE THE SELF TEST SCORE
1476 016556 006304                       ASL    R4         ;WORD INDEX
1477 016560 122764 000006 004356        CMPB   #6,SELF0(R4) ;SELF READ NEG OFFSET < 6 ?
1478 016566 101403                       BLOS   3$        ;BRANCH IF NOT
1479 016570 004537 017052                JSR    R5,PRINT   ;REPORT EXCEPTIONS
1480 016574 000003                       000003          ;COLUMN POSITION ON REPORT
1481 016576 013704 001364                3$:    MOV    CMTRK,R4   ;LOCATE THE SELF TEST SCORE
1482 016602 006304                       ASL    R4         ;WORD INDEX
1483 016604 122764 000006 004357        CMPB   #6,SELF0+1(R4) ;SELF READ POS OFFSET < 6 ?
1484 016612 101403                       BLOS   4$        ;BRANCH IF NOT
1485 016614 004537 017052                JSR    R5,PRINT   ;REPORT EXCEPTIONS
1486 016620 000004                       000004          ;COLUMN POSITION FOR REPORT PRINTING
1487 016622
1488
1489 016622 005201                       LOOP5: INC    R1     ;INCREASE THE LOGICAL DRIVE #
1490 016624 000241                       CLC                    ;ADJUST THE BIT POSITION
1491 016626 006103                       ROL    R3         ;POINTS TO NEXT DRIVE
1492 016630 005202                       INC    R2         ;UPDATE SCORE BOARD BASE ADDRESS
1493 016632 022701 000017                CMP    #15.,R1    ;ALL DRIVES ARE CHECK ?
    
```

```

1494 016636 103264
1495 016640 005237 001364
1496 016644 012702 002056
1497 016650 005001
1498 016652 013703 001364
1499 016656 001404
1500 016660 062702 000020
1501 016664 005303
1502 016666 001374
1503 016670 012703 000001
1504
1505
1506
1507
1508
1509
1510 016674 123737 001364 001372
1511 016702 003642
1512 016704 005737 001342
1513 016710 001402
1514 016712 104401 001207
1518
1519 016716 104401 001207
1520 016722 012777 017344 010550
1521 016730 005077 010546
1522 016734 104401 041650
1523 016740 104401 001274
1524 016744 013746 001224
1525 016750 104403
1526 016752 001
1527 016753 000
1528 016754 104401 041666
1529
1530 016760 104411
1531 016762 012605
1532 016764 105715
1533 016766 001374
1534 016770 005002
1535 016772 012703 000001
1536 016776 020237 001270
1537 017002 001404
1538 017004 000241
1539 017006 006103
1540 017010 005202
1541 017012 000771
1542 017014 040337 002012
1543 017020 001412
1544 017022 005237 001270
1545 017026 013702 001270
1546 017032 006302
1547 017034 016237 004444 001272
1548 017042 000137 012672
1549
1550 017046 000137 023420
1551
1552
1553

      BHPB      LOOP4      ;BRANCH IF NOT ALL DONE
      INC      CMTRK      ;NEXT TRACK
      MOV      #OVWNO,R2  ;RESET SCORE BOARD BASE ADDRESS
      CLR      R1         ;LOGICAL DRIVE START FROM 0
      MOV      CMTRK,R3   ;LOCATE THE SCORE BOARD BASE ADDRESS
      BEQ      2$        ;BRANCH IF ON TRACK 0
      ADD      #16.,R2    ;FOR EACH TRACK, 16 BYTES
      DEC      R3         ;ALL TRACK DONE ?
      BNE      1$        ;BRANCH IF NOT
      MOV      #BIT0,R3   ;BIT MAP POSITION FOR DRIVE 0

;AT THIS POINT:
:      R1         = LOGICAL DRIVE # START FROM 0
:      R2         = ADDRESS OF SCORE BOARD BASE ADDRESS
:      R3         = BIT MAP, INDICATE LOGICAL DRIVE 0=1

      CMPB     CMTRK,TRKLMT ;ALL TRACKS ARE DONE ?
      BLE     LOOP4        ;NO
      TST     BADSEC       ;WAS SCORE PRINTED ?
      BEQ     DISMNT       ;BR IF NO
      TYPE    ,SCLF        ;CR-LF

DISMNT: TYPE    ,SCLF      ;CR-LF
      MOV     #IDLEX,@RMVEC ;RESET THE INTERRUPT VECTOR
      CLR     @RMVEC+2     ;CLEAR THE INTERRUPT VECTOR
      TYPE    ,MSG12       ;UNLOAD AND DISMOUNT MESSAGE
      TYPE    ,SCDW2       ;SYSTEM NAME
      MOV     DRIVE,-(SP)  ;PHYSICAL DRIVE NUMBER
      TYPOS   .BYTE 1
      .BYTE 0
      TYPE    ,MSG13      ;MESSAGE TYPE <CR> WHEN READY

1$:    RDLIN
      MOV     (SP)+,R5     ;LOCATE THE READ IN LINE
      TSTB   (R5)         ;CARRIAGE RETURN ?
      BNE    1$          ;BR IF NO
      CLR    R2          ;LOGICAL DRIVE NUMBER
      MOV    #BIT0,R3    ;BIT MAP OF ASSIGNED DRIVE
      CMP    R2,$DEVMT   ;IS THE LOGICAL DRIVE UNDER TEST ?
      BEQ    2$          ;BRANCH IF IT IS
      CLC
      ROL    R3          ;SHIFT THE BIT MAP FOR
      INC    R2          ;NEXT DRIVE
      BR     1$          ;INCREMENT THE LOGICAL DRIVE #
                        ;LOOPING ,UNTIL THE LOGICAL DRIVE LOCATED
2$:    BIC    R3,ASSGN2  ;CLEAR THE ASSIGNED BIT
      BEQ    XEND2       ;BRANCH IF NO MORE DRIVES
      INC    $DEVMT      ;INCREMENT THE LOGICAL DRIVE #
      MOV    $DEVMT,R2   ;UPDATE SYSTEM BLOCK ADDRESS
      ASL    R2
      MOV    BLKADR(R2),$CDW1 ;SYSTEM BLOCK ADDRESS
      JMP    TST5        ;JUMP TO TEST 5 FOR OTHER DRIVE

XEND2: JMP    $EOP      ;END OF PASS

;PRINT ROUTINE
    
```



```

1554 :NAME PRINT
1555 :PRINT EXCEPTIONS FOR TEST SCORE
1556 :PARAMETER USED
1557 :
1558 :       NULINE = 0      A NEW LINE TO PRINT
1559 :       = 1 - 4 COLUMN NUMBER TO PRINT THE EXCEPTION MARK
1560 :
1561 :       BADSEC = 0      FIRST EXCEPTION DETECTED
1562 :       BADSEC = -1    NOT FIRST EXCEPTION (DON'T PRINT THE TITLE)
1563 :
1564 :CALL
1565 :       JSR      R5,PRINT
1566 :       NUMBER
1567 :       RET
1568 :
1569 :       R1      = LOGICAL DRIVE #
1570 :       $DEVN   = LOGICAL DRIVE UNDER TEST
1571 :       R3      = BIT POSITION OF LOGICAL DRIVE IN R1
1572 :
1573 :PRINT:
1574 017052 010146
1575 017052 010246
1576 017054 010346
1577 017056 010446
1578 017060 010446
1579 017062 005737 001342
1580 017066 100434
1581 017070 104401 001207
1582 017074 104401 042021
1583 017100 104401 001274
1584 017104 013746 001224
1585 017110 104403
1586 017112 001
1587 017113 000
1588 017114 104401 001207
1589 017120 104401 001207
1590 017124 104401 042043
1591 017130 104401 001207
1592 017134 104401 042124
1593 017140 104401 001207
1594 017144 012737 177777 001342
1595 017152 012737 000001 001376
1596 017160 005737 001366
1597 017164 001042
1598 017166 104401 001207
1599 017172 013746 001364
1600 017176 104405
1601 017200 104401 042213
1602 017204 123701 001270
1603 017210 001007
1604 017212 104401 042326
1605 017216 104401 042206
1606 017222 104401 042213
1607 017226 000416
1608 017230 006301
1609 017232 016137 004444 017256
1610 017240 006201

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 , $CRLF      ;;CRLF
TYPE ,MSG16       ;;SCORES FOR DRIVE --
TYPE , $CDW2      ;;SYSTEM NAME
MOV DRIVE,-(SP)   ;;TYPE THE PHYSICAL DRIVE #
TYPOS
.BYTE 1
.BYTE 0
TYPE , $CRLF      ;;CR-LF
TYPE , $CRLF      ;;CR-LF
TYPE ,MSG17       ;;SUB TITLE 1
TYPE , $CRLF      ;;CR-LF
TYPE ,MSG18       ;;SUB TITLE 2
TYPE , $CRLF      ;;CR-LF
MOV #-1,BADSEC    ;;RESET THE ACCEPTANCE FLAG
MOV #1,FAULT      ;;NOT COMPATIBLE FLAG
TST NULINE        ;;NEW LINE ?
BNE 5$           ;;BRANCH IF NOT
TYPE , $CRLF      ;;CR-LF
MOV CMTRK,-(SP)   ;;SAVE CMTRK FOR TYPEOUT
TYPDS             ;;GO TYPE--DECIMAL ASCII WITH SIGN
TYPE ,TAB         ;;TYPE 'TAB' CHARACTER
CMPB $DEVN,R1     ;;SCORE FOR $DEVN ITSELF ?
BNE 2$           ;;BRANCH IF NOT
TYPE ,BLNKS3      ;;TYPE 3 BLANKS
TYPE ,SELF        ;;MESSAGE 'SELF'
TYPE ,TAB         ;;TYPE TAB
BR 4$            ;;NEXT STEP
ASL R1            ;;LOCATE THE SYS HISTORY FILE
MOV BLKADR(R1),3$ ;;LOCATE THE SYSTEM NAME AND DRIVE #
ASR R1           ;;RESTORE DRIVE #
    
```

```

1606 017242 062737 000014 017256      ADD    #$$SYSNM,3$      ;LOCATE THE SYSTEM NAME AND DRIVE #
1607 017250 104401 042326              TYPE    ,BLNKS3        ;TYPE 3 BLANKS
1608 017254 104401              TYPE    ;TYPE THE SYSTEM AND DRIVE
1609 017256 000000              .WORD    0              ;ADDRESS FOR TYPING MESSAGE
1610 017260 104401 042213              TYPE    ,TAB           ;TYPE TAB
1611 017264 012737 000001 001366      4$:    MOV    #1,NULINE  ;INDICATE PRINTER STOPS AT COULMN 1
1612 017272 012504              5$:    MOV    (R5)+,R4    ;RETRIEVE THE DESIRED COLUMN # FROM
1613                                ;CALLING ROUTINE
1614 017274 020437 001366              6$:    CMP    R4,NULINE  ;ON THE RIGHT COLUMN ?
1615 017300 103414              BLO    8$              ;IF LOW NOT PRINT
1616 017302 001405              BEQ    7$              ;BRANCH IF LOCATED
1617 017304 104401 042213              TYPE    ,TAB           ;ADVANCE TO NEXT COLUMN
1618 017310 005237 001366              INC    NULINE          ;INCREMENT COLUMN CTR
1619 017314 000767              BR     6$              ;CHECK AGAIN
1620 017316 104401 042325              7$:    TYPE    ,BLNKS4   ;TYPE 4 BLANKS
1621 017322 104401 042215              TYPE    ,MARKX        ;THE EXCEPTION MARK '* 0'
1622 017326 005237 001366              INC    NULINE          ;NEXT COLUMN
1623 017332              8$:    MOV    (SP)+,R4    ;;POP STACK INTO R4
1624 017332 012604              MOV    (SP)+,R3       ;;POP STACK INTO R3
1625 017334 012603              MOV    (SP)+,R2       ;;POP STACK INTO R2
1626 017336 012602              MOV    (SP)+,R1       ;;POP STACK INTO R1
1627 017340 012601              MOV    (SP)+,R1       ;;POP STACK INTO R1
1628 017342 C00205              RTS     R5             ;EXIT
1629
1630
1631 IDLEX:  NOP              ;RESET ALL RH VECTOR FOR MOUNT AND DISMOUNT
1632 017344 000240              NOP
1633 017346 000240              NOP
1634 017350 000002              RTI              ;EXIT
1635
1636 ;PROCESS THE ORDER TERMINATION
1637
1638 PROCES: MOV B (R0),UNIT ;DRIVE NUMBER FOR ANY ERROR MESSAGES
1639 TST  $STATUS(R0) ;SEE IF DRIVER SIGNALLED AN ERROR
1640 BMI  ERPROC ;BR IF ERROR
1641 BIT  #BIT15,$RMCS1(R0) ;SEE IF 'SC' SET
1642 BEQ  1$ ;BR IF NOT SET
1643 BIT  #BIT14,$RMCS1(R0) ;SEE IF 'TRE' SET
1644 BNE  ERPROC ;BR IF SET
1645 BIT  #BIT14,$RMDS(R0) ;SEE IF 'ERR' SET
1646 BNE  ERPROC ;BR IF SET
1647 JSR  PC,CKERR ;NO ERROR, CHECK ERROR BITS ANYWAY
1648 JSR  PC,CKBUS ;NO ERROR, CHECK BUS ADDR & WC
1649 JSR  PC,RMINIT ;INITIALIZE THE SUB SYSTEM
1650 MOV  #-1,SAVEFG ;CLEAR THE SAVE FLAG
1651 MOV  #-1,SEEKFG ;SET THE NOT IMP FLAG
1652 RTS  PC ;RETURN
1653
1654 ;ORDER TERMINATED WITH AN ERROR - PROCESS THE ERROR
1655
1656 ERPROC: BIT  #BIT07,$STATUS(R0) ;DONE BIT SET ?
1657 BEQ  ERPRC1 ;BR IF ORDER DIDN'T COMPLETE NORMALLY
1658 JMP  DONE ;PROCESS ERROR WITH 'DONE' BIT SET
1659
1660 ;PROCESS ORDER COMPLETION WITH 'ERROR' & 'DONE NOT' BITS
1661
1662 ERPRC1: BIT  #BIT12,$STATUS(R0) ;SEE IF DRIVE WAS UNSAFE
1663 BNE  PUNSAF ;BR IF YES
1664 BIT  #BIT11,$STATUS(R0) ;PARITY ERROR OCCURRED

```

```

1659 017500 001025          BNE      UCPAR          ;BR IF IT DID
1660 017502 032760 002000 000016 BIT      #BIT10,$STATUS(R0) ;FATAL PARITY ERROR?
1661 017510 001025          BNE      FALPAR        ;BR IF THERE IS ONE
1662 017512 032760 001000 000016 BIT      #BIT09,$STATUS(R0) ;TIMEOUT?
1663 017520 001025          BNE      SWTIM         ;BR IF YES
1664 017522 032760 040002 000016 BIT      #BIT14!BIT01,$STATUS(R0) ;DRIVE WENT OFFLINE ?
1665 017530 001025          BNE      OFLIN         ;BR IF IT DID
1666 017532 032760 000004 000016 BIT      #BIT2,$STATUS(R0) ;PORT REQUEST TIME OUT ?
1667 017540 001025          BNE      PRTIM         ;BR IF IT DID
1668 017542 000207          RTS       PC           ;ERROR. RETURN
1669
1670                          ;DRIVE IS PERSISTENTLY UNSAFE
1671
1672 017544          PUNSAF:
1673 017544 104414 036552      DISPLY  ,EM12
1674 017550 000137 017620      JMP     DUMP2
1675
1676                          ;UNCORRECTABLE MASSBUS PARITY ERROR OCCURRED
1677
1678 017554          UCPAR:
1679 017554 104414 036454      DISPLY  ,EM10
1680 017560 000137 017620      JMP     DUMP2
1681
1682                          ;'FATAL' MASSBUS PARITY ERROR OCCURRED
1683
1684 017564          FALPAR:
1685 017564 104414 036517      DISPLY  ,EM11
1686 017570 000137 017620      JMP     DUMP2
1687
1688                          ;SOFTWARE TIMEOUT OCCURRED
1689
1690 017574          SWTIM:
1691 017574 104414 036603      DISPLY  ,EM13
1692 017600 000137 017620      JMP     DUMP2
1693
1694                          ;DRIVE WENT OFFLINE
1695
1696 017604          OFLIN:
1697 017604 104414 036655      DISPLY  ,EM14
1698 017610 000137 017620      JMP     DUMP2
1699                          ;PORT REQUEST TIMEOUT ERROR
1700
1701 017614          PRTIM:
1702 017614 104414 036700      DISPLY  ,EM15
1703 017620 104401 001207      DUMP2:  TYPE  , $CRLF          ;CR-LF
1704 017624 104401 042621      TYPE  ,XFATL
1705 017630 104401 001207      TYPE  , $CRLF          ;CR-LF
1706 017634 104401 042464      TYPE  ,HALTX
1707 017640 000177 161534      JMP     @RSTART        ;JUMP TO RESTART
1708
1709                          ;PROCESS ORDER COMPLETION WITH 'ERROR' & 'DONE' BITS SET
1710
1711 017644 032760 000030 000016 DONE:  BIT      #BIT04!BIT03,$STATUS(R0) ;UNSAFE OCCURRED
1712 017652 001402          BEQ     ,+6             ;BR IF NOT
1713 017654 000137 020430          JMP     UNSAF          ;REPORT UNSAFE
1714 017660 032760 040000 000030 BIT      #BIT14,$RMCS2(R0) ;IS 'WCE' SET ?
1715 017666 001402          BEQ     ,+6             ;BRANCH IF NOT SET
    
```

```

1716 017670 000137 020210          JMP      WCKER          ;WRITE CHECK ERROR
1717 017674 032760 040000 000032  BIT      #BIT14,$RMDS(R0) ;CHECK 'ERR'
1718 017702 001002          BNE     1$             ;BR IF SET
1719 017704 000137 020400          JMP      TRFER          ;PROCESS 'TRE'
1720 017710 032760 000400 000034 1$:  BIT      #BIT08,$RMER1(R0) ;'HCRC' SET?
1721 017716 001402          BEQ     .+6            ;BR IF NOT
1722 017720 000137 020240          JMP      HRCRCER        ;PROCESS 'HCRC'
1723 017724 032760 000020 000034  BIT      #BIT04,$RMER1(R0) ;'FMT' SET?
1724 017732 001402          BEQ     .+6            ;BR IF NOT SET
1725 017734 000137 020260          JMP      CKFMT          ;CHECK FORMAT ERROR
1726 017740 032760 000200 000034  BIT      #BIT07,$RMER1(R0) ;'HCE' SET?
1727 017746 001402          BEQ     .+6            ;BR IF NOT SET
1728 017750 000137 020270          JMP      CKHCE          ;CHECK 'HCE' ERROR
1729 017754 032760 020000 000034  BIT      #BIT13,$RMER1(R0) ;'OPI' SET?
1730 017762 001402          BEQ     .+6            ;BR IF NOT SET
1731 017764 000137 020310          JMP      OPIER          ;REPORT 'OPI'
1732 017770 032760 000010 000034  BIT      #BIT3,$RMER1(R0)  ;'PAR' SET?
1733 017776 001402          BEQ     .+6            ;BR IF NOT SET
1734 020000 000137 020330          JMP      PARER          ;REPORT 'PAR'
1735 020004 032760 000040 000034  BIT      #BIT5,$RMER1(R0)  ;'WCF' SET?
1736 020012 001402          BEQ     .+6            ;BR IF NOT SET
1737 020014 000137 020420          JMP      WCFER          ;REPORT 'WCF'
1738 020020 032760 002000 000034  BIT      #BIT10,$RMER1(R0) ;'IAE' SET?
1739 020026 001402          BEQ     .+6            ;BR IF NOT SET
1740 020030 000137 020340          JMP      IAEER          ;REPORT 'IAE'
1741 020034 032760 004000 000034  BIT      #BIT11,$RMER1(R0) ;'WLE' SET?
1742 020042 001402          BEQ     .+6            ;BR IF NOT SET
1743 020044 000137 020350          JMP      WLEER          ;REPORT 'WLE'
1744 020050 032760 001000 000034  BIT      #BIT9,$RMER1(R0)  ;'AOE' SET?
1745 020056 001405          BEQ     2$             ;BR IF NOT SET
1746 020060 032760 002000 000032  BIT      #BIT10,$RMDS(R0) ;'LST' SET?
1747 020066 001401          BEQ     2$             ;BR IF NOT SET
1748 020070 000207          RTS     PC             ;'AOE' & 'LST' SET, EXIT
1749 020072 032760 010000 000034 2$:  BIT      #BIT12,$RMER1(R0) ;SEE IF 'DTE' SET
1750 020100 001402          BEQ     .+6            ;BR IF NOT
1751 020102 000137 020320          JMP      DTEER          ;REPORT 'DTE' ERROR
1752 020106 005760 000034          TST     $RMER1(R0)     ;SEE IF 'DCK' SET
1753 020112 100002          BPL     .+6            ;BR IF NOT
1754 020114 000137 020200          JMP      DCKER          ;PROCESS 'DCK'
1755 020120 032760 060000 000062  BIT      #BIT14!BIT13,$RMER2(R0) ;'SKI' OR 'OCYL' SET
1756 020126 001006          BNE     3$             ;BR IF IT IS
1757 020130 032760 100000 000062  BIT      #BIT15,$RMER2(R0) ;BAD SPOT ?
1758 020136 001004          BNE     4$             ;BRANCH IF SO
1759 020140 000137 020250          JMP      DRIVER        ;REPORT ERROR
1760 020144 000137 020410          JMP      SKIER          ;REPORT DRIVE ERROR
1761 020150 104401 001207          4$:  TYPE     , $CRLF      ;CR-LF
1762 020154 104401 041565          TYPE     , MSG11
1763 020160 104007          EMT     7
1764 020162 104010          EMT     10
1765 020164 104011          EMT     11
1766 020166 104012          EMT     12
1767 020170 104401 042464          TYPE     , HALTX
1768 020174 000177 161200          JMP      @RSTART       ;JUMP TO RESTART
1769
1770          ;PROCESS DATA ('DCK') CHECK ERROR
1771
1772 020200          DCKER:
    
```

```

1773 020200 104414 036755          DISPLY ,EM21
1774 020204 000137 020440          JMP      DUMP
1775
1776                                ;WRITE CHECK ERROR PROCESSING
1777
1778 020210                                WCKER:
1779 020210 032760 100000 000034      BIT      #BIT15,$RMR1(R0)          ;DCK BIT SET ?
1780 020216 001004 020440          BNE     1$                          ;BRANCH IF SET
1781 020220 104414 037061          DISPLY  ,EM23
1782 020224 000137 020440          JMP     DUMP
1783 020230 104414 037006      1$:   DISPLY  ,EM22
1784 020234 000137 020440          JMP     DUMP
1785
1786                                ;REPORT 'HCRC' ERROR
1787
1788 020240                                HRCRER:
1789 020240 104414 036734          DISPLY  ,EM20
1790 020244 000137 020440          JMP     DUMP
1791
1792                                ;REPORT DRIVE ERROR
1793
1794 020250                                DRVER:
1795 020250 104414 037351          DISPLY  ,EM30
1796 020254 000137 020440          JMP     DUMP
1797
1798                                ;PROCESS FORMAT ('FER') ERROR
1799
1800 020260                                CKFMT:
1801 020260 104414 037140          DISPLY  ,EM24
1802 020264 000137 020440          JMP     DUMP
1803
1804                                ;PROCESS HEADER COMPARE ('HCE') ERROR
1805
1806 020270                                CKHCE:
1807 020270 104414 037206          DISPLY  ,EM25
1808 020274 000137 020440          JMP     DUMP
1809
1810                                ;POSSIBLE POSITIONING ERROR
1811 020300                                POSER:
1812 020304 104414 040477          DISPLY  ,EM51
1813 020304 000137 020440          JMP     DUMP
1814
1815                                ;REPORT 'OPI' ERROR
1816 020310                                OPIER:
1817 020314 104414 037403          DISPLY  ,EM31
1818 020314 000137 020440          JMP     DUMP
1819
1820                                ;REPORT 'DTE' ERROR
1821 020320                                DTEER:
1822 020320 104414 037446          DISPLY  ,EM32
1823 020324 000137 020440          JMP     DUMP
1824
1825                                ;REPORT 'PAR' ERROR
1826
1827 020330                                PARER:
1828 020330 104414 037501          DISPLY  ,EM33
1829 020334 000137 020440          JMP     DUMP
    
```

1830					
1831					:REPORT 'IAE' ERROR
1832					
1833	020340				IAEER:
1834	020340	104414	037620	DISPLY	,EM35
1835	020344	000137	020440	JMP	DUMP
1836					
1837					:REPORT WLE ERROR
1838					
1839	020350				WLEER:
1840	020350	104414	037656	DISPLY	,EM36
1841	020354	000137	020440	JMP	DUMP
1842					
1843					:REPORT FORMAT ERROR
1844					
1845	020360				FMTER:
1846	020360	104414	037267	DISPLY	,EM26
1847	020364	000137	020440	JMP	DUMP
1848					
1849					:REPORT HEADER COMPARE ERROR
1850					
1851	020370	104414	037314	HCEER:	DISPLY ,EM27
1852	020374	000137	020440	JMP	DUMP
1853					
1854					:PROCESS CONTROL/INTERFACE TRANSFER ERROR
1855					
1856	020400				TRFER:
1857	020400	104414	037771	DISPLY	,EM40
1858	020404	000137	020440	JMP	DUMP
1859					
1860					:PROCESS 'SKI' OR 'OCYL'
1861					
1862	020410				SKIER:
1863	020410	104414	040441	DISPLY	,EM50
1864	020414	000137	020440	JMP	DUMP
1865					
1866					:REPORT WRITE CLOCK FAILURE
1867					
1868	020420				WCFER:
1869	020420	104414	037556	DISPLY	,EM34
1870	020424	000137	020440	JMP	DUMP
1871					
1872					:REPORT DRIVE UNSAFE ERROR
1873					
1874	020430				UNSAF:
1875	020430	104414	040542	DISPLY	,EM60
1876	020434	000137	020440	JMP	DUMP
1877					
1878	020440				DUMP:
	020440	104007		EMT	7
1879	020442	104010		EMT	10
1880	020444	104011		EMT	11
1881	020446	104012		EMT	12
1882					
1883	020450	004737	027514	JSR	PC,RMINIT
1884	020454	012737	177777	MOV	#-1,SAVEFG
1885	020462	012737	177777	MOV	#-1,SEEKFG
			027436		
			027440		:CLEAR THE SUB-SYSTEM

```

1886 020470 000207          RTS      PC
1887
1888
1889          ;CHECK ERROR BITS IN THE RH/RM REGISTERS
1890
1891 020472 032760 060000 000020 CKERR: BIT      #60000,$RMCS1(R0)      ;SEE IF 'TRE' OR 'MCPE'
1892 020500 001012          BNE      1$              ;YES
1893 020502 032760 177400 000030          BIT      #177400,$RMCS2(R0)    ;ERROR BITS IN CS2 /
1894 020510 001006          BNE      1$              ;YES
1895 020512 005760 000034          TST      $RMER1(R0)          ;ANY ERROR IN ER1
1896 020516 001003          BNE      1$              ;YES
1897 020520 005760 000062          TST      $RMER2(R0)          ;ANY ERROR IN ER2
1898 020524 001404          BEQ      2$              ;BRANCH IF NO ERROR
1899 020526 104414 040217 1$:      DISPLY  ,EM44
1900 020532 000137 020440          JMP      DUMP              ;TYPE ALL REGISTERS
1901 020536 000207 2$:      RTS      PC
1902
1903          ;CHECK BUS ADDRESS REGISTER AND WORD COUNT REGISTER
1904
1905 020540 005760 000022          CKBUS: TST      $RMWC(R0)      ;WORD COUNT = 0
1906 020544 001011          BNE      1$              ;NO
1907 020546 016046 000004          MOV      $WRDM(R0),-(SP)    ;WORD LENGTH
1908 020552 005416          NEG      (SP)              ;GET THE POSITIVE NUMBER OF WORD COUNT
1909 020554 006316          ASL      (SP)              ;BYTE COUNT
1910 020556 066016 000006          ADD      $BUF(R0),(SP)
1911 020562 022660 000024          CMP      (SP)+,$RMBA(R0)
1912 020566 001404          BEQ      2$
1913 020570 104414 040025 1$:      DISPLY  ,EM41
1914 020574 000137 020440          JMP      DUMP              ;TYPE ALL REGISTERS
1915 020600 000207 2$:      RTS      PC
1916
1917          ;ROUTINE TO DISPLAY THE SECTOR WHICH GAVE THE HARD ERROR
1918 020602 104401 001207          PRTBAD: TYPE     ,SCLF      ;CR-LF
1919 020606 104401 042342          TYPE     ,MSG19
1920 020612 104401 001207          TYPE     ,SCLF      ;CR-LF
1921 020616 016001 000024          MOV      $RMBA(R0),R1      ;PUT THE END ADDRESS INTO R1
1922 020622 016046 000004          MOV      $WRDM(R0),-(SP)    ;FIND THE BEGINNING OF THE SECTOR
1923 020626 005416          NEG      (SP)              ;GET THE POSITIVE NUMBER OF WORD COUNT
1924 020630 066016 000022          ADD      $RMWC(R0),(SP)    ;SUBTRACT THE WORDS NOT TRANSFERED
1925 020634 005046          CLR      -(SP)              ;MAKE THE UPPER DIVIDEND 0
1926 020636 012746 000400          MOV      #256, -(SP)        ;DIVIDE THE WORDS TRANSFERED BY THE SECTOR SIZE
1927 020642 004737 021664          JSR      PC,LINKDV          ;DIVIDE
1928 020646 005716          TST      (SP)              ;REMAINDER = 0 ?
1929 020650 001403          BEQ      1$              ;BR IF IT IS - COMPLETE SECTOR TRANSFERED
1930 020652 006316          ASL      (SP)              ;CONVERT THE RESIDUAL SECTOR SIZE INTO BYTE COUNT
1931 020654 161601          SUB      (SP),R1           ;SUBTRACT IT FROM THE END ADDRESS
1932 020656 000402          BR       2$              ;FINISH THE SIZING
1933 020660 162701 001000 1$:      SUB      #1000,R1          ;SUBTRACT FULL SECTOR SIZE FROM END ADDR
1934 020664 062706 000004 2$:      ADD      #4,SP           ;RESTORE THE STACK POINTER
1935 020670 012702 000007 3$:      MOV      #7,R2           ;R2 CONTAINS THE WORDS/LINE COUNT
1936 020674 020160 000024 4$:      CMP      R1,$RMBA(R0)    ;PRINTED ALL THE SECTOR ?
1937 020700 001415          BEQ      5$              ;BR IF ALL PRINTED
1938 020702 104414 042330          DISPLY  ,BLNKS1          ;TYPE 1 BLANK
1939 020706 012146          MOV      (R1)+, -(SP)      ;PUT THE DATA ON THE STACK
1940 020710 004737 021062          JSR      PC,LINOC        ;TYPE THE DATA
1941 020714 022711 177777          CMP      #-1,(R1)         ;END OF FILE ?
1942 020720 001405          BEQ      5$              ;BRANCH IF SO
    
```

```

1943 020722 005302          DEC      R2          ;DECREMENT THE HORIZONTAL COUNT
1944 020724 001363          BNE      4$          ;BR IF NOT AT THE END OF THE LINE
1945 020726 104414 001207  DISPLY   $CRLF       ;CR-LF
1946 020732 000756          BR       3$          ;RESTORE THE WORDS/LINE COUNT
1947 020734 104414 001207  5$:     DISPLY   $CRLF       ;PRINT WHAT REMAINS IN THE BUFFER
1948 020740 000207          6$:     RTS        PC          ;RETURN
1949
1950
1951
1952 020742 104412          FILBUF: SAVREG       ;SAVE THE REGISTERS
1953 020744 016001 000006  MOV      $BUF(R0),R1 ;BUFFER ADDRESS
1954 020750 016002 000004  MOV      $WRDM(R0),R2 ;POSITIVE WORD COUNT
1955 020754 005402          NEG      R2          ;
1956 020756 012705 005262  MOV      #STNDAT,R5  ;PATTERN ADDRESS
1957 020762 012703 000020  MOV      #20,R3      ;PATTERN COUNT
1958 020766 012521          3$:     MOV      (R5)+,(R1)+ ;MOVE THE PATTERN INTO THE BUFFER
1959 020770 005302          DEC      R2          ;DECREMENT THE WORD COUNT
1960 020772 003407          BLE      4$          ;BR IF DONE (WORD COUNT = 0)
1961 020774 005303          DEC      R3          ;DECREMENT THE PATTERN COUNT
1962 020776 001373          BNE      3$          ;BR IF MORE PATTERN
1963 021000 012703 000020  MOV      #20,R3      ;RESTORE PATTERN COUNT
1964 021004 012705 005262  MOV      #STNDAT,R5  ;RESTORE THE ADDRESS
1965 021010 000766          BR       3$          ;CONTINUE DISTRIBUTING THE PATTERN
1966 021012 104413          4$:     RESREG      ;RESTORE THE REGISTERS
1967 021014 000207          RTS        PC          ;RETURN
1968
1969
1970          .SBTTL  ERROR MESSAGE GENERATION ROUTINES
1971
1972          ;PRINT LINE 1 OF ERROR MESSAGE:
1973          ;'HH:MM:SS'
1974
1975 021016 032777 002000 160130 LINE1: BIT      #SW10,@SWR  ;SWITCH 10 SET ?
1976 021024 001402          BEQ     1$          ;BR IF NOT
1977 021026 104401 001202  TYPE     $BELL      ;RING THE BELL
1978 021032 032777 020000 160114 1$:     BIT      #SW13,@SWR  ;INHIBIT TYPEOUT ?
1979 021040 001403          BEQ     2$          ;BR IF NOT
1980 021042 104414 001207  DISPLY   $CRLF       ;CR-LF
1981 021046 000404          BR       3$          ;EXIT
1982 021050 004737 021330  2$:     JSR      PC,$TIME  ;TYPE THE TIME
1983 021054 104414 042330  DISPLY   ,BLNK$1    ;TYPE 1 BLANK
1984 021060 000207          3$:     RTS        PC          ;RETURN & TYPE DESCRIPTION
1985
1986          ;OCTAL TYPEOUT ROUTINE
1987          ;CALL:
1988          ;
1989          ;     MOV      NUM,-(SP) ;PUT THE NUMBER ON THE STACK
1990          ;     JSR      PC,LINOCT
1991          ;     RETURN
1992 021062 016646 000002          LINOCT: MOV      2(SP),-(SP) ;PUT NUMBER IN PROPER LOCATION ON STACK
1993 021066 004737 022624          JSR      PC,$SB20    ;CONVERT THE NUMBER TO OCTAL
1994 021072 012637 021106          MOV      (SP)+,1$    ;GET THE ADDRESS OF THE ASCII STRING
1995 021076 062737 000005 021106  ADD      #5.,1$      ;ADDRESS THE LAST 6 ASCII DIGITS
1996 021104 104414          DISPLY   ;TYPE IT
1997 021106 000000          1$:     .WORD     0      ;ADDRESS
1998 021110 012616          MOV      (SP)+,(SP)  ;CORRECT THE STACK
1999 021112 000207          RTS        PC          ;RETURN
    
```



```

2000
2001
2002 ;ROUTINE TO CONVERT THE INPUT NUMBER TO DECIMAL AND TYPE IT WITH
2003 ;LEADING ZERO SUPPRESSION
2004 ;CALL:
2005 ;: MOV NUM,-(SP) ;PUT THE NUMBER ON THE STACK
2006 ;: JSR PC,LINDEC
2007 ;: RETURN
2008 021114 016646 000002 LINDEC: MOV 2(SP),-(SP) ;SET UP STACK FOR CONVERT
2009 021120 004737 022574 JSR PC,$SB2D ;CONVERT IT TO DECIMAL
2010 021124 004737 022200 JSR PC,$SUPRS ;TYPE IT (WITH LEADING ZEROS SUPRESSED)
2011 021130 012616 MOV (SP)+,(SP) ;RESTORE STACK POINTER
2012 021132 000207 RTS PC
2013
2014
2015 .SBTTL GENERAL SUPPORT SUBROUTINES
2016
2017
2018 ;ROUTINE TO CHECK FOR KW11-L OR KW11-P CLOCKS
2019
2020 021134 012737 177777 001316 CKCLK: MOV #-1,CLKFLG ;CLEAR CLOCK AVAILABILITY FLAG
2021 021142 012737 177777 001314 MOV #-1,PCLOCK ;CLEAR KW11-P CLOCK AVAILABILITY FLAG
2022 021150 012737 021230 000004 MOV #CKCLK1,ERRVEC ;SET UP VECTOR FOR CLOCK CHECK
2023 021156 005037 000006 CLR @#ERRVEC+2 ;NEW PSW
2024 021162 005777 160114 TST @SLKCSR ;CHECK FOR KW11-P
2025 021166 005037 001316 CLR CLKFLG ;SET CLOCK AVAILABILITY FLAG
2026 021172 005037 001314 CLR PCLOCK ;SET KW11-P CLOCK FLAG
2027 021176 013701 001306 MOV $LPVEC,R1 ;KW11-P VECTOR ADDRESS
2028 021202 012721 021426 MOV #CLOCK,(R1)+ ;SET UP KW11-P VECTOR
2029 021206 012711 000300 MOV #300,(R1) ;PSW - PRI 6
2030 021212 012777 174575 160064 MOV #-1667,@SLKCSB ;LOAD COUNTER BUFFER WITH 16.67
2031 021220 012777 000131 160054 MOV #131,@SLKCSR ;SET CLOCK - CNT UP, 10US. CONT INT
2032 021226 000434 BR CKCLK3
2033 021230 062706 000004 CKCLK1: ADD #4,SP ;RESTORE THE STACK POINTER
2034 021234 012737 021276 000004 MOV #CKCLK2,@#ERRVEC ;CHANGE ERROR VECTOR TO CHECK FOR KW11-L
2035 021242 005777 160042 TST @SLKS ;LOOK FOR KW11-L
2036 021246 005037 001316 CLR CLKFLG ;SET CLOCK FLAG
2037 021252 013701 001312 MOV $LLVEC,R1 ;KW11-L VECTOR ADDRESS
2038 021256 012721 021426 MOV #CLOCK,(R1)+ ;SET UP KW11-L VECTOR
2039 021262 012711 000300 MOV #300,(R1) ;PSW - PRI 6
2040 021266 012777 000100 160014 MOV #100,@SLKS ;SET KW11-L INTERRUPT
2041 021274 000411 BR CKCLK3
2042 021276 062706 000004 CKCLK2: ADD #4,SP ;RESTORE THE STACK POINTER
2043 021302 104401 042635 TYPE ,NEDCLK ;'P OR L CLOCK MUST BE ON SYSTEM'
2044 021306 005737 000042 TST 42 ;UNDER MONITOR CONTROL ?
2045 021312 001400 BEQ 1$ ;BR IF NOT
2046 021314 000177 160060 1$: JMP @RSTART ;JUMP TO RESTART
2047
2048 021320 012737 000006 000004 CKCLK3: MOV #6,@#ERRVEC ;RESTORE THE ERROR VECTOR
2049 021326 000207 RTS PC
2050
2051
2052 ;ROUTINE TO TYPE THE TIME
2053
2054 021330 005737 001316 $TIME: TST CLKFLG ;CLOCK ON THE SYSTEM ?
2055 021334 001033 BNE 1$ ;BR IF NOT
2056 021336 104401 001207 TYPE ,$CRLF ;CR-LF
    
```

```

2057 021342 013746 001344      MOV    HOUR,-(SP)      ;PUT 'HOURS' ON THE STACK
2058 021346 004737 022574      JSR    PC,$SB2D       ;CONVERT TO DECIMAL
2059 021352 004537 022110      JSR    R5,REPLZ      ;TYPE IT
2060 021356 000002              .WORD  2              ;TYPE 2 DIGITS
2061 021360 104401 042332      TYPE   ,COLON        ;:':
2062 021364 013746 001346      MOV    MINUTE,-(SP)   ;PUT 'MINUTES' ON THE STACK
2063 021370 004737 022574      JSR    PC,$SB2D       ;CONVERT TO DECIMAL
2064 021374 004537 022110      JSR    R5,REPLZ      ;TYPE IT
2065 021400 000002              .WORD  2              ;TYPE 2 DIGITS
2066 021402 104401 042332      TYPE   ,COLON        ;:':
2067 021406 013746 001350      MOV    SECOND,-(SP)  ;PUT SECONDS ON THE STACK
2068 021412 004737 022574      JSR    PC,$SB2D       ;CONVERT TO DECIMAL
2069 021416 004537 022110      JSR    R5,REPLZ      ;TYPE IT
2070 021422 000002              .WORD  2              ;TYPE 2 DIGITS
2071 021424 000207              1$:   RTS    PC
2072
2073              ;CLOCK HANDLER ROUTINE
2074
2075 021426 005337 001352      CLOCK: DEC    SIXTEE   ;INCREMENT THE 1/60 SECOND COUNTER
2076 021432 001033              BNE    1$            ;BR IF A SECOND NOT COUNTED
2077 021434 013737 001320 001352      MOV    HZ,SIXTEE     ;RESTORE THE VALUE
2078 021442 005237 001350              INC    SECOND        ;COUNT THE SECOND
2079 021446 022737 000074 001350      CMP    #60.,SECOND   ;AT MAXIMUM ?
2080 021454 001022              BNE    1$            ;BR IF NOT
2081 021456 005037 001350              CLR    SECOND        ;CLEAR THE SECOND'S COUNTER
2082 021462 005237 001346              INC    MINUTE        ;COUNT THE MINUTE
2083 021466 022737 000074 001346      CMP    #60.,MINUTE   ;AT MAXIMUM ?
2084 021474 001012              BNE    1$            ;BR IF NOT
2085 021476 005037 001346              CLR    MINUTE        ;CLEAR THE MINUTE'S COUNTER
2086 021502 005237 001344              INC    HOUR          ;COUNT THE HOURS
2087 021506 022737 001747 001344      CMP    #999.,HOUR    ;AT MAXIMUM
2088 021514 103002              BHIS   1$            ;BR IF NOT
2089 021516 005037 001344              CLR    HOUR          ;CLEAR THE HOURS
2090 021522 012746 000021      1$:   MOV    #17.,-(SP) ;17 MS ON THE STACK
2091 021526 004737 033734      JSR    PC,RMTMR      ;DRIVER TIMER ROUTINE
2092 021532 000002              2$:   RTI
2093
2094              ;COMMAND DECODE ROUTINE
2095              ;CALL:
2096              .:       MOV    #-1,CFLAG ;'CFLAG' IS NORMALLY SET BY THE TTY SERVICE
2097              .:       .:       .:       ;ROUTINE IN INTERRUPT MODE
2098              .:       JSR    PC,KSR
2099              .:       .:       .:       ;SYSTEM BUSY RETURN
2100              .:       RETURN1 ;RETURN AFTER KEYBOARD SERVICED
2101              .:       RETURN2
2102 021534 104412              KSR:   SAVREG        ;SAVE THE REGISTERS
2103 021536 012737 000200 177776      MOV    #PR4,PS       ;SET PRIORITY TO 4
2104 021544 005037 001336              CLR    CFLAG        ;CLEAR THE 'CONTROL C' FLAG
2105 021550 004737 021330              JSR    PC,$TIME      ;TYPE THE TIME
2106 021554 005777 157402              TST    @TKB          ;CLEAR ANY GARBAGE IN THE TTY BUFFER
2107 021560 005737 001336              TST    CFLAG        ;CHECK THE CONTROL C FLAG
2108 021564 001002              BNE    7$            ;EXIT IF 'CONTROL C' ENTERED
2109 021566 000240              NOP
2110 021570 000240              NOP
2111 021572 104413              7$:   RESREG        ;RESTORE R0 - R5
2112 021574 062716 000002      ADD    #2,(SP)       ;INCREMENT THE RETURN ADDRESS
2113 021600 005777 157356      TST    @TKB          ;CLEAR THE TTY BUFFER

```

```

2114 021604 052777 000100 157346      B'S      #BIT06,@$TKS      ;SET TTY INTERRUPT ENABLE
2115 021612 005037 177776      CLR      PS           ;SET PRIORITY BACK TO ZERO
2116 021616 000207                RTS      PC           ;RETURN
2117
2118      ;ROUTINE TO CLEAR THE DPB FOR THE ASSIGNED DRIVE
2119      ;CALL:
2120      ;      MOV      #DPB,R0           ;DPB ADDRESS
2121      ;      JSR      PC,CLRDPB
2122      ;
2123
2124      CLRDPB:
2125      021620 010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
2126      021622 010346      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
2127      021624 010446      MOV      R4,-(SP)      ;;PUSH R4 ON STACK
2128      021626 010546      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
2129      021630 010004      MOV      R0,R4         ;GET THE DPB ADDRESS
2130      021632 062704 000002      ADD      #2,R4         ;ADDRESS OF FIRST LOCN TO BE CLEARED
2131      021636 012703 000020      MOV      #$EMTAB-2,R3 ;NUMBER OF LOCATIONS TO CLEAR
2132      021642 005024      1$: CLR      (R4)+       ;CLEAR THE STORAGE LOCATION
2133      021644 162703 000002      SUB      #2,R3         ;DECREMENT THE BYTE COUNT
2134      021650 001374      BNE      1$           ;LOOPING BACK
2135      021652 012605      MOV      (SP)+,R5      ;;POP STACK INTO R5
2136      021654 012604      MOV      (SP)+,R4      ;;POP STACK INTO R4
2137      021656 012603      MOV      (SP)+,R3      ;;POP STACK INTO R3
2138      021660 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
2139      021662 000207      RTS      PC           ;RETURN
2140
2141      LINKDV: SAVREG
2142      021664 104412      MOV      26(SP),R5     ;STORE R0 - R5
2143      021666 016605 000026      CLR      R4           ;DIVISOR
2144      021672 005004      MOV      30(SP),R2     ;OTHER DIVISOR WORD
2145      021674 016602 000030      MOV      32(SP),R3     ;UPPER DIVIDEND WORD
2146      021700 016603 000032      CLR      R0           ;LOWER DIVIDEND WORD
2147      021704 005000      CLR      R1           ;CLEAR OTHER DIVIDEND REGISTERS
2148      021706 005001
2149      021710 004737 021732      JSR      PC,M.DPID     ;GO TO THE DIVIDE ROUTINE
2150      021714 010166 000030      MOV      R1,30(SP)     ;REMAINDER ON THE STACK
2151      021720 010366 000032      MOV      R3,32(SP)     ;QUOTIENT ON THE STACK
2152      021724 104413      RESREG
2153      021726 012616      MOV      (SP)+,(SP)    ;RESTORE R0 - R5
2154      021730 000207      RTS      PC           ;MOVE RETURN UP THE STACK
2155
2156      ;
2157      ; DIVISION UTILITY SUBROUTINE
2158      ; R0-R1-R2-R3=DIVIDEND
2159      ; R4-R5=DIVISOR
2160      ; R0-R1=REMAINDER AFTER DIVISION
2161      ; R2-R3=QUOTIENT AFTER DIVISION
2162      ; ENTER WITH JSR PC,M.DPID
2163
2164      M.DPID: MOV      #40,-(SP)     ;COUNTER FOR DIVISION CYCLES
2165      021732 012746 000040      MOV      R4,-(SP)     ;HIGH ORDER
2166      021736 010446      MOV      R5,-(SP)     ;LOW ORDER DIVISOR TO THE STACK
2167      021740 010546      NEG      2(SP)         ;FORM NEGATIVE
2168      021742 005466 000002      NEG      @SP          ;VERSION OF THE DIVISOR
2169      021746 005416
2170      021750 005666 000002      SBC      2(SP)
2171      021754 061601      ADD      @SP,R1
2172      021756 005500      ADC      R0
2173      021760 066600 000002      ADD      2(SP),R0     ;PERFORM THE INITIAL SUBTRACTION

```

```

2164 021764 103445          BCS      M.DP50          ;IF CARRY THEN OVERFLOW HAS OCCURRED
2165 021766 005046          CLR      -(SP)          ;THIS IS A LONGER LASTING CARRY BIT
2166 021770 006103          M.DP40: ROL      R3
2167 021772 006102          ROL      R2
2168 021774 006101          ROL      R1
2169 021776 006100          ROL      R0
2170 022000 005716          TST      @SP
2171 022002 001410          BEQ      M.DP41          ;TEST "CARRY" INDICATOR
2172 022004 005016          CLR      @SP          ;IF NO "CARRY" THEN ADD ELSE SUBTRACT
2173 022006 066601 000002  ADD      2(SP),R1      ;CLEAR UP FOR NEXT TIME
2174 022012 005500          ADC      R0
2175 022014 005516          ADC      @SP          ;ADD -(DIVISOR)
2176 022016 066600 000004  ADD      4(SP),R0; <- ;SET "CARRY"
2177 022022 000404          BR       M.DP42
2178 022024 060501          M.DP41: ADD      R5,R1
2179 022026 005500          ADC      R0          ;ADD +(DIVISOR)
2180 022030 005516          ADC      @SP          ;SET "CARRY"
2181 022032 060400          ADD      R4,R0      ; <-
2182 022034 005516          M.DP42: ADC      @SP          ;SET "CARRY"
2183 022036 005716          TST      @SP          ;TEST THE UPDATE INDICATOR
2184 022040 001401          BEQ      .+4          ;-> ;IF ZERO FORGET IT
2185 022042 005203          INC      R3          ; <- ;NO CARRY POSSIBLE HERE
2186 022044 005366 000006  DEC      6(SP)        ; <- ;DECREMENT COUNTER
2187 022050 003347          BGT      M.DP40          ;BRANCH IF MORE TO DO
2188 022052 006003          ROR      R3
2189 022054 103404          BCS      M.DP44
2190 022056 060501          ADD      R5,R1
2191 022060 005500          ADC      R0
2192 022062 060400          ADD      R4,R0
2193 022064 000241          CLC
2194 022066 006103          M.DP44: ROL      R3
2195 022070 062706 000010  ADD      #10,SP      ;ADJUST STACK BY 4 WORDS
2196 022074 000242          CLV
2197 022076 000207          RTS      PC
2198 022100 062706 000006  M.DP50: ADD      #6,SP
2199 022104 000262          SEV
2200 022106 000207          RTS      PC
2201
2202
2203          ;ROUTINE TO REPLACE LEADING ZEROS IN A NUMERIC STRING WITH SPACES
2204          ;CALL
2205          ;      MOV      #ADR,-(SP)          ;ADDRESS OF NUMBER (IN ASCII)
2206          ;      JSR      R5,REPLZ
2207          ;      .WORD      N          ;'N' IS NUMBER OF DIGITS TO BE TYPED
2208
2209 022110 010046          REPLZ: MOV      R0,-(SP)          ;SAVE R0
2210 022112 012746 000012  MOV      #10,-(SP)      ;MAXIMUM NUMBER OF DIGITS TO BE TYPED
2211 022116 162516          SUB      (R5)+,(SP)      ;SUBTRACT DIGITS TO FORM INDEX
2212 022120 016600 000006  MOV      6(SP),R0      ;ADDRESS OF NUMBER TO R0
2213 022124 122710 000060  1$: CMPB   #'0',(R0)      ;BYTE EQUAL TO ASCII '0' ?
2214 022130 001004          BNE      2$          ;BR IF NOT
2215 022132 112710 000040  MOVB   #40,(R0)      ;REPLACE THE ZERO WITH A SPACE
2216 022136 005200          INC      R0          ;INCREMENT THE BYTE ADDRESS
2217 022140 000771          BR       1$          ;GO BACK AND LOOK FOR MORE LEADING ZEROS
2218 022142 105710          2$: TSTB   (R0)          ;SEE IF ZERO BYTE TERMINATOR
2219 022144 001003          BNE      3$          ;BR IF NOT
2220 022146 005300          DEC      R0          ;BACKUP STRING POINTER

```

```
2221 022150 112710 000060
2222 022154 016637 000006 022170 3$:   MOVB   #'0,(R0)      ;PUT A ZERO BACK IN
2223 022162 062637 022170      MOV    6(SP),4$     ;PUT ADDRESS IN LOCATION FOR TYPEOUT
2224 022166 104401      ADD    (SP)+,4$    ;BEGINNING OF SIGNIFICANT DIGITS
2225 022170 000000      TYPE   TYPE        ;TYPE THE NUMBER
2226 022172 012600      4$:   .WORD  0       ;ADDRESS OF NUMBER
2227 022174 012616      MOV    (SP)+,R0    ;RESTORE R0
2228 022176 000205      MOV    (SP)+,(SP) ;MOVE RETURN ADDRESS
2229      RTS    R5      ;RETURN
2230      ;TYPE NUMERICAL ASCIZ STRING SUPPRESS LEADING ZEROS
2231      ;CALL
2232      ;
2233      ;   MOV    #NUMADR,-(SP) ;FIRST ADDRESS OF ASCIZ STRING
2234      ;   JSR    PC,$SUPRS
2235
2236 022200 010046      $SUPRS: MOV   R0,-(SP)    ;SAVE R0
2237 022202 016600 000004      MOV   4(SP),R0    ;PICKUP THE POINTER
2238 022206 105710      1$:   TSTB   (R0)     ;TERMINATOR ?
2239 022210 001403      BEQ   2$          ;BR IF YES
2240 022212 122720 000060      CMPB  #'0,(R0)+   ;IS THIS AN ASCII '0' ?
2241 022216 001773      BEQ   1$          ;BR IF YES
2242 022220 005300      2$:   DEC    R0     ;BACKUP BY '1'
2243 022222 010037 022230      MOV   R0,3$      ;SAVE FOR TYPING
2244 022226 104414      DISPLY ;GO PRINT
2245 022230 000000      3$:   .WORD  0       ;ASCIZ POINTER GOES HERE
2246 022232 012600      MOV   (SP)+,R0    ;RESTORE R0
2247 022234 012616      MOV   (SP)+,(SP) ;RESTORE THE STACK
2248 022236 000207      RTS    PC        ;RETURN
2249
2250      ;ROUTINE TO TYPE AT PRIORITY 4
2251
2252 022240 013746 177776      TYPRI4: MOV   @#PS,-(SP) ;SAVE THE PRESENT STATUS
2253 022244 012737 000200 177776      MOV   #200,@#PS   ;CHANGE THE PRIORITY TO 4
2254 022252 012537 022262      MOV   (R5)+,1$    ;MESSAGE ADDRESS
2255 022256 004737 024116      JSR   PC,$TYPE    ;TYPE THE MESSAGE
2256 022262 000000      1$:   .WORD  0       ;MESSAGE ADDRESS GOES HERE
2257 022264 000205      RTS    R5        ;RETURN
2258
2259      ;ROUTINE TO TYPE ERRORS
2260      ;CALL
2261      ;
2262      ;   DISPLY ;MUST DEFINED IN 'TRAP' TABLE
2263      ;   MESADR ;ADDRESS OF MESSAGE
2264      ;   RETURN
2265 022266 032777 020000 156660 $DSPLY: BIT   #BIT13,@SWR ;INHIBIT ERROR TYPEOUT ?
2266 022274 001002      BNE   1$          ;BR IF YES
2267 022276 000137 024116      JMP   $TYPE       ;TYPE THE MESSAGE
2268 022302 062716 000002      1$:   ADD   #2,(SP) ;INCREMENT THE RETURN
2269 022306 000002      RTI              ;RETURN
2270
2271      ;THIS ROUTINE IS USED TO CHECK IF AN
2272      ;ASCII CHARACTER IS A DIGIT BETWEEN 0 AND 7.
2273      ;CALL
2274      ;
2275      ;   MOV   #ADR,R1 ;ADDRESS OF ASCII CHARACTER
2276      ;   JSR   R5,CK.OCT ;CHECK THE CHARACTER
2277      ;   RETURN1 ;CHARACTER IS NOT BETWEEN 0-7
2277      ;   RETURN2 ;CHARACTER IS IN R2 AS A
```

```

2278                                     :                               ;OCTAL DIGIT
2279                                     :
2280 022310 121127 000060      CK.OCT: CMPB   (R1),#'0      ;LESS THAN ZERO?
2281 022314 103407                BLO     1$          ;YES -- BRANCH
2282 022316 121127 000067      CMPB   (R1),#'7      ;GREATER THAN SEVEN?
2283 022322 101004                BHI     1$          ;YES -- BRANCH
2284 022324 111102                MOVB   (R1),R2      ;GET THE CHARACTER
2285 022326 042702 177770      BIC    #'C7,R2     ;STRIP AWAY THE ASCII
2286 022332 005725                TST    (R5)+       ;ADJUST FOR RETURN
2287 022334 000205      1$:    RTS     R5          ;RETURN
2288
2289                                     ;THIS ROUTINE IS USED TO CHECK AN ASCII CHARACTER
2290                                     ;AND DETERMINE IF IT IS A DIGIT BETWEEN 0 AND 9.
2291                                     ;CALL
2292                                     :
2293                                     :       MOV    #ADR,R1      ;ADDRESS OF ASCII CHARACTER
2294                                     :       JSR    R5,CK.DEC   ;CHECK THE CHARACTER
2295                                     :       RETURN1          ;NOT BETWEEN 0 AND 9
2296                                     :       RETURN2          ;BETWEEN 0 AND 9
2297                                     :       ;R2 = DIGIT
2298 022336 121127 000060      CK.DEC: CMPB   (R1),#'0      ;LESS THAN ZERO?
2299 022342 103407                BLO     1$          ;YES -- BRANCH
2300 022344 121127 000071      CMPB   (R1),#'9      ;GREATER THAN NINE?
2301 022350 101004                BHI     1$          ;YES -- BRANCH
2302 022352 111102                MOVB   (R1),R2      ;GET THE CHARACTER
2303 022354 042702 000060      BIC    #'0,R2     ;STRIP AWAY THE ASCII
2304 022360 005725                TST    (R5)+       ;ADJUST FOR RETURN
2305 022362 000205      1$:    RTS     R5          ;RETURN
2306
2307                                     ;THIS ROUTINE WILL CHECK AN ASCII CHARACTER TO
2308                                     ;DETERMINE WHAT IT IS.
2309                                     ;CALL
2310                                     :
2311                                     :       MOV    #ADR,R1      ;ADDRESS OF ASCII CHARACTER
2312                                     :       JSR    R5,CK.CHR   ;CHECK CHARACTER
2313                                     :       RETURN ADR1       ;UNKNOWN CHARACTER
2314                                     :       RETURN ADR2       ;CARRIAGE RETURN * (R1)=ADR+1
2315                                     :       RETURN ADR3       ;COMMA * (R1)=ADR+1
2316                                     :       RETURN ADR4       ;PERIOD * (R1)=ADR+1
2317                                     :       RETURN ADR5       ;DIGIT BETWEEN 0 AND 7.
2318                                     :       RETURN ADR6       ;DIGIT BETWEEN 8 AND 9.
2319                                     :       ;R2 = DIGIT * (R1)=ADR+1
2320 022364 105711      CK.CHR: TSTB   (R1)          ;'CARRIAGE RETURN'?
2321 022366 001417                BEQ    3$          ;YES -- BRANCH
2322 022370 121127 000054      CMPB   (R1),#',     ;'COMMA'?
2323 022374 001413                BEQ    2$          ;YES -- BRANCH
2324 022376 121127 000056      CMPB   (R1),#'.     ;'PERIOD'?
2325 022402 001407                BEQ    1$          ;YES -- BRANCH
2326 022404 004537 022336      JSR    R5,CK.DEC   ;'DIGIT'?
2327 022410 000410                BR     4$          ;NO -- BRANCH
2328 022412 004537 022310      JSR    R5,CK.OCT   ;OCTAL ?
2329 022416 005725                TST    (R5)+       ;DIGIT BETWEEN 8-9
2330 022420 005725                TST    (R5)+       ;DIGIT BETWEEN 0-7
2331 022422 005725      1$:    TST    (R5)+       ;PERIOD
2332 022424 005725      2$:    TST    (R5)+       ;COMMA
2333 022426 005725      3$:    TST    (R5)+       ;CARRIAGE RETURN
2334 022430 005201                INC    R1          ;MOVE POINTER TO NEXT CHARACTER

```



```

2380 022566 012604          MOV      (SP)+,R4          ;RESTORE R4
2381 022570 011505          MOV      (R5),R5          ;GET RETURN ADDRESS
2382 022572 000205          RTS       R5              ;RETURN
2383
2384          ;THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
2385          ;UNSIGNED DECIMAL ASCIZ NUMBER.
2386          ;CALL
2387          ;
2388          ;      MOV      NUMBER,-(SP)      ;PUT THE NUMBER ON THE STACK
2389          ;      JSR      PC,$SB2D         ;CALL
2390          ;      RETURN                     ;ADDRESS OF THE 1ST ASCIZ CHAR IS ON THE STACK
2391          ;NOTE: THE PROGRAM REQUIRES THIS FORM OF '$SB2D', NOT THE VERSION ON
2392          ;      THE SYSMAC LIBRARY, REV C AND LATER
2393
2394 022574 016637 000002 022620 $SB2D: MOV      2(SP),1$          ;SAVE THE BINARY NUMBER
2395 022602 012746 022620          MOV      #1$,-(SP)        ;SET THE POINTER
2396 022606 004737 026314          JSR      PC,$DB2D         ;CALL THE DOUBLE LENGTH CONVERT
2397 022612 012666 000002          MOV      (SP)+,2(SP)      ;PICKUP THE POINTER
2398 022616 000207          RTS       PC              ;RETURN
2399 022620 000000 000000          1$:      .WORD    0,0
2400
2401          ;THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
2402          ;UNSIGNED OCTAL ASCIZ NUMBER.
2403          ;CALL
2404          ;
2405          ;      MOV      NUMBER,-(SP)      ;PUT THE NUMBER ON THE STACK
2406          ;      JSR      PC,$SB20         ;CALL
2407          ;      RETURN                     ;ADDRESS OF THE 1ST ASCIZ CHAR IS ON THE STACK
2408          ;NOTE: THE PROGRAM REQUIRES THIS FORM OF '$SB20', NOT THE VERSION ON
2409          ;      THE SYSMAC LIBRARY, REV C AND LATER
2410
2411 022624 016637 000002 022650 $SB20: MOV      2(SP),1$          ;SAVE THE BINARY NUMBER
2412 022632 012746 022650          MOV      #1$,-(SP)        ;SET THE POINTER
2413 022636 004737 026510          JSR      PC,$DB20         ;CALL THE DOUBLE LENGTH CONVERT
2414 022642 012666 000002          MOV      (SP)+,2(SP)      ;PICKUP THE POINTER
2415 022646 000207          RTS       PC              ;RETURN
2416 022650 000000 000000          1$:      .WORD    0,0
2417
2418          ;KEYBOARD INTERRUPT INITIALIZATION ROUTINE
2419          ;CALL
2420          ;
2421          ;      JSR      PC,$TKINT        ;
2422          ;      RETURN
2423 022654 012737 022704 000060 $TKINT: MOV      #$TKSRV,TKVEC ;SETUP VECTOR
2424 022662 012737 000240 000062          MOV      #PR5,TKVEC+2   ;PRIORITY TO 5
2425 022670 005777 156266          TST      @TKB            ;CLEAR THE BUFFER
2426 022674 012777 000100 156256          MOV      #BIT06,@TKS     ;SET INTERRUPT ENABLE
2427 022702 000207          RTS       PC              ;RETURN
2428
2429          ;KEYBOARD INTERRUPT SERVICE ROUTINE
2430          ;CALL
2431          ;
2432          ;      ENTER VIA INTERRUPT
2433 022704 104410          $TKSRV: RDCHR              ;READ THE KEYBOARD
2434 022706 112637 023046          MOV      (SP)+,5$        ;GET THE CHARACTER
2435 022712 023727 023046 000003          CMP      5$,#3           ;'CONTROL C' ?
2436 022720 001017          BNE      1$              ;BR IF NOT

```



```

2437 022722 104401 001207          TYPE      , $CRLF          : CR-LF
2438 022726 104401 023352          TYPE      , $CNTLC        : 'AC'
2439 022732 012737 177777 001336  MOV      #-1, CFLAG      : SET THE 'CONTROL C' FLAG
2440 022740 005077 156214          CLR      @ $TKS          : CLEAR THE TTY INTERRUPT
2441 022744 104401 001207          TYPE      , $CRLF          : CR-LF
2442 022750 104401 042464          TYPE      , HALTX         : HALT THE PROGRAM
2443 022754 000177 156420          JMP      @ RSTART        : JUMP TO RESTART
2444
2445 022760 023727 001154 000176 1$:  CMP      SWR, #SWREG      : SOFTWARE SWITCH REGISTER IN USE ?
2446 022766 001024 023046 000007  BNE      3$              : BR IF NOT
2447 022770 023727 023046 000007  CMP      5$, #7          : 'CONTROL G' ?
2448 022776 001020 023046 000007  BNE      3$              : BR IF NOT
2449 023000 104401 001207          TYPE      , $CRLF          : CR-LF
2450 023004 104401 026171          TYPE      , $CNTLG        : 'AG'
2451 023010 013746 177776          MOV      PS, -(SP)       : PUT THE STATUS WORD ON THE STACK
2452 023014 012746 023030          MOV      #2$, -(SP)     : RETURN ADDRESS
2453 023020 005077 156134          CLR      @ $TKS          : CLEAR THE TTY INTERRUPT ENABLE
2454 023024 000137 025632          JMP      $GTSWR          : GET THE SWITCH REGISTER ENTRY
2455 023030 012777 000100 156122 2$:  MOV      #100, @ $TKS    : ENABLE TTY KEYBOARD INTERRUPT
2456 023036 000402 023046 000000  BR      4$              : EXIT
2457 023040 104401 023046 000000 3$:  TYPE      , 5$          : ECHO THE CHARACTER
2458 023044 000002 023046 000000 4$:  RTI                    : RETURN
2459
2460 023046 000000 023046 000000 5$:  .WORD    0              : ENTERED CHARACTER
2461
2462
; THIS ROUTINE WILL INPUT A STRING FROM THE TTY
; CALL:
;
; RDLIN
; RETURN HERE
;
; INPUT A STRING FROM THE TTY
; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
; TERMINATOR WILL BE A BYTE OF ALL 0'S

023050 010346          $RDLIN: MOV      R3, -(SP)      : SAVE R3
023052 005046          CLR      -(SP)         : CLEAR THE RUBOUT KEY
2470 023054 012703 023326 1$:  MOV      # $TTYIN, R3   : GET ADDRESS
2471 023060 022703 023340 2$:  CMP      # $TTYIN+10., R3 : BUFFER FULL?
2472 023064 101467          BLOS     4$            : BR IF YES
2473 023066 104410          RDCHR          : GO READ ONE CHARACTER FROM THE TTY
2474 023070 112613          MOV      (SP)+, (R3)    : GET CHARACTER
2475 023072 122713 000177          CMP      #177, (R3)     : IS IT A RUBOUT
2476 023076 001022          BNE      5$            : BR IF NO
2477 023100 005716          TST      (SP)          : IS THIS THE FIRST RUBOUT?
2478 023102 001007          BNE      6$            : BR IF NO
2479 023104 112737 000134 023324          MOV      #' \, 9$      : TYPE A BACK SLASH
2480 023112 104401 023324          TYPE      , 9$         :
2481 023116 012716 177777          MOV      #-1, (SP)     : SET THE RUBOUT KEY
2482 023122 005303 023324 6$:  DEC      R3             : BACKUP BY ONE
2483 023124 020327 023326          CMP      R3, # $TTYIN  : STACK EMPTY?
2484 023130 103445          BLO      4$            : BR IF YES
2485 023132 111337 023324          MOV      (R3), 9$      : SETUP TO TYPEOUT THE DELETED CHAR.
2486 023136 104401 023324          TYPE      , 9$         : GO TYPE
2487 023142 000746          BR      2$            : GO READ ANOTHER CHAR.
2488 023144 005716 023324 5$:  TST      (SP)          : RUBOUT KEY SET?
2489 023146 001406          BEQ      7$            : BR IF NO
2490 023150 112737 000134 023324          MOV      #' \, 9$      : TYPE A BACK SLASH
2491 023156 104401 023324          TYPE      , 9$         :
2492 023162 005016          CLR      (SP)          : CLEAR THE RUBOUT KEY
2493 023164 122713 000025 7$:  CMP      #25, (R3)     : IS CHARACTER A CTRL U?
    
```

```
2494 023170 001003          BNE      10$          ;BR IF NO
2495 023172 104401 026164    TYPE    ,SCNTLU      ;TYPE A CONTROL 'U'
2496 023176 000726          BR       1$          ;GO START OVER
2497 023200 122713 000003    10$:    CMPB     #3,(R3) ;IS CHARACTER A CTRL C ?
2498 023204 001006          BNE      8$          ;BR IF NOT
2499 023206 012737 177777    001336  MOV     #-1,CFLAG   ;SET CNTRL C FLAG
2500 023214 104401 023352    TYPE    ,SCNTLC      ;ECHO IT
2501 023220 000427          BR       11$         ;EXIT
2502 023222 122713 000012    8$:    CMPB     #12,(R3) ;IS CHARACTER A 'LF'?
2503 023226 001011          BNE      3$          ;BRANCH IF NO
2504 023230 105013          CLRB     (R3)        ;CLEAR THE CHARACTER
2505 023232 104401 001207    TYPE    ,SCRLF       ;TYPE A 'CR' & 'LF'
2506 023236 104401 023326    TYPE    ,STTYIN      ;TYPE THE INPUT STRING
2507 023242 000706          BR       2$          ;GO PICKUP ANOTHER CHACTER
2508 023244 104401 001206    4$:    TYPE    ,SQUES    ;TYPE A '?'
2509 023250 000701          BR       1$          ;CLEAR THE BUFFER AND LOOP
2510 023252 111337 023324    3$:    MOVB     (R3),9$  ;ECHO THE CHARACTER
2511 023256 104401 023324    TYPE    ,9$
2512 023262 122723 000015    CMPB     #15,(R3)+  ;CHECK FOR RETURN
2513 023266 001274          BNE      2$          ;LOOP IF NOT RETURN
2514 023270 105063 177777    CLRB     -1(R3)     ;CLEAR RETURN (THE 15)
2515 023274 104401 001210    TYPE    ,SLF        ;TYPE A LINE FEED
2516 023300 005726          11$:    TST      (SP)+     ;CLEAN RUBOUT KEY FROM THE STACK
2517 023302 012603          MOV      (SP)+,R3   ;RESTORE R3
2518 023304 011646          MOV      (SP),-(SP) ;ADJUST THE STACK AND PUT ADDRESS OF THE
2519 023306 016666 000004 000002  MOV      4(SP),2(SP) ;FIRST ASCII CHARACTER ON IT
2520 023314 012766 023326 000004  MOV      #STTYIN,4(SP)
2521 023322 000002          RTI
2522 023324 000          9$:    .BYTE    0          ;RETURN
2523 023325 000          .BYTE    0          ;STORAGE FOR ASCII CHAR. TO TYPE
2524 023326          .BLKB   20.        ;TERMINATOR
2525 023352 136 103 200 $TTYIN: .BLKB 20. ;RESERVE 20 BYTES FOR TTY INPUT
2526          $CNTLC: .ASCIZ /^C/<CRLF> ;CONTROL 'C'
2527          .EVEN
2528
2529 023356 005737 001376    FINISH: TST      FAULT ;COMPATIBLE ?
2530 023362 001004          BNE      1$          ;BRANCH IF NOT
2531 023364 104401 001207    TYPE    ,SCRLF       ;CR-LF
2532 023370 104401 041760    TYPE    ,MESG15      ;MESSAGE: ALL DRIVE COMPATIBLE
2533 023374 104401 001207    1$:    TYPE    ,SCRLF       ;CR-LF
2534 023400 104401 042362    TYPE    ,MESG20
2535 023404 104401 042400    TYPE    ,STARS       ;TYPE STARS MESSAGE
2536 023410 104401 001207    TYPE    ,SCRLF       ;CR-LF
2537 023414 000177 155760    JMP      @RSTART    ;JUMP TO RESTART
2538
2539          .SBTTL  END OF PASS ROUTINE
```

```
::*****
;*INCREMENT THE PASS NUMBER ($PASS)
;*IF SW12=1 INHIBIT TRACE TRAP
;*IF THERES A MONITOR GO TO IT
;*IF THERE ISN'T JUMP TO FINISH
```

```
023420
023420 000240          $EOP:  NOP
023422 005037 001116    CLR     $STNM      ;;ZERO THE TEST NUMBER
023426 005037 001176    CLR     $TIMES     ;;ZERO THE NUMBER OF ITERATIONS
```



```

023572 013777 001116 155356      MOV      $STNM,@DISPLAY  ;;DISPLAY TEST NUMBER AND ERROR FLAG
023600 032777 002000 155346      BIT      #BIT10,@SWR    ;;BELL ON ERROR?
023606 001402                BEQ      1$             ;;NO - SKIP
023610 104401 001202                TYPE     ,SBELL        ;;RING BELL
023614 005237 001126      1$:      INC      $ERTTL    ;;COUNT THE NUMBER OF ERRORS
023620 011637 001132                MOV      (SP),$ERRPC   ;;GET ADDRESS OF ERROR INSTRUCTION
023624 162737 000002 001132      SUB      #2,$ERRPC
023632 117737 155274 001130      MOV     @ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
023640 032777 020000 155306      BIT      #BIT13,@SWR   ;;SKIP TYPEOUT IF SET
023646 001004                BNE     20$           ;;SKIP TYPEOUTS
023650 004737 023762                JSR     PC,$ERRTYP    ;;GO TO USER ERROR ROUTINE
023654 104401 001207                TYPE     ,SRLF
023660
023660 122737 000001 001232      20$:    CMPB    #APTENV,$ENV   ;;RUNNING IN APT MODE
023666 001007                BNE     2$            ;;NO,SKIP APT ERROR REPORT
023670 113737 001130 023702      MOV     $ITEMB,21$    ;;SET ITEM NUMBER AS ERROR NUMBER
023676 004737 024470                JSR     PC,$ATY4     ;;REPORT FATAL ERROR TO APT
023702      000                21$:    .BYTE   0
023703      000                .BYTE   0
023704 000777                22$:    BR      22$           ;;APT ERROR LOOP
023706 005777 155242      2$:    TST     @SWR        ;;HALT ON ERROR
023712 100002                BPL     3$            ;;SKIP IF CONTINUE
023714 000000                HALT    ;;HALT ON ERROR!
023716 104407                CKSWR   ;;TEST FOR CHANGE IN SOFT-SWR
023720 032777 001000 155226      3$:    BIT      #BIT09,@SWR  ;;LOOP ON ERROR SWITCH SET?
023726 001402                BEQ     4$            ;;BR IF NO
023730 013716 001124                MOV     $LPERR,(SP)   ;;FUDGE RETURN FOR LOOPING
023734 005737 001200      4$:    TST     $ESCAPE     ;;CHECK FOR AN ESCAPE ADDRESS
023740 001402                BEQ     5$            ;;BR IF NONE
023742 013716 001200                MOV     $ESCAPE,(SP)  ;;FUDGE RETURN ADDRESS FOR ESCAPE
023746
023746 022737 023506 000042      5$:    CMP     #SENDAD,@#42  ;;ACT-11 AUTO-ACCEPT?
023754 001001                BNE     6$            ;;BRANCH IF NO
023756 000000                HALT    ;;YES
023760
023760 000002      6$:    RTI             ;;RETURN
    
```

2542
2543

.SBTTL ERROR MESSAGE TYPEOUT ROUTINE

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

```

023762 104401 001207      $ERRTYP:
023762 010046                TYPE     ,SRLF        ;;"CARRIAGE RETURN" & "LINE FEED"
023766 005000                MOV     R0,-(SP)     ;;SAVE R0
023770 153700 001130      CLR     R0            ;;PICKUP THE ITEM INDEX
023772 001004                BISB    @#$ITEMB,R0
023776                                BNE     1$            ;;IF ITEM NUMBER IS ZERO, JUST
                                ;;TYPE THE PC OF THE ERROR
024000 013746 001132      MOV     $ERRPC,-(SP)  ;;SAVE $ERRPC FOR TYPEOUT
                                ;;ERROR ADDRESS
024004 104402                TYPOC   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
024006 000426                BR      6$            ;;GET OUT
024010 005300                1$:    DEC     R0          ;;ADJUST THE INDEX SO THAT IT WILL
024012 006300                ASL     R0            ;;WORK FOR THE ERROR TABLE
    
```



```

024166 000000
024170 132737 000040 001233 61$: .WORD 0 ::MESSAGE ADDRESS
024176 001003 62$: BITB #APTC SUP,$ENVM ::APT CONSOLE SUPPRESSED
024200 112046 BNE 60$ ::YES,SKIP TYPE OUT
024202 001005 2$: MOV B (R0)+,-(SP) ::PUSH CHARACTER TO BE TYPED ONTO STACK
024204 005726 BNE 4$ ::BR IF IT ISN'T THE TERMINATOR
024206 012600 TST (SP)+ ::IF TERMINATOR POP IT OFF THE STACK
024210 062716 000002 60$: MOV (SP)+,R0 ::RESTORE R0
024214 000002 3$: ADD #2,(SP) ::ADJUST RETURN PC
024216 122716 000011 4$: CMPB #HT,(SP) ::RETURN
024222 001430 BEQ 8$ ::BRANCH IF <HT>
024224 122716 000200 CMPB #CRLF,(SP) ::BRANCH IF NOT <CRLF>
024230 001006 BNE 5$
024232 005726 TST (SP)+ ::POP <CR><LF> EQUIV
024234 104401 TYPE ::TYPE A CR AND LF
024236 001207 $CRLF
024240 105037 024446 CLRB $CHARCNT ::CLEAR CHARACTER COUNT
024244 000755 BR 2$ ::GET NEXT CHARACTER
024246 004737 024330 5$: JSR PC,$TYPEC ::GO TYPE THIS CHARACTER
024252 123726 001172 6$: CMPB $FILLC,(SP)+ ::IS IT TIME FOR FILLER CHARS.?
024256 001350 BNE 2$ ::IF NO GO GET NEXT CHAR.
024260 013746 001170 MOV $NULL,-(SP) ::GET # OF FILLER CHARS. NEEDED
::AND THE NULL CHAR.
024264 105366 000001 7$: DECB 1(SP) ::DOES A NULL NEED TO BE TYPED?
024270 002770 BLT 6$ ::BR IF NO--GO POP THE NULL OFF OF STACK
024272 004737 024330 JSR PC,$TYPEC ::GO TYPE A NULL
024276 105337 024446 DECB $CHARCNT ::DO NOT COUNT AS A COUNT
024302 000770 BR 7$ ::LOOP

```

:HORIZONTAL TAB PROCESSOR

```

024304 112716 000040 8$: MOV B #' ,(SP) ::REPLACE TAB WITH SPACE
024310 004737 024330 9$: JSR PC,$TYPEC ::TYPE A SPACE
024314 132737 000007 024446 BITB #7,$CHARCNT ::BRANCH IF NOT AT
024322 001372 BNE 9$ ::TAB STOP
024324 005726 TST (SP)+ ::POP SPACE OFF STACK
024326 000724 BR 2$ ::GET NEXT CHARACTER
024330 $TYPEC:
024330 105777 154624 TSTB @$TKS ::CHAR IN KYBD BUFFER?
024334 100022 BPL 10$ ::BR IF NOT
024336 017746 154620 MOV @$TKB,-(SP) ::GET CHAR
024342 042716 177600 BIC #177600,(SP) ::STRIP EXTRANEIOUS BITS
024346 122716 000023 CMPB #$XOFF,(SP) ::WAS CHAR XOFF
024352 001012 BNE 102$ ::BR IF NOT
024354 105777 154600 101$: TSTB @$TKS ::WAIT FOR CHAR
024360 100375 BPL 101$
024362 117716 154574 MOVB @$TKB,(SP) ::GET CHAR
024366 042716 177600 BIC #177600,(SP) ::STRIP IT
024372 122716 000021 CMPB #$XON,(SP) ::WAS IT XON?
024376 001366 BNE 101$ ::BR IF NOT
024400 102$: TST (SP)+ ::FIX STACK
024402 005726 10$: TST @$TPS ::WAIT UNTIL PRINTER IS READY
024402 105777 154556 BPL 10$
024406 100375 MOV B 2(SP),@$TPB ::LOAD CHAR TO BE TYPED INTO DATA REG.
024410 116677 000002 154550

```

```

024416 122766 000015 000002      CMPB   #CR,2(SP)      ;;IS CHARACTER A CARRIAGE RETURN?
024424 001003          BNE     1$           ;;BRANCH IF NO
024426 105037 024446          CLRB   $CHARCNT     ;;YES--CLEAR CHARACTER COUNT
024432 000406          BR      $TYPEX      ;;EXIT
024434 122766 000012 000002 1$:    CMPB   #LF,2(SP)     ;;IS CHARACTER A LINE FEED?
024442 001402          BEQ    $TYPEX      ;;BRANCH IF YES
024444 105227          INCB  (PC)+        ;;COUNT THE CHARACTER
024446 000000          $CHARCNT:.WORD   0  ;;CHARACTER COUNT STORAGE
024450 000207          $TYPEX: RTS      PC
    
```

2546
2547

.SBTTL APT COMMUNICATIONS ROUTINE

```

*****
024452 112737 000001 024716 $ATY1: MOVB   #1,$FFLG      ;;TO REPORT FATAL ERROR
024460 112737 000001 024714 $ATY3: MOVB   #1,$MFLG      ;;TO TYPE A MESSAGE
024466 000403          BR      $ATYC
024470 112737 000001 024716 $ATY4: MOVB   #1,$FFLG      ;;TO ONLY REPORT FATAL ERROR
024476          $ATYC:
024476 010046          MOV    R0,-(SP)      ;;PUSH R0 ON STACK
024500 010146          MOV    R1,-(SP)      ;;PUSH R1 ON STACK
024502 105737 024714          TSTB  $MFLG         ;;SHOULD TYPE A MESSAGE?
024506 001450          BEQ    5$           ;;IF NOT: BR
024510 122737 000001 001232 CMPB   #APTENV,$ENV     ;;OPERATING UNDER APT?
024516 001031          BNE     3$           ;;IF NOT: BR
024520 132737 000100 001233 BITB   #APTSPOOL,$ENVM  ;;SHOULD SPOOL MESSAGES?
024526 001425          BEQ    3$           ;;IF NOT: BR
024530 017600 000004          MOV    @4(SP),R0     ;;GET MESSAGE ADDR.
024534 062766 000002 000004 ADD    #2,4(SP)       ;;BUMP RETURN ADDR.
024542 005737 001212 1$:    TST    $MSGTYPE     ;;SEE IF DONE W/ LAST XMISSION?
024546 001375          BNE     1$           ;;IF NOT: WAIT
024550 010037 001226          MOV    R0,$MSGAD     ;;PUT ADDR IN MAILBOX
024554 105720          2$:    TSTB  (R0)+      ;;FIND END OF MESSAGE
024556 001376          BNE     2$
024560 163700 001226          SUB    $MSGAD,R0     ;;SUB START OF MESSAGE
024564 006200          ASR    R0           ;;GET MESSAGE LNTH IN WORDS
024566 010037 001230          MOV    R0,$MSGGLT    ;;PUT LENGTH IN MAILBOX
024572 012737 000004 001212 MOV    #4,$MSGTYPE    ;;TELL APT TO TAKE MSG.
024600 000413          BR      5$
024602 017637 000004 024626 3$:    MOV    @4(SP),4$     ;;PUT MSG ADDR IN JSR LINKAGE
024610 062766 000002 000004 ADD    #2,4(SP)       ;;BUMP RETURN ADDRESS
024616 013746 177776          MOV    177776,-(SP)  ;;PUSH 177776 ON STACK
024622 004737 024116          JSR   PC,$TYPE      ;;CALL TYPE MACRO
024626 000000          4$:    .WORD   0
024630          5$:
024630 105737 024716          10$:   TSTB  $FFLG         ;;SHOULD REPORT FATAL ERROR?
024634 001416          BEQ    12$         ;;IF NOT: BR
024636 005737 001232          TST   $ENV         ;;RUNNING UNDER APT?
024642 001413          BEQ    12$         ;;IF NOT: BR
024644 005737 001212 11$:   TST   $MSGTYPE     ;;FINISHED LAST MESSAGE?
024650 001375          BNE     11$        ;;IF NOT: WAIT
024652 017637 000004 001214 MOV    @4(SP),$FATAL  ;;GET ERROR #
024660 062766 000002 000004 ADD    #2,4(SP)       ;;BUMP RETURN ADDR.
024666 005237 001212          INC   $MSGTYPE     ;;TELL APT TO TAKE ERROR
024672 105037 024716          12$:   CLRB  $FFLG         ;;CLEAR FATAL FLAG
024676 105037 024715          CLRB  $LFLG        ;;CLEAR LOG FLAG
024702 105037 024714          CLRB  $MFLG        ;;CLEAR MESSAGE FLAG
    
```

```
024706 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
024710 012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
024712 000207      RTS      PC           ;;RETURN
024714      000      $MFLG: .BYTE      0           ;;MESSG. FLAG
024715      000      $LFLG: .BYTE      0           ;;LOG FLAG
024716      000      $FFLG: .BYTE      0           ;;FATAL FLAG
                                .EVEN
                                APTSIZE = 200
                                APTENV  = 001
                                APTSPOOL= 100
                                APTCSUP = 040
```

2548
2549

.SBTTL POWER DOWN AND UP ROUTINES

```
*****
:POWER DOWN ROUTINE
024720 012737 025072 000024 $PWRDN: MOV      #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
024726 012737 000340 000026      MOV      #340,@#PWRVEC+2 ;;PRIO:7
024734 010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
024736 010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
024740 010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
024742 010346      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
024744 010446      MOV      R4,-(SP)      ;;PUSH R4 ON STACK
024746 010546      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
024750 017746 154200      MOV      @SWR,-(SP)     ;;PUSH @SWR ON STACK
024754 010637 025076      MOV      SP,$SAVR6     ;;SAVE SP
024760 012737 024772 000024      MOV      #$PWRUP,@#PWRVEC ;;SET UP VECTOR
024766 000000      HALT
024770 000776      BR      -2           ;;HANG UP
```

```
*****
:POWER UP ROUTINE
024772 012737 025072 000024 $PWRUP: MOV      #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
025000 013706 025076      MOV      $SAVR6,SP     ;;GET SP
025004 005037 025076      CLR      $SAVR6       ;;WAIT LOOP FOR THE TTY
025010 005237 025076 1$: INC      $SAVR6       ;;WAIT FOR THE INC
025014 001375      BNE     1$           ;;OF WORD
025016 012677 154132      MOV      (SP)+,@SWR    ;;POP STACK INTO @SWR
025022 012605      MOV      (SP)+,R5     ;;POP STACK INTO R5
025024 012604      MOV      (SP)+,R4     ;;POP STACK INTO R4
025026 012603      MOV      (SP)+,R3     ;;POP STACK INTO R3
025030 012602      MOV      (SP)+,R2     ;;POP STACK INTO R2
025032 012601      MOV      (SP)+,R1     ;;POP STACK INTO R1
025034 012600      MOV      (SP)+,R0     ;;POP STACK INTO R0
025036 012737 024720 000024      MOV      #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
025044 012737 000340 000026      MOV      #340,@#PWRVEC+2 ;;PRIO:7
025052 104401      TYPE
025054 025100      $PWRMG: .WORD      $POWER ;;REPORT THE POWER FAILURE
025056 042766 000020 000002      BIC      #20,2(SP)    ;;POWER FAIL MESSAGE POINTER
025064 005037 023560      CLR      $TBIT       ;;CLEAR 'T' BIT
025070 000002      RTI                ;;CLEAR THE 'T' BIT FLAG
025072 000000      $ILLUP: HALT
025074 000776      BR      -2           ;;THE POWER UP SEQUENCE WAS STARTED
025076 000000      $SAVR6: 0           ;;BEFORE THE POWER DOWN WAS COMPLETE
025100      015      012      120 $POWER: .ASCIZ <15><12>'POWER'
                                .EVEN
```

2550

2551

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

025110	017646	000000		\$TYPOS:	MOV	@(SP),-(SP)	;;PICKUP THE MODE
025114	116637	000001	025333		MOVB	1(SP),\$OFILL	;;LOAD ZERO FILL SWITCH
025122	112637	025335			MOVB	(SP)+,\$OMODE+1	;;NUMBER OF DIGITS TO TYPE
025126	062716	000002			ADD	#2,(SP)	;;ADJUST RETURN ADDRESS
025132	000406				BR	\$TYPON	
025134	112737	000001	025333	\$TYPOC:	MOVB	#1,\$OFILL	;;SET THE ZERO FILL SWITCH
025142	112737	000006	025335		MOVB	#6,\$OMODE+1	;;SET FOR SIX(6) DIGITS
025150	112737	000005	025332	\$TYPON:	MOVB	#5,\$OCNT	;;SET THE ITERATION COUNT
025156	010346				MOV	R3,-(SP)	;;SAVE R3
025160	010446				MOV	R4,-(SP)	;;SAVE R4
025162	010546				MOV	R5,-(SP)	;;SAVE R5
025164	113704	025335			MOVB	\$OMODE+1,R4	;;GET THE NUMBER OF DIGITS TO TYPE
025170	005404				NEG	R4	
025172	062704	000006			ADD	#6,R4	;;SUBTRACT IT FOR MAX. ALLOWED
025176	110437	025334			MOVB	R4,\$OMODE	;;SAVE IT FOR USE
025202	113704	025333			MOVB	\$OFILL,R4	;;GET THE ZERO FILL SWITCH
025206	016605	000012			MOV	12(SP),R5	;;PICKUP THE INPUT NUMBER
025212	005003				CLR	R3	;;CLEAR THE OUTPUT WORD
025214	006105			1\$:	ROL	R5	;;ROTATE MSB INTO 'C'
025216	000404				BR	3\$;;GO DO MSB
025220	006105			2\$:	ROL	R5	;;FORM THIS DIGIT
025222	006105				ROL	R5	
025224	006105				ROL	R5	
025226	010503				MOV	R5,R3	
025230	006103			3\$:	ROL	R3	;;GET LSB OF THIS DIGIT
025232	105337	025334			DECB	\$OMODE	;;TYPE THIS DIGIT?
025236	100016				BPL	7\$;;BR IF NO
025240	042703	177770			BIC	#177770,R3	;;GET RID OF JUNK
025244	001002				BNE	4\$;;TEST FOR 0
025246	005704				TST	R4	;;SUPPRESS THIS 0?
025250	001403				BEQ	5\$;;BR IF YES
025252	005204			4\$:	INC	R4	;;DON'T SUPPRESS ANYMORE 0'S

```

025254 052703 000060
025260 052703 000040
025264 110337 025330
025270 104401 025330
025274 105337 025332
025300 003347
025302 002402
025304 005204
025306 000744
025310 012605
025312 012604
025314 012603
025316 016666 000002 000004
025324 012616
025326 000002
025330 000
025331 000
025332 000
025333 000
025334 000000

```

```

5$: BIS #'0,R3      ;;MAKE THIS DIGIT ASCII
    BIS #' ,R3     ;;MAKE ASCII IF NOT ALREADY
    MOVB R3,8$     ;;SAVE FOR TYPING
    TYPE 8$        ;;GO TYPE THIS DIGIT
7$: DECB $OCNT    ;;COUNT BY 1
    BGT 2$         ;;BR IF MORE TO DO
    BLT 6$         ;;BR IF DONE
    INC R4         ;;INSURE LAST DIGIT ISN'T A BLANK
    BR 2$          ;;GO DO THE LAST DIGIT
6$: MOV (SP)+,R5  ;;RESTORE R5
    MOV (SP)+,R4  ;;RESTORE R4
    MOV (SP)+,R3  ;;RESTORE R3
    MOV 2(SP),4(SP) ;;SET THE STACK FOR RETURNING
    MOV (SP)+,(SP)
    RTI           ;;RETURN
8$: .BYTE 0       ;;STORAGE FOR ASCII DIGIT
    .BYTE 0       ;;TERMINATOR FOR TYPE ROUTINE
SOCNT: .BYTE 0    ;;OCTAL DIGIT COUNTER
$OFILL: .BYTE 0   ;;ZERO FILL SWITCH
$OMODE: .WORD 0   ;;NUMBER OF DIGITS TO TYPE

```

2552
2553

.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

```

```

025336
025336 010046
025340 010146
025342 010246
025344 010346
025346 010546
025350 012746 020200
025354 016605 000020
025360 100004
025362 005405
025364 112766 000055 000001
025372 005000
025374 012703 025552
025400 112723 000040
025404 005002
025406 016001 025542
025412 160105
025414 002402
025416 005202
025420 000774
025422 060105
025424 005702
025426 001002
025430 105716

```

```

$TYPDS:
MOV R0,-(SP) ;;PUSH R0 ON STACK
MOV R1,-(SP) ;;PUSH R1 ON STACK
MOV R2,-(SP) ;;PUSH R2 ON STACK
MOV R3,-(SP) ;;PUSH R3 ON STACK
MOV R5,-(SP) ;;PUSH R5 ON STACK
MOV #20200,-(SP) ;;SET BLANK SWITCH AND SIGN
MOV 20(SP),R5 ;;GET THE INPUT NUMBER
BPL 1$ ;;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 # $DBLK,R3 ;;SETUP THE OUTPUT POINTER
    MOVB #' ,(R3)+ ;;SET THE FIRST CHARACTER TO A BLANK
2$: 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
    BR 3$
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?

```

```

025432 100407          BMI      7$          ;;BR IF YES
025434 106316          5$:  ASLB    (SP)          ;;MSD?
025436 103003          BCC     6$          ;;BR IF NO
025440 116663 000001 177777  MOVB   1(SP),-1(R3)  ;;YES--SET THE SIGN
025446 052702 000060          BIS    #'0,R2        ;;MAKE THE BCD DIGIT ASCII
025452 052702 000040          6$:  BIS    #' ,R2        ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
025456 110223          7$:  MOVB   R2,(R3)+      ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
025460 005720          TST    (R0)+          ;;JUST INCREMENTING
025462 020027 000010          CMP    R0,#10        ;;CHECK THE TABLE INDEX
025466 002746          BLT    2$          ;;GO DO THE NEXT DIGIT
025470 003002          BGT    8$          ;;GO TO EXIT
025472 010502          MOV    R5,R2        ;;GET THE LSD
025474 000764          BR     6$          ;;GO CHANGE TO ASCII
025476 105726          8$:  TSTB   (SP)+      ;;WAS THE LSD THE FIRST NON-ZERO?
025500 100003          BPL    9$          ;;BR IF NO
025502 116663 177777 177776  MOVB   -1(SP),-2(R3)  ;;YES--SET THE SIGN FOR TYPING
025510 105013          9$:  CLRB   (R3)          ;;SET THE TERMINATOR
025512 012605          MOV    (SP)+,R5     ;;POP STACK INTO R5
025514 012603          MOV    (SP)+,R3     ;;POP STACK INTO R3
025516 012602          MOV    (SP)+,R2     ;;POP STACK INTO R2
025520 012601          MOV    (SP)+,R1     ;;POP STACK INTO R1
025522 012600          MOV    (SP)+,R0     ;;POP STACK INTO R0
025524 104401 025552          TYPE   $DBLK        ;;NOW TYPE THE NUMBER
025530 016666 000002 000004  MOV    2(SP),4(SP)   ;;ADJUST THE STACK
025536 012616          MOV    (SP)+,(SP)
025540 000002          RTI
025542 023420          $DTBL: 10000.        ;;RETURN TO USER
025544 001750          1000.
025546 000144          100.
025550 000012          10.
025552          $DBLK: .BLKW 4
    
```

2554
2555

.SBTTL TTY INPUT ROUTINE

 .ENABL LSB

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

```

025562 022737 000176 001154 $CKSWR: CMP    #SWREG,SWR  ;;IS THE SOFT-SWR SELECTED?
025570 001074          BNE    15$          ;;BRANCH IF NO
025572 105777 153362          TSTB   @$TKS        ;;CHAR THERE?
025576 100071          BPL    15$          ;;IF NO, DON'T WAIT AROUND
025600 117746 153356          MOVB   @$TKB,-(SP)  ;;SAVE THE CHAR
025604 042716 177600          BIC    #^C177,(SP)  ;;STRIP-OFF THE ASCII
025610 022726 000007          CMP    #7,(SP)+     ;;IS IT A CONTROL G?
025614 001062          BNE    15$          ;;NO, RETURN TO USER
025616 123727 001150 000001  CMPB   $AUTOB,#1    ;;ARE WE RUNNING IN AUTO-MODE?
025624 001456          BEQ    15$          ;;BRANCH IF YES

025626 104401 026171          TYPE   ,$CNTLG      ;;ECHO THE CONTROL-G (^G)
025632 104401 026176          $GTSWR: TYPE  $MSWR   ;;TYPE CURRENT CONTENTS
025636 013746 000176          MOV    SWREG,-(SP)  ;;SAVE SWREG FOR TYPEOUT
025642 104402          TYPOC                ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
    
```

```

025644 104401 026207          TYPE      , $MNEW      :: PROMPT FOR NEW SWR
025650 005046          CLR        -(SP)      :: CLEAR COUNTER
025652 005046          CLR        -(SP)      :: THE NEW SWR
025654 105777 153300      7$:      TSTB      @$TKS      :: CHAR THERE?
025660 100375          BPL        7$      :: IF NOT TRY AGAIN

025662 117746 153274          MOVB     @$TKB, -(SP)  :: PICK UP CHAR
025666 042716 177600          BIC      #^C177, (SP)  :: MAKE IT 7-BIT ASCII

025672 021627 000025      9$:      CMP      (SP), #25      :: IS IT A CONTROL-U?
025676 001005          BNE      10$      :: BRANCH IF NOT
025700 104401 026164          TYPE     , $CNTLU      :: YES, ECHO CONTROL-U (^U)
025704 062706 000006      20$:     ADD      #6, SP      :: IGNORE PREVIOUS INPUT
025710 000757          BR       19$      :: LET'S TRY IT AGAIN

025712 021627 000015      10$:     CMP      (SP), #15      :: IS IT A <CR>?
025716 001022          BNE      16$      :: BRANCH IF NO
025720 005766 000004          TST      4(SP)      :: YES, IS IT THE FIRST CHAR?
025724 001403          BEQ      11$      :: BRANCH IF YES
025726 016677 000002 153220      MOV      2(SP), @SWR      :: SAVE NEW SWR
025734 062706 000006      11$:     ADD      #6, SP      :: CLEAR UP STACK
025740 104401 001207      14$:     TYPE     , $CRLF      :: ECHO <CR> AND <LF>
025744 123727 001151 000001      CMPB     $INTAG, #1      :: RE-ENABLE TTY KBD INTERRUPTS?
025752 001003          BNE      15$      :: BRANCH IF NOT
025754 012777 000100 153176      MOV      #100, @$TKS      :: RE-ENABLE TTY KBD INTERRUPTS
025762 000002          RTI      :: RETURN
025764 004737 024330      16$:     JSR      PC, $TYPEC      :: ECHO CHAR
025770 021627 000060          CMP      (SP), #60      :: CHAR < 0?
025774 002420          BLT      18$      :: BRANCH IF YES
025776 021627 000067          CMP      (SP), #67      :: CHAR > 7?
026002 003015          BGT      18$      :: BRANCH IF YES
026004 042726 000060          BIC      #60, (SP)+      :: STRIP-OFF ASCII
026010 005766 000002          TST      2(SP)      :: IS THIS THE FIRST CHAR
026014 001403          BEQ      17$      :: BRANCH IF YES
026016 006316          ASL      (SP)      :: NO, SHIFT PRESENT
026020 006316          ASL      (SP)      :: CHAR OVER TO MAKE
026022 006316          ASL      (SP)      :: ROOM FOR NEW ONE.
026024 005266 000002      17$:     INC      2(SP)      :: KEEP COUNT OF CHAR
026030 056616 177776          BIS      -2(SP), (SP)  :: SET IN NEW CHAR
026034 000707          BR       7$      :: GET THE NEXT ONE
026036 104401 001206      18$:     TYPE     , $QUES      :: TYPE ?<CR><LF>
026042 000720          BR       20$      :: SIMULATE CONTROL-U
.DSABL  LSB

```

```

::*****
::THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
::CALL:
::*   RDCHR      :: INPUT A SINGLE CHARACTER FROM THE TTY
::*   RETURN HERE :: CHARACTER IS ON THE STACK
::*           :: WITH PARITY BIT STRIPPED OFF
::

```

```

026044 011646      $RDCHR: MOV      (SP), -(SP)      :: PUSH DOWN THE PC

```

```

026046 016666 000004 000002      MOV      4(SP),2(SP)      ;;SAVE THE PS
026054 105777 153100      1$:      TSTB     @$TKS          ;;WAIT FOR
026060 100375          BPL      1$              ;;A CHARACTER
026062 117766 153074 000004      MOVB     @$TKB,4(SP)     ;;READ THE TTY
026070 042766 177600 000004      BIC      #^C<177>,4(SP) ;;GET RID OF JUNK IF ANY
026076 026627 000004 000023      CMP      4(SP),#23      ;;IS IT A CONTROL-S?
026104 001013          BNE      3$              ;;BRANCH IF NO
026106 105777 153046      2$:      TSTB     @$TKS          ;;WAIT FOR A CHARACTER
026112 100375          BPL      2$              ;;LOOP UNTIL ITS THERE
026114 117746 153042      MOVB     @$TKB,-(SP)     ;;GET CHARACTER
026120 042716 177600          BIC      #^C177,(SP)    ;;MAKE IT 7-BIT ASCII
026124 022627 000021      CMP      (SP)+,#21      ;;IS IT A CONTROL-Q?
026130 001366          BNE      2$              ;;IF NOT DISCARD IT
026132 000750          BR       1$              ;;YES, RESUME
026134 026627 000004 000140      3$:      CMP      4(SP),#140     ;;IS IT UPPER CASE?
026142 002407          BLT      4$              ;;BRANCH IF YES
026144 026627 000004 000175      CMP      4(SP),#175     ;;IS IT A SPECIAL CHAR?
026152 003003          BGT      4$              ;;BRANCH IF YES
026154 042766 000040 000004      BIC      #40,4(SP)      ;;MAKE IT UPPER CASE
026162 000002          RTI                      ;;GO BACK TO USER
026164      136      125      015      $CNTLU: .ASCIZ /^U/<15><12> ;;CONTROL 'U'
026171      136      107      015      $CNTLG: .ASCIZ /^G/<15><12> ;;CONTROL 'G'
026176      015      012      123      $MSWR:  .ASCIZ <15><12>/SWR = /
026207      040      040      116      $MNEW:  .ASCIZ / NEW = /
    
```

2556
2557

.SBTTL SAVE AND RESTORE R0-R5 ROUTINES

```

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

```

026220      $SAVREG:
026220 010046      MOV      R0,-(SP)        ;;PUSH R0 ON STACK
026222 010146      MOV      R1,-(SP)        ;;PUSH R1 ON STACK
026224 010246      MOV      R2,-(SP)        ;;PUSH R2 ON STACK
026226 010346      MOV      R3,-(SP)        ;;PUSH R3 ON STACK
026230 010446      MOV      R4,-(SP)        ;;PUSH R4 ON STACK
026232 010546      MOV      R5,-(SP)        ;;PUSH R5 ON STACK
026234 016646 000022      MOV      22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
026240 016646 000022      MOV      22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
026244 016646 000022      MOV      22(SP),-(SP)    ;;SAVE PS OF CALL
026250 016646 000022      MOV      22(SP),-(SP)    ;;SAVE PC OF CALL
026254 000002      RTI
    
```

```

;*RESTORE R0-R5
;*CALL:
    
```

```

    026256          RESREG
    026256 012666 000022 $RESREG: MOV (SP)+,22(SP)  ;;RESTORE PC OF CALL
    026262 012666 000022          MOV (SP)+,22(SP)  ;;RESTORE PS OF CALL
    026266 012666 000022          MOV (SP)+,22(SP)  ;;RESTORE PC OF MAIN FLOW
    026272 012666 000022          MOV (SP)+,22(SP)  ;;RESTORE PS OF MAIN FLOW
    026276 012605          MOV (SP)+,R5           ;;POP STACK INTO R5
    026300 012604          MOV (SP)+,R4           ;;POP STACK INTO R4
    026302 012603          MOV (SP)+,R3           ;;POP STACK INTO R3
    026304 012602          MOV (SP)+,R2           ;;POP STACK INTO R2
    026306 012601          MOV (SP)+,R1           ;;POP STACK INTO R1
    026310 012600          MOV (SP)+,R0           ;;POP STACK INTO R0
    026312 000002          RTI
    
```

2558
 2559

.SBTTL DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

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

```

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

```

    026314 104412          $DB2D: SAVREG           ;;SAVE REGISTERS
    026316 016602 000002   MOV    2(SP),R2       ;;PICKUP THE DATA POINTER
    026322 012700 026474   MOV    #$DECVL,R0      ;;GET ADDRESS OF '$DECVL' STRING
    026326 010066 000002   MOV    R0,2(SP)       ;;PUT ADDRESS OF ASCII STRING ON STACK
    026332 012201          MOV    (R2)+,R1        ;;PICKUP THE BINARY NUMBER
    026334 012202          MOV    (R2)+,R2
    026336 012737 000012 026412 MOV    #10.,4$        ;;SET UP TO DO 10 CONVERSIONS
    026344 012704 026424   MOV    #$TNPWR,R4      ;;ADDRESS OF TEN POWER
    026350 012705 026426   MOV    #$TNPWR+2,R5
    026354 005003          1$: CLR    R3           ;;CLEAR PARTIAL
    026356 161401          2$: SUB    (R4),R1        ;;SUBTRACT TEN POWER
    026360 005602          SBC    R2
    026362 161502          SUB    (R5),R2
    026364 002402          BLT    3$           ;;BR IF TEN POWER TO LARGE
    026366 005203          INC    R3           ;;ADD 1 TO PARTIAL
    026370 000772          BR     2$           ;;LOOP
    026372 062401          3$: ADD    (R4)+,R1      ;;RESTORE SUBTRACTED VALUE
    026374 005502          ADC    R2
    026376 062402          ADD    (R4)+,R2
    026400 022525          CMP    (R5)+,(R5)+
    026402 052703 000060   BIS    #'0,R3         ;;MOVE TO NEXT TEN POWER
    026406 110320          MOVB   R3,(R0)+      ;;CHANGE PARTIAL TO ASCII
    026410 005327          DEC    (PC)+       ;;SAVE IT
    026412 000000          4$: .WORD 0       ;;DONE?
    026414 001357          BNE    1$           ;;BR IF NO
    026416 105020          CLRB  (R0)+       ;;TERMINATOR
    026420 104413          RESREG
    026422 000207          RTS    PC         ;;RESTORE REGISTERS
    026424 145000          $TNPWR: 145000    ;;RETURN
    026426 035632          35632         ;;1.0E09
    
```

026430	160400	160400	:::1.0E08
026432	002765	2765	
026434	113200	113200	:::1.0E07
026436	000230	230	
026440	041100	041100	:::1.0E06
026442	000017	17	
026444	103240	103240	:::1.0E05
026446	000001	1	
026450	023420	23420	:::1.0E04
026452	000000	0	
026454	001750	1750	:::1.0E03
026456	000000	0	
026460	000144	144	:::1.0E02
026462	000000	0	
026464	000012	12	:::1.0E01
026466	000000	0	
026470	000001	1	:::1.0E00
026472	000000	0	
026474			

2560
2561

\$DECVL: .BLKB 12. ::RESERVE STORAGE FOR ASCIZ STRING

.SBTTL DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE

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

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

026510	104412		\$DB20: SAVREG	:::SAVE ALL REGISTERS
026512	016601	000002	MOV 2(SP),R1	:::PICKUP THE POINTER TO LOW WORD
026516	012705	026627	MOV #\$OCTVL+13.,R5	:::POINTER TO DATA TABLE
026522	012704	000014	MOV #12.,R4	:::DO ELEVEN CHARACTERS
026526	012703	177770	MOV #^C7,R3	:::MASK
026532	012100		MOV (R1)+,R0	:::LOWER WORD
026534	012101		MOV (R1)+,R1	:::HIGH WORD
026536	005002		CLR R2	:::TERMINATOR
026540	110245		1\$: MOV B R2,-(R5)	:::PUT CHARACTER IN DATA TABLE
026542	010002		MOV R0,R2	:::GET THIS DIGIT
026544	005304		DEC R4	:::COUNT THIS CHARACTER
026546	003007		BGT 3\$:::BR IF NOT THE LAST DIGIT
026550	001405		BEQ 2\$:::BR IF IT IS THE LAST DIGIT
026552	005205		INC R5	:::ALL DIGITS DONE-ADJUST POINTER FOR FIRST
026554	010566	000002	MOV R5,2(SP)	:::ASCIZ CHAR. & PUT IT ON THE STACK
026560	104413		RESREG	:::RESTORE ALL REGISTERS
026562	000207		RTS PC	:::RETURN TO USER
026564	006203		2\$: ASR R3	:::POSITION THE MASK FOR THE LAST DIGIT
026566	006001		3\$: ROR R1	:::POSITION THE BINARY NUMBER FOR
026570	006000		ROR R0	::: THE NEXT OCTAL DIGIT
026572	006001		ROR R1	
026574	006000		ROR R0	
026576	006001		ROR R1	
026600	006000		ROR R0	
026602	040302		BIC R3,R2	:::MASK OUT ALL JUNK
026604	062702	000060	ADD #'0,R2	:::MAKE THIS CHAR. ASCII

026610 000753
 026612
 2562
 2563

BR 1\$::GO PUT IT IN THE DATA TABLE
 \$OCTLV: .BLKB 14. ::RESERVE DATA TABLE

.SBTTL SCOPE HANDLER ROUTINE

 *THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
 *AND LOAD THE TEST NUMBER(\$TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
 *AND LOAD THE ERROR FLAG (\$ERFLG) INTO DISPLAY<15:08>
 *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
 *SW14=1 LOOP ON TEST
 *SW11=1 INHIBIT ITERATIONS
 *SW09=1 LOOP ON ERROR
 *SW08=1 LOOP ON TEST IN SWR<7:0>
 *CALL
 * SCOPE ;;SCOPE=IOT

026630				\$SCOPE:			
026630	104407				CKSWR		::TEST FOR CHANGE IN SOFT-SWR
026632	032777	040000	152314	1\$:	BIT #BIT14,@SWR		::LOOP ON PRESENT TEST?
026640	001114				BNE \$OVER		::YES IF SW14=1
					#####START OF CODE FOR THE XOR TESTER#####		
026642	000416			\$XTSTR:	BR 6\$::IF RUNNING ON THE 'XOR' TESTER CHANGE
							::THIS INSTRUCTION TO A 'NOP' (NOP=240)
026644	013746	000004			MOV @#ERRVEC,-(SP)		::SAVE THE CONTENTS OF THE ERROR VECTOR
026650	012737	026670	000004		MOV #5\$,@#ERRVEC		::SET FOR TIMEOUT
026656	005737	177060			TST @#177060		::TIME OUT ON XOR?
026662	012637	000004			MOV (SP)+,@#ERRVEC		::RESTORE THE ERROR VECTOR
026666	000463				BR \$SVLAD		::GO TO THE NEXT TEST
026670	022626			5\$:	CMP (SP)+,(SP)+		::CLEAR THE STACK AFTER A TIME OUT
026672	012637	000004			MOV (SP)+,@#ERRVEC		::RESTORE THE ERROR VECTOR
026676	000423				BR 7\$::LOOP ON THE PRESENT TEST
026700				6\$:	#####END OF CODE FOR THE XOR TESTER#####		
026700	032777	000400	152246		BIT #BIT08,@SWR		::LOOP ON SPEC. TEST?
026706	001404				BEQ 2\$::BR IF NO
026710	127737	152240	001116		CMPB @SWR,\$TSTNM		::ON THE RIGHT TEST? SWR<7:0>
026716	001465				BEQ \$OVER		::BR IF YES
026720	105737	001117		2\$:	TSTB \$ERFLG		::HAS AN ERROR OCCURRED?
026724	001421				BEQ 3\$::BR IF NO
026726	123737	001131	001117		CMPB \$ERMAX,\$ERFLG		::MAX. ERRORS FOR THIS TEST OCCURRED?
026734	101015				BHI 3\$::BR IF NO
026736	032777	001000	152210		BIT #BIT09,@SWR		::LOOP ON ERROR?
026744	001404				BEQ 4\$::BR IF NO
026746	013737	001124	001122	7\$:	MOV \$LPERR,\$LPADR		::SET LOOP ADDRESS TO LAST SCOPE
026754	000446				BR \$OVER		
026756	105037	001117		4\$:	CLRB \$ERFLG		::ZERO THE ERROR FLAG
026762	005037	001176			CLR \$TIMES		::CLEAR THE NUMBER OF ITERATIONS TO MAKE
026766	000415				BR 1\$::ESCAPE TO THE NEXT TEST
026770	032777	004000	152156	3\$:	BIT #BIT11,@SWR		::INHIBIT ITERATIONS?
026776	001011				BNE 1\$::BR IF YES
027000	005737	001220			TST \$PASS		::IF FIRST PASS OF PROGRAM
027004	001406				BEQ 1\$:: INHIBIT ITERATIONS
027006	005237	001120			INC \$ICNT		::INCREMENT ITERATION COUNT
027012	023737	001176	001120		CMP \$TIMES,\$ICNT		::CHECK THE NUMBER OF ITERATIONS MADE
027020	002024				BGE \$OVER		::BR IF MORE ITERATION REQUIRED
027022	012737	000001	001120	1\$:	MOV #1,\$ICNT		::REINITIALIZE THE ITERATION COUNTER
027030	013737	027106	001176		MOV \$MXCNT,\$TIMES		::SET NUMBER OF ITERATIONS TO DO


```
027036 105237 001116    $SVLAD: INCB  $STNM          ;;COUNT TEST NUMBERS
027042 113737 001116 001216    MOV  $STNM,$TESTN  ;;SET TEST NUMBER IN APT MAILBOX
027050 011637 001122    MOV  (SP),$LPADR   ;;SAVE SCOPE LOOP ADDRESS
027054 011637 001124    MOV  (SP),$LPERR   ;;SAVE ERROR LOOP ADDRESS
027060 005037 001200    CLR  $ESCAPE       ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
027064 112737 000001 001131    MOV  #1,$ERMAX     ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
027072 013777 001116 152056 $OVER: MOV  $STNM,@DISPLAY ;;DISPLAY TEST NUMBER
027100 013716 001122    MOV  $LPADR,(SP)   ;;FUDGE RETURN ADDRESS
027104 000002    RTI                ;;FIXES PS
027106 003720    $MXCNT: 2000.     ;;MAX. NUMBER OF ITERATIONS
```

2564
2565

.SBTTL ROUTINE TO SIZE MEMORY

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

```
027110 010046    $SIZE: MOV  R0,-(SP)      ;;SAVE R0 ON THE STACK
027112 010146    MOV  R1,-(SP)      ;;SAVE R1 ON THE STACK
027114 013746 000114    MOV  @#114,-(SP)   ;;SAVE MEMORY ERROR VECTOR PS & PC
027120 013746 000116    MOV  @#116,-(SP)
027124 012737 000116 000114    MOV  #116,@#114    ;;IGNORE PARITY ERRORS WHILE SIZING
027132 012737 000002 000116    MOV  #RTI,@#116
027140 013746 000004    MOV  @#ERRVEC,-(SP) ;;SAVE PRESENT ERROR VECTOR PS & PC
027144 013746 000006    MOV  @#ERRVEC+2,-(SP)
027150 010600    MOV  SP,R0        ;;SAVE THE STACK POINTER
;;SET THE ERRVEC PS TO THE PRESENT PS
027152 104400    TRAP
027154 012637 000006    MOV  (SP)+,@#ERRVEC+2 ;;PUSH OLD PSW AND PC ON STACK
027160 012737 027200 000004    MOV  #2$,@#ERRVEC   ;;SAVE THE PSW IN @#ERRVEC+2
027166 012701 020000    MOV  #20000,R1     ;;SET FOR TIMEOUT
027172 005711    1$:  TST  (R1)        ;;FIRST ADDRESS
027174 005721    TST  (R1)+         ;;TEST THIS ADDRESS
027176 000775    BR   1$           ;;STEP TO NEXT ADDRESS
027200 162701 000002    2$:  SUB  #2,R1      ;;TRY ANOTHER
027204 010006    MOV  R0,SP        ;;DROP BACK
027206 012637 000006    MOV  (SP)+,@#ERRVEC+2 ;;RESTORE THE STACK
027212 012637 000004    MOV  (SP)+,@#ERRVEC ;;RESTORE ERROR VECTOR
027216 012637 000116    MOV  (SP)+,@#116   ;;RESTORE MEMORY ERROR VECTOR
027222 012637 000114    MOV  (SP)+,@#114
027226 010137 027240    MOV  R1,$LSTAD    ;;RESTORE MEMORY ERROR VECTOR
027232 012601    MOV  (SP)+,R1     ;;LAST ADDRESS
027234 012600    MOV  (SP)+,R0     ;;RESTORE R1
027236 000207    RTS  PC          ;;RESTORE R0
027240 000000    $LSTAD: .WORD 0   ;;CONTAINS THE LAST ADDRESS
```

2566
2567

.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.
```

```
027242 010046    $TRAP: MOV  R0,-(SP) ;;SAVE R0
```

```

027244 016600 000002      MOV      2(SP),R0      ;;GET TRAP ADDRESS
027250 005740             TST      -(R0)        ;;BACKUP BY 2
027252 111000             MOV      (R0),R0      ;;GET RIGHT BYTE OF TRAP
027254 006300             ASL      R0           ;;POSITION FOR INDEXING
027256 016000 027276      MOV      $TRPAD(R0),R0 ;;INDEX TO TABLE
027262 000200             RTS      R0          ;;GO TO ROUTINE

```

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

```

027264 011646             $TRAP2: MOV      (SP),-(SP) ;;MOVE THE PC DOWN
027266 016666 000004 000002  MOV      4(SP),2(SP) ;;MOVE THE PSW DOWN
027274 000002             RTI                    ;;RESTORE THE PSW

```

.SBTTL TRAP TABLE

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

```

:          ROUTINE
:          -----
027276 027264             $TRPAD: .WORD      $TRAP2
027300 024116             $TYPE  ;;CALL=TYPE      TRAP+1(104401) TTY TYPEOUT ROUTINE
027302 025134             $TYPOC ;;CALL=TYPOC     TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
027304 025110             $TYPOS ;;CALL=TYPOS     TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
027306 025150             $TYPON ;;CALL=TYPON      TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
027310 025336             $TYPDS ;;CALL=TYPDS     TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)

027312 025632             $GTSWR ;;CALL=GTSWR     TRAP+6(104406) GET SOFT-SWR SETTING

027314 025562             $CKSWR ;;CALL=CKSWR     TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
027316 026044             $RDCHR ;;CALL=RDCHR     TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
027320 023050             $RDLIN ;;CALL=RDLIN     TRAP+11(104411) TTY TYPEIN STRING ROUTINE
027322 026220             $$SAVREG ;;CALL=SAVREG    TRAP+12(104412) SAVE R0-R5 ROUTINE
027324 026256             $RESREG ;;CALL=RESREG    TRAP+13(104413) RESTORE R0-R5 ROUTINE
2568 027326 022266             $DSPLY ;;CALL=DISPLY   TRAP+14(104414) ROUTINE TO TYPE ERROR MESSAGES
2569 000032
2570
2577
$TERM=-.$TRPAD

```

```
1      .SBTTL SINGLE/DUAL PORT RH/RM DRIVER (REV 6.2) FEBRUARY 1980
2
3      ;NEW DRIVE TYPE ID FOR RM02 *****
4      ;10-AUG-77 *****
5      ;10-MAR-78 THE SC, SC5 CHANGES
6      ;NEW DRIVE TYPE ID FOR RM05 *****
7      ;FEBRUARY 1980 *****
8
9      ;COPYRIGHT (C) 1977
10     ;DIGITAL EQUIPMENT CORP.
11     ;MAYNARD, MA 01754
12     ;AUTHOR(S): JIM LACEY/CHUCK HESS/
13
14     ;*****
15
16     ;STORAGE FOR RMD5, RMER1, RMER2, AND RMMR2 ON AN ERROR '2'
17     ;RMERRS = RMD5
18     ;RMERRS+2 = RMER1
19     ;RMERRS+4 = RMER2
20     ;RMERRS+6 = RMMR2
21
22 027330 000000 000000 000000 RMERRS: .WORD 0,0,0,0
23
24     ;TABLE OF DRIVE ACTIVE INDICATORS (DRVACT=8 BYTES)
25     ;DRVACT=0 IF DRIVE IS IDLE
26     ;DRVACT>0 IF DRIVE IS ACTIVE WITH A COMMAND
27     ;DRVACT<0 IF DRIVE IS ACTIVE WITH AN ERROR RECOVERY OPERATION
28
29 027340      000      DRVACT: .BYTE 0      ;DRIVE 0
32 027341      000      .BYTE 0      ;DRIVE 1
   027342      000      .BYTE 0      ;DRIVE 2
   027343      000      .BYTE 0      ;DRIVE 3
   027344      000      .BYTE 0      ;DRIVE 4
   027345      000      .BYTE 0      ;DRIVE 5
   027346      000      .BYTE 0      ;DRIVE 6
   027347      000      .BYTE 0      ;DRIVE 7
33
34     ;TABLE OF DRIVE STATUS INDICATORS (DRVSTA=8 BYTES)
35     ;DRVSTA=0 IF DRIVE IS OFFLINE OR NONEXISTENT
36     ;DRVSTA>0 IF DRIVE IS ONLINE
37     ;DRVSTA<0 IF DRIVE IS UNSAFE
38
39 027350      000      DRVSTA: .BYTE 0      ;DRIVE 0
42 027351      000      .BYTE 0      ;DRIVE 1
   027352      000      .BYTE 0      ;DRIVE 2
   027353      000      .BYTE 0      ;DRIVE 3
   027354      000      .BYTE 0      ;DRIVE 4
   027355      000      .BYTE 0      ;DRIVE 5
   027356      000      .BYTE 0      ;DRIVE 6
   027357      000      .BYTE 0      ;DRIVE 7
43
44     ;TABLE OF DRIVE TYPES (DRVTYP=8 BYTES)
45     ;DRVTYP=0 IF DRIVE IS NONEXISTENT (DRVSTA=0, ALSO)
46     ;DRVTYP=7 IF DRIVE IS RM05 *****
47     ;DRVTYP=5 IF DRIVE IS RM02 *****
48     ;DRVTYP=4 IF DRIVE IS RM03
49     ;DRVTYP=-1 IF NOT RM05/3/2
```

```

50
51 027360      000      DRVTYP: .BYTE 0          ;DRIVE 0
54 027361      000          .BYTE 0          ;DRIVE 1
    027362      000          .BYTE 0          ;DRIVE 2
    027363      000          .BYTE 0          ;DRIVE 3
    027364      000          .BYTE 0          ;DRIVE 4
    027365      000          .BYTE 0          ;DRIVE 5
    027366      000          .BYTE 0          ;DRIVE 6
    027367      000          .BYTE 0          ;DRIVE 7
55
56      ;TABLE OF DUAL PORT INITIALIZATION INDICATORS
57      ;DPINT=0 IF INITIALIZATION IS NOT ACTIVE ON THE DRIVE
58      ;DPINT<0 IF INITIALIZATION IS IN PROGRESS
59
60 027370      000      DPINT: .BYTE 0          ;DRIVE 0
63 027371      000          .BYTE 0          ;DRIVE 1
    027372      000          .BYTE 0          ;DRIVE 2
    027373      000          .BYTE 0          ;DRIVE 3
    027374      000          .BYTE 0          ;DRIVE 4
    027375      000          .BYTE 0          ;DRIVE 5
    027376      000          .BYTE 0          ;DRIVE 6
    027377      000          .BYTE 0          ;DRIVE 7
64
65      ;TABLE OF PENDING DUAL PORT REQUESTS
66      ;DPRQS=0 IF THAT A DUAL PORT REQUEST IS NOT PENDING FOR THAT DRIVE
67      ;DPRQS<0 IF THAT A DUAL PORT REQUEST IS PENDING FOR THAT DRIVE
68
69 027400      000      DPRQS: .BYTE 0          ;DRIVE 0
72 027401      000          .BYTE 0          ;DRIVE 1
    027402      000          .BYTE 0          ;DRIVE 2
    027403      000          .BYTE 0          ;DRIVE 3
    027404      000          .BYTE 0          ;DRIVE 4
    027405      000          .BYTE 0          ;DRIVE 5
    027406      000          .BYTE 0          ;DRIVE 6
    027407      000          .BYTE 0          ;DRIVE 7
73
74      ;TRANSFER WAIT FLAG (TRNSWT=1 WORD)
75      ;THIS IS A ONE WORD QUEUE. IT WILL CONTAIN THE ADDRESS OF
76      ;'DPB' OF THE I/O OPERATION.
77
78 027410      000000    TRNSWT: .WORD 0
79
80      ;SEARCH WAIT KEYS (SRCHWT=1 WORD)
81      ;THIS IS A ONE WORD QUEUE THAT WILL CONTAIN A KEY FOR EACH OF
82      ;THE DRIVES THAT ARE PERFORMING A SEARCH COMMAND FOR THE I/O
83      ;REQUEST THAT IS AT THE TOP OF THEIR REQUEST QUEUE.
84      ;EACH DRIVE IS ASSIGNED ONE BIT, STARTING AT BIT00 FOR DRIVE 0.
85
86 027412      000000    SRCHWT: .WORD 0
87
88      ;RM DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)
89      ;ACTDRV=0 IF DRIVER IS INACTIVE
90      ;ACTDRV>0 IF DRIVER IS ACTIVE
91
92 027414      000      ACTDRV: .BYTE 0
93
94      ;SOFTWARE TIMER ROUTINE ACTIVE FLAG (ACTSTR=1 BYTE)
    
```

```

95                                     ;ACTSTR=0 IF SOFTWARE TIMER ROUTINE IS INACTIVE
96                                     ;ACTSTR>0 IF SOFTWARE TIMER ROUTINE IS ACTIVE
97
98 027415      000      ACTSTR: .BYTE  0
99
100                                     ;UNLOAD FLAG (ULDFLG=8 BYTES)
101                                     ;ULDFLG=0 IF NO UNLOAD COMMAND
102                                     ;ULDFLG>0 IF UNLOAD COMMAND IN PROGRESS
103                                     ;ULDFLG<0 IF UNLOAD COMMAND IN WAIT QUEUE
104
105 027416      000      ULDFLG: .BYTE  0          ;DRIVE 0
108 027417      000          .BYTE  0          ;DRIVE 1
      027420      000          .BYTE  0          ;DRIVE 2
      027421      000          .BYTE  0          ;DRIVE 3
      027422      000          .BYTE  0          ;DRIVE 4
      027423      000          .BYTE  0          ;DRIVE 5
      027424      000          .BYTE  0          ;DRIVE 6
      027425      000          .BYTE  0          ;DRIVE 7
109
110                                     ;LOOK AHEAD COUNT (LACNT=8 BYTES)
111                                     ;LACNT WILL INDICATE THE NUMBER OF LOOK AHEADS PERFORMED
112
113 027426      000      LACNT:  .BYTE  0          ;DRIVE 0
116 027427      000          .BYTE  0          ;DRIVE 1
      027430      000          .BYTE  0          ;DRIVE 2
      027431      000          .BYTE  0          ;DRIVE 3
      027432      000          .BYTE  0          ;DRIVE 4
      027433      000          .BYTE  0          ;DRIVE 5
      027434      000          .BYTE  0          ;DRIVE 6
      027435      000          .BYTE  0          ;DRIVE 7
117
118                                     ;SAVE REGISTERS FLAG (SAVEFG =1 WORD)
119                                     ;SAVEFG <0 IF SAVE THE RH/RM REGISTERS WHEN THE
120                                     ;OPERATION IS COMPLETED AS PER (DPB+14).
121                                     ;SAVEFG=0 IF SAVE THE RH/RM REGISTERS, AS PER
122                                     ;(DPB+14), AFTER AN ERROR.
123
124 027436      000000    SAVEFG: .WORD  0
125
126                                     ;SEEK FLAG (SEEKFG=1 WORD)
127                                     ;SEEKFG=0 IF WHEN THE DISK ADDRESS ISN'T IN THE WINDOW
128                                     ;FOR A DATA TRANSFER START A SEARCH COMMAND
129                                     ;SEEKFG<0 IF DATA TRANSFER WILL DO IMPLIED SEEKS,
130                                     ;DISREGARD THE WINDOW
131
132 027440      177777    SEEKFG: .WORD -1
133
134                                     ;TIMEOUT TABLE (TIMER=8 WORDS)
135                                     ;THIS TABLE CONTAINS THE TIME ALLOWED FOR AN OPERATION
136
137 027442      177777    TIMER:  .WORD -1          ;DRIVE 0
140 027444      177777          .WORD -1          ;DRIVE 1
      027446      177777          .WORD -1          ;DRIVE 2
      027450      177777          .WORD -1          ;DRIVE 3
      027452      177777          .WORD -1          ;DRIVE 4
      027454      177777          .WORD -1          ;DRIVE 5
      027456      177777          .WORD -1          ;DRIVE 6
    
```

```

027460 177777          .WORD  -1          ;DRIVE 7
141
142          ;DATA TRANSFER UNDERWAY INDICATOR (DTUW=1 WORD)
143          ;DTUW<0 IF NO DATA TRANSFER UNDERWAY
144          ;DTUW=+N (WHERE N=0 TO 7) IMPLIES DATA TRANSFER UNDERWAY ON DRIVE N
145
146 027462 177777      DTUW:  .WORD  -1
147
148          ;ATTENTION BITS TABLE (ATABIT=8 BYTES)
149          ;THIS TABLE CONTAINS THE CORRESPONDING BIT TO EACH DRIVES
150          ;ATTENTION BIT
151
152 027464      001      ATABIT:  .BYTE  1          ;DRIVE 0
153 027465      002          .BYTE  2          ;DRIVE 1
154 027466      004          .BYTE  4          ;DRIVE 2
155 027467      010          .BYTE  10         ;DRIVE 3
156 027470      020          .BYTE  20         ;DRIVE 4
157 027471      040          .BYTE  40         ;DRIVE 5
158 027472      100          .BYTE  100        ;DRIVE 6
159 027473      200          .BYTE  200        ;DRIVE 7
160
161          ;FSRM TO RH11/RH70 'MASSBUS CONTROL BUS PARITY ERRORS' (MCPE) ALLOWED BEFORE
162          ;CALLING IT FATAL (MCPEMX=1 WORD)
163
164 027474      000003      MCPEMX: .WORD  3
165
166          ;STORAGE FOR RMADR (THE FIRST ADDRESS (776700) OF THE RH/RM),
167          ;RMVEC (THE VECTOR ADDRESS (254)), AND RMVEC+2 (THE BR LEVEL (5)).
168
169 027476      176700      RMADR:  .WORD  176700
170 027500      000254      000240      RMVEC:  .WORD  254,5*32.
171
172          ;MAXIMUM NUMBER OF LOOK AHEADS ALLOWED IS 4 (MXLACT=1 WORD)
173 027504      000004      MXLACT: .WORD  4
174
175          ;MAXIMUM DELTA DELAY IS 8 SECTORS (MXDLTA=1 WORD)
176 027506      001000      MXDLTA: .WORD  8.*64.
177
178          ;MINIMUM DELTA DELAY IS 2 SECTORS (MNDLTA=1 WORD)
179 027510      000200      MNDLTA: .WORD  2*64.
180
181          ;MAXIMUM SEARCH FOR I/O WINDOW IS 5 SECTORS (MXWNDW=1 WORD)
182 027512      000005      MXWNDW: .WORD  5
183
184          ;DEFINITIONS OF THE RH/RM ADDRESS INDEXES
185
186          000000      RMCS1=0          ;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 00)
187          000002      RMWC=2          ;WORD COUNT REGISTER (NOT A DRIVE REG)
188          000004      RMBA=4          ;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)
189          000006      RMDA=6          ;DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 05)
190          000010      RMCS2=10        ;CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
191          000012      RMD5=12          ;DRIVE STATUS REGISTER (DRIVE REG 01)
192          000014      RMER1=14        ;ERROR REGISTER #1 (DRIVE REG. 02)
193          000016      RMAS=16          ;ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 04)
194          000020      RMLA=20          ;LOOK AHEAD REGISTER (DRIVE REG. 07)
195          000022      RMDB=22          ;DATA BUFFER REGISTER (NOT A DRIVE REG.)
196          000024      RMMR1=24        ;MAINTAINABILITY REGISTER (DRIVE REG. 03)
    
```

197	000026	RMDT=26	:DRIVE TYPE REGISTER (DRIVE REG. 06)
198	000030	RMSN=30	:SERIAL NUMBER REGISTER (DRIVE REG. 10)
199	000032	RMOF=32	:OFFSET REGISTER (DRIVE REG. 11)
200	000034	RMDC=34	:DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
201	000036	RMHR=36	:DUMMY ADDRESS REGISTER (DRIVE REG. 13)
202	000040	RMMR2=40	:MAINTENANCE REGISTER #2
203	000042	RMER2=42	:ERROR REGISTER #2 (DRIVE REG. 15)
204	000044	RMEC1=44	:ECC POSITION REGISTER (DRIVE REG. 16)
205	000046	RMEC2=46	:ECC PATTERN REGISTER (DRIVE REG. 17)

206
 207 .SBTTL RH/RM DRIVER INITIALIZATION CODE
 208
 209 ;THIS ROUTINE WILL DETERMINE WHICH RM DRIVES ARE
 210 ;AVAILABLE FOR TESTING AND SET THE DRVSTA INDICATOR
 211 ;TO THE PROPER STATE FOR EACH DRIVE.
 212 ;NOTE: THIS ROUTINE CALLS DRVINT

213
 214 ;CALL
 215
 216 ; JSR PC,RMINIT
 217 ; RETURN
 218
 219 ;NOTE: THE 'P' OR 'L' CLOCK MUST BE STARTED
 220 ;

221	027514	104412	RMINIT: SAVREG	:SAVE R0 - R5
222	027516	013746	MOV PS,-(SP)	:SAVE THE PRESENT PROCESSOR STATUS
223	027522	012737	MOV #<5*32.>,PS	:CHANGE THE PRIORITY TO 5
224	027530	004737	JSR PC,CLRQUE	:CLEAR ALL REQUEST QUEUES
225	027534	012701	MOV #RMERRS,R1	:FIRST ADDRESS TO BE CLEARED
226	027540	012702	MOV #SEEKFG,R2	:LAST ADDRESS TO BE CLEARED
227	027544	005021	1\$: CLR (R1)+	:CLEAR
228	027546	020102	CMP R1,R2	:ARE WE DONE?
229	027550	103775	BLO 1\$:BR IF NO
230	027552	012702	MOV #DTUW,R2	:LAST ADDRESS
231	027556	012721	2\$: MOV #-1,(R1)+	:INITIALIZE
232	027562	020102	CMP R1,R2	:DONE?
233	027564	101774	BLOS 2\$:LOOP IF NO
234	027566	005037	CLR DRVSTA	:SET ALL DRIVES TO OFFLINE
235	027572	005037	CLR DRVSTA+2	
236	027576	005037	CLR DRVSTA+4	
237	027602	005037	CLR DRVSTA+6	
238	027606	013703	MOV RMVEC,R3	:SETUP THE RH/RM VECTOR
239	027612	012723	MOV #ISR,(R3)+	
240	027616	013713	MOV RMVEC+2,(R3)	
241	027622	013704	MOV RMADR,R4	:FIRST ADDRESS OF RH/RM
242	027626	012764	MOV #CLR,RMCS2(R4)	:MASSBUS INIT
243	027634	005001	CLR R1	:START WITH DRIVE 0
244	027636	004037	3\$: JSR R0,DRVINT	:INIT THE DRIVE
245	027642	000401	BR 4\$: 'DVA' NOT SET OR PARITY ERROR
246	027644	000402	BR 5\$:NORMAL RETURN
247	027646	105061	4\$: CLRB DRVSTA(R1)	:SET DRIVE STATUS TO OFFLINE
248	027652	005201	5\$: INC R1	:GO TO NEXT DRIVE
249	027654	042701	BIC #^C7,R1	:MASK OUT UNUSED BITS
250	027660	001366	BNE 3\$:BR IF MORE DRIVES TO GO
251	027662	012701	MOV #7,R1	:START WITH DRIVE 7
252	027666	005037	CLR PS	:CLEAR THE PROCESSOR STATUS
253	027672	105761	6\$: TSTB DPINT(R1)	:WAITING FOR DRIVE TO SWITCH PORTS ?

```

254 027676 001405          BEQ      8$          ;BR NOT WAITING
255 027700 004737 035036   JSR      PC,SET.IE   ;SET INTERRUPT
256 027704 105761 027370   7$:     TSTB     DP,INT(R1) ;DRIVE SWITCHED PORTS ?
257 027710 001375          BNE      7$          ;BR IF NOT
258 027712 005301          DEC      R1          ;GO TO THE NEXT DRIVE
259 027714 100366          BPL      6$          ;CHECK NEXT DRIVE
260 027716 012637 177776   MOV      (SP)+,PS    ;RESTORE THE PROCESSOR STATUS
261 027722 104413          RESREG                    ;RESTORE R0 - R5
262 027724 000207          RTS       PC          ;BYE-BYE
263
264
265 ;DRIVE INITIALIZATION ROUTINE
266 ;THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
267 ;AN RM05/3/2. IF IT IS, A 'READ-IN PRESET' IS ISSUED AND FMT16
268 ;IS SET TO A '1'. THEN MOL, DPR, DRY, AND VV ARE CHECKED TO
269 ;INSURE THEY ARE ALL ON A '1'. AND DEPENDING ON THEIR STATE,
270 ;DRVSTA IS SET TO THE PROPER CONDITION.
271 ;CALL
272 :       MOV      #DRVNUM,R1 ;DRIVE NUMBER TO R1
273 :       MOV      RMADR,R4    ;UNIBUS ADDRESS OF RH/RM (RMCS1)
274 :       JSR      R0,DRVINT   ;CALLED BY A JSR
275 :       RETURN1 ;ERROR OCCURRED (PARITY)
276 :       RETURN2 ;NORMAL RETURN
277
278 027726 010546          DRVINT: MOV      R5,-(SP) ;SAVE R5
279 027730 105061 027350   DULP:  CLRB     DRVSTA(R1) ;START DRIVE STATUS AS OFFLINE
280 027734 105061 027360   CLRB     DRVSTYP(R1)    ;CLEAR THE DRIVE TYPE INDICATOR
281 027740 105061 027416   CLRB     ULDFLG(R1)     ;CLEAR THE UNLOAD FLAG
282 027744 010164 000010   MOV      R1,RMCS2(R4)   ;SELECT A DRIVE
283 027750 112764 000111 000000   MOV      #111,RMCS1(R4) ;DO A DRIVE CLEAR COMMAND (& SEIZE DRIVE)
284 027756 032764 010000 000010   BIT      #BIT12,RMCS2(R4) ;NONEXISTENT DRIVE?
285 027764 001403          BEQ      1$          ;NO
286 027766 004737 035036   JSR      PC,SET.IE   ;GO SET 'IE' WITHOUT A 'TRE'
287 027772 000520          BR       6$          ;LEAVE THIS ROUTINE
288
289 027774 105061 027350   1$:     CLRB     DRVSTA(R1) ;SET DRIVE STATUS TO OFFLINE
290 030000 032764 004000 000000   BIT      #BIT11,RMCS1(R4) ;SEE IF DRIVE AVAILABLE
291 030006 001750          BEQ      DULP        ;BR IF DRIVE NOT AVAILABLE
292 030010 004037 034346   JSR      R0,RD.RM     ;READ THE DRIVE TYPE REG.
293 030014 000026          RMDT
294 030016 030260          8$
295 030020 012605          MOV      (SP)+,R5     ;ERROR RETURN ADDRESS
296 030022 112761 000004 027360   MOV      #4,DRVSTYP(R1) ;PUT DRIVE TYPE IN R5
297 030030 022705 020024   MOV      #20024,R5    ;SET RM03 INDICATOR
298 030034 001431          CMP      #20024,R5    ;SINGLE PORT RM03 ?
299 030036 022705 024024   BEQ      2$          ;BR IF YES
300 030042 001426          CMP      #24024,R5    ;DUAL PORT RM03 ?
301 030044 112761 000005 027360   BEQ      2$          ;BR IF YES
302 030052 022705 020025   MOV      #5,DRVSTYP(R1) ;SET RM02 INDICATOR
303 030056 001420          CMP      #20025,R5    ;SINGLE PORT RM02 ?
304 030060 022705 024025   BEQ      2$          ;BR IF SO
305 030064 001415          CMP      #24025,R5    ;DUAL PORT RM02 ?
306 030066 112761 000007 027360   BEQ      2$          ;BR IF SO
307 030074 022705 020027   MOV      #7,DRVSTYP(R1) ;SET RM05 INDICATOR
308 030100 001407          CMP      #20027,R5    ;SINGLE PORT RM05 ?
309 030102 022705 024027   BEQ      2$          ;BR IF YES
310 030106 001404          CMP      #24027,R5    ;DUAL PORT RM05 ?
          BEQ      2$          ;BR IF YES
  
```



```

311 030110 112761 177777 027360      MOVB  #-1,DRVSTYP(R1)  ;SET INDICATOR TO 'OTHER'
312 030116 000446                    BR    6$              ;EXIT
313
314 030120 012746 000121      2$:  MOV    #121,-(SP)    ;DO A 'READ-IN PRESET''
315 030124 004037 034526      JSR    R0,WRT.RM
316 030130 000000                    RMCS1
317 030132 030260                    8$
318 030134 012746 010000      MOV    #BIT12,-(SP)    ;SET FMT16=1
319 030140 004037 034526      JSR    R0,WRT.RM
320 030144 000032                    RMOF
321 030146 030260                    8$
322 030150 004037 034346      JSR    R0,RD.RM      ;READ RMDS
323 030154 000012                    RMDS
324 030156 030260                    8$
325 030160 012605                    MOV    (SP)+,R5      ;AND SAVE IT IN R5
326 030162 100015                    BPL    4$            ;BR IF ATA=0
327 030164 116164 027464 000016      MOVB  ATABIT(R1),RMAS(R4) ;CLEAR ATTENTION BIT
328 030172 004037 034346      JSR    R0,RD.RM      ;FIND OUT WHY ATA=1
329 030176 000014                    RMER1
330 030200 030260                    8$
331 030202 006126                    ROL    (SP)+        ;IS IT UNSAFE?
332 030204 100004                    BPL    4$            ;BR IF NOT
333 030206 112761 177777 027350      MOVB  #-1,DRVSTA(R1)  ;SET UNSAFE INDICATOR
334 030214 000407                    BR    6$              ;EXIT
335
336 030216 005105      4$:  COM    R5          ;CHECK MOL, DPR, DRY, AND VV
337 030220 042705 167077      BIC    #^C<BIT12!BIT08!BIT07!BIT06>,R5
338 030224 001003                    BNE    6$            ;BR IF MOL, DPR, DRY, OR VV IS CLEAR
339 030226 112761 000001 027350      MOVB  #1,DRVSTA(R1)  ;SET DRIVE STATUS TO ONLINE
340 030234 005720      6$:  TST    (R0)+        ;STEP OVER THE ERROR RETURN
341 030236 000410                    BR    8$              ;EXIT
342 030240 006301      7$:  ASL    R1          ;CHANGE INDEX TO ADDRESS WORDS
343 030242 012761 060000 027442      MOV    #60000,TIMER(R1) ;START 2 SEC TIMER
344 030250 006201                    ASR    R1            ;RESTORE R1
345 030252 112761 177777 027370      MOVB  #-1,DPINT(R1)  ;SET PORT INITIALIZE INIDICATOR
346 030260 012605      8$:  MOV    (SP)+,R5      ;RESTORE R5
347 030262 000200                    RTS    R0            ;EXIT!
348
349      ;REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
350
351      ;CALL
352
353      ;
354      JSR    R0,RM05    ;CALL THE RM05 DRIVER
355      PNTADR ;ADDRESS OF POINTER OF DRIVES PARAMETER BLOCK
356      RETURN1 ;RETURN HERE IF QUEUE IS FULL
357      ;
358      RETURN2 ;RETURN HERE IF REQUEST IS IN QUEUE OR THERE
359      ;IS AN ERROR CONDITION
360
361 RM05: MOV    PS,-(SP)    ;SAVE THE CALLING STATUS
362      MOV    RMVEC+2,PS ;DON'T ALLOW ANY RM INTERRUPTS
363      MOVB  #1,ACTDRV  ;SET 'ACTIVE DRIVER' FLAG
364      SAVREG ;SAVE R0 - R5
365      MOV    (R0),R2   ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
366      CLR    16(R2)   ;CLEAR THE STATUS/ERROR INDICATOR
367      MOVB  (R2),R1   ;PICKUP THE DRIVE NUMBER
368      MOV    RMADR,R4 ;UNIBUS ADDRESS OF RMCS1
369      TSTB  DRVSTA(R1) ;CHECK DRIVES STATUS

```

```

368 030326 003014          BGT      1$          ;BR IF ONLINE
369 030330 105761 027416  TSTB    ULDFLG(R1) ;UNLOAD COMMAND IN QUEUE?
370 030334 001036          BNE      3$          ;BR IF YES
371 030336 105761 027370  TSTB    DPINT(R1)  ;TRYING TO INIT THE DRIVE
372 030342 001042          BNE      5$          ;BR IF YES
373 030344 004037 027726  JSR     R0,DRVINT  ;GO INIT. THE DRIVE
374 030350 000434          BR       4$          ;ERROR RETURN
375 030352 105761 027350  TSTB    DRVSTA(R1) ;IS DRIVE STATUS ONLINE?
376 030356 003445          BLE      6$          ;BR IF NOT
377 030360 105761 027400  1$:    TSTB    DPRQS(R1) ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
378 030364 001031          BNE      5$          ;BR IF YES
379 030366 010164 000010  MOV     R1, RMCS2(R4) ;SELECT THE DRIVE
380 030372 004037 035500  JSR     R0,DRVQUE  ;PUT THIS REQUEST IN QUEUE
381 030376 000460          BR       9$          ;QUEUE IS FULL
382 030400 122762 000103 000002  CMPB    #103,2(R2) ;IS THIS REQ. FOR AN UNLOAD?
383 030406 001003          BNE      2$          ;BR IF NO
384 030410 112761 177777 027416  MOVB    #-1,ULDFLG(R1) ;SET THE 'UNLOAD IN QUEUE' FLAG
385 030416 105761 027340  2$:    TSTB    DRVACT(R1) ;IS THIS DRIVE ACTIVE?
386 030422 001043          BNE      8$          ;BR IF YES
387 030424 004737 030556  JSR     PC,OPT     ;CALL THE OPTIMIZER
388 030430 000440          BR       8$
389 030432 012762 120000 000016  3$:    MOV     #BIT15!BIT13,16(R2) ;SET THE 'UNLOAD IN QUEUE' ERROR FLAG
390 030440 000434          BR       8$          ;EXIT
391 030442 004737 031636  4$:    JSR     PC,C17    ;GO HANDLE THE PARITY ERROR
392 030446 000431          BR       8$
393 030450 004037 035500  5$:    JSR     R0,DRVQUE  ;PUT REQUEST IN QUEUE
394 030454 000431          BR       9$          ;QUEUE IS FULL
395 030456 032714 000100  BIT     #BIT06,(R4) ;IE BIT SET ?
396 030462 001023          BNE      8$          ;YES
397 030464 004737 035036  JSR     PC,SET.IE  ;SET THE INTERRUPT
398 030470 000420          BR       8$          ;RETURN
399 030472 105761 027350  6$:    TSTB    DRVSTA(R1) ;SEE IF DRIVE OFFLINE OR UNSAFE
400 030476 002412          BLT     7$          ;BR IF UNSAFE
401 030500 012762 140000 000016  MOV     #BIT15!BIT14,16(R2) ;SET OFFLINE ERROR INDICATOR
402 030506 105761 027360  TSTB    DRVSTYP(R1) ;SEE IF OFFLINE OR NONEXISTENT
403 030512 001007          BNE      8$          ;BR IF OFFLINE
404 030514 012762 100002 000016  MOV     #BIT15!BIT01,16(R2) ;REPORT DRIVE NONEXISTENT
405 030522 000403          BR       8$          ;GO TO EXIT
406 030524 012762 110000 000016  7$:    MOV     #BIT15!BIT12,16(R2) ;DRIVE IS UNSAFE
407 030532 104413          8$:    RESREG          ;RESTORE R0 - R5
408 030534 005720          TST     (R0)+      ;SETUP FOR NORMAL RETURN
409 030536 000401          BR      10$         ;FINISH UP, THEN EXIT
410 030540 104413          9$:    RESREG          ;RESTORE R0 - R5
411 030542 005720          10$:   TST     (R0)+      ;CORRECT THE RETURN ADDRESS
412 030544 105037 027414  CLRB    ACTDRV    ;CLEAR 'ACTIVE DRIVER' FLAG
413 030550 012637 177776  MOV     (SP)+,PS   ;RETURN 'PS' TO USER LEVEL
414 030554 000200          RTS     R0         ;RETURN TO CALLER
415
416          ;OPTIMIZER-CALLED FOR A PARTICULAR DRIVE
417          ;CALL
418          ;
419          ;
420          ;
421          ;
422 030556 104412          OPT:   SAVREG          ;SAVE R0 - R5
423 030560 013746 177776  MOV     PS,-(SP)   ;SAVE PROC. STATUS
424 030564 146137 027464 027412  BICB    ATABIT(R1),SRCHWT ;CLEAR LA SEACH FLAG

```

```

425 030572 105061 027400          CLRB   DPRQS(R1)      ;RESET THE PORT REQ FLAG ****
426 030576 004737 035554          JSR    PC,GETREQ     ;GET 'DPB' POINTER OF REQUEST
427 030602 005702                TST    R2            ;IS THERE A REQUEST IN QUEUE?
428 030604 001472                BEQ    7$            ;NO--BR TO EXIT
429 030606 010164 000010          MOV    R1, RMCS2(R4) ;LOAD THE DRIVE ADDRESS *****
430 030612 012764 000111 000000    MOV    #111, RMCS1(R4) ;CLEAR THE DRIVE
431 030620 032764 004000 000000    BIT    #BIT11, RMCS1(R4) ;DVA SET ?
432 030626 001446                BEQ    5$            ;TO PROT REQUEST ,IF NOT
433 030630 105761 027350          10$:  TSTB   DRVSTA(R1)  ;IS DRIVE ONLINE?
434 030634 003014                BGT    1$            ;YES
435 030636 004737 035576          JSR    PC,POPQUE     ;NO--REMOVE REQUEST FROM QUEUE
436 030642 012762 140000 000016    MOV    #BIT15!BIT14,16(R2) ;SET OFFLINE STATUS/ERROR INDICATOR
437 030650 105761 027350          TSTB   DRVSTA(R1)  ;IS DRIVE UNSAFE ?
438 030654 100053                BPL    8$            ;BR TO EXIT IF NOT
439 030656 012762 110000 000016    MOV    #BIT15!BIT12,16(R2) ;SET UNSAFE STATUS/ERROR INDICATOR
440 030664 000447                BR     8$            ;BR TO EXIT
441 030666                1$:
442                :
443                :
444                :
445                :
446                :
447                :
448 030666 122762 000150 000002    MOV    #111, -(SP)   ;LOAD COMMAND ONTO THE STACK
449 030674 002403                JSR    R0, WRT.RM    ;LOAD THE REGISTER
450 030676 004737 031222          RMCS1  ;REGISTER INCREMENT
451 030702 000440                6$
452 030704 005737 027462          BIT    #BIT11, (R4) ;DRIVE AVAILABLE ?
453 030710 002012                BEQ    9$            ;BR IF NOT
454 030712 005737 027440          CMPB   #150,2(R2)   ;IS THE REQUEST FOR I/O?
455 030716 100404                BLT    2$            ;YES
456 030720 004037 032174          JSR    PC, CI4       ;CALL THE COMMAND INITIATOR
457 030724 000427                BR     8$            ;BR TO EXIT
458 030726 000403                TST    DTUW         ;DATA TRANSFER UNDERWAY?
459 030730 004737 031014          2$:  BGE    4$            ;YES--GO START A SEARCH
460 030734 000423                TST    SEEKFG       ;DO IMPLIED SEEKS?
461 030736 004737 031122          BMI    3$            ;YES
462 030742 000420                JSR    R0, LA        ;NO--DO LOOK AHEAD
463 030744 112761 177777 027400    BR     8$            ;RETURN HERE ON A PARITY ERROR
464 030752 010103                BR     4$            ;GO START A SEARCH
465 030754 006303                JSR    PC, CI1       ;START A DATA TRANSFER
466 030756 012763 060000 027442    BR     8$            ;START A SEARCH
467 030764 000402                BR     8$            ;START A SEARCH
468 030766 004737 031636          3$:  JSR    PC, CI1       ;START A SEARCH
469 030772 032714 000100          4$:  BR     8$            ;GO TO THE EXIT
470 030776 001002                BR     8$            ;GO TO THE EXIT
471 031000 004737 035036          5$:  MOVB   #-1, DPRQS(R1) ;SET PORT REQUEST INDICATOR
472 031004 012637 177776          6$:  MOV    R1, R3       ;SET UP TO ADDRESS WORDS
473 031010 104413                ASL    R3            ;CONVERT TO WORD INDEX
474 031012 000207                MOV    #60000, TIMER(R3) ;START 10 SEC TIMER
475                BR     7$            ;EXIT
476                6$:  JSR    PC, CI7       ;PROCESS THE PARITY ERROR
477                7$:  BIT    #BIT06, (R4) ;SEE IF 'IE' ALREADY SET
478                BNE    8$            ;BR IF SET
479                JSR    PC, SET.IE ;SET 'IE' WITHOUT A 'TRE'
480                8$:  MOV    (SP)+, PS    ;RESTORE PROC. STATUS
481                RESREG ;RESTORE R0 - R5
RTS    PC

;COMMAND INITIATOR
:
:CALL
:
:   MOV    #DRVNUM, R1 ;DRIVE NUMBER
:   MOV    #DPB, R2    ;ADDRESS OF DPB
:   JSR    PC, CI?    ;CI?= CI1, CI3, OR CI4

```

```

482
483
484
485
486
487 031014 004737 035576
488 031020 010237 027410
489 031024 010203
490 031026 013704 027476
491 031032 010164 000010
492 031036 062703 000004
493 031042 062704 000002
494 031046 012324
495 031050 012324
496 031052 012346
497 031054 004037 034526
498 031060 000006
499 031062 031636
500 031064 012346
501 031066 004037 034526
502 031072 000034
503 031074 031636
504 031076 016246 000002
505 031102 004037 034526
506 031106 000000
507 031110 031636
508 031112 010137 027462
509 031116 000137 031600
510 031122 013704 027476
511 031126 010164 000010
512 031132 016246 000012
513 031136 004037 034526
514 031142 000034
515 031144 031636
516 031146 116203 000010
517
518
519
520 031152 010346
521 031154 042716 177740
522 031160 116266 000011 000001
523 031166 004037 034526
524 031172 000006
525 031174 031636
526 031176 012746 000131
527 031202 004037 034526
528 031206 000000
529 031210 031636
530 031212 156137 027464 027412
531 031220 000567
532 031222 013704 027476
533 031226 010164 000010
534 031232 116203 000002
535 031236 122703 000131
536 031242 001007
537 031244 016246 000010
538 031250 004037 034526
    
```

```

:
:
:
:
:WHERE:
:CI1=DATA TRANSFER
:CI2=SEARCH REQUESTED BY DATA XFER
:CI4=NOT DATA TRANSFER
:REMOVE REQUEST FROM 'DRIVES WAIT' QUEUE
:PUT REQ. IN TRANSFER WAIT QUEUE
:DPB ADDRESS TO R3
:RMCS1 ADDRESS
:SELECT DRIVE
:DESIRED WORD COUNT
:RMWC ADDRESS
:LOAD WORD COUNT
:LOAD BUFFER ADDRESS
:LOAD SECTOR AND TRACK
:CALL THE LOAD(WRITE) ROUTINE
:INDEX OF REGISTER TO LOAD
:ERROR RETURN ADDRESS
:LOAD CYLINDER ADDRESS
:LOAD 'COMMAND+GO', 'A17&A16', AND 'PSEL'
:SET 'DATA TRANSFER UNDERWAY'
:RMCS1 ADDRESS
:SELECT DRIVE
:DESIRED CYLINDER ADDRESS
:PICKUP SECTOR ADDRESS
:BACKUP BY MAX. SEARCH FOR I/O WINDOW
:COMBINE THE ADJUSTED SECTOR WITH
:CHOP OF ALL EXENTED BITS ,IF ANY
:THE DESIRED TRACK
:LOAD DESIRED TRACK & SECTOR
:START A SEARCH
:SET 'SEARCH WAIT' KEY
:RMCS1 ADDRESS
:SELECT DRIVE
:PICKUP THE REQUESTED COMMAND
:IS IT A SEARCH COMMAND?
:BR IF NO
:LOAD DESIRED TRACK & SECTOR
    
```

539	031254	000006								
540	031256	031636								
541	031260	000403								
542	031262	122703	000105	1\$:	BR	2\$:GO LOAD CYLINDER	
543	031266	001007			CMPB	#105,R3			:IS IT A SEEK COMMAND	
544	031270	016246	000012	2\$:	BNE	3\$:BR IF NO	
545	031274	004037	034526		MOV	12(R2),-(SP)			:LOAD DESIRED CYLINDER	
546	031300	000034			JSR	R0,WRT.RM				
547	031302	031636			RMDC					
548	031304	000546			CI7					
549	031306	122703	000115	3\$:	BR	CI6				
550	031312	001013			CMPB	#115,R3			:IS IT AN 'OFFSET' COMMAND?	
551	031314	004037	034346		BNE	4\$:BR IF NO	
552	031320	000032			JSR	R0,RD.RM			:MERGE THE OFFSET VALUE INTO RMOF	
553	031322	031636			RMOF				:BUT DON'T CHANGE THE UPPER	
554	031324	116216	000001		CI7					
555	031330	004037	034526		MOVB	1(R2),(SP)			:BYTE WHEN LOADING THE	
556	031334	000032			JSR	R0,WRT.RM			:REGISTER (RMOF)	
557	031336	031636			RMOF					
558	031340	000530			CI7					
559	031342	122703	000107	4\$:	BR	CI6			:GO START THE COMMAND	
560	031346	001525			CMPB	#107,R3			:IS IT A 'RECALIBRATE' COMMAND?	
561	031350	122703	000117		BEQ	CI6			:BR IF YES	
562	031354	001522			CMPB	#117,R3			:IS IT A RETURN TO CENTER?	
563	031356	122703	000103		BEQ	CI6			:BR IF YES	
564	031362	001016			CMPB	#103,R3			:IS IT AN 'UNLOAD' COMMAND?	
565	031364	112761	000001	027340	BNE	5\$:BR IF NO	
566	031372	105061	027350		MOVB	#1,DRVACT(R1)			:SET THE DRIVE ACTIVE INDICATOR	
567	031376	112761	000001	027416	CLRB	DRVSTA(R1)			:PUT DRIVE STATUS TO OFFLINE	
568	031404	010346			MOVB	#1,ULDFLG(R1)			:SET 'UNLOAD IN PROGRESS' FLAG	
569	031406	004037	034526		MOV	R3,-(SP)			:START THE 'UNLOAD' COMMAND	
570	031412	000000			JSR	R0,WRT.RM				
571	031414	031636			RMCS1					
572	031416	000207			CI7					
573	031420	122703	000143	5\$:	RTS	PC			:RETURN TO USER	
574	031424	001014			CMPB	#143,R3			:IS IT A 'SET FORMAT' COMMAND?	
575	031426	004037	034346		BNE	6\$:BR IF NO	
576	031432	000032			JSR	R0,RD.RM			:READ THE OFFSET REGISTER	
577	031434	031636			RMOF					
578	031436	116266	000001	000001	CI7					
579	031444	004037	034526		MOVB	1(R2),1(SP)			:COMBINE 'FMT16','ECI', AND 'HCI'	
580	031450	000032			JSR	R0,WRT.RM			:LOAD 'FMT16','ECI', AND/OR 'HCI'.	
581	031452	031636			RMOF					
582	031454	000436			CI7					
583	031456	122703	000141	6\$:	BR	12\$				
584	031462	001023			CMPB	#141,R3			:IS IT A 'GET REGISTER' COMMAND?	
585	031464	016203	000006	7\$:	BNE	10\$:BR IF NO	
586					MOV	6(R2),R3			:POINTS TO 1ST ADDRESS OF WHERE	
587	031470	116237	000010	031506					:TO PUT THE REGISTER(S)	
588	031476	116205	000011		MOVB	10(R2),9\$:INIT. THE INDEX FOR THE FIRST REG.	
589	031502	004037	034346		MOVB	11(R2),R5			:INDEX OF LAST REG. TO MOVE	
590	031506	000000		8\$:	JSR	R0,RD.RM			:READ RH/RM REGISTER	
591	031510	031636		9\$:	RMCS1				:INDEX OF REG. TO READ	
592	031512	012623			CI7					
593	031514	023705	031506		MOV	(SP)+,(R3)+			:GET THE CONTENTS OF RH/RM REG.	
594	031520	001414			CMP	9\$,R5			:LAST REG. BEEN READ?	
595	031522	062737	000002	031506	BEQ	12\$:GET OUT IF YES	
					ADD	#2,9\$:INCREASE THE INDEX BY 2	

```
596 031530 000764 BR 8$ ;LOOP--MORE TO READ
597 031532 122703 000145 10$: CMPB #145,R3 ;IS IT A 'SELECT DRIVE' COMMAND?
598 031536 001405 BEQ 12$ ;BR IF YES
599 031540 010346 11$: MOV R3,-(SP) ;LOAD THE COMMAND
600 031542 004037 034526 JSR R0,WRT.RM
601 031546 000000 RMCS1
602 031550 031636 CI7
603 031552 004737 035576 12$: JSR PC,POPQUE ;REMOVE REQ. FROM QUEUE
604 031556 052762 000200 000016 BIS #BIT07,16(R2) ;SET THE 'DONE' BIT
605 031564 005737 027436 TST SAVEFG ;SAVE THE RH/RM REGISTERS?
606 031570 100002 BPL 13$ ;BR IF NO
607 031572 004737 034720 JSR PC,SVRH70 ;YES--GO SAVE THE REGISTERS
608 031576 000207 13$: RTS PC ;RETURN TO USER
609 031600 006301 CI5: ASL R1
610 031602 012761 060000 027442 MOV #60000,TIMER(R1) ;SET A ONE SECOND TIMER
611 031610 006201 ASR R1
612 031612 112761 000001 027340 MOVB #1,DRVACT(R1) ;SET THE DRIVE ACTIVE
613 031620 000207 RTS PC ;RETURN TO THE USER
614 031622 010346 CI6: MOV R3,-(SP) ;LOAD THE COMMAND
615 031624 004037 034526 JSR R0,WRT.RM
616 031630 000000 RMCS1
617 031632 031636 CI7
618 031634 000761 BR CI5
619 031636 032764 010000 000010 CI7: BIT #BIT12,RMCS2(R4) ;DRIVE NON-EXISTENT ?
620 : BNE CI8 ;BR IF YES
621 031644 005702 1$: TST R2 ;ANYTHING IN QUEUE ?
622 : BEQ CI7B ;BR IF NOT
623 031646 001001 BNE 2$ ;BR IF QUEUE IS THERE
624 031650 000207 RTS PC ;OTHERWISE EXIT
625 031652 012762 104000 000016 2$: MOV #BIT15!BIT11,16(R2) ;SET 'PARITY' ERROR INDICATOR
626 : JSR PC,SVRH70 ;GO SAVE THE RH/RM REGISTERS
627 031660 012746 000111 CI7B: MOV #111,-(SP) ;DO A 'DRIVE CLEAR'
628 031664 004037 034526 JSR R0,WRT.RM
629 031670 000000 RMCS1
630 031672 031736 CI8
631 031674 004737 035460 2$: JSR PC,EMPTYQ ;EMPTY THE QUEUE
632 031700 105061 027400 CLRB DPRQS(R1) ;CLEAR THE PORT REQUEST FLAG
633 031704 105061 027416 CLRB ULDFLG(R1) ;CLEAR THE UNLOAD IN QUEUE FLAG
634 031710 105061 027340 CLRB DRVACT(R1) ;DRIVE IS IDLE
635 031714 020237 027410 CMP R2,TRNSWT ;IF THIS DRIVE HAD AN I/O REQUEST
636 : CMP R1,DTUW ;IF THIS DRIVE HAD AN I/O REQUEST
637 031720 001005 BNE 1$ ;IN PROGRESS CLEAR ALL OF THE FLAGS
638 031722 005037 027410 CLR TRNSWT
639 031726 012737 177777 027462 MOV #-1,DTUW
640 031734 000207 1$: RTS PC
641 031736 104412 CI8: SAVREG ;SAVE R0 - R5
642 031740 032764 010000 000010 BIT #BIT12,RMCS2(R4) ;IS 'NED' SET ?
643 : BNE 1$ ;BR IF YES
644 031746 005001 CLR R1
645 031750 005003 CLR R3
646 031752 105761 027340 1$: TSTB DRVACT(R1) ;DRIVE ACTIVE?
647 : BEQ 5$ ;BR IF NO
648 031756 001003 BNE 22$ ;BR IF IN ACTIVE
649 031760 105761 027400 TSTB DPRQS(R1) ;PORT REQUEST
650 031764 001443 BEQ 5$ ;BR IF NOT
651 031766 013702 027410 22$: MOV TRNSWT,R2 ;GET THE 'TRANSFER WAIT' QUEUE
652 031772 020137 027462 CMP R1,DTUW ;DID THIS DRIVE HAVE AN I/O IN PROGRESS?
```

```

653 031776 001402          BEQ      2$          :BR IF YES
654 032000 004737 035554  JSR      PC,GETREQ  :GET THE DPB POINTER
655 032004 005702          2$:   TST      R2      :QUEUE ENTRY FOR DRIVE ?
656 032006 001413          BEQ      4$          :BR IF NOT
657 032010 032764 010000 000010  BIT      #BIT12,RMCS2(R4) :'NED' SET ?
658 032016 001404          BEQ      3$          :BR IF NOT
659 032020 012762 100002 000016  MOV      #BIT15!BIT01,16(R2) :SET 'DRIVE NON-EXISTENT' INDICATOR
660 032026 000403          BR       4$          :CONTINUE
661 032030 012762 102000 000016  3$:   MOV      #BIT15!BIT10,16(R2) :SET 'NON-CLEARABLE PARITY' ERROR INDICATOR
662          JSR      PC,SVRH70 :SAVE RH/RM REGISTERS
663 032036 012763 177777 027442  4$:   MOV      #-1,TIMER(R3) :STOP THE TIMER
664 032044 105061 027340          CLRB     DRVACT(R1) :SET 'DRIVE ACTIVE' TO IDLE
665 032050 105061 027400          CLRB     DPRQS(R1) :CLEAR PORT REQUEST FLAG
666 032054 020137 027462          CMP      R1,DTUW    :IS THIS DRIVE SETUP FOR A TRANSFER
667 032060 001005          BNE     5$          :BR IF NOT
668 032062 012737 177777 027462  MOV      #-1,DTUW    :RESET THE INDICATOR
669 032070 005037 027410          CLR     TRNSWT      :CLEAR THE TRANSFER QUEUE
670 032074 105061 027416          5$:   CLRB     ULDFLG(R1) :CLEAR UNLOAD FLAG
671 032100 032764 010000 000010  BIT      #BIT12,RMCS2(R4) :'NED' SET ?
672          BNE     6$          :BR IF YES
673 032106 005201          INC     R1          :MOVE TO THE NEXT DRIVE
674 032110 062703 000002          ADD     #2,R3
675 032114 042701 177770          BIC     #^C7,R1
676 032120 001314          BNE     1$          :BR IF MORE DRIVES
677 032122 012737 177777 027462  MOV      #-1,DTUW    :NO DATA TRANSFERS UNDERWAY
678 032130 005037 027410          CLR     TRNSWT      :CLEAR THE 'TRANSFER WAIT' QUEUE
679 032134 004737 035402          JSR     PC,CLRQUE   :CLEAR ALL OF THE REQUEST QUEUES
680 032140 012764 005000 000010  MOV      #CLR,RMCS2(R4) :DO A MASSBUS INIT.
681 032146 000406          BR      7$          :CONTINUE
682 032150 004737 035460          6$:   JSR     PC,EMPTYQ   :CLEAR THE DRIVE'S QUEUE
683 032154 105061 027350          CLRB     DRVSTA(R1) :SET DRIVE TO OFFLINE
684 032160 105061 027360          CLRB     DRVTP(R1) :CLEAR THE DRIVE TYPE INDICATOR
685 032164 004737 035036          7$:   JSR     PC,SET.IE   :SET 'IE' WITHOUT 'TRE'
686 032170 104413          RESREG :RESTORE R0 - R5
687 032172 000207          RTS     PC          :RETURN
688
689          :LOOK AHEAD ROUTINE
690
691          :CALL
692
693          :
694          :
695          :
696          :
697          :
698          :
699 032174 013704 027476          LA:   MOV     RMADR,R4 :GET RMCS1'S ADDRESS
700 032200 010164 000010          MOV     R1,RMCS2(R4) :SELECT DRIVE
701 032204 004037 034346          JSR     R0,RD.RM    :READ DRIVE STATUS
702 032210 000012          RMDS   4$          :
703 032212 032342          4$     :ERROR RETURN ADDRESS
704 032214 042716 157577          BIC     #^C020200,(SP) :ON CYLINDER ?
705 032220 022726 000200          CMP     #200,(SP)+   :PIP=0,DRY=1?
706 032224 001044          BNE     3$          :NO
707 032226 105261 027426          INCB   LACNT(R1)    :INCREMENT THE LOOK AHEAD COUNT
708 032232 126137 027426 027504  CMPB   LACNT(R1),MXLACT :EXCEED MAX?
709 032240 003033          BGT     2$          :BR IF YES

```



```

767 032502 100002          BPL      1$          ;BR IF NO
768 032504 004737 034720   JSR      PC,SVRH70   ;YES--SAVE THE REGISTERS
772 032510          1$:          ;
774 032510 004737 035554   JSR      PC,GETREQ   ;GET DPB POINTER
775 032514 005702          TST      R2          ;ENTRY FOR DRIVE ?
776 032516 001403          BEQ      2$          ;BR IF NOT
777 032520 004737 030556   JSR      PC,OPT      ;CALL OPTIMIZER
778 032524 000417          BR       SC          ;CHECK OTHER DRIVES
779          ;THE RELEASE DRIVE COMMAND IS FORECD TO ENTER FOR DUAL PORT OPERATION
780 032526 012714 000113   2$:      MOV      #113,(R4) ;RELEASE THE DRIVE
781 032532 000414          BR       SC          ;CHECK FOR OTHER DRIVES
782 032534 052762 100100 000016 3$:      BIS      #BIT15!BIT06,16(R2) ;SET DATA ERROR FLAG
783 032542 004737 035460   JSR      PC,EMPTYQ   ;EMPTY THE 'DRIVE'S WAIT' QUEUE
784 032546 004737 034720   JSR      PC,SVRH70   ;SAVE THE RH/RM REGISTERS
785 032552 012714 040111   MOV      #40111,(R4) ;ISSUE A 'DRIVE CLEAR'
786 032556 012714 000113   MOV      #113,(R4)   ;ISSUE A RELEASE TO THE DRIVE
787 032562 000400          BR       SC          ;CHECK FOR OTHER DRIVES
788
789
806
807          ;SPECIAL CONDITION ROUTINE
808
809 032564 116403 000016   SC:      MOVB     RMAS(R4),R3 ;READ 'RMAS'
810 032570 001014          BNE     2$          ;BR IF ANY 'ATA' BITS SET
811 032572 004037 034346   JSR      R0,RD.RM    ;READ CONTROL AND STATUS REGISTER
812 032576 000000          RMCS1
813 032600 031736          CIB
814          ;
815 032602 106126          1$:      ROLB     (SP)+      ;EXIT IF FAIL TO READ
816 032604 100405          BMI     1$          ;IS 'IE'=1?
817 032606 004037 035644   JSR      R0,ES.SAV   ;YES, NO DRIVES TO CHECK
818          032612 104001          EMT     1           ;SAVE THE ADDRESS IN '$ESCAPE'
819          032614 004737 035036   JSR      PC,SET.IE  ;REPORT AN ILLEGAL INTERRUPT
820 032620 000207          1$:      RTS     PC          ;SET INTERRUPT ENABLE
821 032622 005046          2$:      CLR     -(SP)      ;RETURN
822 032624 110316          MOVB    R3,(SP)     ;PROCESS ALL DRIVES THAT HAVE
823 032626 012703 000001   MOV     #1,R3       ;AN 'ATA'=1
824 032632 005001          CLR     R1
825 032634 030316          SC3:    BIT     R3,(SP) ;ATA=1?
826 032636 001005          BNE     SC5         ;YES
827 032640 005201          SC4:    INC     R1     ;MOVE TO THE NEXT DRIVE
828 032642 106303          ASLB   R3
829 032644 001373          BNE     SC3         ;BR IF MORE TO CHECK?
830 032650 000207          TST    (SP)+       ;CLEAN OFF THE STACK
831 032652 105761 027370   SC5:    RTS     PC          ;RETURN TO USER
832 032656 001402          TSTB   DPINT(R1)   ;INITIALIZING THE DRIVE ?
833 032660 000137 033576   BEQ     1$          ;BR IF NOT
834 032664 105761 027400   JMP     SC13        ;PROCESS THE DRIVE
835 032670 001402          1$:      TSTB   DPRQS(R1) ;PORT REQUEST OUTSTANDING ?
836 032672 000137 033576   BEQ     2$          ;BR IF NOT
837 032676 105761 027350   JMP     SC13        ;START THE OUTSTANDING COMMAND
838 032702 003023          2$:      TSTB   DRVSTA(R1) ;CHECK THE DRIVE STATUS
839 032704 105761 027416   BGT     5$          ;BR IF ONLINE
840 032710 003420          TSTB   ULDFLG(R1) ;UNLOAD IN PROGRESS?
841 032712 004737 035554   BLE     5$          ;BR IF NOT
842 032716 004737 034720   JSR     PC,GETREQ   ;GET DPB POINTER
                        JSR     PC,SVRH70   ;SAVE THE RH/RM REGISTERS

```

```

843 032722 004737 033526      JSR      PC,SC12      ;SAVE RMDS, RMER1, RMER2, AND RMMR2
844                                ;ALSO DO A DRIVE INIT (DRVINT)
845 032726 105761 027350      TSTB     DRVSTA(R1)  ;DID DRIVE COME ONLINE?
846 032732 003414                                BLE      6$          ;NO
847 032734 032737 040000 027330  BIT      #BIT14,RMERRS ;WAS THERE AN ERROR?
848 032742 001000                                BNE      3$          ;BR IF ERROR
849                                JMP      SC11        ;NO ERROR
850 032744 013705 027332      3$:     MOV      RMERRS+2,R5 ;YES -- PICKUP RMER1 AND
851 032750 000504                                BR       SC6A        ;GO PROCESS THE ERROR
852 032752 105761 027340      5$:     TSTB     DRVACT(R1) ;DRIVE ACTIVE WITH COMMAND OR ERROR RECOVERY ?
853 032756 001033                                BNE      SC6         ;BR IF EITHER
854 032760 004737 033526      JSR      PC,SC12      ;SAVE RMDS, RMER1, RMER2, AND RMMR2
855                                ;ALSO DO A DRVINT
856 032764 105761 027370      6$:     TSTB     DPINT(R1) ;TRYING TO INIT THE DRIVE ?
857 032770 001323                                BNE      SC4         ;BR IF YES, CHECK ON MORE DRIVES
858 032772 105761 027350      TSTB     DRVSTA(R1)  ;CHECK ON DRIVE'S STATUS
859 032776 100412                                BMI      7$          ;BR IF UNSAFE
860 033000 032737 020000 027334  BIT      #BIT13,RMERRS+4 ;ADDRESS PLUG CHANGED ?
861 033006 001013                                BNE      8$          ;BR IF YES
862                                ;
863 033010 012746 000111      MOV      #113,-(SP)   ;RELEASE COMMAND
864 033014 004037 034526      MOV      #111,-(SP)   ;DRIVE CLEAR
865 033020 000000                                JSR      RO,WRT.RM   ;WRITE THE COMMAND INTO RMCS1
866 033022 033366                                RMCS1
867 033024 011605                                SC8
868 033026 004037 035644      7$:     MOV      (SP),R5    ;PARITY EXIT ADDRESS
869 033032 104002                                JSR      RO,ES.SAV   ;PICKUP (RMAS) BEFORE THE ERROR CALL
870 033034 000701                                EMT      2           ;SAVE THE ADDRESS IN '$ESCAPE'
871 033036 004037 035644      8$:     BR       SC4      ;REPORT THE UNEXPECTED ATTENTION
872                                ;GO CHECK FOR MORE ATA'S
873 033042 104005                                JSR      RO,ES.SAV   ;SAVE THE ADDRESS IN '$ESCAPE'
874 033044 000675                                EMT      5           ;REPORT THE ADDRESS PLUG CHANGE
875 033046 006301                                BR       SC4         ;CHECK FOR MORE DRIVES
876 033050 012761 177777 027442  5C6:    ASL      R1          ;SETUP TO ADDRESS WORDS
877 033056 006201                                MOV      #-1,TIMER(R1) ;STOP THE TIMER
878 033060 004737 035554      ASR      R1          ;RESTORE THE DRIVE ADDRESS
879 033064 010164 000010      JSR      PC,GETREQ   ;GET THE DPB POINTER FROM THE QUEUE
880 033070 000137 033416      MOV      R1,RMCS2(R4) ;SELECT DRIVE
881 033074 004037 034346      JMP      SC11        ;PROCESS THE SEARCH
882 033100 000012      JSR      RO,RD.RM   ;READ THE RM'S STATUS REG.
883 033102 033366      RMDS
884 033104 011605      SC8
885 033106 006126      MOV      (SP),R5    ;AND PUT IT IN R5
886 033110 100407      ROL      (SP)+      ;WAS THERE AN ERROR?
887 033112 105761 027340      BMI      1$          ;BR IF ERROR
888 033116 003137      TSTB     DRVACT(R1)  ;CHECK DRIVE'S STATE
889 033120 052762 100210 000016  BGT      SC11        ;BR IF DRIVE ACTIVE WITH ORDER
890 033126 000470      BIS      #BIT15!BIT07!BIT03,16(R2) ;INFORM USER OF ERROR RECOVER COMPLETION
891 033130 004037 034346      1$:     JSR      RO,RD.RM   ;READ ERROR REGISTER #1
892 033134 000014      RMER1
893 033136 033366      SC8
894 033140 012605      MOV      (SP)+,R5   ;AND SAVE IT IN R5
895 033142 004737 034720      JSR      PC,SVRH70  ;SAVE RH/RM REGISTERS
896 033146 012746 000111      MOV      #111,-(SP) ;ISSUE A DRIVE CLEAR
897 033152 004037 034526      JSR      RO,WRT.RM
898 033156 000000      RMCS1
899 033160 033366      SC8
  
```

```

897 033162 006105          SC6A:  ROL    R5          ;WAS 'UNSAFE' CONDITION =1?
898 033164 100406          BMI    1$          ;BR IF YES
899 033166 005702          TST    R2          ;ANYTHING IN QUEUE ?
900 033170 001447          BEQ    SC7          ;BR IF NOT
901 033172 052762 100240 000016  BIS    #BIT15!BIT07!BIT05,16(R2) ;INFORM USER OF ERROR
902 033200 000443          BR     SC7
903 033202 004037 034346 1$:  JSR    R0,RD.RM    ;READ DRIVE STATUS REG. #1
904 033206 000012          RMDS
905 033210 033366          SC8
906 033212 011605          MOV    (SP),R5     ;SAVE RMDS IN R5
907 033214 006126          ROL    (SP)+       ;'ERR'=1?
908 033216 100011          BPL    2$          ;BR IF NO--UNSAFE CLEARED
909 033220 112761 177777 027350  MOVB  #-1,DRVSTA(R1) ;DRIVE IS UNSAFE
910 033226 004737 034720  JSR    PC,SVRH70   ;SAVE RH/RM REGISTERS
911 033232 052762 110000 000016  BIS    #BIT15!BIT12,16(R2) ;INFORM USER OF UNSAFE ERROR
912 033240 000423          BR     SC7
913 033242 032705 010000 2$:  BIT    #BIT12,R5   ;'MOL' = 1 ?
914 033246 001015          BNE    3$          ;BR IF YES
915 033250 112761 177777 027340  MOVB  #-1,DRVACT(R1) ;ACTIVE ERROR RECOVER
916 033256 112761 000001 027350  MOVB  #1,DRVSTA(R1) ;ONLINE
917 033264 006301          ASL    R1
918 033266 012761 072460 027442  MOV    #30000.,TIMER(R1) ;START 30 SECOND TIMER
919 033274 006201          ASR    R1
920 033276 000137 032640  JSR    PC,CI7
921 033302 052762 100220 000016 3$:  BIS    #BIT15!BIT07!BIT04,16(R2) ;INFORM USER OF ERROR
922 033310 105061 027340  SC7:  CLR    DRVACT(R1) ;DRIVE IS IDLE
923          ;          JSR    PC,EMPTYQ   ;DUMP THE QUEUE
924 033314 004737 035576  JSR    PC,POPQUE   ;REMOVE THE QUEUE
925 033320 105761 027416  TSTB  ULDFLG(R1)  ;UNLOAD IN PROGRESS OR QUEUE?
926 033324 003002          BGT    1$          ;BR IF NOT
927 033326 105061 027416  CLR    ULDFLG(R1) ;CLEAR UNLOAD FLAG
928 033332 116164 027464 000016 1$:  MOVB  ATABIT(R1),RMAS(R4) ;CLEAR ATTENTION BIT
929 033340 105761 027350  TSTB  DRVSTA(R1)  ;IS THE DRIVE UNSAFE ?
930 033344 100406          BMI    2$          ;BR IF IT IS
931          ;          MOV    #113,-(SP)  ;RELEASE COMMAND
932 033346 012746 000111  MOV    #111,-(SP)  ;DRIVE CLEAR COMMAND
933 033352 004037 034526  JSR    R0,WRT.RM   ;WRITE THE COMMAND INTO RPCS1
934 033356 000000          RMCS1
935 033360 033366          SC8
936 033362 000137 032640 2$:  JMP    SC4          ;CHECK FOR MORE DRIVES
937 033366 105761 027340  SC8:  TSTB  DRVACT(R1)  ;IS DRIVE IDLE?
938 033372 001405          BEQ    1$          ;YES
939 033374 004737 035554  JSR    PC,GETREQ   ;GET DPB POINTER
940 033400 004737 031636  JSR    PC,CI7      ;PROCESS THE PARITY ERROR
941 033404 000402          BR     2$          ;CONTINUE
942 033406          1$:
943          ;
944 033406 004737 031660  JSR    PC,CI7      ;PROCESS THE PARITY ERROR
945 033412 000137 032640  JSR    PC,CI7B     ;PROCESS THE UNCORRECTABLE PARITY ERROR
946 033416 105761 027416 2$:  JMP    SC4          ;CHECK MORE DRIVES
947 033422 003402          SC11: TSTB  ULDFLG(R1)  ;'UNLOAD IN PROGRESS'?
948 033424 105061 027416  BLE    1$          ;BR IF NO
949 033430 105061 027340  CLR    ULDFLG(R1) ;CLEAR UNLOAD FLAG
950 033434 136137 027464 027412 1$:  CLR    DRVACT(R1) ;SET DRIVE IDLE
951          BITB  ATABIT(R1),SRCHWT ;DOING A SEARCH OPERATION FOR
952          ;          ;AN I/O COMMAND?
953 033442 001012          BNE    2$          ;BR IF YES
954 033444 004737 035576  JSR    PC,POPQUE   ;REMOVE REQUEST FROM QUEUE
  
```

```

954 033450 052762 000200 000016 BIS #BIT07,16(R2) ;SET 'DONE' BIT
955 033456 005737 027436 TST SAVEFG ;SAVE THE REGISTERS?
956 033462 100002 BPL 2$ ;BR IF NO
957 033464 004737 034720 JSR PC,SVRH70 ;YES--SAVE ALL OF THE RH/RM REG'S
958 033470 116164 027464 000016 2$: MOVB ATABIT(R1),RMAS(R4) ;CLEAR ATTENTION BIT
959 033476 146137 027464 027412 BICB ATABIT(R1),SRCHWT ;CLEAR IMPLIED SEEK SET
960 033504 006301 ASL R1 ;WORD INDEX
961 033506 012761 177777 027442 MOV #-1,TIMER(R1) ;STOP CLOCK
962 033514 006201 ASR R1 ;RESTORE R1
963 033516 004737 030556 JSR PC,OPT ;START A REQUEST
964 033522 000137 032640 JMP SC4 ;CHECK FOR MORE DRIVES
965 033526 010164 000010 SC12: MOV R1,RMCS2(R4) ;SELECT DRIVE
966 033532 016437 000012 027330 MOV RMD5(R4),RMERRS ;SAVE THE FOUR REGISTERS THAT
967 033540 016437 000014 027332 MOV RMER1(R4),RMERRS+2 ;WILL TELL US SOMETHING
968 033546 016437 000042 027334 MOV RMER2(R4),RMERRS+4
969 033554 016437 000040 027336 MOV RMMR2(R4),RMERRS+6
970 033562 004037 027726 JSR RO,DRVINT ;INIT. THE STATE OF THE DRIVE
971 033566 000401 BK 1$ ;TAKE ERROR EXIT
972 033570 000207 RTS PC ;RETURN
973 033572 005726 1$: TST (SP)+ ;POP PC OFF OF THE STACK
974 033574 000674 BR SC8 ;PROCESS THE PARITY ERROR
975 033576 006301 SC13: ASL R1 ;SETUP TO ADDRESS WORDS
976 033600 012761 177777 027442 MOV #-1,TIMER(R1) ;STOP THE TIMER
977 033606 006201 ASR R1 ;
978 033610 010164 000010 MOV R1,RMCS2(R4) ;SELECT THE DRIVE
979 033614 116164 027464 000016 MOVB ATABIT(R1),RMAS(R4) ;CLEAR THE ATTENTION BIT
980 033622 105761 027370 1$: TSTB DPINT(R1) ;INITIALIZING THE DRIVE ?
981 033626 001424 BEQ 2$ ;BR IF NOT
982 033630 105061 027370 CLR B DPINT(R1) ;CLEAR THE INIT INDICATOR
983 033634 004037 027726 JSR RO,DRVINT ;GO INIT THE DRIVE
984 033640 000240 NOP ;DUMMY PARITY ERROR RETURN
985 033642 105761 027350 TSTB DRVSTA(R1) ;DRIVE ONLINE ?
986 033646 003014 BGT 2$ ;BR IF YES -- START ORDER
987 033650 005702 TST R2 ;QUEUE ENTRY FOR THE DRIVE
988 033652 001426 BEQ 3$ ;BR IF NOT
989 033654 004737 035554 JSR PC,GETREQ ;GET DPB ADDRESS
990 033660 052762 140000 000016 BIS #BIT15!BIT14,16(R2) ;INFORM USER THAT DRIVE OFFLINE
991 033666 004737 034720 JSR PC,SVRH70 ;SAVE THE REGISTERS
992 JSR PC,EMPTYQ ;EMPTY THE REQUEST QUEUE
993 033672 004737 035576 JSR PC,POPQUE ;REMOVE THE QUEUE
994 033676 000414 BR 3$
995 033700 032764 004000 000000 2$: BIT #BIT11,RMCS1(R4) ;DVA SET ?
996 033706 001006 BNE 4$ ;SET THEN CALL OPT
997 033710 006301 ASL R1
998 033712 012761 060000 027442 MOV #60000,TIMER(R1)
999 033720 006201 ASR R1
1000 033722 000402 BR 3$
1001 033724 004737 030556 4$: JSR PC,OPT ;START THE PENDING REQUEST
1002 033730 000137 032640 3$: JMP SC4 ;PROCESS OTHER DRIVES
1003
1004 ;RM TIMER ROUTINE
1005 ;CALL
1006 ;
1007 ; MOV #TIME,-(SP) ;ELAPSED TIME IN MILLISECONDS ON THE STACK
1008 ; JSR PC,RMTMR ;CALL RM05 TIME ROUTINE
1009 033734 005737 027414 RMTMR: TST ACTDRV ;CHECK 'ACTDRV & ACTSTR'
1010 033740 001027 BNE 4$ ;IF NON ZERO EXIT

```

```

1011 033742 112737 000001 027415      MOVB    #1,ACTSTR      ;SET 'ACTSTR'
1012 033750 104412                    SAVREG                    ;SAVE R0 - R5
1013 033752 005001                    CLR     R1                ;START WITH DRIVE 0
1014 033754 005003                    CLR     R3
1015 033756 005763 027442      1$:    TST     TIMER(R3)    ;IS THE TIMER RUNNING?
1016 033762 002406                    BLT     2$                ;BR IF NO
1017 033764 166663 000002 027442      SUB     2(SP),TIMER(R3) ;COUNT THE INTERVAL
1018 033772 003002                    BGT     2$                ;BR IF NO SOFTWARE TIMEOUT
1019 033774 004737 034024      JSR     PC,STO           ;CALL SOFTWARE TIMEOUT ROUTINE
1020 034000 005201 2$:    INC     R1                ;MOVE TO NEXT DRIVE
1021 034002 005723                    TST     (R3)+
1022 034004 022701 000010      CMP     #8.,R1          ;OUT OF DRIVES?
1023 034010 003362                    BGT     1$                ;BR IF NO
1024 034012 104413 3$:    RESREG                    ;RESTORE R0 - R5
1025 034014 105037 027415      CLRB   ACTSTR          ;ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
1026 034020 012616 4$:    MOV     (SP)+,(SP)     ;ADJUST THE STACK
1027 034022 000207      RTS     PC                ;RETURN
1028
1029      ;SOFTWARE TIMEOUT ROUTINE
1030
1031      ;NOTE: THIS ROUTINE MUST BE ENTERED AT PRIORITY 6
1032      ;OR GREATER
1033
1034      ;CALL:
1035      ;STO
1036      ;MOV     #DRVNUM,R1      ;DRIVE NUMBER
1037      ;JSR     PC,STO         ;CALL
1038      ;RETURN
1039 034024 010146  STO:   MOV     R1,-(SP)    ;SAVE R1
1040 034026 010246      MOV     R2,-(SP)    ;SAVE R2
1041 034030 010346      MOV     R3,-(SP)    ;SAVE R3
1042 034032 010446      MOV     R4,-(SP)    ;SAVE R4
1043 034034 013704 027476      MOV     RMADR,R4    ;GET ADDRESS OF 'RMCS1'
1044 034040 010164 000010      MOV     R1,RMCS2(R4) ;SELECT THE DRIVE
1045 034044 004037 034346      JSR     R0,RD.RM    ;READ 'DRIVE STATUS REG'
1046 034050 000012      RMDS
1047      ;
1048 034052 034334      STOS
1049 034054 105726      STO9
1050 034056 100436      TSTB   (SP)+        ;IS 'DRY'=1?
1051 034060 105761 027370      BMI    STO2         ;BR IF YES
1052 034064 001033  STO1:  TSTB   DPINT(R1)   ;TRYING TO INTIALIZE THE DRIVE ?
1053 034066 105761 027400      BNE    STO2         ;BR IF YES
1054 034072 001030      TSTB   DPRQS(R1)   ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
1055 034074 013702 027410      BNE    STO2         ;BR IF YES
1056 034100 020137 027462      MOV     TRNSWT,R2   ;PICKUP TRANSFER WAIT QUEUE
1057 034104 001404      CMP     R1,DTUW     ;TRANSFER UNDERWAY ON THIS DRIVE?
1058 034106 000137 034334      BEQ    1$          ;BR IF YES
1059 034112 004737 035554      JMP    STO9         ;IF NOT DON'T BOTHER DRIVES
1060 034116 052762 101000 000016 1$:    JSR     PC,GETREQ    ;GET DPB ADDRESS
1061 034124 004737 034720      BIS    #BIT15!BIT09,16(R2) ;SET THE ERROR FLAGS
1062      ;
1063 034130 105061 027340      JSR     PC,SVRH70   ;SAVE RH/RM REGISTERS
1064 034134 105061 027416      MOV     #CLR,RMCS2(R4) ;'INIT' THE MASS BUS
1065 034140 005037 027410      CLRB   DRVACT(R1)  ;DRIVE IS IDLE
1066 034144 012737 177777 027462      CLRB   ULDFLG(R1)  ;CLEAR THE UNLOAD FLAG
1067 034152 000470      CLR    TRNSWT      ;CLEAR DPB ADDRESS
                                MOV     #-1,DTUW     ;CLEAR THE TRANSFER DRIVE #
                                BR     STO9         ;DON'T BOTHER OTHER DRIVES

```

```

1068 034154 116405 000016      ST02:  MOV B  RMAS(R4),R5      ;READ ATTENTION REG
1069 034160 136105 027464      BIT B  ATABIT(R1),R5      ;IS ATTENTION FOR THIS DRIVE UP ?
1070 034164 001007              BNE    ST03              ;YES
1071 034166 105761 027370      TST B  DPINT(R1)        ;TRYING TO INTIALIZE THE DRIVE ?
1072 034172 001021              BNE    ST06              ;BR IF YES - DRIVE NOT ONLINE
1073 034174 105761 027400      TST B  DPRQS(R1)        ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
1074 034200 001035              BNE    ST07              ;BR IF YES - NO RESPONSE TO REQUEST
1075 034202 000454              BR     ST09              ;OTHER WISE EXIT
1076 034204 105761 027370      ST03:  TST B  DPINT(R1)        ;INITIALIZING THE DRIVE ?
1077 034210 001003              BNE    1$                ;BR IF INIT PENDING
1078 034212 105761 027400      TST B  DPRQS(R1)        ;PORT REQUEST PENDING ?
1079 034216 001446              BEQ    ST09              ;BR IF NOT
1080 034220 012763 177777 027442 1$:  MOV    #-1,TIMER(R3)      ;STOP THE TIMER
1081 034226 000442              BR     ST09              ;EXIT
1082 034230 004737 031736      ST05:  JSR    PC,C18        ;GO HANDLE THE PARITY ERROR
1083 034234 000437              BR     ST09
1084 034236 105061 027370      ST06:  CLRB  DPINT(R1)        ;CLEAR THE INITIALIZE INDICATOR
1085 034242 105061 027350      CLRB  DRVSTA(R1)        ;SET UNIT OFFLINE
1086 034246 012763 177777 027442  MOV    #-1,TIMER(R3)      ;STOP THE TIMER
1087 034254 004737 035554      JSR    PC,GETREQ        ;GET THE DPB ADDRESS
1088 034260 005702              TST   R2                ;REQUEST IN QUEUE ?
1089 034262 001424              BEQ    ST09              ;BR IF NOT
1090 034264 052762 140000 000016  BIS    #BIT15!BIT14,16(R2) ;INFORM THE USER DRIVE NOT AVAILABLE
1091 034272 000414              BR     ST08              ;FINISH
1092 034274 012763 177777 027442 ST07:  MOV    #-1,TIMER(R3)      ;STOP THE TIMER
1093 034302 105061 027400      CLRB  DPRQS(R1)        ;CLEAR PORT REQUEST INDICATOR
1094 034306 004737 035554      JSR    PC,GETREQ        ;GET DPB ADDRESS
1095 034312 005702              TST   R2                ;QUEUE ENTRY FOR DRIVE ?
1096 034314 001407              BEQ    ST09              ;BR IF NONE
1097 034316 012762 100004 000016  MOV    #BIT15!BIT2,16(R2) ;INFORM USER OF PORT REQUEST ERROR
1098 034324 004737 035460      ST08:  JSR    PC,EMPTYQ     ;CLEAR THE QUEUE FOR THE DRIVE
1099 034330 004737 034720      JSR    PC,SVRH70        ;SAVE THE REGISTERS
1100 034334 012604      ST09:  MOV    (SP)+,R4        ;RESTORE R4
1101 034336 012603              MOV    (SP)+,R3        ;RESTORE R3
1102 034340 012602              MOV    (SP)+,R2        ;RESTORE R2
1103 034342 012601              MOV    (SP)+,R1        ;RESTORE R1
1104 034344 000207              RTS    PC                ;RETURN
1105
1106      ;ROUTINE TO READ A RH/RM REGISTER
1107
1108      ;CALL
1109      ;
1110      JSR    R0,RD.RM      ;GO READ A REGISTER
1111      INDEX  ;REG. INDEX FROM BASE
1112      ERRADR ;ERROR ADDRESS--PROCESS ERROR STARTING
1113      ;AT THIS ADDRESS
1114      ;CONTENTS OF REG. IS ON THE STACK
1115 034346 013737 027474 034514 RD.RM:  MOV    MCPEMX,RD.RM2     ;MAX. RETRYS ALLOWED
1116 034354 011646              MOV    (SP),-(SP)      ;SAVE R0 FOR RETURN
1117 034356 013737 027476 034372  MOV    RMADR,RD.ADR     ;FORM THE DESIRED ADDRESS
1118 034364 062037 034372      ADD    (R0)+,RD.ADR    ;USING THE BASE AND THE INDEX
1119 034370 013727      RD.RM1: MOV    @(PC)+,(PC)+ ;READ THE DESIRED REGISTER OF THE RM DRIVE
1120 034372 000000      RD.ADR: .WORD 0        ;ADDRESS IS FORMED HERE
1121 034374 000000      RD.WRD: .WORD 0        ;REG. CONTENTS PUT HERE
1122 034376 013766 034374 000002  MOV    RD.WRD,2(SP)     ;RETURN IT TO THE USER
1123 034404 013746 027476      MOV    RMADR,-(SP)     ;PUT THE ADDRESS ON THE STACK
1124 034410 062716 000010      ADD    #RMCS2,(SP)     ;FORM THE ADDRESS OF RMCS2
  
```

```

1125 034414 032736 010000          BIT    #BIT12,@(SP)+    ;CHECK THE 'NED' BIT
1126 034420 001037          BNE    RD.RM3          ;BR IF DRIVE NON-EXISTENT
1127 034422 017746 173050          MOV    @RMADR,-(SP)    ;READ RMCS1
1128 034426 032716 020000          BIT    #BIT13,(SP)    ;DID MCPE SET?
1129 034432 001002          BNE    1$             ;BR IF YES
1130 034434 022620          CMP    (SP)+,(RO)+    ;ADJUST FOR RETURN
1131 034436 000432          BR     RD.RM4         ;EXIT
1132 034440          1$:
    034440 004037 035644          JSR    RO,ES.SAV      ;SAVE THE ADDRESS IN '$ESCAPE'
    034444 104003          EMT    3              ;REPORT 'MCPE' ERROR
1133 034446 005737 027462          TST    DTUW          ;DATA TRANSFER UNDERWAY?
1134 034452 100405          BMI    2$            ;NO
1135 034454 032716 040000          BIT    #BIT14,(SP)    ;'TRE' = 1 ?
1136 034460 001402          BEQ    2$            ;NO
1137 034462 005726          TST    (SP)+         ;YES--CLEAN OFF THE STACK AND
1138 034464 000415          BR     RD.RM3         ;TAKE THE FATAL ERROR EXIT
1139 034466 052716 040000          2$: BIS    #BIT14,(SP) ;CLEAR 'MCPE' BY SENDING A '1' TO 'TRE'
1140 034472 000316          SWAB   (SP)          ;POSITION BEFORE WRITING
1141 034474 013737 027476 034510  MOV    RMADR,3$      ;FORM ADDRESS OF HIGH BYTE
1142 034502 005237 034510          INC    3$
1143 034506 112637          MOVB   (SP)+,@(PC)+  ;WRITE THE HIGH BYTE OF RMCS1
1144 034510 000000          3$: .WORD 0          ;ADDRESS STORAGE
1145 034512 005327          DEC    (PC)+         ;EXCEEDED MAX. RETRYS
1146 034514 000003          RD.RM2: .WORD 3
1147 034516 002324          BGE    RD.RM1        ;BR IF NO
1148 034520 011000          RD.RM3: MOV    (RO),RO ;FATAL ERROR EXIT
1149 034522 012616          MOV    (SP)+,(SP)
1150 034524 000200          RD.RM4: RTS    RO
1151
1152          ;ROUTINE TO WRITE A REGISTER
1153          ;
1154          ;CALL
1155          ;
1156          ;   MOV    DATA,-(SP)    ;DATA TO BE LOADED ON THE STACK
1157          ;   JSR    RO,WRT.RM    ;CALL THE ROUTINE TO LOAD(WRITE) THE REG.
1158          ;   INDEX          ;INDEX OF THE REGISTER TO BE LOADED
1159          ;   ERRADR        ;ADDRESS TO RETURN TO ON AN ERROR
1160          ;   RETURN       ;ERROR FREE RETURN
1161 034526 013737 027474 034704  WRT.RM: MOV    MCPEMX,WRT.R2 ;MAX RETRYS ALLOWED
1162 034534 016637 000002 034614  MOV    2(SP),WRT.WD    ;SAVE THE WORD TO WRITE
1163 034542 012616          MOV    (SP)+,(SP)    ;ADJUST THE STACK
1164 034544 012037 034616          MOV    (RO)+,WRT.AD   ;GET INDEX OF REGISTER TO BE WRITTEN
1165 034550 001015          BNE    1$             ;BR IF NOT RMCS1
1166 034552 122737 000150 034614  CMPB   #150,WRT.WD    ;IS THE COMMAND FOR DATA TRANSFERS?
1167 034560 002411          BLT    1$             ;YES--DON'T GET THE OLD A16 & A17, & PSEL
1168 034562 004037 034346          JSR    RO,RD.RM      ;NO---COMBINE A16&A17, & PSEL WITH
1169 034566 000000          RMCS1                ;THE COMMAND BEFORE SENDING IT TO
1170 034570 034710          WRT.R3                ;THE RH/RM
1171 034572 000316          SWAB   (SP)
1172 034574 042716 177770          BIC    #^C7,(SP)
1173 034600 112637 034615          MOVB   (SP)+,WRT.WD+1
1174 034604 063737 027476 034616  1$: ADD    RMADR,WRT.AD   ;FORM THE ADDRESS OF THE DISK REG.
1175 034612 012737          WRT.R1: MOV    (PC)+,@(PC)+ ;LOAD THE DESIRED REG.
1176 034614 000000          WRT.WD: .WORD 0      ;WORD TO WRITE GOES HERE
1177 034616 000000          WRT.AD: .WORD 0      ;ADDRESS IS FORMED HERE
1178 034620 013746 027476          MOV    RMADR,-(SP)   ;PUT THE ADDRESS ON THE STACK
1179 034624 062716 000010          ADD    #RMCS2,(SP)   ;FORM THE ADDRESS OF RMCS2
  
```

```

1180 034630 032736 010000          BIT    #BIT12,@(SP)+    ;CHECK THE 'NED' BIT
1181 034634 001025          BNE    WRT.R3          ;BR IF DRIVE NON-EXISTENT
1182 034636 004037 034346          JSR    RO,RD.RM       ;CHECK FOR PARITY ERROR ON WRITE
1183 034642 000014          RMER1
1184 034644 034710          WRT.R3
1185 034646 032726 000010          BIT    #BIT03,(SP)+
1186 034652 001420          BEQ    WRT.R4          ;BR IF 'PAR=0'
1187 034654 016037 177776 034666          MOV    -2(RO),1$      ;PICKUP THE INDEX
1188 034662 004037 034346          JSR    RO,RD.RM       ;READ THE REG.
1189 034666 000000          1$: .WORD 0           ;REG. INDEX
1190 034670 034710          WRT.R3
1191 034672 004037 035644          JSR    RO,ES.SAV      ;RETURN TO THIS ADDRESS ON ERROR
1191 034676 104004          EMT    4              ;SAVE THE ADDRESS IN '$ESCAPE'
1192 034700 005726          TST    (SP)+          ;REPORT THE PARITY ON WRITE ERROR
1193 034702 005327          DEC    (PC)+          ;CLEAR OFF THE STACK
1194 034704 000003          WRT.R2: .WORD 3      ;DECREMENT THE ERROR COUNT
1195 034706 002341          BGE    WRT.R1         ;RETRY COUNTER
1196 034710 011000          WRT.R3: MOV (RO),RO   ;TRY AGAIN IF NOT FINISHED
1197 034712 000401          BR     WRT.R5         ;TAKE THE 'PARITY ON WRITE' ERROR EXIT
1198 034714 005720          WRT.R4: TST (RO)+    ;EXIT
1199 034716 000200          WRT.R5: RTS    RO     ;ADJUST FOR ERROR FREE EXIT

1200
1201          ;ROUTINE TO SAVE THE RH/RM REGISTERS AS PER DPB+14
1202          ;CALL
1203          ;
1204          ;   MOV    #DPBNUM,R2    ;DPB POINTER TO R2
1205          ;   JSR    PC,SVRH70   ;SAVE THE DRIVES REG'S
1206
1207 034720 104412          SVRH70: SAVREG       ;SAVE R0 - R5
1208 034722 005702          TST    R2             ;QUEUE ENTRY FOR THE DRIVE ?
1209 034724 001442          BEQ    6$             ;BR IF NONE
1210 034726 013704 027476          MOV    RMADR,R4
1211 034732 111264 000010          MOV    (R2),RMCS2(R4) ;SELECT DRIVE
1212 034736 016203 000014          MOV    14(R2),R3     ;GET THE ERROR TABLE POINTER
1213 034742 001433          BEQ    6$             ;EXIT IF NO ADDRESS
1214 034744 005037 035000          CLR    3$            ;COUNTER & POINTER
1215 034750 023727 035000 000022 1$: CMP    3$,#RMDB       ;REACHED THE BUFFER REGISTER ?
1216 034756 001006          BNE    2$            ;BR IF NOT
1217 034760 032764 000200 000010          BIT    #BIT07,RMCS2(R4) ;'OR' SET ?
1218 034766 001002          BNE    2$            ;BR IF SET
1219 034770 005023          CLR    (R3)+         ;STORE RMDB AS ZEROES
1220 034772 000405          BR     4$            ;CONTINUE
1221 034774 004037 034346          2$: JSR    RO,RD.RM    ;READ THE SELECTED REGISTER
1222 035000 000000          3$: .WORD 0           ;REGISTER INDEX
1223 035002 035026          5$          ;ERROR RETURN ADDRESS
1224 035004 012623          MOV    (SP)+,(R3)+   ;STORE THE REGISTER CONTENTS
1225 035006 023727 035000 000046 4$: CMP    3$,#RMEC2    ;REACHED THE END ?
1226 035014 001406          BEQ    6$            ;BR IF YES
1227 035016 062737 000002 035000          ADD    #2,3$         ;INCREMENT THE REGISTER INDEX
1228 035024 000751          BR     1$            ;CONTINUE READING THE REGISTERS
1229 035026 004737 031636          5$: JSR    PC,C17      ;PROCESS THE UNCORRECTABLE PARITY ERROR
1230 035032 104413          6$: RESREG
1231 035034 000207          RTS    PC            ;RESTORE R0 - R5
1232
1233          ;ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A 'TRE'
1234          ;CALL
1235          ;   MOV    #DRVNUM,R1    ;DRIVE NUMBER '

```



```

1236      :      JSR      PC,SET.IE      ;SET 'IE'
1237      :      RETURN
1238
1239 035036 010446      SET.IE: MOV      R4,-(SP)      ;SAVE R4
1240 035040 013704 027476  MOV      RMADR,R4      ;PICKUP ADDRESS OF RMCS1
1241 035044 010164 000010  MOV      R1,RMCS2(R4)      ;SELECT DRIVE
1242 035050 011446      MOV      (R4),-(SP)      ;READ RMCS1
1243 035052 052716 040000  BIS      #BIT14,(SP)      ;SET THE 'TRE' BIT OF THE WORD READ
1244 035056 000316      SWAB      (SP)      ;ADJUST FOR DATO
1245 035060 112714 000100  MOVB     #BIT06,(R4)      ;SET 'IE'
1246 035064 032764 010000 000010  BIT      #BIT12,RMCS2(R4) ;IS 'NED'=1?
1247 035072 001002      BNE      1$      ;YES--CLEAR 'TRE'
1248 035074 005726      TST      (SP)+      ;CLEAN OFF THE STACK
1249 035076 000402      BR       2$
1250 035100 112664 000001  1$: MOVB  (SP)+,1(R4)      ;CLEAR 'TRE'
1251 035104 012604      2$: MOV      (SP)+,R4      ;RESTORE R4
1252 035106 000207      RTS      PC      ;RETURN TO CALLER
1253
1254      ;QUEUE COUNT
1255 035110      000      QCNT: .BYTE 0      ;DRIVE 0
1256 035111      000      .BYTE 0      ;DRIVE 1
1257 035112      000      .BYTE 0      ;DRIVE 2
1258 035113      000      .BYTE 0      ;DRIVE 3
1259 035114      000      .BYTE 0      ;DRIVE 4
1260 035115      000      .BYTE 0      ;DRIVE 5
1261 035116      000      .BYTE 0      ;DRIVE 6
1262 035117      000      .BYTE 0      ;DRIVE 7
1263
1264      ;QUEUE INPUT POINTERS
1265
1266 035120 035202      QINPT: .WORD  QDRV0      ;DRIVE 0
1267 035122 035222      .WORD  QDRV1      ;DRIVE 1
1268 035124 035242      .WORD  QDRV2      ;DRIVE 2
1269 035126 035262      .WORD  QDRV3      ;DRIVE 3
1270 035130 035302      .WORD  QDRV4      ;DRIVE 4
1271 035132 035322      .WORD  QDRV5      ;DRIVE 5
1272 035134 035342      .WORD  QDRV6      ;DRIVE 6
1273 035136 035362      .WORD  QDRV7      ;DRIVE 7
1274
1275      ;QUEUE OUTPUT POINTERS
1276
1277 035140 035202      QOUTPT: .WORD  QDRV0      ;DRIVE 0
1278 035142 035222      .WORD  QDRV1      ;DRIVE 1
1279 035144 035242      .WORD  QDRV2      ;DRIVE 2
1280 035146 035262      .WORD  QDRV3      ;DRIVE 3
1281 035150 035302      .WORD  QDRV4      ;DRIVE 4
1282 035152 035322      .WORD  QDRV5      ;DRIVE 5
1283 035154 035342      .WORD  QDRV6      ;DRIVE 6
1284 035156 035362      .WORD  QDRV7      ;DRIVE 7
1285
1286 035160 035202      QSTART: .WORD  QDRV0      ;DRIVE 0 START ADDRESS
1287 035162 035222      QSTOP:  .WORD  QDRV1      ;DRIVE 0 STOP ADDRESS & DRIVE 1 START ADDRESS
1288 035164 035242      .WORD  QDRV2      ;STOP DRIVE 1--START DRIVE 2
1289 035166 035262      .WORD  QDRV3      ;STOP DRIVE 2--START DRIVE 3
1290 035170 035302      .WORD  QDRV4      ;STOP DRIVE 3--START DRIVE 4
1291 035172 035322      .WORD  QDRV5      ;STOP DRIVE 4--START DRIVE 5
1292 035174 035342      .WORD  QDRV6      ;STOP DRIVE 5--START DRIVE 6
  
```

```

1293 035176 035362          .WORD  QDRV7          ;STOP DRIVE 6--START DRIVE 7
1294 035200 035402          .WORD  QTERM           ;STOP DRIVE 7
1295
1296          ;DRIVE REQUEST QUEUES
1297
1298 035202  QDRV0:  .BLKW  10
1299 035222  QDRV1:  .BLKW  10
1300 035242  QDRV2:  .BLKW  10
1301 035262  QDRV3:  .BLKW  10
1302 035302  QDRV4:  .BLKW  10
1303 035322  QDRV5:  .BLKW  10
1304 035342  QDRV6:  .BLKW  10
1305 035362  QDRV7:  .BLKW  10
1306          QTERM=.
1307          035402
1308
1309          ;ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
1310          ;
1311          ;CALL
1312          ;      JSR      PC,CLRQUE
1313 035402 104412 CLRQUE: SAVREG          ;SAVE R0 - R5
1314 035404 012702 035110      MOV      #QCNT,R2      ;ZERO THE QUEUE COUNTS
1315 035410 005022          CLR      (R2)+        ;DRIVES 0 & 1
1316 035412 005022          CLR      (R2)+        ;DRIVES 2 & 3
1317 035414 005022          CLR      (R2)+        ;DRIVES 4 & 5
1318 035416 005022          CLR      (R2)+        ;DRIVES 6 & 7
1319 035420 012703 000010      MOV      #8.,R3        ;MOVE THE STARTING
1320 035424 012701 035160      MOV      #QSTART,R1    ;ADDRESS OF THE QUEUE INTO
1321 035430 012122          1$:  MOV      (R1)+,(R2)+    ;THE QUEUE INPUT POINTER
1322 035432 005303          DEC      R3
1323 035434 001375          BNE     1$
1324 035436 012703 000010      MOV      #8.,R3        ;MOVE THE STARTING ADDRESS
1325 035442 012701 035160      MOV      #QSTART,R1    ;OF THE QUEUE INTO THE
1326 035446 012122          2$:  MOV      (R1)+,(R2)+    ;QUEUE OUTPUT POINTER
1327 035450 005303          DEC      R3
1328 035452 001375          BNE     2$
1329 035454 104413          RESREG
1330 035456 000207          RTS      PC          ;RESTORE R0 - R5
1331
1332          ;EMPTY THE QUEUE SPECIFIED BY R1
1333          ;
1334          ;CALL
1335          ;      MOV      DRVNUM,R1      ;DRIVE NUMBER TO R1
1336          ;      JSR      PC,EMPTYQ
1337
1338 035460 105061 035110 EMPTYQ: CLRB   QCNT(R1)      ;CLEAR NUMBER OF ITEMS IN QUEUE
1339 035464 006301          ASL      R1
1340 035466 016161 035120 035140      MOV      QINPT(R1),QOUTPT(R1) ;SET OUTPUT QUEUE POINTER=INPUT POINTER
1341 035474 006201          ASR      R1
1342 035476 000207          RTS      PC
1343
1344          ;ROUTINE TO PUT A REQUEST IN QUEUE
1345          ;
1346          ;CALL
1347          ;      MOV      #DRVNUM,R1      ;DRIVE NUMBER
1348          ;      MOV      #DPB,R2        ;ADDRESS OF PARAMETER BLOCK
1349          ;      JSR      R0,DRVQUE     ;GO PUT REQUEST IN QUEUE

```

```

1350      :      RETURN1      ;RETURN HERE IF QUEUE IS FULL
1351      :      RETURN2      ;RETURN HERE IF REQUEST IS IN QUEUE
1352
1353 035500 122761 000010 035110 DRVQUE: CMPB   #10,QCNT(R1)  ;IS QUEUE FULL?
1354 035506 001421          BEQ     2$          ;BR IF YES-TAKE RETURN1
1355 035510 105261 035110          INCB   QCNT(R1)    ;INCREMENT QUEUE COUNT
1356 035514 006301          ASL     R1
1357 035516 010271 035120          MOV    R2,@QINPT(R1) ;PUT THIS REQUEST IN QUEUE
1358 035522 062761 000002 035120          ADD    #2,QINPT(R1) ;UPDATE THE QUEUE POINTER
1359 035530 026161 035120 035162          CMP    QINPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER
1360 035536 001003          BNE    1$          ;BR IF NO
1361 035540 016161 035160 035120          MOV    QSTART(R1),QINPT(R1) ;YES--RESET POINTER
1362 035546 006201          1$: ASR    R1
1363 035550 005720          TST   (R0)+        ;TAKE RETURN 2
1364 035552 000200          2$: RTS    R0          ;RETURN TO USER
1365
1366      ;ROUTINE TO GET THE 'DPB' ADDRESS OF NEXT REQUEST IN QUEUE
1367      ;CALL
1368      ;
1369      ;      MOV    #DRVNUM,R1      ;DRIVE NUMBER TO R1
1370      ;      JSR    PC,GETREQ      ;GO GET THE REQUEST
1371      ;      RETURN                   ;R2='DPB' ADDRESS OF THE REQUEST
1372      ;                               ;R2=0 IF NO REQUEST IN QUEUE
1373
1374 035554 005002          GETREQ: CLR    R2
1375 035556 105761 035110          TSTB  QCNT(R1)    ;IS THERE ANY REQUEST IN QUEUE?
1376 035562 001404          BEQ    2$          ;NO
1377 035564 006301          1$: ASL    R1
1378 035566 017102 035140          MOV    @QOUTPT(R1),R2 ;PICKUP 'DPB' POINTER FOR THIS DRIVE
1379 035572 006201          ASR    R1
1380 035574 000207          2$: RTS    PC          ;RETURN TO USER
1381
1382      ;ROUTINE TO 'POP' THE REQUEST FROM QUEUE
1383      ;CALL
1384      ;
1385      ;      MOV    #DRVNUM,R1      ;DRIVE NUMBER TO R1
1386      ;      JSR    PC,POPQUE      ;CALL TO REMOVE REQUEST
1387      ;      RETURN                   ;R2=ADDRESS OF DPB REMOVED
1388
1389 035576 105361 035110          POPQUE: DECB  QCNT(R1) ;DECREMENT QUEUE COUNT
1390 035602 006301          ASL    R1
1391 035604 017102 035140          MOV    @QOUTPT(R1),R2 ;GET THE 'DPB' POINTER
1392 035610 005071 035140          CLR    @QOUTPT(R1) ;REMOVE DPB ADDRESS FROM THE QUEUE
1393 035614 062761 000002 035140          ADD    #2,QOUTPT(R1) ;UPDATE THE QUEUE POINTER
1394 035622 026161 035140 035162          CMP    QOUTPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER?
1395 035630 001003          BNE    1$          ;NO--BR TO EXIT
1396 035632 016161 035160 035140          MOV    QSTART(R1),QOUTPT(R1) ;YES--RESET THE POINTER
1397 035640 006201          1$: ASR    R1
1398 035642 000207          RTS    PC          ;RETURN TO USER
1399
1401      ;ROUTINE TO SAVE THE CONTENTS OF '$ESCAPE' WHEN THE DRIVER
1402      ;REPORTS AN ERROR DIRECTLY.
1403      ;CALL
1404      ;
1405      ;      JSR    R0,ES.SAV      ;:THE ERROR CALL
1406      ;      ERROR  N              ;:THE RETURN IS PAST THE ERROR CALL
1407      ;      RETURN

```

```

1408
1409 035644 012037 035660      ES.SAV: MOV      (R0)+,1$      ;GET THE ERROR CALL
1410 035650 013746 001200      MOV      $ESCAPE,-(SP)      ;SAVE THE ADDRESS IN '$ESCAPE'
1411 035654 005037 001200      CLR      $ESCAPE            ;CLEAR THE ESCAPE RETURN
1412 035660 000000 000000      1$:      .WORD    0            ;THE ERROR CALL IS MOVED HERE
1413 035662 012637 001200      MOV      (SP)+,$ESCAPE      ;RESTORE THE ESCAPE ADDRESS
1414 035666 000200 000200      RTS      R0                  ;RETURN
1416
1417
1418      .SBTTL  BUSADR - GET BUS ADDRESS AND VECTOR ADDRESS
1419
1420      ;THIS ROUTINE IS USED TO INSURE THE BUS ADDRESS
1421      ;OF THE RH/RM IS SETUP FOR THE PROPER ADDRESS.
1422      ;IT WILL ALSO READ THE ADDRESS FROM THE TTY IF
1423      ;REQUIRED.
1424      ;NOTE: THIS ROUTINE DESTROYS R0-R4
1425      ;CALL
1426      :
1427      :      JSR      PC,BUSADR
1428      :      RETURN
1429 035670 104412 000000      BUSADR: SAVREG      ;SAVE ALL REG
1430 035672 012700 001276      1$:      MOV      #SRMADR,R0      ;FIRST ADDRESS
1431 035676 104401 036040      TYPE    ,MRMCS1           ;'RMCS1='
1432 035702 011046 000000      MOV      (R0),-(SP)        ;PRESENT RMCS1 ADDRESS
1433 035704 104402 000000      TYPOC   ;TYPE IT
1434 035706 104401 042330      TYPE    ,BLNKS1           ;TYPE 1 BLANK
1435 035712 104411 000000      RDLIN   ;GET THE ENTRY
1436 035714 012601 000000      MOV      (SP)+,R1          ;ADDRESS OF ASCII TEXT
1437 035716 004537 036056      JSR      R5,CK.NUM        ;ENTER AND STORE THE NEW ADDRESS
1438 035722 000763 000000      BR      1$                ;ERROR EXIT
1439 035724 012700 001300      2$:      MOV      #SRMVEC,R0      ;VECTOR ADDRESS
1440 035730 104401 036047      TYPE    ,MRMVEC           ;'RMVEC='
1441 035734 011046 000000      MOV      (R0),-(SP)        ;PRESENT RH/RM VECTOR ADDRESS ON THE STACK
1442 035736 104402 000000      TYPOC   ;TYPE IT
1443 035740 104401 042330      TYPE    ,BLNKS1           ;TYPE 1 BLANK
1444 035744 104411 000000      RDLIN   ;READ THE ENTRY
1445 035746 012601 000000      MOV      (SP)+,R1          ;ASCII TEXT ADDRESS
1446 035750 004537 036056      JSR      R5,CK.NUM        ;ENTER AND STORE NEW ADDRESS
1447 035754 000763 000000      BR      2$                ;ERROR EXIT
1448 035756 012700 001276      3$:      MOV      #SRMADR,R0      ;FIRST ADDRESS OF NEW PARAMETERS
1449 035762 012701 027476      MOV      #RMADR,R1        ;FIRST ADDRESS OF WHERE TO PUT THEM
1450 035766 013705 000004      MOV      @#ERRVEC,R5      ;SAVE ERROR VECTOR
1451 035772 012737 036016 000004      MOV      #4$,@#ERRVEC     ;LOAD NEW VECTOR ADDRESS
1452 036000 005777 143272      TST     @SRMADR           ;LEGAL I/O ADDRESS ?
1453 036004 010537 000004      MOV      R5,@#ERRVEC      ;YES, IF NOT TRAP TO TIMEOUT
1454 036010 012021 000000      MOV      (R0)+,(R1)+      ;LOAD THE BUS ADDRESS
1455 036012 012021 000000      MOV      (R0)+,(R1)+      ;LOAD VECTOR ADDRESS
1456 036014 000407 000000      BR      6$                ;COMMON EXIT
1457 036016 012716 036024      4$:      MOV      #5$,(SP)        ;SET RETURN ADDRESS
1458 036022 000002 000000      RTI
1459 036024 104006 000000      5$:      EMT      6
1460 036026 010537 000004      MOV      R5,@#ERRVEC      ;RESTORE THE TIME OUT TRAP
1461 036032 000717 000000      BR      1$                ;TRY AGAIN
1462 036034 104413 000000      6$:      RESREG
1463 036036 000207 000000      RTS      PC
1464
  
```

```

1465 036040 122 115 103 MRMCS1: .ASCIZ @RMCS1=@
1466 036047 122 115 126 MRMVEC: .ASCIZ @RMVEC=@
1467
1468 .SBTTL CK.NUM - CHECK NUMBER (OCTAL)
1469 :THIS ROUTINE CHECKS AN ASCIZ STRING FOR LEGAL CHARACTERS
1470 :AND FORMS AN OCTAL NUMBER IN R2
1471 :CALL:
1472 :      MOV      #ADR,R1      ;ADDRESS OF ASCIZ STRING
1473 :      JSR      R5,CK.NUM    ;R5 CHANGED
1474 :      RET      ;ERROR EXIT
1475 :      RET      ;NORMAL EXIT
1476 036056 010246 CK.NUM: MOV      R2,-(SP)    ;SAVE R2
1477 036060 010346      MOV      R3,-(SP)    ;SAVE R3
1478 036062 010446      MOV      R4,-(SP)    ;SAVE R4
1479 036064 012703      MOV      #6,R3      ;MAX OCTAL DIGITS IN THE NUMBER
1480 036070 005002      CLR      R2      ;FINAL OCTAL VALUE
1481 036072 112104      1$:  MOVB     (R1)+,R4    ;GET CURRENT POINTED BYTE
1482 036074 001424      BEQ      3$      ;BRANCH,IF TERMINATOR DETECTED
1483 036076 120427 000060      CMPB     R4,#'0    ;SMALLER THAN ASCII-0 ?
1484 036102 103425      BLO      5$      ;YES,ERROR EXIT
1485 036104 120427 000067      CMPB     R4,#'7    ;LARGER THAN ASCII-7 ?
1486 036110 101022      BHI      5$      ;YES,ERROR EXIT
1487 036112 006302      ASL      R2      ;SHIFT LEFT
1488 036114 103420      BCS      5$      ;
1489 036116 006302      ASL      R2      ;ONE
1490 036120 103416      BCS      5$      ;
1491 036122 006302      ASL      R2      ;OCTAL DIGIT
1492 036124 103414      BCS      5$      ;ERROR IF CARRY BIT SET
1493 036126 042704 177770      BIC      #177770,R4 ;CHOP OFF HIGHER BITS
1494 036132 060402      ADD      R4,R2    ;APPENDING CURRENT DIGIT TO NUMBER
1495 036134 005303      DEC      R3      ;DECREMENT BYTE COUNT
1496 036136 001401      BEQ      2$      ;BRANCH,IF LAST BYTE
1497 036140 000754      BR       1$      ;LOOPING BACK
1498 036142 112104      2$:  MOVB     (R1)+,R4    ;CHECK TERMINATOR
1499 036144 001004      BNE      5$      ;ERROR EXIT
1500 036146 005702      3$:  TST      R2      ;FINAL VALUE = 0
1501 036150 001402      BEQ      5$      ;YES, TAKE ERROR EXIT
1502 036152 010210      MOV      R2,(R0)  ;REPLACE THE ORIGINAL VALUE
1503 036154 005725      4$:  TST      (R5)+    ;ADJUST FOR NORMAL RETURN
1504 036156 012604      5$:  MOV      (SP)+,R4  ;RESTORE R4
1505 036160 012603      MOV      (SP)+,R3  ;RESTORE R3
1506 036162 012602      MOV      (SP)+,R2  ;RESTORE R2
1507 036164 000205      RTS      R5      ;EXIT
1508

```

```

1          .SBTTL  ERROR MESSAGES
2
3 036166   122   110   040  EM1:   .ASCIZ  /RH INTERRUPT OCCURRED (RMAS = 0)/
4 036227   125   116   105  EM2:   .ASCIZ  /UNEXPECTED ATTENTION OCCURRED/
5 036265   115   101   123  EM3:   .ASCIZ  /MASSBUS PARITY ERROR (MCPE=1)/
6 036323   115   101   123  EM4:   .ASCIZ  /MASSBUS PARITY ERROR (PAR=1)/
7 036360   101   104   104  EM5:   .ASCIZ  /ADDRESS PLUG CHANGE BIT SET/
8 036414   122   110   040  EM6:   .ASCIZ  /RH DIDN'T RESPOND TO ADDRESSING/
9 036454   125   116   103  EM10:  .ASCIZ  /UNCORRECTABLE MASSBUS PARITY ERROR/
10 036517  106   101   124  EM11:  .ASCIZ  /FATAL MASSBUS PARITY ERROR/
11 036552  120   105   122  EM12:  .ASCIZ  /PERSISTENT DEVI'E UNSAFE/
12 036603  117   120   105  EM13:  .ASCIZ  /OPERATION NOT COMPLETED WITHIN TIME LIMIT/
13 036655  104   122   111  EM14:  .ASCIZ  /DRIVE WENT OFFLINE/
14 036700  116   117   040  EM15:  .ASCIZ  /NO RESPONSE TO PORT REQUEST/
15 036734  110   105   101  EM20:  .ASCIZ  /HEADER CRC ERROR/
16 036755  104   101   124  EM21:  .ASCIZ  /DATA CHECK ('DCK') ERROR/
17 037006  127   122   111  EM22:  .ASCIZ  /WRITE CHECK ERROR - DATA CHECK ('DCK') SET/
18 037061  127   122   111  EM23:  .ASCIZ  /WRITE CHECK ERROR - DATA CHECK ('DCK') NOT SET/
19 037140  110   105   101  EM24:  .ASCIZ  /HEADER READ ERROR - 'FMT' BIT DROPPED/
20 037206  110   105   101  EM25:  .ASCIZ  /HEADER READ ERROR - HEADER COMPARE ('HCE') ERROR/
21 037267  106   117   122  EM26:  .ASCIZ  /FORMAT ERROR ('FER')/
22 037314  110   105   101  EM27:  .ASCIZ  /HEADER COMPARE ('HCE') ERROR/
23 037351  115   111   123  EM30:  .ASCIZ  /MISCELLANEOUS DRIVE ERROR/
24 037403  117   120   105  EM31:  .ASCIZ  /OPERATION INCOMPLETE ('OPI') ERROR/
25 037446  104   122   111  EM32:  .ASCIZ  /DRIVE TIMING ('DTE') ERROR/
26 037501  120   101   122  EM33:  .ASCIZ  /PARITY ('PAR') ERROR AFTER OPERATION STARTED/
27 037556  127   122   111  EM34:  .ASCIZ  /WRITE CLOCK FAILURE ('WCF') ERROR/
28 037620  111   116   126  EM35:  .ASCIZ  /INVALID ADDRESS ('IAE') ERROR/
29 037656  127   122   111  EM36:  .ASCIZ  /WRITE LOCK ('WLE') ERROR/
30 037707  104   101   124  EM37:  .ASCIZ  /DATA CHECK ('DCK') SET DURING WRITE CHECK COMMAND/
31 037771  122   110   040  EM40:  .ASCIZ  /RH OR UNIBUS TRANSFER ERROR/
32 040025  102   125   123  EM41:  .ASCIZ  /BUS ADDRESS OR WORD COUNT INCORRECT/
33 040071  104   101   124  EM42:  .ASCIZ  /DATA COMPARE ERRORS - NO OTHER ERROR(S) DETECTED/
34 040152  103   101   116  EM43:  .ASCIZ  /CAN'T MATCH DATA READ WITH A PATTERN/
35 040217  105   122   122  EM44:  .ASCIZ  /ERROR BIT(S) SET, BUT NO ERROR SIGNALLED BY THE RH/
36 040301  105   103   103  EM45:  .ASCIZ  /ECC LOGIC FAILURE - POSITION REGISTER VALUE NOT VALID/
37 040367  102   125   123  EM46:  .ASCIZ  /BUS ADDRESS AND WORD COUNT NOT CONSISTENT/
38 040441  123   105   105  EM50:  .ASCIZ  /SEEK INCOMPLETE ('SKI') ERROR/
39 040477  120   122   117  EM51:  .ASCIZ  /PROGRAM DETECTED POSITIONING ERROR/
40 040542  104   122   111  EM60:  .ASCIZ  /DRIVE UNSAFE ERROR/
41 040565  040   040   000  EM61:  .ASCIZ  / /
42
43          .EVEN
44 040570   122   115   101  DH1:   .ASCIZ  /RMAS /
45 040577   104   122   111  DH2:   .ASCIZ  /DRIVE  RMDS  RMER1  RMER2  RMMR2  RMAS  /
46 040661   104   122   111  DH3:   .ASCIZ  /DRIVE  REG ADR  DATA  /
47 040712   104   122   111  DH4:   .ASCIZ  /DRIVE  REG ADR  GOOD  BAD  /
48 040754   122   115   101  DH6:   .ASCIZ  /RMADR /
49 040765   104   122   111  DH7:   .ASCIZ  /DRIVE  RMCS1  RMWC  RMBA  RMDA  /
50 041036   122   115   103  DH10:  .ASCIZ  /RMCS2  RMDS  RMER1  RMAS  RMDB  /
51 041107   122   115   115  DH11:  .ASCIZ  /RMMR1  RMDT  RMOF  RMDC  RMMR2  /
52 041160   122   115   105  DH12:  .ASCIZ  /RMER2  RMEC1  RMEC2  /
53          .EVEN
54
55 041212  001326  000000  DT1:   .WORD  ATTN,0
56 041216  001224  027330  027332  DT2:   .WORD  DRIVE,RMERRS,RMERRS+2,RMERRS+4,RMERRS+6,ATTN,0
57 041234  001224  034372  034374  DT3:   .WORD  DRIVE,RD.ADR,RD.WRD,0
    
```

58	041244	001224	034616	034614	DT4:	.WORD	DRIVE,WRT.AD,WRT.WD,RD.WRD,0
59	041256	027476	000000		DT6:	.WORD	RMADR,0
60	041262	001224	004526	004530	DT7:	.WORD	DRIVE,RM.REG,RM.REG+2,RM.REG+4,RM.REG+6,0
61	041276	004536	004540	004542	DT10:	.WORD	RM.REG+10,RM.REG+12,RM.REG+14,RM.REG+16,RM.REG+22,0
62	041312	004552	004554	004560	DT11:	.WORD	RM.REG+24,RM.REG+26,RM.REG+32,RM.REG+34,RM.REG+40,0
63	041326	004570	004572	004574	DT12:	.WORD	RM.REG+42,RM.REG+44,RM.REG+46,0
64							
65	041336	000			DF1:	.BYTE	0
66	041337	000	000	000	DF2:	.BYTE	0,0,0,0,0,0
67	041345	000	000	000	DF3:	.BYTE	0,0,0
68	041350	000	000	000	DF4:	.BYTE	0,0,0,0
69						.EVEN	
70							
71	041354	123	125	102	MESG1:	.ASCIZ	/SUBSYS /
72	041364	104	122	111	MESG2:	.ASCIZ	/DRIVE(S): /
73	041377	052	052	040	MESG3:	.ASCIZ	/** STARTING PASS 1 /
74	041423	200	115	117	MESG4:	.ASCIZ	<CRLF>/MOUNT PACK ON DRIVE /
75	041451	101	116	104	MESG8:	.ASCIZ	/AND LOAD. /
76	041464	124	131	120	MESG9:	.ASCIZ	/TYPE <CR> WHEN DRIVE READY./
77	041520	200	104	122	MESG10:	.ASCIZ	<CRLF>/DRIVE IS NOT READY, TEST IS ABORTED/
78	041565	200	120	101	MESG11:	.ASCIZ	<CRLF>/PACK IS NOT ACCEPTABLE, CHANGE PACK AND TRY AGAIN/
79	041650	125	116	114	MESG12:	.ASCIZ	/UNLOAD DRIVE /
80	041666	040	101	116	MESG13:	.ASCII	/ AND REMOVE PACK/<CRLF>
81	041707	124	131	120		.ASCIZ	/TYPE <CR> WHEN DONE./
82	041734	052	052	040	MESG14:	.ASCIZ	/** STARTING PASS 2 /
83	041760	052	052	040	MESG15:	.ASCIZ	/** ALL DRIVES ARE COMPATIBLE **/<CRLF>
84	042021	123	103	117	MESG16:	.ASCIZ	/SCORES FOR DRIVE /
85	042043	040	040	040	MESG17:	.ASCIZ	/ TRACK DRIVE OVRWRT OVRWRT READ READ/
86	042124	040	040	040	MESG18:	.ASCIZ	/ NUMBR READ OFST- OFST+ OFST- OFST+ /
87	042206	123	105	114	SELFX:	.ASCIZ	/SELF/
88	042213	011	000		TAB:	.ASCIZ	<HT>
89	042215	052	040	060	MARKX:	.ASCIZ	/* 0/<HT>
90	042222	042	000		QUOTM:	.ASCIZ	/'/'
91	042224	054	040	000	COMMA:	.ASCIZ	/, /
92	042227	127	111	114	MESG5:	.ASCIZ	/WILL TEST /
93	042242	117	116	040	MESG6:	.ASCIZ	/ON /
94	042246	200	111	123	MESG7:	.ASCIZ	<CRLF>/IS THERE ANOTHER SUB-SYSTEM (Y OR N) ? /
95	042317	131	200	000	Y:	.ASCIZ	/Y/<CRLF>
96	042322	116	200	000	N:	.ASCIZ	/N/<CRLF>
97	042325	040			BLNKS4:	.ASCII	//
98	042326	040			BLNKS3:	.ASCII	//
99	042327	040			BLNKS2:	.ASCII	//
100	042330	040	000		BLNKS1:	.ASCIZ	//
101	042332	072	000		COLON:	.ASCIZ	/:/
102	042334	040	057	040	SLASH:	.ASCIZ	@ / @
103	042340	077	000		QUES:	.ASCIZ	/?/
104	042342	040	102	101	MESG19:	.ASCIZ	/ BAD SPOT FILE /
105	042362	124	105	123	MESG20:	.ASCIZ	/TEST COMPLETE/
106	042400	200	052	052	STARS:	.ASCIZ	<CRLF>/*****/<CRLF>
107	042422	104	122	111	MESG21:	.ASCII	/DRIVE NOT ON-LINE OR NOT ASSIGNED/<CRLF>
111	042464	007	120	122	HALTX:	.ASCIZ	<07>/PROGRAM HALT/<CRLF>
112	042503	104	122	111	HALT1:	.ASCIZ	/DRIVE FAILED ON BASIC READ & WRITE TEST/<CRLF>
113	042554	104	122	111	HALT2:	.ASCIZ	/DRIVE FAILED ON WRITE TEST (TEST 4)/<CRLF>
114	042621	106	101	124	XFATL:	.ASCIZ	/FATAL ERROR/
115	042635	200	040	047	NEDCLK:	.ASCIZ	<CRLF>/ 'L' OR 'P' CLOCK REQUIRED ON SYSTEM/<CRLF>
116	042704	040	077	104	QDRIV:	.ASCIZ	/ ?DRIVE/
117	042714	040	111	123	LODEV:	.ASCIZ	/ IS LOAD DEVICE/

```
118 042734      040      077      i03 NOTST: .ASCIZ @ ?CANNOT SELECT RM03/2'S AND RM05'S TOGETHER(NOT COMPATIBLE)@<CRLF>
119
120              .EVEN
121
122 043032      BUFFER:
123
124              .END      200
```


ABASE = 176700	AUSWR = 000000	CK.CHR 022364	DRIV6 004776	EM44 040217
ACDW1 = 000000	AVECT1= 000254	CK.DEC 022336	DRIV7 005020	EM45 040301
ACDW2 = 000000	AVECT2= 000000	CK.DIG 022436	DRVACT 027340	EM46 040367
ACK = 000123	BADSEC 001342	CK.NUM 036056	DRVCLR= 000111	EM5 036360
ACPUOP= 000000	BIT0 = 000001	CK.OCT 022310	DRVER 020250	EM50 040441
ACTDRV 027414	BIT00 = 000001	CLKFLG 001316	DRVINT 027726	EM51 040477
ACTSTR 027415	BIT01 = 000002	CLOCK 021426	DRVQUE 035500	EM6 036414
ADDW0 = 000000	BIT02 = 000004	CLRDPB 021620	DRVSTA 027350	EM60 040542
ADDW1 = 000000	BIT03 = 000010	CLRQUE 035402	DRVSTYP 027360	EM61 040565
ADDW10= 000000	BIT04 = 000020	CMCNT 001354	DSWR = 177570	ENDX1 010704
ADDW11= 000000	BIT05 = 000040	CMCYL 001356	DTEER 020320	ERPRC1 017462
ADDW12= 000000	BIT06 = 000100	CMSEC 001362	DTUW 027462	ERPROC 017446
ADDW13= 000000	BIT07 = 000200	CMTRK 001364	DTYP 001402	ERROR = 104000
ADDW14= 000000	BIT08 = 000400	COLON 042332	DT1 041212	ERRVEC= 000004
ADDW15= 000000	BIT09 = 001000	COMMA 042224	DT10 041276	ES.SAV 035644
ADDW2 = 000000	BIT1 = 000002	CR = 000015	DT11 041312	FALPAR 017564
ADDW3 = 000000	BIT10 = 002000	CRLF = 000200	DT12 041326	FAULT 001376
ADDW4 = 000000	BIT11 = 004000	CYLIMT 001374	DT2 041216	FILBUF 020742
ADDW5 = 000000	BIT12 = 010000	CYLNR 004616	DT3 041234	FINISH 023356
ADDW6 = 000000	BIT13 = 020000	DCKER 020200	DT4 041244	FMTDPB 004506
ADDW7 = 000000	BIT14 = 040000	DDISP = 177570	DT6 041256	FMTER 020360
ADDW8 = 000000	BIT15 = 100000	DEASG 016770	DT7 041262	GENDPB 004576
ADDW9 = 000000	BIT2 = 000004	DF1 041336	DULP 027730	GETREG= 000141
ADEVCT= 000000	BIT3 = 000010	DF2 041337	DUMP 020440	GETREQ 035554
ADEVM = 000000	BIT4 = 000020	DF3 041345	DUMP2 017620	GTSWR = 104406
AENV = 000000	BIT5 = 000040	DF4 041350	EMPTYQ 035460	HALTX 042464
AENVM = 000000	BIT6 = 000100	DH1 040570	EMTVEC= 000030	HALT1 042503
AFATAL= 000000	BIT7 = 000200	DH10 041036	EM1 036166	HALT2 042554
AMADR1= 000000	BIT8 = 000400	DH11 041107	EM10 036454	HCEER 020370
AMADR2= 000000	BIT9 = 001000	DH12 041160	EM11 036517	HCR CER 020240
AMADR3= 000000	BLKADR 004444	DH2 040577	EM12 036552	HOUR 001344
AMADR4= 000000	BLNKS1 042330	DH3 040661	EM13 036603	HT = 000011
AMAMS1= 000000	BLNKS2 042327	DH4 040712	EM14 036655	HZ 001320
AMAMS2= 000000	BLNKS3 042326	DH6 040754	EM15 036700	IAEER 020340
AMAMS3= 000000	BLNKS4 042325	DH7 040765	EM2 036227	IDLEX 017344
AMAMS4= 000000	BPTVEC= 000014	DISMNT 016716	EM20 036734	INDST 004424
AMSGAD= 000000	BUFFER 043032	DISPLA 001156	EM21 036755	IOTVEC= 000020
AMSGLG= 000000	BUSADR 035670	DISPLY= 104414	EM22 037006	ISR 032350
AMSGTY= 000000	CFLAG 001336	DISPRE 000174	EM23 037061	KPATH 016430
AMTYP1= 000000	CHGADR 001334	DONE 017644	EM24 037140	KSR 021534
AMTYP2= 000000	CI1 031014	DPINT 027370	EM25 037206	LA 032174
AMTYP3= 000000	CI3 031122	DPRQS 027400	EM26 037267	LABAD 014734
AMTYP4= 000000	CI4 031222	DRIVE = 001224	EM27 037314	LACNT 027426
APASS = 000000	CI5 031600	DRIVO 004622	EM3 036265	LF = 000012
APRIOR= 000000	CI6 031622	DRIV0 004644	EM30 037351	LINDEC 021114
APTCSU= 000040	CI7 031636	DRIV1 005042	EM31 037403	LINE1 021016
APTENV= 000001	CI7B 031660	DRIV10 005064	EM32 037446	LINKDV 021664
APTSIZ= 000200	CI8 031736	DRIV11 005106	EM33 037501	LINOCT 021062
APTSPO= 000100	CKBUS 020540	DRIV12 005130	EM34 037556	LODEV 042714
ASNLST 002006	CKCLK 021134	DRIV13 005152	EM35 037620	LOGO 001406
ASSGN1 002010	CKCLK1 021230	DRIV14 005174	EM36 037656	LOG1 001426
ASSGN2 002012	CKCLK2 021276	DRIV15 005216	EM37 037707	LOG10 001646
ASWREG= 000000	CKCLK3 021320	DRIV16 005240	EM4 036323	LOG11 001666
ATABIT 027464	CKERR 020472	DRIV17 004666	EM40 037771	LOG12 001706
AATESTN= 000000	CKFMT 020260	DRIV2 004710	EM41 040025	LOG13 001726
ATTN 001326	CKHCE 020270	DRIV3 004732	EM42 040071	LOG14 001746
AUNIT = 000000	CKSWR = 104407	DRIV4 004754	EM43 040152	LOG15 001766

LOG2	001446	M.DP42	022034	POSER	020300	RDN9	003436	RMWC	= 000002
LOG3	001466	M.DP44	022066	PRINT	017052	RDP0	003676	RM.REG	004526
LOG4	001506	M.DP50	022100	PROCES	017352	RDP1	003716	RM05	030264
LOG5	001526	N	042322	PRTBAD	020602	RDP10	004136	RNOP	= 000101
LOG6	001546	NEDCLK	042635	PRTIM	017614	RDP11	004156	RSTART	001400
LOG7	001566	NOTST	042734	PR0	= 000000	RDP12	004176	RTC	= 000117
LOG8	001606	NULINE	001366	PR1	= 000040	RDP13	004216	R6	= %000006
LOG9	001626	OFFCOD	004504	PR2	= 000100	RDP14	004236	R7	= %000007
LOOP1	013530	OFFSET=	000115	PR3	= 000140	RDP15	004256	SAVEFG	027436
LOOP2	013546	OFFST	014144	PR4	= 000200	RDP16	004276	SAVREG=	104412
LOOP3	015616	OFLIN	017604	PR5	= 000240	RDP17	004316	SC	032564
LOOP4	016410	OPIER	020310	PR6	= 000300	RDP18	004336	SCOPE	= 000004
LOOP5	016622	OPT	030556	PR7	= 000340	RDP2	003736	SCORE	014514
LOOP6	016640	OVWNO	002056	PS	= 177776	RDP3	003756	SC11	033416
LSTAD	001332	OVWN1	002076	PSEUDO	005322	RDP4	003776	SC12	033526
MAKEUP	014376	OVWN10	002316	PSW	= 177776	RDP5	004016	SC13	033576
MARKX	042215	OVWN11	002336	PUNSAF	017544	RDP6	004036	SC3	032634
MCPEMX	027477	OVWN12	002356	PWRVEC=	000024	RDP7	004056	SC4	032640
MESG1	041354	OVWN13	002376	QCNT	035110	RDP8	004076	SC5	032652
MESG10	041520	OVWN14	002416	QDRIV	042704	RDP9	004116	SC6	033046
MESG11	041565	OVWN15	002436	QDRV0	035202	RD.ADR	034372	SC6A	033162
MESG12	041650	OVWN16	002456	QDRV1	035222	RD.RM	034346	SC7	033310
MESG13	041666	OVWN17	002476	QDRV2	035242	RD.RM1	034370	SC8	033366
MESG14	041734	OVWN18	002516	QDRV3	035262	RD.RM2	034514	SEARCH=	000131
MESG15	041760	OVWN2	002116	QDRV4	035302	RD.RM3	034520	SECLMT	001370
MESG16	042021	OVWN3	002136	QDRV5	035322	RD.RM4	034524	SECOND	001350
MESG17	042043	OVWN4	002156	QDRV6	035342	RD.WRD	034374	SEEK	= 000105
MESG18	042124	OVWN5	002176	QDRV7	035362	READIN=	000121	SEEKFG	027440
MESG19	042342	OVWN6	002216	QINPT	035120	RECAL	= 000107	SELDRV=	000145
MESG2	041364	OVWN7	002236	QOUTPT	035140	RELSE	= 000113	SELFX	042206
MESG20	042362	OVWN8	002256	QSTART	035160	REPLZ	022110	SELF0	004356
MESG21	042422	OVWN9	002276	QSTOP	035162	RESREG=	104413	SELF1	004360
MESG3	041377	OVWP0	002536	QTERM	= 035402	RESVEC=	000010	SELF10	004402
MESG4	041423	OVWP1	002556	QUES	042340	RMADR	027476	SELF11	004404
MESG5	042227	OVWP10	002776	QUOTM	042222	RMAS	= 000016	SELF12	004406
MESG6	042242	OVWP11	003016	RDCHR	= 104410	RMBA	= 000004	SELF13	004410
MESG7	042246	OVWP12	003036	RDDAT	= 000171	RMCS1	= 000000	SELF14	004412
MESG8	041451	OVWP13	003056	RDHD	= 000173	RMCS2	= 000010	SELF15	004414
MESG9	041464	OVWP14	003076	RDLIN	= 104411	RMDA	= 000006	SELF16	004416
MINUTE	001346	OVWP15	003116	RDN0	003216	RMDB	= 000022	SELF17	004420
MINDLTA	027510	OVWP16	003136	RDN1	003236	RMDC	= 000034	SELF18	004422
MOD00	007036	OVWP17	003156	RDN10	003456	RMDS	= 000012	SELF2	004362
MOD11	007344	OVWP18	003176	RDN11	003476	RMDT	= 000026	SELF3	004364
MOD12	007524	OVWP2	002576	RDN12	003516	RMEC1	= 000044	SELF4	004366
MOD21	007062	OVWP3	002616	RDN13	003536	RMEC2	= 000046	SELF5	004370
MOD22	007330	OVWP4	002636	RDN14	003556	RMERRS	027330	SELF6	004372
MOD23	007264	OVWP5	002656	RDN15	003576	RMER1	= 000014	SELF7	004374
MOD30	007356	OVWP6	002676	RDN16	003616	RMER2	= 000042	SELF8	004376
MONTR	007006	OVWP7	002716	RDN17	003636	RMHR	= 000036	SELF9	004400
MRMCS1	036040	OVWP8	002736	RDN18	003656	RMINIT	027514	SETFMT=	000143
MRMVEC	036047	OVWP9	002756	RDN2	003256	RMLA	= 000020	SETVEC	006774
MXDLTA	027506	PACK	001324	RDN3	003276	RMMR1	= 000024	SET.IE	035036
MXLACT	027504	PARER	020330	RDN4	003316	RMMR2	= 000040	SIXTEE	001352
MXWINDW	027512	PCLOCK	001314	RDN5	003336	RMOF	= 000032	SIZMEM	006730
M.DPID	021732	PIRQ	= 177772	RDN6	003356	RMFSN	= 000030	SKIER	020410
M.DP40	021770	PIRQVE=	000240	RDN7	003376	RMTMR	033734	SLASH	042334
M.DP41	022024	POPQUE	035576	RDN8	003416	RMVEC	027500	SPATH	016520

SPOTX	010536	TIMER	027442	\$BDDAT	001142	\$ILLUP	025072	\$RMCS1=	000020
SRCHWT	027412	TKVEC =	000060	\$BELL	001202	\$INTAG	001151	\$RMCS2=	000030
STACK =	001100	TPVEC =	000064	\$BUF =	000006	\$ITEMB	001130	\$RMDB =	000026
STARS	042400	TRAPVE=	000034	\$CDW1	001272	\$LF	001210	\$RMDC =	000042
STARSC	001360	TRFER	020400	\$CDW2	001274	\$LFLG	024715	\$RMDS =	000054
START	005502	TRKLMT	001372	\$CHARC	024446	\$LKCSB	001304	\$RMDS =	000032
START1	005520	TRNSWT	027410	\$CKSWR	025562	\$LKCSR	001302	\$RMDT =	000046
START2	005536	TRTVEC=	000014	\$CLR.T	023476	\$LKS	001310	\$RMEC1=	000064
START3	005550	TSTNM	001340	\$CMTAG	001114	\$LLVEC	001312	\$RMEC2=	000066
STATIN	001322	TST1	010720	\$CM3 =	000000	\$LOOP	023554	\$RMER1=	000034
STKLMT=	177774	TST10	015554	\$CM4 =	000001	\$LPADR	001122	\$RMER2=	000062
STNDAT	005262	TST11	016242	\$CNTLC	023352	\$LPERR	001124	\$RMHR =	000056
STO	034024	TST2	011274	\$CNTLG	026171	\$LPVEC	001306	\$RMLA =	000040
STO1	034060	TST3	011550	\$CNTLU	026164	\$LSTAD	027240	\$RMMR1=	000044
STO2	034154	TST4	012174	\$COMND=	000002	\$MADR1	001244	\$RMMR2=	000060
STO3	034204	TST5	012672	\$CPUOP	001240	\$MADR2	001250	\$RMOF =	000052
STO5	034230	TST6	013244	\$CRLF	001207	\$MADR3	001254	\$RMSN =	000050
STO6	034236	TST7	015040	\$CYL =	000012	\$MADR4	001260	\$RMVEC	001300
STO7	034274	TYPDS =	104405	\$DBLK	025552	\$MAIL	001212	\$RMWC =	000022
STO8	034324	TYPE =	104401	\$DB2D	026314	\$MAMS1	001242	\$RTNAD	023556
STO9	034334	TYPOC =	104402	\$DB20	026510	\$MAMS2	001246	\$RTRN	023552
SVRH70	034720	TYPON =	104404	\$DECVL	026474	\$MAMS3	001252	\$SAVRE	026220
SWR	001154	TYPOS =	104403	\$DEVCT	001222	\$MAMS4	001256	\$SAVR6	025076
SWREG	000176	TYPRI4	022240	\$DEVVM	001270	\$MBADR	001102	\$SB2D	022574
SWTIM	017574	UCPAR	017554	\$DOAGN	023516	\$MFLG	024714	\$SB20	022624
SW0 =	000001	ULDFLG	027416	\$DSPLY	022266	\$MNEW	026207	\$SCOPE	026630
SW00 =	000001	UNIT	001330	\$DTBL	025542	\$MSGAD	001226	\$SEC =	000010
SW01 =	000002	UNSAF	020430	\$EMTAB=	000022	\$MSGLG	001230	\$SETUP=	000137
SW02 =	000004	WCFER	020420	\$ENDAD	023506	\$MSGTY	001212	\$SIZE	027110
SW03 =	000010	WCKD =	000151	\$ENDCT	023454	\$MSWR	026176	\$STUP =	177777
SW04 =	000020	WCKER	020210	\$ENV	001232	\$MTYP1	001243	\$SUPRS	022200
SW05 =	000040	WCKHD =	000153	\$ENVM	001233	\$MTYP2	001247	\$SVLAD	027036
SW06 =	000100	WLEER	020350	\$EOP	023420	\$MTYP3	001253	\$SVPC =	000214
SW07 =	000200	WRTDAT=	000161	\$EOPCT	023446	\$MTYP4	001257	\$SWR =	177600
SW08 =	000400	WRTHD =	000163	\$ERFLG	001117	\$MXCNT	027106	\$SWREG	001234
SW09 =	001000	WRT.AD	034616	\$ERMAX	001131	\$NULL	001170	\$SWRMK=	000000
SW1 =	000002	WRT.RM	034526	\$ERROR	023562	\$NWTST=	000000	\$SYSNM=	000014
SW10 =	002000	WRT.R1	034612	\$ERRPC	001132	\$OCNT	025332	\$STATUS=	000016
SW11 =	004000	WRT.R2	034704	\$ERRTB	005362	\$OCTVL	026612	\$TBIT	023560
SW12 =	010000	WRT.R3	034710	\$ERRTY	023762	\$OMODE	025334	\$TERM =	000032
SW13 =	020000	WRT.R4	034714	\$ERTTL	001126	\$OVER	027072	\$TESTN	001216
SW14 =	040000	WRT.R5	034716	\$ESCAP	001200	\$PASS	001220	\$TIME	021330
SW15 =	100000	WRT.WD	034614	\$ETABL	001232	\$PASTM	001106	\$TIMES	001176
SW2 =	000004	XEND2	017046	\$ETEND	001276	\$PHYDR=	000015	\$TKB	001162
SW3 =	000010	XFATL	042621	\$FATAL	001214	\$POWER	025100	\$TKINT	022654
SW4 =	000020	XPASS1	007754	\$FFLG	024716	\$PSEL =	000003	\$TKS	001160
SW5 =	000040	XPASS2	012360	\$FILLC	001172	\$PWDRN	024720	\$TKSRV	022704
SW6 =	000100	XXDP	001404	\$FILLS	001171	\$PWRMG	025054	\$TMP0	001174
SW7 =	000200	Y	042317	\$FMT =	000001	\$PWRUP	024772	\$TN =	000012
SW8 =	000400	\$APTHD	001100	\$GAP =	000016	\$QUES	001206	\$TNPWR	026424
SW9 =	001000	\$ATYC	024476	\$GDADR	001134	\$RDCHR	026044	\$TPB	001166
SYSADR	002014	\$ATY1	024452	\$GDDAT	001140	\$RDLIN	023050	\$TPFLG	001173
TAB	042213	\$ATY3	024460	\$GET42	023460	\$RDSZ =	000001	\$TPS	001164
TABLEX	002054	\$ATY4	024470	\$GTSWR	025632	\$RESRE	026256	\$TRAP	027242
TAB.XY=	001114	\$AUTOB	001150	\$HD =	000000	\$RMADR	001276	\$TRAP2	027264
TBITVE=	000014	\$BASE	001266	\$HIBTS	001100	\$RMAS =	000036	\$TRK =	000011
TD	032414	\$BDADR	001136	\$ICNT	001120	\$RMBA =	000024	\$TRP =	000015

STRPAD 027276
STSTM 001104
STSTNM 001116
STTYIN 023326
STYPDS 025336

\$TYPE 024116
\$TYPEC 024330
\$TYPEX 024450
\$TYPOC 025134
\$TYPON 025150

\$TYPOS 025110
\$UNIT 001224
\$UNITM 001110
\$USWR 001236
\$VECT1 001262

\$VECT2 001264
\$WRDM = 000004
\$XOFF = 000023
\$XON = 000021

\$XTSTR 026642
\$\$GET4= 000001
\$OFILL 025333
.\$X = 001100

. ABS. 043032 000
 000000 001
ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 51016 WORDS (200 PAGES)
DYNAMIC MEMORY AVAILABLE FOR 69 PAGES
CZRMTO.BIN,CZRMTO/C=CZRMTO.DOC,CZRMTO,SYSMAC/M

SMTYP2 5-0#

M 11

SEQ 0142

STYPOC 8-151# 8-167 8-167

D 12

SEQ 0146

00000000 02222222 33333333 44444444 55555555 66666666 77777777 88888888 99999999

M.DP44 8-E89 8-E94#

C 13

SEQ 0158

OVWN12 6-0#

E 13

SEQ 0160

PUNSAF 8-a57 8-a72#

G 13

SEQ 0162

[Faint handwritten scribble]

RDP13 6-0#

I 13

SEQ 0164

RDP14	6-0#													
RDP15	6-0#													
RDP16	6-0#													
RDP17	6-0#													
RDP18	6-0#	8-=13												
RDP2	6-0#													
RDP3	6-0#													
RDP4	6-0#													
RDP5	6-0#													
RDP6	6-0#													
RDP7	6-0#													
RDP8	6-0#													
RDP9	6-0#													
READIN	4-104#													
RECAL	4-99#													
RELSE	4-101#													
REPLZ	8-D59	8-D64	8-D69	8-F09#										
RESREG	8-C66	8-E11	8-E44	8-I59	8-I61	8-I67#	9-261	9-407	9-410	9-473	9-686	9-746	9-:24	9-<30
	9-=29	9->62												
RESVEC	4-67#	8-19	8-19*	8-19*										
RM.REG	6-0	6-0	6-0#	8-336	8-548	8-609	8-897	8-941	8-970	8-<29	8-=21	10-60	10-60	10-60
	10-60	10-61	10-61	10-61	10-61	10-61	10-62	10-62	10-62	10-62	10-62	10-63	10-63	10-63
RM05	8-343	8-552	8-562	8-633	8-910	8-942	8-:33	8-:55	8-<31	8-<60	8-=26	9-359#		
RMADR	8-57*	8-289*	8-505*	8-732*	8-807*	8-:42	8-:01	9-169#	9-241	9-366	9-490	9-510	9-532	9-699
	9-740	9-:43	9-:17	9-:23	9-:27	9-:41	9-:74	9-:78	9-<10	9-<40	9->49	10-59		
RMAS	9-193#	9-327*	9-809	9-928*	9-958*	9-979*	9-:68							
RMBA	8-:48*	9-188#												
RMCS1	8-:06*	9-186#	9-283*	9-290	9-316	9-430*	9-431	9-506	9-528	9-570	9-590	9-601	9-616	9-629
	9-762	9-812	9-865	9-895	9-934	9-995	9-:69							
RMCS2	8-:44*	9-190#	9-242*	9-282*	9-284	9-379*	9-429*	9-491*	9-511*	9-533*	9-619	9-642	9-657	9-671
	9-680*	9-700*	9-760*	9-876*	9-965*	9-978*	9-:44*	9-:24	9-:79	9-<11*	9-<17	9-<41*	9-<46	
RMDA	8-:46*	9-189#	9-498	9-524	9-539									
RMDB	9-195#	9-<15												
RMDC	8-:45*	9-200#	9-502	9-514	9-546									
RMDS	9-191#	9-323	9-702	9-879	9-904	9-966	9-:46							
RMDT	9-197#	9-293												
RMEC1	9-204#													
RMEC2	9-205#	9-<25												
RMER1	9-192#	9-329	9-889	9-967	9-:83									
RMER2	9-203#	9-968												
RMERRS	9-22#	9-225	9-847	9-850	9-860	9-966*	9-967*	9-968*	9-969*	10-56	10-56	10-56	10-56	
RMHR	9-201#													
RMINIT	8-319	8-507	8-762	8-833	8-919	8-952	8-982	8-<11	8-<40	8-=36	8-=64	8-a43	8-B83	9-221#
RMLA	9-194#	9-716	9-718											
RMMR1	9-196#													
RMMR2	9-202#	9-969												
RMOF	9-199#	9-320	9-552	9-556	9-576	9-580								
RMSN	9-198#													
RMTMR	8-D91	9-:09#												
RMVEC	8-58*	8-290*	8-292*	8-293*	8-474*	8-475*	8-506*	8-680*	8-681*	8-733*	8-735*	8-736*	8-800*	8-801*
	8-808*	8-?20*	8-?21*	9-170#	9-238	9-240	9-360							
RMWC	8-:47*	9-187#												
RNOP	4-97#													
RSTART	6-0#	8-3*	8-7*	8-11*	8-449	8-524	8-579	8-669	8-850	8-A07	8-A68	8-D46	8-H43	8-I37
RTC	4-103#													
SAVEFG	8-85*	8-320*	8-508*	8-763*	8-834*	8-920*	8-953*	8-983*	8-<12*	8-<41*	8-=37*	8-=65*	8-a44*	8-B84*

9-124# 9-605 9-766 9-955

K 13

SEQ 0166

START1 4-124 8-7#

M 13

SEQ 0168

TRFER 8-A19 8-B56#

B 14

SEQ 0170

XFATL 8-A04 10-114#

D 14

SEQ 0172

XPASS1	8-269	8-282#	8-386							
XPASS2	8-271	8-712	8-725#							
XXDP	6-0#	8-25*	8-28*	3-29	8-31*	8-36	8-47	8-145	8-147	
Y	8-264	10-95#								

\$\$CMRE	4-372#																				
\$\$CMTM	4-372#	5-0																			
\$\$ESCA	4-67#																				
\$\$NEWT	4-67#	8-465	8-538	8-594	8-677	8-791	8-880	8-<10	8-=08	8->10											
\$\$SET	8-167	8-167	8-167	8-167	8-167	8-167	8-167	8-167	8-167	8-167	8-167	8-167	8-167	8-167	8-167#	8-168					
\$\$SETM	8-19	8-19#																			
\$\$SETU	8-19	8-19#																			
\$\$SKIP	4-67#																				
.\$ACT1	4-55#	4-129																			
.\$APT8	4-59#	5-0	5-0#																		
.\$APTH	4-59#	4-132																			
.\$APTY	4-59#	8-147																			
.\$CATC	4-56#	4-122																			
.\$CMTA	4-56#	4-372																			
.\$DB2D	4-57#	8-159																			
.\$DB2O	4-57#	8-161																			
.\$EOP	4-59#	8-139																			
.\$ERRO	4-56#	8-141																			
.\$ERRT	4-56#	8-143																			
.\$POWE	4-59#	8-149																			
.\$READ	4-55#	8-155																			
.\$SAVE	4-57#	8-157																			
.\$SCOP	4-58#	8-163																			
.\$SIZE	4-57#	8-165																			
.\$STRAP	4-57#	8-167																			
.\$TYPD	4-56#	8-153																			
.\$TYPE	4-55#	8-145																			
.\$TYPO	4-56#	8-151																			
.\$EQUAT	4-55#	4-67																			
.\$HEADE	4-55#	4-63																			
.\$SETUP	4-55#	4-120																			
.\$SWRHI	4-55#	4-64																			
.\$SWRLO	4-55#	4-64#	4-65																		
CKCHR	4-16#	8-G57	8-G65																		
CKDIG	4-26#																				
CKNUM	4-39#																				
COMMEN	4-67#																				
ENDCOM	4-67#																				
ERRCAL	8-171#	9-817	9-868	9-870	9-;32	9-;91															
ERROR	4-8#	8-A63	8-A64	8-A65	8-A66	8-B78	8-B79	8-B80	8-B81	9->59											
ESCAPE	4-67#																				
GETPRI	4-67#	8-139	8-165																		
GETSWR	4-58#	4-67#	8-20	8-20#																	
MORETA	4-135#	5-0																			
MULT	4-67#																				
NEWTST	4-67#	8-465	8-538	8-594	8-677	8-791	8-880	8-<10	8-=08	8->10											
POP	4-67#	8-;67	8-<00	8-a23	8-E31	8-147	8-147	8-149	8-149	8-153	8-157										
PUSH	4-67#	8-;24	8-;83	8-?73	8-E24	8-147	8-147	8-147	8-149	8-149	8-153	8-157									
REPORT	4-67#																				
SETPRI	4-67#																				
SETTRA	8-167	8-167	8-167	8-167	8-167	8-167	8-167	8-167	8-167	8-167	8-167	8-167	8-167	8-167#	8-168						
SETUP	4-67#	8-19																			
SKIP	4-67#																				
SLASH	4-67#																				
STARS	4-67#	4-129	4-132	4-132	4-132	5-0	5-0	5-0	8-465	8-538	8-594	8-677	8-791	8-880							
	8-<10	8-=08	8->10	8-139	8-141	8-143	8-145	8-147	8-149	8-149	8-151	8-153	8-155	8-155							

SWRSU	8-155	8-157	8-159	8-161	8-163	8-165	8-167	9-14
TRMTRP	4-67#	8-19	8-19#					
TYPBIN	8-167#	8-169						
TYPDEC	4-67#	8-?95						
TYPNAM	4-67#	8-20						
TYPNUM	4-67#							
TYPOCS	4-67#	8-150						
TYPOCT	4-67#	8-143	8-143	8-155				
TYPTXT	4-67#	8-34	8-45	8-51	8-52			