

RM02/03/05

RM05/3/2 DRV CMPT TST
CZRMTB0

AH-F943B-MC
FICHE 1 OF 1

AUG 1981
COPYRIGHT © 80-81
MADE IN USA



.REM 2

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

IDENTIFICATION

PRODUCT CODE: AC-F942B-MC
PRODUCT NAME: CZRMTBO RM05/3/2 DRIVE COMPATIBILITY TEST
PRODUCT DATE: APRIL 1981
MAINTAINER: CX DIAGNOSTIC GROUP
AUTHOR: MIKE LEAVITT

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

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

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

COPYRIGHT (C) 1980,1981 DIGITAL EQUIPMENT CORPORATION

CONTENTS

1	
2	
3	
4	1.0 ABSTRACT
5	
6	2.0 HARDWARE REQUIREMENTS
7	
8	3.0 PRELIMINARY PROGRAM REQUIREMENTS
9	
10	4.0 GENERAL PROGRAM CONSIDERATIONS
11	
12	4.1 SYSMAC
13	4.2 XXDP
14	4.3 ACT
15	4.4 APT
16	4.5 DUAL-ACCESS
17	4.6 MEMORY MANAGEMENT
18	4.7 MEMORY PARITY OPTION
19	4.8 BAD SECTORS
20	4.9 EXECUTION TIME
21	
22	5.0 PROGRAM LOAD MEDIA
23	
24	6.0 PROGRAM OPTIONS
25	
26	6.1 STARTING ADDRESSES
27	6.2 'SOFTWARE' SWITCH REGISTER
28	6.3 OPERATIONAL SWITCH SETTINGS
29	
30	7.0 RUNNING THE PROGRAM
31	
32	8.0 OPERATIONAL DIALOGUE
33	
34	8.1 DIALOGUE FOR ADDRESS 200 START
35	8.2 DIALOGUE FOR ADDRESS 204 START
36	8.3 DIALOGUE FOR ADDRESS 210 START
37	8.4 PASS 1 DIALOGUE
38	8.5 PASS 2 DIALOGUE
39	
40	9.0 DESCRIPTION OF TESTS
41	
42	9.1 DESCRIPTION OF PASS 1 TESTS
43	9.2 DESCRIPTION OF PASS 2 TESTS
44	
45	10.0 PRINTOUT OF TEST RESULTS
46	
47	10.1 OVERWRITE AND DRIVE COMPATIBILITY DATA TEST RESULTS
48	
49	11.0 ERROR REPORTING
50	
51	11.1 COMMON ERRORS
52	11.2 ERROR HANDLING
53	11.3 ERROR PRINTOUT EXAMPLE
54	
55	12.0 TABLE DESCRIPTIONS
56	
57	12.1 TABLE A - BASIC READ/WRITE TEST SECTORS

58
59
60
61
62
63
64
65

12.2 TABLE B - WORSE CASE DATA PATTERN
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

1.0 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 PROCESSOR
12K 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 DISKLESS TESTS AND THE THE FUNCTIONAL TESTS. 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.

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 'SOFTWARE' SWITCH REGISTER

IF THE PROGRAM IS BEING RUN ON A SWITCHLESS PROCESSOR THE PROGRAM WILL DETERMINE THAT THE HARDWARE SWITCH REGISTER IS NOT PRESENT AND WILL USE A 'SOFTWARE' SWITCH REGISTER. THE 'SOFTWARE' SWITCH REGISTER IS LOCATED AT LOCATION 176 (8). THE SETTINGS OF THE 'SOFTWARE' SWITCHES ARE CONTROLLED THROUGH A KEYBOARD ROUTINE WHICH IS CALLED BY TYPING A 'CONTROL G'. THE PROGRAM WILL RECOGNIZE THE 'CONTROL G' AT ANY TIME EXCEPT WHEN THE PROGRAM IS AT A HIGHER PRIORITY PROCESSING AN RM80 INTERRUPT. THE 'SOFTWARE' SWITCH VALUES ARE ENTERED AS AN OCTAL NUMBER IN RESPONSE TO THE PROMPT FROM THE SWITCH ENTRY ROUTINE:

'SWR = NNNNNN NEW ='

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

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

6.3 OPERATIONAL SWITCH SETTINGS

WITH ALL SWITCHES SET TO ZERO, THE PROGRAM WILL TYPE ALL ERRORS AND CONTINUE TESTING.

THE SWITCH SETTINGS ARE:

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:

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

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

THE PROGRAM FIRST IDENTIFIES ITSELF AS FOLLOWS:

CZRMTB0 - 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

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:

CZRMTB0 - 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

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

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 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:

CZRMTB0 - 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

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

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.

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.

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 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

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

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
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

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

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 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:

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

** 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	OWRVRT OFST-	OWRVRT 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

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

742	HEADER READ ERROR - HEADER COMPARE 'HCE' ERROR
743	
744	FORMAT ERROR 'FER'
745	
746	HEADER COMPARE 'HCE' ERROR
747	
748	MISCELLANEOUS DRIVE ERROR
749	
750	OPERATION INCOMPLETE 'OPI' ERROR
751	
752	DRIVE TIMING 'DTE' ERROR
753	
754	PARITY 'PAR' ERROR AFTER OPERATION STARTED
755	
756	WRITE CLOCK FAILURE 'WCF' ERROR
757	
758	INVALID ADDRESS 'IAE' ERROR
759	
760	WRITE LOCK 'WLE' ERROR
761	
762	DATA CHECK 'DCK' SET DURING WRITE CHECK COMMAND
763	
764	RH OR UNIBUS TRANSFER ERROR
765	
766	BUS ADDRESS OR WORD COUNT INCORRECT
767	
768	DATA COMPARE ERRORS - NO OTHER ERROR(S) DETECTED
769	
770	CAN'T MATCH DATA READ WITH A PATTERN
771	
772	ERROR BIT(S) SET, BUT NO ERROR SIGNALLED BY THE RH
773	
774	ECC LOGIC FAILURE - POSITION REGISTER VALUE NOT VALID
775	
776	BUS ADDRESS AND WORD COUNT NOT CONSISTENT
777	
778	SEEK INCOMPLETE 'SKI' ERROR
779	
780	PROGRAM DETECTED POSITIONING ERROR
781	
782	DRIVE UNSAFE ERROR
783	
784	
785	
786	
787	
788	
789	
790	
791	
792	
793	
794	
795	
796	
797	
798	

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, I/STERS, ARE REPORTED IN A GIVEN ERROR. OTHER ERROR REPORTS MAY CONSIST OF A SINGLE DESCRIPTIVE LINE.

11.3 ERROR PRINTOUT EXAMPLE

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

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		

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

856	6	066633
857	7	155466
858	8	133155
859	9	066333
860	10	154666
861	11	131555
862	12	063333
863	13	146666
864	14	115555
865	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
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

913	->	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
914	->	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0
915	->	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1	2
916	->	6	7	8	9	10	11	12	13	14	15	0	1	2	3	4	5
917	->	10	11	12	13	14	15	0	1	2	3	4	5	6	7	8	9
918	->	15	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
919	->	5	6	7	8	9	10	11	12	13	14	15	0	1	2	3	4
920	->	12	13	14	15	0	1	2	3	4	5	6	7	8	9	10	11
921	->	4	5	6	7	8	9	10	11	12	13	14	15	0	1	2	3
922	->	13	14	15	0	1	2	3	4	5	6	7	8	9	10	11	12
923	->	7	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6
924	->	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1
925	->	14	15	0	1	2	3	4	5	6	7	8	9	10	11	12	13
926	->	11	12	13	14	15	0	1	2	3	4	5	6	7	8	9	10
927	->	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7	8
928	->	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
1	1,129,257,385,513,641,769
2	2,130,258,386,514,642,770
3	3,131,259,387,515,643,771
4	4,132,260,388,516,644,772
5	5,133,261,389,517,645,773
6	6,134,262,390,518,646,774
7	7,135,263,391,519,647,775
8	8,136,264,392,520,648,776
9	9,137,265,393,521,649,777
10	10,138,266,394,522,650,778
11	11,139,267,395,523,651,779
12	12,140,268,396,524,652,780
13	13,141,269,397,525,653,781
14	14,142,270,398,526,654,782
15	15,143,271,399,527,655,783

12.6 TABLE F - SELF-TEST CYLINDERS

DRIVE #	CYLINDERS
-----	-----
0	17,145,273,401,529,657,785
1	18,146,274,402,530,658,786
2	19,147,275,403,531,659,787
3	20,148,276,404,532,660,788
4	21,149,277,405,533,661,789
5	22,150,278,406,534,662,790
6	23,151,279,407,535,663,791
7	24,152,280,408,536,664,792
8	25,153,281,409,537,665,793
9	26,154,282,410,538,666,794
10	27,155,283,411,539,667,795

970	11	28,156,284,412,540,668,796
971	12	29,157,285,413,541,669,797
972	13	30,158,286,414,542,670,798
973	14	31,159,287,415,543,671,799
974	15	32,160,288,416,544,672,800
975		
976		
977		
978		
979		

12.8 TABLE G - PSEUDO-RANDOM DATA PATTERN

	WORD #	DATA (OCTAL)
980	-----	-----
981	0	040135
982	1	177070
983	2	070414
984	3	064531
985	4	174473
986	5	062422
987	6	114352
988	7	036620
989	8	010031
990	9	052336
991	10	017310
992	11	011347
993	12	102367
994	13	152567
995	14	001246
996	15	160073
997		
998		
999		

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:

1026

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

DRV TYP	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 R0,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)

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

1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140

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  
      .WORD    WRTBUF       ;BUFFER ADDRESS  
      .BYTE    3            ;SECTOR  
      .BYTE    5            ;TRACK  
      .WORD    400          ;CYLINDER  
      .WORD    ERRTB5       ;ERROR TABLE  
      .WORD    0            ;STATUS/ERROR INDICATOR
```

ALTERNATE DPB SETUP

```
WRTDPB: .WORD    5            ;THIS SETUP ACHIEVED  
      .WORD    WRITE        ;EVERYTHING THE  
      .WORD    -1000.       ;ABOVE TABLE DID, BUT  
      .WORD    WRTBUF       ;IN A CLEANER FORMAT  
      .BYTE    3,5  
      .WORD    400,ERRTB5,0
```

13.5 RH/RM REGISTERS

MNEMONIC	INDEX
-----	-----
RMCS1	0
RMWC	2
RMBA	4
RMDA	6
RMCS2	10
RMDS	12
RMER1	14
RMAS	16
RMLA	20
RMDB	22
RMMR1	24
RMDT	26

1141	RMSN	30
1142	RMOF	32
1143	RMDC	34
1144	RMHR	36
1145	RMMR2	40
1146	RMER2	42
1147	RMEC1	44
1148	RMEC2	46

13.6 COMMANDS PERFORMED BY THE DRIVER

COMMAND	CODE	COMMAND TYPE
SEEK	105	P
RECALIRATE	107	P
DRIVE CLEAR	111	N
RELEASE	113	N
OFFSET	115	P
RETURN TO CENTER	117	P
READIN PRESET	121	N
PACK ACKNOWLEDGE	123	N
SEARCH	131	P
GET REGISTER(S)	141	S
SET FORMAT	143	S
SELECT DRIVE	145	S
WRITE CHECK DATA	151	D
WRITE CHK HEADER & DATA	153	D
WRITE DATA	161	D
WRITE HEADER & DATA	163	D
READ DATA	171	D
READ HEADER & DATA	173	D

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

13.7 DPB STATUS/ERROR INDICATOR WORD

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

BIT NO.	MEANING IF ON A '1'
15	ERROR OCCURRED DONE (BIT07=0); BITS 14-9 SPECIFIES TYPE DONE (BIT07=1); BITS 6-3 SPECIFIES TYPE
14(1)	USER MADE A REQUEST FOR A FUNCTION TO BE PERFORMED ON AN OFFLINE OR UNSAFE DRIVE
13(1)	USER MADE A REQUEST FOR A FUNCTION TO BE PERFORMED ON A DRIVE THAT HAS AN

1198		UNLOAD REQUEST IN QUEUE.
1199		
1200	12(2)	PERSISTENT UNSAFE CONDITION EXIST.
1201		
1202	11(2)	UNCORRECTABLE PARITY ERROR OCCURRED
1203		
1204	10(2)(4)	FATAL PARITY ERROR. A MASSBUS CLEAR WAS
1205		PERFORMED, ALL QUEUES WERE EMPTIED, AND
1206		ALL DRVACT'S SET TO THE IDLE STATE
1207		
1208	9(3)(4)	SOFTWARE TIMEOUT OCCURRED ON THIS DRIVE
1209		
1210	8(4)	SOFTWARE TIMEOUT OCCURRED ON ANOTHER DRIVE
1211		
1212	7	DONE
1213		
1214	6(2)	ERROR OCCURRED DURING AN I/O OPERATION
1215		
1216	5(2)	ERROR OCCURRED DURING AN OPERATION
1217		OTHER THAN I/O.
1218		
1219	4(2)	CORRECTABLE UNSAFE CONDITION OCCURRED
1220		
1221	3(2)	DRIVE ERROR OCCURRED THAT CAUSED AN
1222		AUTOMATIC 'RECALIBRATE' SEQUENCE
1223		
1224	2	PORT REQUEST TIMEOUT. THE DRIVER REQUESTED
1225		THE DRIVE BUT THE OPPOSITE PORT DID NOT
1226		RELEASE THE DRIVE WITHIN 15. SECONDS.
1227		
1228	1	NON-EXISTENT DRIVE REQUESTED. USER MADE
1229		A REQUEST FOR A NON-EXISTENT DRIVE.
1230		

NOTES FOR ABOVE

1231		
1232		
1233		
1234	(1) =	REQUEST WASN'T PUT IN QUEUE. (RH/RM
1235		REGISTERS WERE NOT SAVED)
1236		
1237	(2) =	REQUEST QUEUE HAS BEEN EMPTIED. THE DRIVER
1238		ISSUED A 'DRIVE CLEAR' TO THE DRIVE.
1239		NOTE: ALL RH/RM REGISTERS ARE SAVED
1240		AS PER DPB+14 BEFORE THE 'DRIVE CLEAR'.
1241		
1242	(3) =	REQUEST QUEUE HAS BEEN EMPTIED. THE
1243		DRIVER ISSUED A MASSBUS INIT. ALL
1244		RH/RM REGISTERS FOR THE DRIVE WERE
1245		SAVED AS PER DPB+14 BEFORE THE INIT.
1246		
1247	(4) =	A 'RECALIBRATE' SHOULD BE ISSUED
1248		BEFORE ANY OTHER COMMAND.
1249		

13.8 ERROR CALLS MADE BY THE DRIVER.

THERE ARE A FEW ERRORS THAT CAN OCCUR THAT CAN NOT BE INDICATED IN A DPB.

1250
1251
1252
1253
1254

1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293

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 =RMD5 RMERRS+2=RMER1 RMERRS+4=RMER2 RMERRS+6=RMER2
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
5	ADDRESS PLUG CHANGE BIT SET ('OPE' ERROR)	R1= DRIVE NUMBER R3= ATA BIT *R4= RMCS1'S ADDRESS R5= (RMAS) RMERRS =RMD5 RMERRS+2=RMER1 RMERRS+4=RMER2 RMERRS+6=RMER2

* THIS IS THE ACTUAL UNIBUS ADDRESS (176700)

1
53
54

;*LAST REVISION 04-APR-81

.TITLE CZRMTBO RM05/3/2 DR CMPT TST

;*COPYRIGHT (C) 1981

;*DIGITAL EQUIPMENT CORPORATION

;*COLORADO SPGS., CO. 80919

;*

;*PROGRAM BY MIKE LEAVITT

;*

;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC

;*PACKAGE (MAINDEC-11-DZQAC-C5), 18-MAR-81

;*

55

.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

56
57
58

.SBTTL BASIC DEFINITIONS

;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***

001100

STACK = 1100

104000

ERROR = EMT

;;BASIC DEFINITION OF ERROR CALL

000004

SCOPE = IOT

;;BASIC DEFINITION OF SCOPE CALL

;*MISCELLANEOUS DEFINITIONS

000011

HT = 11

;;CODE FOR HORIZONTAL TAB

000012

LF = 12

;;CODE FOR LINE FEED

000015

CR = 15

;;CODE FOR CARRIAGE RETURN

000200

CRLF = 200

;;CODE FOR CARRIAGE RETURN-LINE FEED

177776

PS = 177776

;;PROCESSOR STATUS WORD

177776

PSW=PS

177774

STKLMT = 177774

;;STACK LIMIT REGISTER

177772

PIRQ = 177772

;;PROGRAM INTERRUPT REQUEST REGISTER

177570

DSWR = 177570

;;HARDWARE SWITCH REGISTER

177570

DDISP = 177570

;;HARDWARE DISPLAY REGISTER

;*GENERAL PURPOSE REGISTER DEFINITIONS

000000

R0 - %0

;;GENERAL REGISTER

000001

R1 - %1

;;GENERAL REGISTER

000002

R2 - %2

;;GENERAL REGISTER

000003

R3 - %3

;;GENERAL REGISTER

000004

R4 - %4

;;GENERAL REGISTER

000005

R5 - %5

;;GENERAL REGISTER

000006

R6 - %6

;;GENERAL REGISTER

000007

R7 - %7

;;GENERAL REGISTER

000006

SP - %6

;;STACK POINTER

000007

PC - %7

;;PROGRAM COUNTER

```

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
ERRVEC = 4 ;TIME OUT AND OTHER ERRORS
000010 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

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

.SBTTL RM REGISTERS

;INDEX OF STATUS AND REGISTER WORDS RELATIVE TO FMTDPB

000016 STATUS = 16
000020 SRMCS1 = STATUS+2
000022 SRMJC = SRMCS1+2
000024 SRMBA = SRMJC+2
000026 SRMDA = SRMBA+2
000030 SRMCS2 = SRMDA+2
000032 SRMDS = SRMCS2+2
000034 SRMER1 = SRMDS+2
000036 SRMAS = SRMER1+2
000040 SRMLA = SRMAS+2
000042 SRMDB = SRMLA+2
000044 SRMMR1 = SRMDB+2
000046 SRMDT = SRMMR1+2
000050 SRMSN = SRMDT+2
000052 SRMOF = SRMSN+2
000054 SRMDC = SRMOF+2
000056 SRMHR = SRMDC+2
000060 SRMMR2 = SRMHR+2
000062 SRMER2 = SRMMR2+2
000064 SRMEC1 = SRMER2+2
000066 SRMEC2 = SRMEC1+2

.SBTTL RM DRIVER COMMANDS

RNOP = 101 ;NO OPERATION
SEEK = 105 ;SEEK
RECAL = 107 ;RECALIBRATE
DRVCLR = 111 ;DRIVE CLEAR
RELSE = 113 ;RELEASE
OFFSET = 115 ;OFFSET

94	000117	RTC	= 117	;RETURN TO CENTER LINE
95	000121	READIN	= 121	;READ IN PRESET
96	000123	ACK	= 123	;PACK ACKNOWLEDGE
97	000131	SEARCH	= 131	;SEARCH
98	000141	GETREG	= 141	;GET REGISTERS
99	000143	SETFMT	= 143	;SET FORMAT (& ECI OR HCI)
100	000145	SELDRV	= 145	;SELECT DRIVE
101	000151	WCKD	= 151	;WRITE CHECK DATA
102	000153	WCKHD	= 153	;WRITE CHECK HEADER & DATA
103	000161	WRTDAT	= 161	;WRITE DATA
104	000163	WRTHD	= 163	;WRITE HEADER & DATA
105	000171	RDDAT	= 171	;READ DATA
106	000173	RDHD	= 173	;READ HEADER & DATA
107				
108	176700	ABASE	= 176700	
109	000254	AVECT1	= 254	
110				

1

U00000

.SBTTL TRAP CATCHER

```

      . = 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
000176 000000
000176 000000

```

```

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

```

.SBTTL STARTING ADDRESS(ES)

```

2 000200 000137 005566
3 000204 000137 005604
4 000210 000137 005622
5
6
7
8

```

```

      JMP      @#START      ;;JUMP TO STARTING ADDRESS OF PROGRAM
      JMP      @#START1     ;CHANGE THE RH/RM UNIBUS ADDRESS
      JMP      @#START2     ;SECOND PASS STARTING ADDRESS

```

.SBTTL ACT11 HOOKS

```

000046 000214
000046 000046
000052 023544
000052 000052
000052 040000
000052 000214
9
10 001100
11

```

```

;*****
;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

```

```

      .-1100
      .SBTTL APT PARAMETER BLOCK

```

```

000024 001100
000024 000024
000024 000200
000044 000044
000044 001100
000044 001100

```

```

;*****
;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
      . = 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-PDP*1 DIAGNOSTIC
;INTERFACE SPEC.

```

```

001100
001100 000000
001102 001212
001104 000454
001106 000454
001110 000454
001112 000032
12 001114
13

```

```

$APTHD:
$HIBTS: .WORD 0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$MBAADR: .WORD $MAIL  ;;ADDRESS OF APT MAILBOX (BITS 0-15)
$TSTM:   .WORD 300.   ;;RUN TIM OF LONGEST TEST
$PASTM:  .WORD 300.   ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
$UNITM:  .WORD 300.   ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDED UNIT
          .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE (WORDS)
TAB.XY .

```

6

.SBTTL COMMON TAGS

```

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

```

Address	Hex	ASCII	Symbolic	Description
001114	001114		SCMTAG: . = 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	\$BELL: .ASCII <207><377><377>	:: CODE FOR BELL
001206	077		\$QUES: .ASCII /?/	:: QUESTION MARK
001207	015		\$CRLF: .ASCII <15>	:: CARRIAGE RETURN
001210	012	000	\$LF: .ASCII <12>	:: LINE FEED

SBTTL APT MAILBOX-ETABLE

ADDRESS	DATA	FIELD	DESCRIPTION
001212		SMAIL:	APT MAILBOX
001212	000000	SMSGTY: .WORD	MESSAGE TYPE CODE
001214	000000	SFATAL: .WORD	FATAL ERROR NUMBER
001216	000000	STESTN: .WORD	TEST NUMBER
001220	000000	SPASS: .WORD	PASS COUNT
001222	000000	SDEVCT: .WORD	DEVICE COUNT
001224	000000	SUNIT: .WORD	I/O UNIT NUMBER
001226	000000	SMSGAD: .WORD	MESSAGE ADDRESS

001230	000000	\$MSGLG: .WORD	AMSGLG	::MESSAGE LENGTH
001232		\$ETABLE:		::APT ENVIRONMENT TAB. E
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#2, BUS PRIORITY#2
001266	176700	\$BASE: .WORD	ABASE	::BASE ADDRESS OF EQUIPMENT UNDER TEST
001270	000000	\$DEVN: .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		

.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	: DELTA 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.

```
;SCORE BOARD TABLES
;TABLE OF OVERWRITE SCORE,NEGATIVE OFFSET SCORE
```

002056	000	000	000	OVWNO:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;	TRACK 0
002076	000	000	000	OVWN1:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;	TRACK 1
002116	000	000	000	OVWN2:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;	TRACK 2
002136	000	000	000	OVWN3:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;	TRACK 3
002156	000	000	000	OVWN4:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;	TRACK 4
002176	000	000	000	OVWN5:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;	TRACK 5
002216	000	000	000	OVWN6:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;	TRACK 6
002236	000	000	000	OVWN7:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;	TRACK 7
002256	000	000	000	OVWN8:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;	TRACK 8
002276	000	000	000	OVWN9:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;	TRACK 9
002316	000	000	000	OVWN10:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;	TRACK 10
002336	000	000	000	OVWN11:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;	TRACK 11
002356	000	000	000	OVWN12:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;	TRACK 12
002376	000	000	000	OVWN13:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;	TRACK 13
002416	000	000	000	OVWN14:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;	TRACK 14
002436	000	000	000	OVWN15:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;	TRACK 15
002456	000	000	000	OVWN16:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;	TRACK 16
002476	000	000	000	OVWN17:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;	TRACK 17
002516	000	000	000	OVWN18:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;	TRACK 18

;TABLE OF OVERWRITE SCORE, POSITIVE OFFSET SCORE

002536	000	000	000	OVWP0:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	: TRACK	0
002556	000	000	000	OVWP1:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	: TRACK	1
002576	000	000	000	OVWP2:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	: TRACK	2
002616	000	000	000	OVWP3:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	: TRACK	3
002636	000	000	000	OVWP4:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	: TRACK	4
002656	000	000	000	OVWP5:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	: TRACK	5
002676	000	000	000	OVWP6:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	: TRACK	6
002716	000	000	000	OVWP7:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	: TRACK	7
002736	000	000	000	OVWP8:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	: TRACK	8
002756	000	000	000	OVWP9:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	: TRACK	9
002776	000	000	000	OVWP10:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	: TRACK	10
003016	000	000	000	OVWP11:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	: TRACK	11
003036	000	000	000	OVWP12:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	: TRACK	12
003056	000	000	000	OVWP13:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	: TRACK	13
003076	000	000	000	OVWP14:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	: TRACK	14

```
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

[illegible]

;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	043456		.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	C	,RMDS
004542	000000		.WORD	0	,RMER1
004544	000000		.WORD	0	,RMAS
004546	000000		.WORD	0	,RMLA
004550	000000		.WORD	0	,RMD8
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		\$BLF	=	\$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	780,0	:HISTORY BLOCK OF LOGICAL DRIVE 0
004624			.BLKB	\$EMTAB-\$COMND		
004644	001	000	DRIV1:	.BYTE	781,0	:HISTORY BLOCK OF LOGICAL DRIVE 1
004646			.BLKB	\$EMTAB-\$COMND		
004666	002	000	DRIV2:	.BYTE	782,0	:HISTORY BLOCK OF LOGICAL DRIVE 2

004670			.BLKB	SEMTAB-\$COMND	
004710	003	000	DRIV3:	.BYTE 783,0	;HISTORY BLOCK OF LOGICAL DRIVE 3
004712			.BLKB	SEMTAB-\$COMND	
004732	004	000	DRIV4:	.BYTE 784,0	;HISTORY BLOCK OF LOGICAL DRIVE 4
004734			.BLKB	SEMTAB-\$COMND	
004754	005	000	DRIV5:	.BYTE 785,0	;HISTORY BLOCK OF LOGICAL DRIVE 5
004756			.BLKB	SEMTAB-\$COMND	
004776	006	000	DRIV6:	.BYTE 786,0	;HISTORY BLOCK OF LOGICAL DRIVE 6
005000			.BLKB	SEMTAB-\$COMND	
005020	007	000	DRIV7:	.BYTE 787,0	;HISTORY BLOCK OF LOGICAL DRIVE 7
005022			.BLKB	SEMTAB-\$COMND	
005042	000	000	DRIV10:	.BYTE 7810,0	;HISTORY BLOCK OF LOGICAL DRIVE 10
005044			.BLKB	SEMTAB-\$COMND	
005064	001	000	DRIV11:	.BYTE 7811,0	;HISTORY BLOCK OF LOGICAL DRIVE 11
005066			.BLKB	SEMTAB-\$COMND	
005106	002	000	DRIV12:	.BYTE 7812,0	;HISTORY BLOCK OF LOGICAL DRIVE 12
005110			.BLKB	SEMTAB-\$COMND	
005130	003	000	DRIV13:	.BYTE 7813,0	;HISTORY BLOCK OF LOGICAL DRIVE 13
005132			.BLKB	SEMTAB-\$COMND	
005152	004	000	DRIV14:	.BYTE 7814,0	;HISTORY BLOCK OF LOGICAL DRIVE 14
005154			.BLKB	SEMTAB-\$COMND	
005174	005	000	DRIV15:	.BYTE 7815,0	;HISTORY BLOCK OF LOGICAL DRIVE 15
005176			.BLKB	SEMTAB-\$COMND	
005216	006	000	DRIV16:	.BYTE 7816,0	;HISTORY BLOCK OF LOGICAL DRIVE 16
005220			.BLKB	SEMTAB-\$COMND	
005240	007	000	DRIV17:	.BYTE 7817,0	;HISTORY BLOCK OF LOGICAL DRIVE 17
005242			.BLKB	SEMTAB-\$COMND	

;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

.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

1	005362			
2				
3				
4	005362	036576	EM1	;RH CONTROLLER INTERRUPT OCCURRED (RMAS 0)
5	005364	041212	DH1	
6	005366	041634	DT1	
7	005370	041760	DF1	
8				
9				
10				
11	005372	036635	EM2	;UNEXPECTED ATTENTION OCCURRED
12	005374	041221	DH2	
13	005376	041640	DT2	
14	005400	041761	DF2	
15				
16				
17				
18	005402	036673	EM3	;MASSBUS PARITY ERROR (MCPE 1)
19	005404	041303	DH3	
20	005406	041656	DT3	
21	005410	041767	DF3	
22				
23				
24				
25	005412	036731	EM4	;MASSBUS PARITY ERROR (PAR-1)
26	005414	041334	DH4	
27	005416	041666	DT4	
28	005420	041772	DF4	
29				
30				
31				
32	005422	036766	EM5	;ADDRESS PLUG BIT CHANGED
33	005424	041221	DH2	
34	005426	041640	DT2	
35	005430	041761	DF2	
36				
37				
38				
39	005432	037022	EM6	;RH CONTROLLER DIDN'T RESPOND TO ADDRESSING
40	005434	041376	DH6	
41	005436	041700	DT6	
42	005440	041760	DF1	

```
43
44
45      ; * ERRORS 7 - 12 ARE PART OF THE 'DUMP' SUBROUTINE
46      ; ERROR 7
47
48      005442 000000      0
49      005444 041407      DH7
50      005446 041704      DT7
51      005450 000000      0
52
53      ; ERROR 10
54
55      005452 000000      0
56      005454 041460      DH10
57      005456 041720      DT10
58      005460 000000      0
59
60      ; ERROR 11
61
62      005462 000000      0
63      005464 041531      DH11
64      005466 041734      DT11
65      005470 000000      0
66
67      ; ERROR 12
68
69      005472 000000      0
70      005474 041602      DH12
71      005476 041750      DT12
72      005500 000000      0
```

```

1      ; THIS ROUTINE HANDLES UNEXPECTED TIMEOUTS
2
3 005502 011600      BADTMO: MOV      (SP),R0      ;SAVE PC WHERE THE TIME OUT OCCURED
4 005504 005740      TST      -(R0)      ;ADJUST PC -2
5 005506 022626      CMP      (SP)+,(SP)+    ;RESTORE STACK POINTER
6 005510 104401 005516  TYPE      ,65$      ;:TYPE ASCIZ STRING
   005514 000417      BR       64$      ;:GET OVER THE ASCIZ
   ;:65$: .ASCIZ <CRLF>/UNEXPECTED BUS TIMEOUT, PC=/
   64$:
7 005554 010046      MOV      R0,-(SP)      ;SETUP FOR TYPING OUT PC
8 005556 104402      TYPOC
9 005560 000240      NOP
   ;PUT 'HALT(0)' INSTRUCTION HERE IF YOU WISH
   ;TO STOP ON UNEXPECTED TIMEOUT.
10
11 005562 000177 173612  JMP      @RSTART    ;JUMP TO RESTART
12
13      .SBTTL  START OF PROGRAM
14
15 005566 012737 005566 001400  START:  MOV      #.,RSTART    ;SETUP RESTART ADDRESS
16 005574 012737 000400 001334      MOV      #400,CHGADR    ;200 START ADDRESS FLAG
17 005602 000414      BR       START3
18
19 005604 012737 005604 001400  START1: MOV      #.,RSTART    ;SETUP RESTART ADDRESS
20 005612 012737 177777 001334      MOV      #-1,CHGADR    ;204 START ADDRESS FLAG
21 005620 000405      BR       START3
22
23 005622 012737 005622 001400  START2: MOV      #.,RSTART    ;SETUP RESTART ADDRESS
24 005630 005037 001334      CLR      CHGADR      ;210 START ADDRESS FLAG
25
26 005634 000240      START3: NOP
27 005636 005227 000000      INC      #0      ;TTY LOOP, WAIT FOR INCREMENT
28 005642 001375      BNE      -4      ;OF WORD
29 005644 000005      RESET      ;CLEAR THE WORLD
30
31      .SBTTL  INITIALIZE THE COMMON TAGS
   ;:CLEAR THE COMMON TAGS ($CMTAG) AREA
   MOV      #SCMTAG,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      #TRAP,@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      $ENDCT,$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

```

CRMTB0 RM05/3/2 DR CMPT IST
INITIALIZE THE COMMON TAGS

MACRO V04.00 4-APR-81 18:12:15 PAGE 9-1

SEQ 0044

```

006006 012737 000002 023610      MOV      #RTI,$RTRN      ;;SET $RTRN TO A RTI
006014 012737 006042 000010      MOV      #65$,@#RESVEC  ;;TRY TO DO A RTT
006022 005046                CLR      -(SP)          ;;DUMMY PS
006024 012746 006032                MOV      #64$,-(SP)      ;;AND PC
006030 000006                RTT                    ;;TRY THE RTT
006032 012737 000006 023610 64$:  MOV      #RTT,$RTRN      ;;RTT IS LEGAL--SET $RTRN TO A RTT
006040 000402                BR       66$
006042 062706 000010 65$:      ADD      #10,SP          ;;RTT ILLEGAL--CLEAN OFF THE STACK
006046 012737 000012 000010 66$:  MOV      #RESVEC+2,@#RESVEC  ;;RESTORE TRAP CATCHER
006054 005037 023616                CLR      $TBIT        ;;CLEAR 'T' BIT SWITCH
006060 012737 006060 001122                MOV      #,$LPADR      ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
006066 012737 006066 001124                MOV      #,$LPERR      ;;SETUP THE ERROR LOOP ADDRESS
                                ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
                                ;;EQUAL TO A '-1', SETUP FOR A SOFTWARE SWITCH REGISTER.
006074 013746 000004                MOV      @#ERRVEC,-(SP)    ;;SAVE ERROR VECTOR
006100 012737 006134 000004                MOV      #67$,@#ERRVEC  ;;SET UP ERROR VECTOR
006106 012737 177570 001154                MOV      #DSWR,SWR      ;;SETUP FOR A HARDWARE SWICH REGISTER
006114 012737 177570 001156                MOV      #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
006122 022777 177777 173024                CMP      #-1,@SWR      ;;TRY TO REFERENCE HARDWARE SWR
006130 001012                BNE      69$                ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                ;;AND THE HARDWARE SWR IS NOT = -1
006132 000403                BR       68$                ;;BRANCH IF NO TIMEOUT
006134 012716 006142 67$:      MOV      #68$,(SP)          ;;SET UP FOR TRAP RETURN
006140 000002                RTI
006142 012737 000176 001154 68$:  MOV      #SWREG,SWR      ;;POINT TO SOFTWARE SWR
006150 012737 000174 001156                MOV      #DISPREG,DISPLAY
006156 012637 000004 69$:      MOV      (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR

006162 005037 001220                CLR      $PASS        ;;CLEAR PASS COUNT
006166 132737 000200 001233                BITB     #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
006174 001403                BEQ      70$                ;;YES,USE NON-APT SWITCH
006176 012737 001234 001154                MOV      #SSWREG,SWR  ;;NO,USE APT SWITCH REGISTER
006204                70$:
32      ;;SETUP 'TIMEOUT' TRAP VECTOR FOR UNEXPECTED BUS TIMEOUTS
33      MOV      #BADTMO,ERRVEC  ;;SETUP FOR UNEXPECTED TIMEOUT
34      MOV      #PR6,ERRVEC+2  ;;LEVEL 6
35
36      .SBTTL  TYPE PROGRAM NAME
      ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
006220 005227 177777                INC      #-1          ;;FIRST TIME?
006224 001036                BNE      71$                ;;BRANCH IF NO
006226 022737 023544 000042                CMP      #SENDAD,@#42  ;;ACT-11?
006234 001432                BEQ      71$                ;;BRANCH IF YES
006236 104401 006244                TYPE     72$                ;;TYPE ASCII STRING
006242 000427                BR       71$                ;;GET OVER THE ASCII
      ;;72$: .ASCIIZ <CRLF>@CRMTB0 - RM05/3/2 DRIVE COMPATIBILITY IST@<CRLF>
      71$:
      .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
006322 005737 000042                TST      @#42          ;;ARE WE RUNNING UNDER XXDP/ACT?
006326 001012                BNE      73$                ;;BRANCH IF YES
006330 123727 001232 000001                CMPB    $ENV,#1      ;;ARE WE RUNNING UNDER APT?
006336 001406                BEQ      73$                ;;BRANCH IF YES
006340 023727 001154 000176                CMP      SWR,#SWREG  ;;SOFTWARE SWITCH REG SELECTED?
006346 001005                BNE      74$                ;;BRANCH IF NO
006350 104406                GTSWR      ;;GET SOFT-SWR SETTINGS
006352 000403                BR       74$
006354 112737 000001 001150 73$:  MOVB     #1,$AUTOB      ;;SET AUTO-MODE INDICATOR

```

```

006362
37
38
39
40
41 006362 005037 001404 CLR XXDP ;CLEAR 'XXDP' LOAD DEVICE STORAGE
42 006366 122737 000016 000041 CMPB #16,@#41 ;LOADED FROM AN RM05/3/2 ?
43 006374 001160 BNE 3$ ;ER IF NOT
44 006376 013737 000040 001404 MOV @#40,XXDP ;GET DEVICE INDICATOR AND NUMBER
45 006404 122737 000007 001404 CMPB #7,XXDP ;IS IT A VALID NUMBER ?
46 006412 103002 BHIS 1$ ;YES
47 006414 105037 001404 CLRB XXDP ;NO, DEFAULT TO DRIVE 0
48 006420 005737 000042 1$: TST @#42 ;CHAIN MODE OR ACT11 AUTO ACCEPT ?
49 006424 001425 BEQ 2$ ;BR IF NEITHER
50 006426 104401 006434 TYPE ,76$ ;TYPE ASCII STRING
006432 000412 BR 75$ ;GET OVER THE ASCII
;:76$: .ASCIIZ <CRLF>/NOT TESTING DRIVE /
75$:
51 006460 005046 CLR -(SP) ;CLEAR WORD ON STACK
52 006462 113716 001404 MOVB XXDP,(SP) ;GET DRIVE ADDRESS
53 006466 104403 TYPOS ;TYPE THE ADDRESS
54 006470 001 .BYTE 1 ;ONLY 1 CHARACTER
55 006471 000 .BYTE 0 ;SUPRESS LEADING ZEROS
56 006472 104401 001207 TYPE ,5CRLF ;CR-LF
57 006476 000517 BR 3$ ;GET NUMBER OF DRIVES
58
59 006500 005227 177777 2$: INC #-1 ;FIRST TIME THRU HERE ?
60 006504 001114 BNE 3$ ;NO
61 006506 104401 006514 TYPE ,78$ ;TYPE ASCII STRING
006512 000410 BR 77$ ;GET OVER THE ASCII
;:78$: .ASCIIZ <CRLF>/TO TEST DRIVE /
77$:
62 006534 005046 CLR -(SP) ;CLEAR WORD ON STACK
63 006536 113716 001404 MOVB XXDP,(SP) ;GET DRIVE ADDRESS
64 006542 104403 TYPOS ;TYPE DRIVE ADDRESS
65 006544 001 .BYTE 1 ;ONLY 1 CHARACTER
66 006545 000 .BYTE 0 ;SUPRESS LEADING ZEROS
67 006546 104401 006554 TYPE ,80$ ;TYPE ASCII STRING
006552 000431 BR 79$ ;GET OVER THE ASCII
;:80$: .ASCIIZ /, HALT PROGRAM, REMOVE RRDP PACK AND REPLACE IT/<CRLF>
79$:
68 006636 104401 006644 TYPE ,81$ ;TYPE ASCII STRING
006642 000435 BR 3$ ;GET OVER THE ASCII
;:81$: .ASCIIZ /WITH A WORK PACK, CLEAR LOCATION 40 AND RESTART PROGRAM./<CRLF>
3$:
72 006736 004737 022754 JSR PC,$TKINT ;TURN ON THE KEYBOARD INTERRUPT
73 006742 013737 001276 030324 MOV $RMADR,RMADR ;RH/RM ADDRESS
74 006750 013737 001300 030326 MOV $RMVEC,RMVEC ;VECTOR ADDRESS
75 006756 012705 002006 MOV #ASNLST,R5 ;START OF AREA TO CLEAR
76 006762 005025 4$: CLR (R5)+
77 006764 022705 004424 CMP #INDST,R5 ;LOOK FOR END OF CLEAR AREA
78 006770 001374 BNE 4$ ;BR IF NOT FINISHED
79 006772 012706 001100 MOV #STACK,SP ;SETUP THE STACK POINTER
80 006776 005037 177776 CLR PS ;CLEAR THE PROCESSOR STATUS WORD
81 007002 013737 001320 001352 MOV HZ,SIXTEE ;1/60 TH OR 1/50 TH SECOND COUNTER VALUE
82 007010 005037 001344 CLR HOUR ;CLEAR THE HOUR'S COUNTER
83 007014 005037 001346 CLR MINUTE ;CLEAR THE MINUTE'S COUNTER

```

```

84 007020 005037 001350          CLR      SECOND      ;CLEAR THE SECOND'S COUNTER
85 007024 005037 001336          CLR      CFLAG        ;CLEAR THE 'CONTROL C' FLAG
86
87                                ;ROUTINE TO DETERMINE BUFFER AREA SIZE
88
89 007030 005227 177777          SIZMEM: INC      #-1      ;SEE IF TIME TO SIZE MEMORY
90 007034 001002                  BNE      1$            ;BR IF NOT
91 007036 004737 027746          JSR      PC,$SIZE      ;SEE HOW MUCH MEMORY ON SYSTEM
92 007042 013737 030076 001332  1$: MOV      $LSTAD,LSTAD ;SAVE THE LAST ADDRESS
93 007050 023727 001332 160000    CMP      LSTAD,#160C00 ;OVER 28K ?
94 007056 101403                  BLOS     2$            ;NO, THEN DON'T SET THE NEW LIMIT
95 007060 012737 160000 001332    MOV      #160000,LSTAD ;SET NEW LIMIT
96 007066 162737 005670 001332  2$: SUB      #1500.*2,LSTAD ;SAVE XXDP LOADER AND ABSOLUTE LOADER
97
98                                ;SET UP THE OTHER SYSTEM DEVICES THAT THE PROGRAM WILL USE
99
100 007074 004737 021234          SETVEC: JSR      PC,CKCLK ;START THE CLOCK
101 007100 012737 177777 030264    MOV      #-1,SAVEFG ;SET THE SAVE REGISTERS FLAG
102
103                                ;SETUP IF 'XXDP' OR 'ACT11' OPERATION
104
105 007106 005001                  MONTR: CLR      R1      ;DRIVE #
106 007110 005002                  CLR      R2      ;AVAIL TABLE INDEX
107 007112 005003                  CLR      R3      ;DRIVE# X 2
108 007114 016300 004444          1$: MOV      BLKADR(R3),R0 ;LOAD DPB ADDRESS
109 007120 004737 021720          JSR      PC,CLRDPB ;CLEAR DPB BLOCK
110 007124 022322                  2$: CMP      (R3)+,(R2)+ ;INCREMENT INDEX
111 007126 005201                  INC      R1      ;NEXT DRIVE
112 007130 022701 000007          CMP      #7,R1      ;ALL DRIVE ASSIGN ?
113 007134 002367                  BGE      1$          ;NO
114
115
116                                ;ASSIGN LOGICAL DRIVES TO BE TEST IN THE PASS1 AND PASS 2
117                                ;THREE WORDS ARE USED IN THE BIT MAPS:
118                                ;ASNLST = SPECIFIES THE LOGICAL DRIVES ASSIGNED
119                                ;ASSGN1 = SPECIFIES THE LOGICAL DRIVES WILL BE TESTED IN PASS1.
120                                ;ASSGN2 = SPECIFIES THE LOGICAL DRIVES WILL BE TESTED IN PASS2.
121
122                                ;EACH LOGICAL DRIVE HAS A HISTORY FILE LABELED DIRV'Z (Z=0 TO 17)
123                                ;THE LOCATIONS LABELED $SYSNM AND $PHYRD IN THE HISTORY FILE
124                                ;STORE THE SYSTEM NAME (A TO H) AND PHYSICAL DRIVE NUMBER
125                                ; (0 TO 7).
126                                ;THE SUB-SYSTEM ADDRESS AND INTERRUPT VECTOR ARE STORED
127                                ;IN THE TABLE LABELED 'SYSADR:'.
128
129                                ;THE LOCATIONS SYSADR AND SYSADR+2 FOR SUB-SYSTEM A, SYSADR+4
130                                ;AND SYSADR+6 FOR SUBSYSTEM B , ETC, THE FIRST WORD
131                                ;IS THE SUB SYSTEM ADDRESS WHILE THE SECOND WORD IS THE VECTOR
132
133                                ;THE LOGICAL DRIVES ARE ASSIGNED FROM CONSOLE KEYBOARD.
134
135 007136 012700 002006          MOD00: MOV      #ASNLST,R0 ;ADDRESS OF 1ST BIT MAPS IN R0
136 007142 012701 004424          MOV      #INDST,R1      ;LAST ADDRESS TO CLEAR
137 007146 005020                  1$: CLR      (R0)+      ;CLEAR CURRENT POINTED ADDRESS
138 007150 020100                  CMP      R1,R0          ;ALL DONE ?
139 007152 101375                  BHI      1$            ;NO, THEN BRANCH BACK
140 007154 012737 000101 001274    MOV      #'A,$CDW2 ;TEMP STORAGE OF SYS NAME

```

```

141
142
143 007162 005037 001224 MOD21: CLR DRIVE ;TEMP STORAGE OF PHYSICAL DRIVE BIT MAP
144 007166 104401 001207 TYPE ,%CRLF ;CR-LF
145 007172 104401 041776 TYPE ,%MSG1 ;SUB-SYSTEM
146 007176 104401 042644 TYPE ,%QUOTM ;TYPE '""' QUOTATION MARK
147 007202 104401 001274 TYPE ,%SCDW2 ;SYS-NAME(A TO H)
148 007206 104401 042644 TYPE ,%QUOTM ;TYPE '""' QUOTATION MARK
149 007212 104401 042752 TYPE ,%BLNKS1 ;TYPE 1 BLANK
150 007216 104401 042006 TYPE ,%MSG2 ;DRIVE(S)
151
152 007222 104411 RDLIN ;READ IN THE DRIVE NUMBERS
153 007224 012601 MOV (SP)+,R1 ;GET THE INPUT LINE ADDRESS
154 007226 105711 1$: TSTB (R1) ;END OF STRING <CR> ?
155 007230 001437 BEQ 3$ ;YES
156 007232 004537 022410 JSR R5,CK.OCT ;CHECK THE DIGIT MUST 0 TO 7,RETURN VALUE IN R2
157 007236 000751 BR MOD21 ;INCORRECT DRIVE NUMBER,ENTER AGAIN
158 007240 156237 030312 001224 BISB ATABIT(R2),DRIVE ;SET THE PHYS. DRIVE BIT, R2 - DRIVE NUMBER
159 007246 022737 176700 001276 CMP #176700,%RMADR ;IS IT STANDARD RH/RM ADDRESS ?
160 007254 001016 BNE 2$ ;BR IF NO
161 007256 005737 001404 TST XXDP ;IS THIS LOAD DEVICE ?
162 007262 001413 BEQ 2$ ;BR IF NO
163 007264 123702 001404 CMPB XXDP,R2 ;IS THIS THE DRIVE ?
164 007270 001010 BNE 2$ ;BR IF NO
165 007272 104401 043326 TYPE ,%QDRIV ;TYPE ' ?DRIVE'
166 007276 010246 MOV R2,-(SP) ;SAVE R2 FOR TYPEOUT
    007300 104403 ;GO TYPE--OCTAL ASCII
    007302 002 ;TYPE 2 DIGIT(S)
    007303 000 ;SUPPRESS LEADING ZEROS
167 007304 104401 043336 TYPE ,%LODEV ;TYPE 'IS LOAD DEVICE'
168 007310 000724 BR MOD21 ;TRY AGAIN
169 007312 005201 2$: INC R1 ;
170 007314 105711 TSTB (R1) ;END OF STRING <CR> ?
171 007316 001404 BEQ 3$ ;YES
172 007320 122721 000054 CMPB #'',(R1)+ ;MUST BE A COMMA
173 007324 001316 BNE MOD21 ;ENTER AGAIN ,IF NOT
174 007326 000737 BR 1$ ;LOCATE NEXT DRIVE
175
176 007330 005737 001334 3$: TST CHGADR ;START AT 200 ?
177 007334 003035 BGT MOD22 ;BRANCH IF SO
178 007336 013701 001274 MOV %SCDW2,R1 ;SYS NAME ASCII FROM A TO H
179 007342 042701 177760 BIC #177760,R1 ;LEFT ONLY 4 BITS
180 007346 005301 DEC R1 ;ADJUST INDEX VALUE
181 007350 006301 ASL R1 ;2 WORD INDEX VALUE
182 007352 006301 ASL R1 ;
183 007354 062701 002014 ADD #SYSADR,R1 ;SYS ADDRESS TABLE ADDRESS
184 007360 010137 001272 MOV R1,%CDW1 ;SYS ADDRESS TABLE'S ENTRY TO CDW1
185
186 007364 005737 001224 MOD23: TST DRIVE ;CHECK IF ANY PHYSICAL DRIVE(S) ASSIGNED
187 007370 001416 BEQ 2$ ;BRANCH IF NONE
188 007372 013700 001272 1$: MOV %CDW1,R0 ;SYS ADDRESS TABLE ENTRY
189 007376 011037 001276 MOV (R0),%RMADR ;SYS ADDRESS
190 007402 016037 000002 001300 MOV 2(R0),%RMVEC ;SYS VECTOR
191 007410 004737 036300 JSR PC,BUSADR ;CHECK THE ADDRESS WITH THE OPERATOR
192 007414 013710 001276 MOV %RMADR,(R0) ;NEW RH/RM ADDRESS INTO TABLE
193 007420 013760 001300 000002 MOV %RMVEC,2(R0) ;NEW VECTOR OF RH/RM INTO TABLE
194 007426 000406 2$: BR MOD11 ;BRANCH TO NEXT MODULE

```

```

195
196 007430 013737 001276 002014 MOD22: MOV $RMADR,SYSADR ;LOAD THE SYSTEM ADDRESS TABLE
197 007436 013737 001300 002016 MOV $RMVEC,SYSADR+2 ;LOAD THE VECTOR
198
199 ;$CDW2 = ASCII NAME OF SUB SYSTEM (A TO H)
200 ;$CDW1 = ENTRY TO THE SYS ADDRESS TABLE
201 ;DRIVE = PHYSICAL DRIVES TO BE ASSIGNED
202
203 ;THIS SECTION OF CODING USES THE ABOVE PARAMETERS
204 ;TO SET UP THE BIT MAP OF ASNLST,ASSGN1,ASSGN2
205
206 007444 005737 001224 MOD11: TST DRIVE ;ANY DRIVE ASSIGN ?
207 007450 001002 BNE MOD30 ;BR IF YES
208 007452 000137 007624 JMP MOD12 ;BRANCH,IF NONE
209
210 007456 022737 177777 002006 MOD30: CMP #-1,ASNLST ;HAVE ALL 16 LOGICAL DRIVES BEEN ASSIGNED ?
211 007464 001457 BEQ 5$ ;BR IF YES
212 007466 013700 002006 MOV ASNLST,R0 ;FOUND THE AVAILABLE LOGICAL DRIVE LOCATION
213 007472 012701 000001 MOV #1,R1 ;START FROM LOGICAL DRIVE 0
214 007476 005002 CLR R2 ;INDEX VALUE
215 007500 030100 1$: BIT R1,R0 ;IS THE LOGICAL DRIVE AVAILABLE ?
216 007502 001406 BEQ 2$ ;YES
217 007504 000241 CLC
218 007506 006101 ROL R1 ;NEXT LOGICAL DRIVE
219 007510 103445 BCS 5$ ;BRANCH IF NONE IS AVAILABLE
220 007512 062702 000002 ADD #2,R2 ;INCREMENT INDEX VALUE
221 007516 000770 BR 1$ ;LOCATE NEXT LOGICAL DRIVE
222 007520 016204 004444 2$: MOV BLKADR(R2),R4 ;GET THE LOGICAL DRIVE'S HISTORY FILE
223 007524 113764 001274 000014 MOVB $CDW2,$SYSNM(R4) ;LOAD THE ASCII SYS NAME
224 007532 050137 002006 BIS R1,ASNLST ;SET LOGICAL DRIVE ASSIGN BIT
225 007536 050137 002010 BIS R1,ASSGN1 ;SET PASS 1 BIT
226 007542 050137 002012 BIS R1,ASSGN2 ;SET PASS 2 BIT
227 007546 005002 CLR R2 ;DRIVE #
228 007550 136237 030312 001224 3$: BITB ATABIT(R2),DRIVE ;IS THIS DRIVE ASSIGNED ?
229 007556 001005 BNE 4$ ;BR IF YES
230 007560 005202 INC R2 ;PHYSICAL DRIVE NUMBER
231 007562 020227 000007 CMP R2,#7 ;ALL DRIVES DONE YET ?
232 007566 003770 BLE 3$ ;BR IF NO
233 007570 000415 BR 5$ ;YES, EXIT
234 007572 110214 4$: MOVB R2,(R4) ;LOAD THE PHYSICAL DRIVE # INTO HISTORY FILE
235 007574 111464 000015 MOVB (R4),$PHYDR(R4) ;GET PHYSICAL DRIVE NUMBER AND
236 007600 152764 000060 000015 BISB #0,$PHYDR(R4) ;MAKE IT ASCII.
237 007606 112764 000011 000016 MOVB #HT,$GAP(R4) ;MAKE UP FOR SCORE TYPE
238 007614 146237 030312 001224 BICB ATABIT(R2),DRIVE ;DEASSIGN DRIVE BIT FROM LIST
239 007622 001315 BNE MOD30 ;BRANCH IF NOT ALL DONE
240 007624 5$:
241
242 007624 005737 001334 MOD12: TST CHGADR ;200 START ?
243 007630 003100 BGT 7$ ;YES, THEN EXIT
244 007632 022737 177777 002006 1$: CMP #-1,ASNLST ;FULL HOUSE
245 007640 001474 BEQ 7$ ;YES, THEN EXIT
246 007642 104401 042651 2$: TYPE ,MSG5 ;WILL TEST
247 007646 104401 042006 TYPE ,MSG2 ;DRIVE(S)
248 007652 005003 CLR R3 ;INDEX TO LOGICAL DRIVE HISTORY FILE
249 007654 012704 000001 MOV #1,R4 ;BIT MAP OF ASNLST
250 007660 030437 002006 3$: BIT R4,ASNLST ;ASSIGNED LOGICAL DRIVE ?
251 007664 001421 BEQ 5$ ;NO

```

```

252 007666 016305 004444      MOV      BLKADR(R3),R5      ;LOAD THE HISTORY FILE ADDRESS
253 007672 126537 000014 001274  CMPB     $SYSNM(R5),$CDW2      ;ON THE SAME SYSTEM ?
254 007700 001005              BNE      4$              ;NO
255 007702 111546              MOVB     (R5),-(SP)      ;TYPE THE PHYSICAL DRIVE #
256 007704 104403              TYPOS    1
257 007706 001                .BYTE     1
258 007707 000                .BYTE     0
259 007710 104401 042752              TYPE     ,BLNKS1      ;TYPE 1 BLANK
260 007714 062703 000002 4$:      ADD      #2,R3      ;INCREMENT TO NEXT LOGICAL DRIVE
261 007720 000241              CLC
262 007722 006104              ROL      R4      ;BIT MAP OF THE NEXT LOGICAL DRIVE
263 007724 103401              BCS      5$      ;BRANCH IF ALL LOGICAL DRIVE'S CHECKED
264 007726 000754              BR       3$      ;BRANCH BACK
265 007730 104401 042664 5$:      TYPE     ,MSG6      ;ON
266 007734 104401 041776              TYPE     ,MSG1      ;SUB SYSTEM
267 007740 104401 042644              TYPE     ,QUOTM      ;TYPE '"' QUOTATION MARK
268 007744 104401 001274              TYPE     ,SCDW2      ;A TO H
269 007750 104401 042644              TYPE     ,QUOTM      ;TYPE '"' QUOTATION MARK
270
271 007754 005237 001274      INC      $CDW2      ;CHECK NEXT SUB SYSTEM
272 007760 122737 000110 001274  CMPB     #'H,$CDW2      ;ALL EIGHT SUB SYSTEMS CHECKED?
273 007766 103421              BLO      7$      ;YES
274 007770 104401 001207              TYPE     ,$CRLF      ;CR-LF
275 007774 104401 042670              TYPE     ,MSG7      ;ASK FOR OTHER SUB SYSTEM
276 010000 104410              RDCHR
277 010002 012637 001174      MOV      (SP)+,$TMP0      ;GET INPUT CHARACTER
278 010006 023727 001174 000131  CMP      $TMP0, #'Y      ;IS IT INPUT 'YES' ?
279 010014 001004              BNE      6$      ;BR IF NO
280 010016 104401 042741              TYPE     ,Y      ;TYPE 'Y' CHARACTER
281 010022 000137 007162              JMP      MOD21      ;SET UP OTHER SUB SYSTEM
282 010026 104401 042744 6$:      TYPE     ,N      ;TYPE 'N' CHARACTER
283 010032 005737 001334 7$:      TST      CHGADR      ;START AT 210
284 010036 001402              BEQ      8$      ;YES,EXECUTE PASS2 ONLY
285 010040 000137 010054              JMP      XPASS1      ;TO PASS1
286 010044 005037 002010 8$:      CLR      ASSGN1      ;CLEAR THE PASS1 BIT MAP
287 010050 000137 012460              JMP      XPASS2      ;BRANCH TO PASS2
288
289      ;PASS ONE, THE VARIABLES ARE ASSIGNED AS FOLLOWS:
290      ;      $CDW1      = ADDRESS OF THE CURRENT LOGICAL DRIVE HISTORY FILE
291      ;      $CDW2      = SYSTEM NAME A THROUGH H
292      ;      $DEVN      = CURRENT LOGICAL DRIVE #
293      ;      ASNLST      = ASSIGNED LOGICAL DRIVES
294      ;      ASSGN1      = ASSIGNED LOGICAL DRIVES IN THIS PASS
295      ;      R0          = ADDRESS OF DPB BLOCK FEEDED INTO DRIVER-HANDLER
296      ;      R1          = TEMP STORAGE OF ADDRESS OF THE LOGICAL BLOCK
297
298 010054 005737 002010  XPASS1: TST      ASSGN1      ;ANY DRIVES ASSIGNED FOR PASS1
299 010060 001002              BNE      1$      ;BR IF YES, ELSE GO TO
300 010062 000137 011004              JMP      ENDX1      ;END OF PASS 1
301
302 010066 005037 001270 1$:      CLR      $DEVN      ;INDEX OF LOGICAL BLOCK
303 010072 013737 004444 001272      MOV      BLKADR,$CDW1      ;ADDRESS OF LOGICAL BLOCK 0
304 010100 112737 000101 001274      MOVB     #'A,$CDW2      ;SYS NAME STARTS FROM A
305 010106 013737 002014 030324      MOV      SYSADR,RMADR      ;LOAD SYS-TEM A INTO
306 010114 013737 002016 030326      MOV      SYSADR+2,RMVEC      ;DIRVER-HANDLER
307 010122 013701 001272      MOV      $CDW1,R1      ;R1=ADDRESS OF LOGICAL BLOCK 1
308 010126 012777 017444 020172      MOV      #IDLEX,$RMVEC      ;RESET ALL INTERRUPT VECTOR

```

309	010134	005077	020170	CLR	ARMVEC+2	;CLEAR THE INTERRUPT LEVEL
310	010140	104401	001207	TYPE	,SCLF	;CR-LF
311	010144	104401	042021	TYPE	,MSG3	;** STARTING PASS 1
312	010150	104401	042664	TYPE	,MSG6	;ON
313	010154	104401	041776	TYPE	,MSG1	;SUB-SYSTEM
314	010160	104401	042644	TYPE	,QUOTM	;TYPE ''' QUOTATION MARK
315	010164	104401	001274	TYPE	,SCDW2	;SYS-NAME(A TO H)
316	010170	104401	042644	TYPE	,QUOTM	;TYPE ''' QUOTATION MARK
317	010174	111137	001224	MOVB	(R1),DRIVE	;LOAD THE PHYSICAL DRIVE #
318	010200	104401	001207	TYPE	,SCLF	;CR-LF
319	010204	104401	042045	TYPE	,MSG4	;MOUNT PACK ON THE DRIVE
320	010210	104401	001274	TYPE	,SCDW2	;SYS-NAME(A - H)
321	010214	113746	001224	MOVB	DRIVE,-(SP)	;TYPE THE DRIVE #
322	010220	104403		TYPOS		
323	010222	001		.BYTE	1	
324	010223	000		.B TE	0	
325	010224	104401	042752	TYPE	,BLNKS1	;TYPE 1 BLANK
326	010230	104401	042073	TYPE	,MSG8	
327	010234	104401	001207	TYPE	,SCLF	;CR-LF
328	010240	104401	042106	TYPE	,MSG9	
329	010244	104401	042752	TYPE	,BLNKS1	;TYPE 1 BLANK
330						
331	010250	104411		2\$: RDLIN		;CHECK IF O.P. READY
332	010252	012602		MOV	(SP)+,R2	;LOCATE THE INPUT LINE
333	010254	105712		TSTB	(R2)	;FOLLOW BY <CR> ?
334	010256	001374		BNE	2\$	;BRANCH IF NOT
335	010260	004737	030334	JSR	PC,RMINIT	;INITIALIZE THE SUB SYSTEM
336	010264	012737	177777 030264	MOV	#-1,SAVEFG	;SAVE ALL RH/RM REGISTERS
337	010272	012737	177777 030266	MOV	#-1,SEEKFG	;DON'T DO IMPLY SEEK
338	010300	013700	001224	MOV	DRIVE,R0	;R0=PHYSICAL DRIVE # OF THE SUB SYSTEM
339	010304	105760	030206	TSTB	DRVSTA(R0)	;DRIVE EXIST AND ON LINE ?
340	010310	003467		BLE	6\$	;BRANCH IF NOT
341	010312	105760	030216	TSTB	DRVTYP(R0)	;DRIVE IS AN RM05/3/2 ?
342	010316	003464		BLE	6\$	;BRANCH IF NOT
343	010320	116037	030216 001402	MOVB	DRVTYP(R0),DTYP	;GET DRIVE TYPE TO BE TESTED
344	010326	012737	000022 001372	MOV	#18.,TRKLMT	;GET LAST TRACK FOR AN RM05
345	010334	122760	000007 030216	CMPB	#7,DRVTYP(R0)	;IS DRIVE AN RM05 ?
346	010342	001403		BEQ	3\$	;BR IF YES
347	010344	012737	000004 001372	MOV	#4,TRKLMT	;GET LAST TRACK FOR AN RM03/2
348						
349	010352	012700	004506	3\$: MOV	#FMTDPB,R0	;DPB ADDRESS
350	010356	113710	001224	MOVB	DRIVE,(R0)	;LOAD THE DRIVE NUMBER
351	010362	012760	043456 000006	MOV	#BUFFER,\$BUF(R0)	;LOAD BUFFER ADDRESS
352	010370	012760	004526 000014	MOV	#RM.REG,14(R0)	;AREA TO SAVE ALL RH/RM REG'S
353	010376	012760	177400 000004	MOV	#-256.,\$WRDM(R0)	;WORD COUNT (NEG)
354	010404	013760	001374 000012	MOV	CYLIMT,\$CYL(R0)	;CYLINDER 822.
355	010412	113760	001372 000011	MOVB	TRKLMT,\$TRK(R0)	;GET TRACK ADDRESS
356	010420	105060	000010	CLRB	\$SEC(R0)	;SEC 0
357	010424	112760	000171 000002	MOVB	#RDDAT,\$COMND(R0)	;READ DATA COMMAND
358						
359	010432	004037	031062	4\$: JSR	R0,RM05	;CALL THE DRIVER-HANDLER
360	010436	004506		FMTDPB		;PARAMETER ADDRESS
361	010440	000774		BR	4\$	;LOOPING IF QUEUE IS NOT SUCCESSFUL
362	010442	005737	004524	5\$: TST	FMTDPB+16	;COMMAND DONE ?
363	010446	001775		BEQ	5\$	;BRANCH IF NOT
364	010450	100015		BPL	7\$	;BRANCH IF DONE, WITHOUT ERROR
365	010452	062737	000002 004516	ADD	#2,FMTDPB+10	;TRY NEXT SECTOR (0,2,4,6,8)

```

366 010460 122737 000010 004516      CMPB    #8.,FMTDPB+10    ;ALL FIVE SECTORS CHECKED ?
367 010466 101361                      BHI      4$              ;NO,THEN TRY AGAIN
368
369 010470 104401 001207      6$:    TYPE    ,SCRLF          ;CR-LF
370 010474 104401 042142      TYPE    ,MSG10         ;DRIVE IS NOT READY
371 010500 000137 011004      JMP      ENDX1          ;STOP THE TEST
372
373      ;BAD SPOT FILE IS RETRIVED, IS STORED FROM BUFFER+4 TO BUFFER+256. FIRST
374      ;WORD CYLINDER #, SECOND WORD TRK AND SEC NUMBERS, FILE IS TERMINATED BY
375      ;A -1 IN THE CYLINDER NUMBER.
376
377 010504 012704 043466      7$:    MOV      #BUFFER+10,R4      ;R4 ADDRESS OF BAD SPOT FILE
378 010510 022714 177777      8$:    CMP      #-1,(R4)        ;END OF BAD SPOT FILE ?
379 010514 001411                      BEQ      9$              ;YES
380 010516 022704 044456      CMP      #BUFFER+1000,R4      ;END OF BAD SPOT FILE ?
381 010522 101406                      BLOS     9$              ;BRANCH IF IT IS
382 010524 004537 010636      JSR      R5,SPOTX          ;CHECK THE CYLINDER POINTED BY (R4)
383 010530 000422                      BR       11$           ;BRANCH IF BAD SPOT IN THE TEST ZONES
384 010532 062704 000004      ADD      #4,R4              ;NEXT BAD SPOT ADDRESS
385 010536 000764                      BR       8$              ;LOOPING BACK
386
387 010540 032777 000200 170406 9$:    BIT      #BIT7,@SWR      ;SWITCH 7 SET ?
388 010546 001404                      BEQ      10$           ;BRANCH IF NOT SET
389 010550 012700 004506      MOV      #FMTDPB,R0          ;DPB ADDRESS
390 010554 004737 020702      JSR      PC,PRTBAD        ;PRINT THE BAD SPOT FILE
391 010560 105760 000010      10$:   TSTB     $SEC(R0)        ;SECTOR 10 HAS BEEN READ ?
392 010564 001022                      BNE      13$           ;BRANCH IF SO
393 010566 112760 000012 000010      MOVB    #10.,$SEC(R0)    ;READ SECTOR 10
394 010574 000716                      BR       4$              ;LOOPING BACK
395
396 010576 104401 042207      11$:   TYPE    ,MSG11         ;PACK NOT ACCEPTABLE
397 010602 104401 001207      TYPE    ,SCRLF          ;CR-LF
398 010606 032777 000200 170340      BIT      #BIT7,@SWR      ;SWITCH 7 SET ?
399 010614 001404                      BEQ      12$           ;BRANCH IF NOT SET
400 010616 012700 004506      MOV      #FMTDPB,R0          ;DPB ADDRESS
401 010622 004737 020702      JSR      PC,PRTBAD        ;TYPE THE BAD SPOT FILE
402 010626 000137 010054      12$:   JMP      XPASS1          ;RESTART PASS 1
403 010632 000137 011020      13$:   JMP      TST1              ;PROCEED TO TEST 1
404
405
406      ;SUBROUTINE SPOTX
407      ;SEE IF THE CYLINDER POINTED BY (R4) IS IN THE TESTING ZONES
408      ;BELOW:
409      ;
410      ;(0-15, 128-143, 256-271, 384-399, 512-527, 640-655, 768-783,
411      ;112-127, 240-255, 368-383, 496-511, 624-639, 752-767, 17-32,
412      ;145-160, 273-288, 401-416, 529-544, 657-672, 785-800 AND 620)
413      ;
414      ;CALL
415      ;      JSR      R5,SPOTX          ;R4=POINT TO CYLINDER NUMBER
416      ;      RET1     ;ERROR RET
417      ;      RET2     ;NORMAL RET1
418
419 010636 010146      SPOTX:  MOV      R1,-(SP)          ;SAVE R1 THROUGH R3
420 010640 010246      MOV      R2,-(SP)
421 010642 010346      MOV      R3,-(SP)
422

```

```

423 010644 005003          CLR      R3          ;ERROR FLAG
424 010646 005046          CLR      -(SP)        ;DUMMY PAIR
425 010650 005046          CLR      -(SP)        ;DUMMY PAIR
426 010652 005046          CLR      -(SP)        ;ZONE STARTING ADDRESS
427 010654 012746 000007    MOV      #7, -(SP)    ;SEGMENT NUMBER
428 010660 012746 000021    MOV      #17, -(SP)   ;ZONE STARTING ADDRESS
429 010664 012746 000007    MOV      #7, -(SP)    ;SEGMENT NUMBER
430 010670 012701 000160    MOV      #12, R1      ;R1=ZONE STARTING ADDRESS
431 010674 012702 000006    MOV      #6, R2      ;R2=SEGMENT NUMBER
432 010700 021401          1$:  CMP      (R4), R1  ;CYL IN THE ZONE ?
433 010702 103410          BLO      3$           ;BRANCH IF NOT
434 010704 062701 000017    ADD      #15, R1      ;CHECK WITH THE UPPER BOND
435 010710 021401          CMP      (R4), R1      ;CYL IN THE ZONE ?
436 010712 101002          BHI      2$           ;BRANCH IF NOT
437 010714 052703 000002    BIS      #BIT1, R3    ;SET THE ERROR FLAG
438
439 010720 162701 000017    2$:  SUB      #15, R1  ;RESTORE TO THE LOWER BOND
440 010724 005302          3$:  DEC      R2        ;DECREMENT THE SEGMENT COUNT
441 010726 001403          BEQ      4$           ;ALL SEGMENT CHECKED ?
442 010730 062701 000200    ADD      #128, R1     ;ADJUST ZONE STARTING ADDRESS
443 010734 000761          BR       1$           ;LOOPING BACK UNTIL ALL SEGMENTS ARE CHECKED
444
445 010736 012602          4$:  MOV      (SP)+, R2 ;POP THE NEXT SET OF ZONE PARAMETERS
446 010740 012601          MOV      (SP)+, R1
447 010742 005702          TST      R2           ;DUMMY PAIR
448 010744 001355          BNE      1$           ;BRANCH IF NOT
449 010746 005701          TST      R1           ;DUMMY PAIR
450 010750 001353          BNE      1$           ;BRANCH IF NOT
451
452 010752 021427 001154    5$:  CMP      (R4), #620 ;ON CYLINDER 620 ?
453 010756 001002          BNE      6$           ;NO
454 010760 052703 000002    BIS      #BIT1, R3    ;SET ERROR FLAG
455 010764 005703          6$:  TST      R3        ;ANY ERROR ?
456 010766 001002          BNE      7$           ;YES
457 010770 062705 000002    ADD      #2, R5       ;ADJUST FOR NORMAL RETURN
458 010774 012603          7$:  MOV      (SP)+, R3 ;RESTORE R3 THROUGH R1
459 010776 012602          MOV      (SP)+, R2
460 011000 012601          MOV      (SP)+, R1
461 011002 000205          RTS      R5           ;EXIT
462
463 011004 104401 001207    ENDX1: TYPE    , $CRLF   ;CR-LF
464 011010 104401 043044    TYPE      , MSG21    ;DRIVE NOT ONLINE OR NOT ASSIGNED
465 011014 000177 170360    JMP      @RSTART ;JUMP TO RESTART
466
467      ;THE FOLLOWING CODING FOR TEST 1 THROUGH TEST 4
468      ;PARAMETER IN TST 1
469      ;      $CDW1 = ADDRESS OF LOGICAL DRIVE BLOCK
470      ;      DRIVE = PHYSICAL DRIVE #
471      ;      $DEVM = LOGICAL DRIVE # 0-17
472      ;      ASSGN1 = ASSIGN LOGICAL DRIVE BIT MAP
473      ;      ASNLST = ASSIGNED LOGICAL DRIVE MAP INDICATOR
474      ;      $CDW2 = SYS-NAME
475
476      ;$CDW2, DRIVE ARE ONLY CHANGED IN TST1 DURING PASS 1
477      ;$DEVM, ASSGN1, $CDW1 ARE CHANGED BY THE TEST 4
478
479      ;TST 1: DIRECT OPERATOR TO MOUNT AND LOAD PACKS

```

480
481

```

011020 000004
011022 012737 000001 001176
482 011030 012737 000001 001340
483 011036 012706 001100
484 011042 023737 001272 004444
485 011050 001551
486 011052 013701 001272
487 011056 126137 000014 001274
488 011064 001426
489 011066 116137 000014 001274
490 011074 012777 017444 017224
491 011102 005077 017222
492 011106 104401 001207
493 011112 104401 042021
494 011116 104401 042664
495 011122 104401 041776
496 011126 104401 042644
497 011132 104401 001274
498 011136 104401 042644
499 011142 111137 001224
500 011146 104401 001207
501 011152 104401 042045
502 011156 104401 001274
503 011162 113746 001224
504 011166 104403
505 011170 001
506 011171 000
507 011172 104401 042752
508 011176 104401 042073
509 011202 104401 001207
510 011206 104401 042106
511
512 011212 104411
513 011214 012605
514 011216 105715
515 011220 001374
516 011222 113701 001274
517 011226 042701 177760
518 011232 005301
519 011234 006301
520 011236 006301
521 011240 016137 002014 030324
522 011246 016137 002016 030326
523 011254 004737 030334
524 011260 012737 177777 030264
525 011266 012737 177777 030266
526 011274 013700 001224
527 011300 105760 030206
528 011304 003426
529 011306 105760 030216
530 011312 003423
531 011314 122737 000007 001402
532 011322 001005
533 011324 122760 000007 030216
534 011332 001420

```

```

*****
TST1: SCOPE
MOV #1,STIMES ;DO 1 ITERATION
MOV #1,TSTNM ;LOAD THE TEST NUMBER
MOV #STACK,SP ;INITIALIZE THE STACK POINT
CMP $CDW1,BLKADR ;LOGICAL DRIVE 0
BEQ 6$ ;THEN EXIT
MOV $CDW1,R1 ;R1=LOGICAL DRIVE BLOCK ADDRESS
CMPB $SYSNM(R1),$CDW2 ;STILL ON THE SAME SYSTEM ?
BEQ 1$ ;THEN ,EXIT
MOVB $SYSNM(R1),$CDW2 ;LOAD THE NEW SUB SYSTEM NAME
MOV #IDLEX,@RMVEC ;RESET THE INTERRUPT VECTOR
CLR @RMVEC+2 ;CLEAR THE INTERRUPT LEVEL
TYPE ,SCRLF ;CR-LF
TYPE ,MSG3 ;** STARTING PASS 1
TYPE ,MSG6 ;ON
TYPE ,MSG1 ;SUB-SYSTEM
TYPE ,QUOTM ;TYPE "" QUOTATION MARK
TYPE ,SCDW2 ;SYS-NAME(A TO H)
TYPE ,QUOTM ;TYPE "" QUOTATION MARK
1$: MOVB (R1),DRIVE ;LOAD THE PHYSICAL DRIVE #
TYPE ,SCRLF ;CR-LF
TYPE ,MSG4 ;MOUNT PACK ON THE DRIVE
TYPE ,SCDW2 ;SYS-NAME(A TO H)
MOVB DRIVE,-(SP) ;THE PHYSICAL DRIVE #
TYP0S
.BYTE 1
.BYTE 0
TYPE ,BLNKS1 ;TYPE 1 BLANK
TYPE ,MSG8 ;AND LOAD
TYPE ,SCRLF ;CR-LF
TYPE ,MSG9

2$: RDLIN
MOV (SP)+,R5 ;LOCATE THE READIN LINE
TSTB (R5) ;NOT CORRECT INPUT LINE FORMAT
BNE 2$ ;BRANCH IF NOT
MOVB $CDW2,R1 ;LOCATE THE SYSTEM ADDRESS TABLE
BIC #177760,R1 ;LEFT ON 4 BITS
DEC R1 ;ADJUST THE INDEX VALUE
ASL R1 ;FOUR WORD INDEX VALUE
ASL R1
MOV SYSADR(R1),RMADR ;LOAD THE SYSTEM ADDRESS
MOV SYSADR+2(R1),RMVEC ;LOAD THE SYSTEM INTERRUPT VECTOR
JSR PC,RMINIT ;INITIALIZE THE SYSTEM
MOV #-1,SAVEFG ;SAVE ALL RH/RM REGISTER
MOV #-1,SEEKFG ;DON'T DO ANY IMPLY SEEK
MOV DRIVE,R0 ;R0=PHYSICAL DRIVE #
TSTB DRVSTA(R0) ;ON-LINE ?
BLE 5$ ;BRANCH IF NOT
TSTB DRVTP(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,DRVTP(R0) ;SEE IF DRIVE IS STILL AN RM05.
BEQ 6$ ;BR IF YES

```


0042

0055

```

590 011624 003730      BLE      2$      ;NO
591 011626 000410      BR       TST3     ;BRANCH TO NEXT TEST
592 011630 104401 001207 6$:      TYPE    ,SCRLF   ;CR-LF
593 011634 104401 043125      TYPE    ,HALT1
594 011640 104401 043106      TYPE    ,HALTX
595 011644 000177 167530      JMP      @RSTART ;JUMP TO RESTART
596
597      ;TEST 3
598      ;WRITE 7 ZONES FOR WRITE TEST IN PASS 2
599      ;WRITE 6 ZONES FOR READ TEST IN PASS2
600      :      $DEV#    = LOGICAL DRIVE #
601      :      $CDW1    = ADDRESS OF LOGICAL DRIVE HISTORY BLOCK FILE
602      :      $CDW2    = SYS NAME
603      :      DRIVE    = PHYSICAL DRIVE # OF THIS LOGICAL DRIVE
604      :      PACK     = ENTRY OF TABLE-D
605      :      CMCNT    = ZONE COUNT
606      :      CMSEC    = DELTA CYLINDER COUNT
607      :      R4       = ENTRY POINTER OF TABLE-D, CANNOT BE DESTROYED.
608      :      R0       = ADDRESS OF FMTDPB
609
610      ;*****
611      TST3:  SCOPE
612      MOV     #1,$TIMES      ;;DO 1 ITERATION
613      MOV     #3,TSTNM      ;LOAD THE TEST NUMBER
614      MOV     #STACK,SP     ;INITIAL THE STACK POINTER
615      MOV     #LOGO,R4      ;ADDRESS OF TABLE-D
616      MOV     $DEV# ,R0     ;LOGICAL DRIVE #
617      BEQ     2$           ;BRANCH IF LOGICAL DRIVE 0
618      ADD     #16.,R4       ;EACH LOGICAL DRIVE TAKES 16 BYTES IN THE TABLE
619      DEC     R0            ;DECREMENT THE DRIVE # COUNT
620      BNE     1$           ;BRANCH ,UNTIL THE ENTRY IS LOCATED
621      MOV     R4,PACK       ;SAVE THE TABLE-D ENTRY IN PACK
622      SETUP   THE FMTDPB BLOCK
623      MOV     #FMTDPB,R0    ;ADDRESS OF FMTDPB
624      MOV     DRIVE,(R0)    ;PHYSICAL DRIVE NUMBER
625      MOV     #WRTDAT,$CMD(R0) ;WRITE DATA COMMAND
626      MOV     #BUFFER,$BUF(R0) ;BUFFER ADDRESS
627      MOV     #RM.REG,14(R0) ;ADDRESS TO SAVE ALL RH/RM REG'S
628      MOV     #-256.,$WRDM(R0) ;NEG WORD COUNT
629      MOV     $DEV# ,R1     ;LOGICAL DRIVE #
630      ASL     R1            ;WORD INDEX
631      MOV     PSEUDO(R1),R4 ;LOAD THE DATA PATTERN
632      MOV     $BUF(R0),R2   ;BUFFER ADDRESS IN R2
633      MOV     #256.,R3      ;POS WORD COUNT
634      MOV     R4,(R2)+      ;FULL THE BUFFER WITH SIGLE WORD PATTERN
635      DEC     R3            ;DECREMENT THE WORD COUNT
636      BNE     3$           ;BRANCH,UNTIL IT IS FULL
637
638      CLR     -(SP)         ;DUMY PAIR OF ZONE COUNT
639      CLR     -(SP)         ;DUMY STARTING CYLINDER NUMBER
640      MOV     #6,-(SP)      ;ZONE COUNT
641      MOV     #112.,-(SP)   ;STARTING CYLINDER NUMBER
642      MOV     #7,CMCNT      ;ZONE COUNT
643      CLR     CMCYL         ;STARTING CYLINDER NUMBER
644      MOV     #16.,CMSEC    ;DELTA CYLINDER NUMBER
645      MOV     #FMTDPB,R0    ;R0 DPB ADDRESS

```

```

645 012050 013704 001324      MOV      PACK,R4      ;R4 ENTRY TO TABLE-D
646 012054 013760 001356 000012 5$:  MOV      (CMCYL,$CYL(R0)) ;STARTING CYLINDER
647 012062 105060 000011      CLRB     $TRK(R0)      ;STARTS FROM TRACK 0
648 012066 111460 000010      MOVB     (R4),$SEC(R0) ;LOAD SECTOR NUMBER FROM TABLE-D
649 012072 004037 031062      JSR      R0,RM05      ;CALL THE DRIVER
650 012076 004506      FMTDPB
651 012100 000774      BR      7$      ;BRANCH IF QUEU IS NOTSUCCESSFUL
652 012102 005737 004524      TST      FMTDPB+16      ;DONE ?
653 012106 001775      BEQ      8$      ;BRANCH IF NOT
654 012110 012700 004506      MOV      #FMTDPB,R0
655 012114 004737 017452      JSR      PC,PROCES      ;PROCESS TO CHECK IF ANY ERROR
656 012120 005760 000016      TST      16(R0)      ;ERROR FLAG SET ?
657 012124 100453      BMI      9$      ;BRANCH IF SO
658 012126 062760 000020 000010      ADD      #16,$SEC(R0) ;WRITE TWO SECTORS ON ONE CYLINDER
659 012134 122760 000037 000010      CMPB     #31,$SEC(R0) ;ALL DONE-TWO SECTORS
660 012142 103353      BHS      7$      ;BRANCH IF NOT
661 012144 111460 000010      MOVB     (R4),$SEC(R0) ;RESTORE SECTOR #
662 012150 105260 000011      INCB     $TRK(R0)      ;INCREMENT TO NEXT TRACK
663 012154 126037 000011 001372      CMPB     $TRK(R0),TRKLMT ;LAST TRACK ?
664 012162 003743      BLE      7$      ;NO, THEN BRANCH
665 012164 105060 000011      CLRB     $TRK(R0)      ;RESTORE TO TRACK-0
666 012170 005260 000012      INC      $CYL(R0)      ;INCREMENT CYLINDER NUMBER
667 012174 005204      INC      R4      ;INCREMENT TABLE-D ENTRY
668 012176 005337 001362      DEC      CMSEC      ;DECREMENT THE DELTA CYLINDER COUNT
669 012202 001331      BNE      6$      ;BRANCH IF NOT END OF THIS BLOCK
670 012204 062760 000160 000012      ADD      #112,$CYL(R0) ;INCREMENT THE CYLINDER NUMBER TO NEXT ZONE
671 012212 016037 000012 001356      MOV      $CYL(R0),CMCYL ;INITAIL THE STARTING CYLINDER IN THE BLOCK
672 012220 005337 001354      DEC      CMCNT      ;DECREMENT THE ZONE COUNT
673 012224 001304      BNE      4$      ;LOOPING IF NOT END OF ZONE
674 012226 012637 001356      MOV      (SP)+,CMCYL      ;LOAD NEW PAIR OF STARTING CYLINDER
675 012232 012637 001354      MOV      (SP)+,CMCNT      ;AND ZONE COUNT
676 012236 005737 001356      TST      CMCYL      ;NOT END YET ?
677 012242 001275      BNE      4$      ;BRANCH IF NOT
678 012244 005737 001354      TST      CMCNT      ;BRANCH IF NOT END
679 012250 001272      BNE      4$      ;LOOPING BACK
680
681 012252 000410      BR      TST4      ;BRANCH TO THE NEXT TEST
682 012254 104401 001207 9$:  TYPE     ,$CRLF      ;CR-LF
683 012260 104401 043176      TYPE     ,HALT2
684 012264 104401 043106      TYPE     ,HALTX
685 012270 000177 167104      JMP      @RSTART      ;JUMP TO RESTART
686
687      ;TEST 4
688      ;UPDATE THE PARAMETERS,$CDW1,$DEVM,ASSGN1
689      ;DIRECT THE OPERATOR TO DISMOUNT PACK AND LOAD TO OTHER DRIVE
690
691      ;$CDW2,DRIVE ARE CHANGED BY TEST ONE ONLY AFTER THE TEST LOOPING TO TEST1
692
693      ;*****
TST4:  SCOPE
694 012274 000004      MOV      #1,$TIMES      ;;DO 1 ITERATION
695 012276 012737 000001 001176      MOV      #4,TSTNM      ;LOAD THE TEST NUMBER
696 012304 012737 000004 001340      MOV      #STACK,SP      ;LOAD THE STACK POINTER
697 012312 012706 001100      MOV      #IDLE,@RMVEC      ;RESET THE INTERRUPT VECTOR
698 012316 012777 017444 016002      CLR      @RMVEC+2      ;CLEAR THE INTERRUPT LEVEL
699 012324 005077 016000      TYPE     , $CRLF      ;CR-LF
012330 104401 001207      TYPE     ,MSG12      ;UNLOAD DRIVE
012334 104401 042272

```

```

700 012340 104401 001274      TYPE      ,SCDW2      ;SYS-NAME(A - H)
701 012344 013746 001224      MOV        DRIVE,-(SP) ;PHYSICAL DRIVE #
702 012350 104403              TYPOS
703 012352      001          .BYTE      1
704 012353      000          .BYTE      0
705 012354 104401 042310      TYPE      ,MSG13
706 012360 012701 000001      MOV        #1,R1
707 012364 005002              CLR        R2
708 012366 020237 001270      1$:      CMP        R2,$DEVM      ;LOCATE THE CORESPONDING BIT MAP
709 012372 001404              BEQ        2$      ;BRANCH IF LOCATED
710 012374 000241              CLC              ;LOCATE NEXT DRIVE
711 012376 006101              ROL              ;
712 012400 005202              INC              ;NEXT DRIVE #
713 012402 000771              BR          1$      ;LOCATE THE BIT MAP
714 012404 040137 002010      2$:      BIC        R1,ASSGN1      ;DEASSIGN THE LOGICAL DRIVE FOR PASS 1
715 012410 001410              BEQ        4$      ;NO MORE DRIVES
716 012412 005202              3$:      INC        R2      ;GET NEXT LOGICAL DRIVE #
717 012414 006302              ASL        R2      ;WORD INDEX
718 012416 016237 004444 001272      MOV        BLKADR(R2),SCDW1      ;LOAD THE NEW DPB ADDRESS
719 012424 006202              ASR        R2      ;RESTORE R2
720 012426 010237 001270      MOV        R2,$DEVM      ;LOAD THE NEW LOGICAL DRIVE #
721
722 012432 104411              4$:      RDLIN      ;WAIT UNTIL IT IS DONE
723 012434 012605              MOV        (SP)+,R5      ;LOCATE THE INPUT LINE
724 012436 105715              TSTB       (R5)      ;TERMINATOR ?
725 012440 001374              BNE        4$      ;BRANCH IF NOT
726 012442 005737 002010      TST        ASSGN1      ;OTHER DRIVES ?
727 012446 001002              BNE        5$      ;BRANCH IF MORE DRIVES IN TEST
728 012450 000137 012460      JMP        XPASS2      ;BRANCH TO PASS 2
729 012454 000137 011020      5$:      JMP        TST1      ;JUMP TO TEST 1
730
731
732      ;XPASS2      INITILIZE FOR PASS 2 TEST
733      :
734      :      $CDW1      = ADDRESS OF THE CURRENT LOGICAL DRIVE HISTORY FILE
735      :      $CDW2      = SYSTEM NAME A THROUGH H
736      :      $DEVM      = CURRENT LOGICAL DRIVE # 0 TO 15.
737      :      ASSGN2      = ASSIGNED LOGICAL DRIVE FOR PASS 2
738      :      ASNLST      = ASSIGNED LOGICAL DRIVE
739      :      DRIVE      = PHYSICAL DRIVE # OF CURRENT RH/RM SYSTEM
740
741 012460 005737 002012      XPASS2: TST        ASSGN2      ;ANYTHING IN TEST FOR PASS 2
742 012464 001002              BNE        1$      ;YES, THEN GO ON
743 012466 000137 017146      JMP        XEND2      ;JUMP TO END OF PASS 2
744
745 012472 005037 001270      1$:      CLR        $DEVM      ;START FROM LOGICAL DRIVE 0
746 012476 013737 004444 001272      MOV        BLKADR,$CDW1      ;ADDRESS OF LOGICAL BLOCK DRIVE 0
747 012504 112737 000101 001274      MOVB       #'A,$CDW2      ;LOAD SYSTEM NAME 'A'
748 012512 013737 000114 030324      MOV        SYSADR,RMADR      ;LOAD SYSTEM-A ADDRESS TO DRIVER
749 012520 013737 002016 030326      MOV        SYSADR+2,RMVEC      ;LOAD SYSTEM-A VECTOR TO DRIVER
750 012526 013701 001272      MOV        $CDW1,R1      ;R1=ADDRESS OF LOGICAL BLOCK
751 012532 012777 017444 015566      MOV        #IDLEX,@RMVEC      ;RESET THE INTERRUPT VECTOR
752 012540 005077 015564      CLR        @RMVEC+2      ;CLEAR THE INTERRUPT LEVEL
753 012544 005037 001376      CLR        FAULT      ;RESET THE NOT COMPATIBLE FLAG
754 012550 104401 001207      TYPE      ,SCRLF      ;CR-LF
755 012554 104401 042356      TYPE      ,MSG14      ;START THE PASS 2
756 012560 104401 042664      TYPE      ,MSG6      ;ON

```

```

757 012564 104401 041776      TYPE      ,MSG1      ;SUB-SYSTEM
758 012570 104401 042644      TYPE      ,QUOTM      ;TYPE "" QUOTATION MARK
759 012574 104401 001274      TYPE      ,SCDW2      ;SYS-NAME(A TO H)
760 012600 104401 042644      TYPE      ,QUOTM      ;TYPE "" QUOTATION MARK
761 012604 111137 001224      MOV      (R1),DRIVE ;LOCATE THE PHYSICAL DRIVE #
762 012610 104401 001207      TYPE      ,$CRLF      ;CR-LF
763 012614 104401 042045      TYPE      ,MSG4      ;MOUNT PACK ON DRIVE
764 012620 104401 001274      TYPE      ,SCDW2      ;SYSTEM NAME
765 012624 111146      MOV      (R1),-(SP)
766 012626 104403      TYP      ,POS
767 012630      001      .BYTE      1
768 012631      C00      .BYTE      0
769 012632 104401 042752      TYPE      ,BLNKS1      ;TYPE 1 BLANK
770 012636 104401 042073      TYPE      ,MSG8      ;AND LOAD
771 012642 104401 001207      TYPE      ,$CRLF      ;CR-LF
772 012646 104401 042106      TYPE      ,MSG9      ;TYPE <CR> WHEN DRIVE IS READY
773
774 012652 104411      2$:      RDLIN
775 012654 012602      MOV      (SP)+,R2      ;LOCATE THE HEAD IN LINE
776 012656 105712      TST      (R2)          ;CARRIAGE RETURN ?
777 012660 001374      BNE      2$            ;BR IF NO
778 012662 004737 030334      JSR      PC,RMINIT
779 012666 012737 177777 030264      MOV      #-1,SAVEFG      ;SAVE ALL RH/RM REGISTERS
780 012674 012737 177777 030266      MOV      #-1,SEEKFG      ;DON'T DO IMPLY SEEK
781 012702 013700 001224      MOV      DRIVE,R0      ;LOAD THE PHYSICAL DRIVE NUMBER
782 012706 105760 030206      TST      DRVSTA(R0)      ;DRIVE EXISTS AND ON-LINE ?
783 012712 003421      BLE      4$            ;BRANCH IF NOT
784 012714 105760 030216      TST      DRVTP(R0)        ;CHECK DRIVE TYPE
785 012720 003416      BLE      4$            ;BR IF NOT AN RM05/3/2
786 012722 116037 030216 001402 3$:      MOV      DRVTP(R0),DTYP      ;GET DRIVE TYPE TO BE TESTED
787 012730 012737 000022 001372      MOV      #18,TRKLMT      ;GET LAST TRACK FOR AN RM05
788 012736 122760 000007 030216      CMP      #7,DRVTP(R0)      ;IS DRIVE AN RM05 ?
789 012744 001412      BEQ      5$            ;BR IF YES
790 012746 012737 000004 001372      MOV      #4,TRKLMT      ;GET LAST TRACK FOR AN RM03/2
791 012754 000406      BR      5$
792
793 012756 104401 001207      4$:      TYPE      ,CRLF      ;CR-LF
794 012762 104401 042142      TYPE      ,MSG10      ;DRIVE IS NOT READY
795 012766 000137 011004      JMP      ENDX1
796 012772      5$:
797
798      ;      $CDW1      = ADDRESS OF CURRENT LOGICAL DRIVE BLOCK,ONLY CHANGED BY TST9
799      ;      $CDW2      = SYSTEM-NAME, ONLY CHANGED BY TEST 5
800      ;      $DEVM      = CURRENT LOGICAL DRIVE #
801      ;      ASSGN2      = ASSIGNED LOGICAL DRIVE'S BIT MAP FOR PASS 2
802      ;      DRIVE      = PHYSICAL DRIVE #
803      ;      ASNLST      = ASSIGNED LOGICAL DRIVE IN THE TEST
804
805      ;IN TEST 5 DIRECT OPERATOR TO CHANGE, LOAD THE PACK
806
807      ;*****
808      TST5:      SCOPE
809      MOV      #1,$TIMES      ;DO 1 ITERATION
810      MOV      #5,TSTNM      ;LOAD THE TEST NUMBER
811      MOV      #STACK,SP      ;LOAD THE STACK POINTER
      CMP      $CDW1,BLKADR      ;LOGICAL DRIVE 0
      BEQ      6$              ;THEN EXIT

```

812	013024	013701	001272		MOV	\$CDW1,R1	:ADDRESS OF THE HISTORY FILE
813	013030	126137	000014	001274	CMPS	\$SYSNM(R1),\$CDW2	:ON THE SAME SUB-SYSTEM
814	013036	001443			BEQ	1\$	:THEN DON'T UPDATE SYSTEM ADDRESS
815	013040	116137	000014	001274	MOVB	\$SYSNM(R1),\$CDW2	
816	013046	012777	017444	015252	MOV	#IDLEX,@RMVEC	:RESET THE INTERRUPT VECTOR
817	013054	005077	015250		CLR	@RMVEC+2	:CLEAR THE INTERRUPT LEVEL
818	013060	013700	001274		MOV	\$CDW2,R0	:LOCATE SYSTEM ADDRESS TABLE
819	013064	005300			DEC	R0	:ADJUST FOR INDEX FORM 0
820	013066	042700	177760		BIC	#177760,R0	:LEFT ON FOUR BITS
821	013072	006300			ASL	R0	
822	013074	006300			ASL	R0	:INDEX FOR TWO WORD
823	013076	016037	002014	030324	MOV	SYSADR(R0),RMADR	:SYSTEM ADDRESS
824	013104	016037	002016	030326	MOV	SYSADR+2(R0),RMVEC	:SYSTEM INTERRUPT VECTOR
825	013112	104401	001207		TYPE	,\$CRLF	:CR-LF
826	013116	104401	042356		TYPE	,\$MSG14	:START THE PASS 2
827	013122	104401	042664		TYPE	,\$MSG6	:ON
828	013126	104401	041776		TYPE	,\$MSG1	:SUB-SYSTEM
829	013132	104401	042644		TYPE	,\$QUOTM	:TYPE ''' QUOTATION MARK
830	013136	104401	001274		TYPE	,\$CDW2	:SYS-NAME(A TO H)
831	013142	104401	042644		TYPE	,\$QUOTM	:TYPE ''' QUOTATION MARK
832	013146	111137	001224	1\$:	MOVB	(R1),DRIVE	:LOCATE THE PHYSICAL DRIVE #
833	013152	104401	001207		TYPE	,\$CRLF	:CR-LF
834	013156	104401	042045		TYPE	,\$MSG4	:MOUNT PACK ON DRIVE
835	013162	104401	001274		TYPE	,\$CDW2	:SYSTEM NAME
836	013166	111146			MOVB	(R1),-(SP)	
837	013170	104403			TYPOS		
838	013172	001			.BYTE	1	
839	013173	000			.BYTE	0	
840	013174	104401	042752		TYPE	,\$BLNKS1	:TYPE 1 BLANK
841	013200	104401	042073		TYPE	,\$MSG8	:AND LOAD
842	013204	104401	001207		TYPE	,\$CRLF	:CR-LF
843	013210	104401	042106		TYPE	,\$MSG9	:TYPE <CR> WHEN DRIVE IS READY
844							
845	013214	104411		2\$:	RDLIN		
846	013216	012602			MOV	(SP)+,R2	:LOCATE THE READ IN LINE
847	013220	105712			TSTB	(R2)	:CARRIAGE RETURN ?
848	013222	001374			BNE	2\$	:BR IF NO
849	013224	004737	030334		JSR	PC,RMINIT	
850	013230	012737	177777	030264	MOV	#-1,SAVEFG	:SAVE ALL RH/RM REGISTERS
851	013236	012737	177777	030266	MOV	#-1,SEEKFG	:DON'T DO IMPLY SEEK
852	013244	013700	001224		MOV	DRIVE,R0	:LOAD THE PHYSICAL DRIVE NUMBER
853	013250	105760	030206		TSTB	DRVSTA(R0)	:DRIVE EXISTS AND ON-LINE ?
854	013254	003426			BLE	5\$	:BRANCH IF NOT
855	013256	105760	030216		TSTB	DRVTYP(R0)	:CHECK DRIVE TYPE
856	013262	003423			BLE	5\$	:BR IF NOT AN RM05/3/2
857	013264	122737	000007	001402	CMPS	#7,DTYP	:WHAT WAS FIRST DRIVE TESTED ?
858	013272	001005			BNE	3\$	:BRANCH IF AN RM02 OR RM03, ELSE
859	013274	122760	000007	030216	CMPS	#7,DRVTYP(R0)	:SEE IF DRIVE IS STILL AN RM05.
860	013302	001420			BEQ	6\$	:BR IF YES
861	013304	000404			BR	4\$	:ERROR ENCOUNTERED
862	013306	122760	000007	030216	CMPS	#7,DRVTYP(R0)	:SEE IF DRIVE IS STILL AN RM02 OR RM03.
863	013314	001013			BNE	6\$	:BR IF YES
864	013316	104401	001207	4\$:	TYPE	,\$CRLF	:CR-LF
865	013322	104401	043356		TYPE	,\$NOTST	:CANNOT SELECT RM03/2'S AND RM05'S TOGETHER
866	013326	000177	165046		JMP	@RSTART	:JUMP TO RESTART
867							
868	013332	104401	001207	5\$:	TYPE	,\$CRLF	:CR-LF

```

869 013336 104401 042142      TYPE      .MSG10      ;DRIVE IS NOT READY
870 013342 000613      BR      IST5      ;TRY AGAIN
871 013344      6$:
872
873      :      DRIVE      = PHYSICAL DRIVE NUMBER
874      :      $CDW1      = DPB BLOCK OF THIS LOGICAL DRIVE
875      :      $DEVN      = LOGICAL DRIVE #
876      :      $CDW2      = SUB-SYSTEM NAME
877      :      RMADR      = SUB-SYSTEM BASE REGISTER ADDRESS
878      :      RMVEC      = SUB-SYSTEM INTERRUPT VECTOR
879
880      :THE ABOVE PARAMETERS CANNOT BE MODIFIED BY THIS TEST (TST6)
881      :THE FOLLOWING REG'S ARE ASSIGNED AS:
882      :      R0      = ADDRESS OF DPB FIELD INTO DRIVER HANDLER=FMTDPB
883      :      R1      = ADDRESS OF THE HISTORY FILE BLOCK OF THE LOGICAL DRIVE-$CDW1
884
885      :IN TEST 6, EACH LOGICAL DRIVE WRITES 7 CYLINDERS ON EACH TRACK WITH
886      :A UNIQUE DATA PATTERN. THESE 7 CYLINDERS HAVE BEEN WRITTEN BY OTHER
887      :DRIVES IN PASS 1.
888      :THEN, THIS LOGICAL DRIVE EXECUTES 'WRITE CHECK DATA' TO SEE IF THIS
889      :LOGICAL DRIVE CAN OVER-WRITE ALL DATA WRITTEN BY OTHER DRIVES.
890
891      :THESE 7 CYLINDERS ARE SPECIFIED AS $DEVN, $DEVN+128, $DEVN+256, $DEVN+384,
892      :$DEVN+512, $DEVN+640 AND $DEVN+768.
893
894      :THE OVER-WRITE TEST IS PERFORMED WITH OFFSETS IN BOTH DIRECTIONS.
895
896      :*****
      TST6:  SCOPE
897      MOV      #1,$TIMES      ;;DO 1 ITERATION
898      MOV      #6,TS1NM      ;LOAD TEST NUMBER
899      MOV      #STACK,SP      ;INITIAL THE STACK POINT
900      MOV      #OVWNO,R3      ;1 ST ADDRESS TO CLEAR
901      MOV      #RDN0,R4      ;LAST ADDRESS+?
902      1$:      CLR      (R3)+      ;RESET ALL SCORE BOARD
903      CMP      R4,R3      ;ALL LOCATIONS ARE CLEARED ?
904      BHI      1$      ;BRANCH IF NOT
905      MOV      #FMTDPB,R0      ;SET UP STARTING ADDRESS OF DPB
906      MOV      $CDW1,R1      ;HISTORY FILE BLOCK
907      MOV      DRIVE,(R0)      ;LOAD THE PHYSICAL DRIVE #
908      MOV      #-8192,$WRDM(R0) ;LOAD THE WORD COUNT (ONE TRACK)
909      CLR      $SEC(R0)      ;START FROM TRACK = 0, SECTOR 0
910      MOV      #WRDAT,$COMND(R0) ;WRITE DATA COMMAND
911      MOV      $DEVN,$CYL(R0) ;LOAD THE STARTING CYLINDER
912      MOV      #BUFFER,$BUF(R0) ;LOAD THE BUFFER ADDRESS
913      MOV      #RM.REG,14(R0) ;REG'S SAVE ADDRESS
914      MOV      $DEVN,R2      ;LOCATE THE DATA PATTERN
915      ASL      R2      ;WORD INDEX
916      MOV      PSEUDO(R2),R2 ;LOAD R2 WITH THE DATA PATTERN POINTER
917      MOV      $BUF(R0),R3 ;BUFFER ADDRESS
918      MOV      #8192,R4      ;WORD COUNT
919      2$:      MOV      R2,(R3)+      ;FILL THE BUFFER
920      DEC      R4      ;DECREMENT WORD CTR
921      BNE      2$      ;BRANCH IF NOT DONE
922
923      ;START TO WRITE ONE CYLINDER ON EACH TRACK OF EACH WRITE CURRENT ZONE

```

```

924                                     ;(7 ZONES ON EACH PACK)
925
926 013512 004037 031062      3$: JSR      R0,RM05      ;CALL THE DRIVER
927 013516 004506             FMTDPB      ;PARAMETER BLOCK
928 013520 000774             BR          3$          ;BRANCH IF QUEUE FAIL
929 013522 005737 004524      4$: TST      FMTDPB+16    ;WRITE COMMAND DGNF ?
930 013526 001775             BEQ        4$          ;NO, THEN WAIT
931 013530 012700 004506      MOV      #FMTDPB,R0      ;PROCESS IF ANY ERROR
932 013534 004737 017452      JSR      PC,PROCES
933 013540 005760 000016      TST      16(R0)          ;ERROR FLAG SET ?
934 013544 100010             BPL        22$          ;BRANCH IF NOT
935 013546 004737 030334      JSR      PC,RMINIT        ;INITIAL THE DRIVE
936 013552 012737 177777 030264      MOV      #-1,SAVEFG
937 013560 012737 177777 030266      MOV      #-1,SEEKFG
938 013566 105260 000011      22$: INCB     $TRK(R0)      ;NEXT TRACK
939 013572 126037 000011 001372      CMPB    $TRK(R0),TRKLMT ;LAST TRACK IS DONE ?
940 013600 003744             BLE        3$          ;NO, THEN BRANCH
941 013602 105060 000011      CLRB     $TRK(R0)      ;RESET TRACK NUMBER
942 013606 062760 000200 000012      ADD     #128,$CYL(R0) ;MOVE TO NEXT ZONE
943 013614 022760 001417 000012      CMP     #783,$CYL(R0) ;LAST ZONE IS DONE ?
944 013622 103333             BHS        3$          ;BRANCH IF NOT
945
946                                     ;RESET THE FMTDPB BLOCK AND EXECUTE WRITE-CHECK COMMAND TO DETECT ANY
947                                     ;COMPATIBLE PROBLEM. THE FOLLOWING SUBROUTINE ARE CALLED SCORE, OFFST
948                                     ;AND MAKEUP.
949
950 013624 005037 004504      LOOP1: CLR      OFFCOD      ;SET NEGATIVE OFFSET DIRECTION FLAG
951 013630 012700 004506      MOV      #FMTDPB,R0      ;R0=FMTDPB ADDRESS
952 013634 005060 000010      CLR      $SEC(R0)        ;START FROM SECTOR 0, TRACK 0
953 013640 013760 001270 000012      MOV      $DEVM,$CYL(R0) ;STARTING CYLINDER NUMBER
954                                     ;TOTAL 7 CYLINDERS ON ONE TRACK
955 013646 112760 000151 000002      LOOP2: MOVB   #WCKD,$COMMAND(R0) ;WRITE-CHECK-DATA COMMAND
956 013654 012760 043456 000006      MOV      #BUFFER,$BUF(R0) ;RESET BUFFER ADDRESS
957 013662 012760 004526 000014      MOV      #RM.REG.14(R0) ;ADDRESS TO SAVE RH/RM REG'S
958 013670 004037 031062      1$: JSR      R0,RM05      ;CALL THE DRIVER
959 013674 004506             FMTDPB      ;PARAMETER BLOCK ADDRESS
960 013676 000774             BR          1$          ;BRANCH IF QUEUE FAILURE
961 013700 005737 004524      2$: TST      FMTDPB+16    ;TEST IF COMMAND IS DONE ?
962 013704 001775             BEQ        2$          ;BRANCH IF NOT
963 013706 012700 004506      MOV      #FMTDPB,R0      ;LOAD THE PARAMETER BLOCK ADDRESS
964 013712 004737 017452      JSR      PC,PROCES
965 013716 004737 015034      JSR      PC,LABAD
966 013722 005760 000016      TST      16(R0)          ;ANY ERROR ?
967 013726 100011             BPL        3$          ;BRANCH IF NONE
968 013730 004737 030334      JSR      PC,RMINIT        ;INITIAL THE SYSTEM
969 013734 012737 177777 030264      MOV      #-1,SAVEFG
970 013742 012737 177777 030266      MOV      #-1,SEEKFG
971 013750 000414             BR          5$          ;NOT UPDATE THE SCORE
972 013752 004737 014614      3$: JSR      PC,SCORE      ;INCREMENT SCORE
973 013756 005760 000022      4$: TST      $RMWC(R0)    ;WORD COUNT = 0 /
974 013762 001407             BEQ        5$          ;BRANCH IF WORD COUNT IS 0
975 013764 116060 000026 000010      MOVB   $RMDA(R0),$SEC(R0) ;UPDATE STARTING SECTOR
976 013772 016060 000022 000004      MOV     $RMWC(R0),$WRDM(R0) ;UPDATE WORD COUNT
977 014000 000733             BR          1$          ;TO READ THE REST SECTORS
978
979                                     ;THE FOLLOWING CODING TEST THE COMPATIBLE PROBLEM IN OFFSET MODE OFFCOD = 0
980                                     ;(NEGATIVE) AND OFFCOD = 1(POSITIVE).

```

```

981
982 014002 012700 004506 5$: MOV #FMTPDB,R0 ;RESET THE DPB BLOCK
983 014006 012760 160000 000004 MOV #-8192,SWRDM(R0) ;FULL TRACK WORD COUNT
984 014014 105060 000010 CLR $SEC(R0) ;RESET TO SECTOR 0, TRACK NOT CHANGED
985 014020 012760 043456 000006 MOV #BUFFER,$BUF(R0) ;BUFFER ADDRESS
986 014026 012760 004526 000014 MOV #RM.REG,14(R0) ;ADDRESS TO SAVE ALL RH/RM REG'S
987 014034 004537 014244 6$: JSR R5,OFFST ;CALL OFFSET
988 014040 000443 BR 10$ ;BRANCH TO NEXT CYLINDER, IF OFFSET FAILS
989 014042 004737 014476 JSR PC,MAKEUP ;CALL WRITE CHECK IN OFFSET MODE
990 014046 005737 004524 7$: TST FMTPDB+16 ; OFFSET WRITE CHECK IS DONE ?
991 014052 001775 BEQ 7$ ;BRANCH IF NOT
992
993 014054 012700 004506 MOV #FMTPDB,R0 ;LOAD THE DPB ADDRESS
994 014060 004737 017452 JSR PC,PROCES ;REPORT , IF ANY ERROR
995 014064 004737 015034 JSR PC,LABAD ;LOCATE STARTING AND ENDING SECTORS
996 014070 005760 000016 TST 16(R0) ;ANY ERROR ?
997 014074 100011 BPL 8$ ;BRANCH, IF NONE
998 014076 004737 030334 JSR PC,RMINIT ;INITIAL THE SYSTEM
999 014102 012737 177777 030264 MOV #-1,SAVEFG
1000 014110 012737 177777 030266 MOV #-1,SEEKFG
1001 014116 000414 RR 10$ ;NOT UPDATE THE SCORE
1007 014120 004737 014614 8$: JSR PC,SCORE ;UPDATE THE TEST SCORE
1008 014124 005760 000022 9$: TST $RMWC(R0) ;WORD CTR = 0 ?
1009 014130 001407 BEQ 10$ ;IF WORD CTR = 0 ,BRANCH TO NEXT OP
1010 014132 116060 000026 000010 MOV $RMDC(R0),$SEC(R0) ;UPDATE THE NEW STARTING ADDRESS
1011 014140 016060 000022 000004 MOV $RMWC(R0),SWRDM(R0) ;UPDATE THE NEW WORD COUNT
1012 014146 000732 BR 6$
1013 014150 062760 000200 000012 10$: ADD #128,$CYL(R0) ;ADJUST CYLINDER ADDRESS TO NEXT ZONE
1014 014156 022760 001417 000012 CMP #783,$CYL(R0) ;ALL 7 ZONES HAVE BEEN TESTED ?
1015 014164 103402 BLO 11$ ;BRANCH, IF ALL DONE
1016 014166 000137 013646 JMP LOOP2 ;TO NEXT WRITE CURRENT ZONE
1017 014172 013760 001270 000012 11$: MOV $DEVM,$CYL(R0) ;RESET CYLINDER ADDRESS
1018 014200 105260 000011 INCB $TRK(R0) ;INCREMENT TO NEXT TRACK
1019 014204 126037 000011 001372 CMPB $TRK(R0),TRKLMT ;ALL TRACKS ARE TESTED ?
1020 014212 003002 BGT 12$ ;BRANCH IF ALL DONE
1021 014214 000137 013646 JMP LOOP2 ;TO NEXT TRACK
1022 014220 005737 004504 12$: TST OFFCOD ;FINISHING TEST THE POSITIVE OFFSET ?
1023 014224 001005 BNE 13$ ;BRANCH IF ALL DONE
1024 014226 012737 000001 004504 MOV #1,OFFCOD ;SET POSITIVE OFFSET DIRECTION FLAG
1025 014234 000137 013630 JMP LOOP1 ;RESTART LOCATION
1026 014240 000137 015140 13$: JMP TST7 ;BRANCH TO NEXT TEST
1027
1028 ;OFFST ROUTINE
1029 ;OFFSET THE HEAD IN THE DIRECTION TOWARD SPINDLE OR AWAY FROM THE SPINDLE
1030 ;
1031 ; OFFCOD = 1, TOWARD SPINDLE (POSITIVE)
1032 ; OFFCOD = 0, AWAY FROM SPINDLE (NEGATIVE)
1033 ; DRIVE = PHYSICAL DRIVE NUMBER
1034 ; R0 = DPB ADDRESS
1035 ; RETRY = 3 TIMES
1036 ;CALL
1037 ; JSR R5,OFFST
1038 ; RET1 OFFSET FAIL RETURN ADDRESS
1039 ; RET2 OFFSET SUCCESSFUL RETURN
1040
1041 014244 010146 OFFST: MOV R1,-(SP) ;SAVE ALL REGISTERS
1042 014246 010246 MOV R2,-(SP)

```

```

1043 014250 010346      MOV      R3,-(SP)
1044 014252 010446      MOV      R4,-(SP)
1045 014254 012700 004506      MOV      #FMTDPB,R0      ;R0 DPB ADDRESS
1046 014260 116046 000002      MOV      $COMND(R0),-(SP)      ;SAVE THE I/O COMMAND
1047 014264 113710 001224      MOV      DRIVE,(R0)      ;LOAD THE DRIVE NUMBER
1048 014270 112760 000117 000002      MOV      #117,$COMND(R0)      ;RETURN TO CENTER COMMAND
1049 014276 004037 031062      JSR      R0,RM05      ;CALL THE DRIVE HANDLE
1050 014302 004506      FMTDPB
1051 014304 000465      BR      6$      ;BRANCH IF QUEUE FAILS
1052 014306 005737 004524      1$:      TST      FMTDPB+16      ;COMMAND DONE ?
1053 014312 001775      BEQ      1$      ;BRANCH IF NOT
1054 014314 100461      BMI      6$      ;BRANCH IF ERROR EXIST
1055 014316 013746 177776      MOV      @#PS,-(SP)      ;SAVE THE PSW
1056 014322 012737 000240 177776      MOV      #<5*32>,@#PS      ;LOAD PS 5
1057 014330 012700 004506      MOV      #FMTDPB,R0      ;DPB ADDRESS
1058 014334 013704 030324      MOV      RMADR,R4      ;MUSS BUS ADDRESS
1059 014340 013701 001224      MOV      DRIVE,R1      ;DRIVE NUMBER
1060 014344 010164 000010      MOV      R1,RMCS2(R4)      ;LOAD THE DRIVE NUMBER INTO CONTROLLER
1061 014350 016064 000012 000034      MOV      $CYL(R0),RMDC(R4)      ;CYLINDER NUMBER
1062 014356 016064 000010 000006      MOV      $SEC(R0),RMDA(R4)      ;SECTOR AND TRACK NUMBER
1063 014364 016064 000004 000002      MOV      $WRDM(R0),RMWC(R4)      ;WORD COUNT
1064 014372 016064 000006 000004      MOV      $BUF(R0),RMBR(R4)      ;BUFFER ADDRESS
1065 014400 012637 177776      MOV      (SP)+,@#PS      ;LOAD THE PSW BACK
1066 014404 112760 000200 000001      MOV      #BIT7,$FMT(R0)      ;LOAD THE OFFSET DIRECTION
1067 014412 005737 004504      TST      OFFCOD      ;NEG ?
1068 014416 001003      BNE      2$      ;BRANCH IF NOT
1069 014420 112760 000000 000001      MOV      #0,$FMT(R0)      ;CHANGE TO OTHER DIRECTION
1070 014426 112760 000115 000002 2$:      MOV      #115,$COMND(R0)      ;LOAD THE OFFSET COMMAND
1071 014434 004037 031062      JSR      R0,RM05      ;CALL THE DRIVE HANDLE
1072 014440 004506      FMTDPB
1073 014442 000406      BR      6$      ;BRANCH IF QUEUE FAILS
1074 014444 005737 004524      3$:      TST      FMTDPB+16      ;OFFSET DONE ?
1075 014450 001775      BEQ      3$      ;BRANCH IF SO
1076 014452 100402      BMI      6$      ;BRANCH IF ERROR
1077 014454 062705 000002      ADD      #2,R5      ;ADJUST RETURN ADDRESS
1078 014460      6$:
1079 014460 112637 004510      MOV      (SP)+,FMTDPB+$COMND      ;RESTORE THE I/O COMMAND
1080 014464 012604      MOV      (SP)+,R4      ;RESTORE REG
1081 014466 012603      MOV      (SP)+,R3
1082 014470 012602      MOV      (SP)+,R2
1083 014472 012601      MOV      (SP)+,R1
1084 014474 000205      RTS      R5      ;EXIT
1085
1086
1087      ;MAKEUP ROUTINE
1088      ;THIS ROUTINE ISSUES A WRITE CHECK OR READ COMMAND TO THE SELECTED DRIVE
1089      ;IN OFFSET MODE.
1090      ;AND SET UP THE FOLLOWING PARAMETERS
1091      :
1092      :      DTUW      = PHYSICAL DRIVE NUMBER
1093      :      TRNSWT     = FMTDPB
1094      :      DRVACT     = 1
1095      :      TIMER      = 1 SECOND
1096      :
1097      ;CALL
1098      :      JSR      PC,MAKEUP
1099      :      RET

```

```

1100
1101      ; MAIN PURPOSE OF THIS ROUTINE, TO EXECUTE A COMMAND WHILE ASSURE THAT
1102      ; THIS READ OR WRITE-CHECK COMMAND BEING EXECUTED IN OFFSET MODE.
1103      ; ROUTINES USED TD, SC, STO, ( IN DRIVE HANDLER )
1104
1105 014476 010146      MAKEUP: MOV      R1, -(SP)
1106 014500 010246      MOV      R2, -(SP)
1107 014502 010446      MOV      R4, -(SP)
1108 014504 012702 000151      MOV      #WCKD, R2      ; WRITE CHECK DATA IN TEST 6
1109 014510 022737 000010 001340      CMP      #10, TSTNM      ; ON TEST 8 ?
1110 014516 001002      BNE      1$      ; BRANCH IF NOT
1111 014520 012702 000171      MOV      #RDDAT, R2      ; READ DATA COMMAND IN TEST 8
1112 014524 005037 004524      1$: CLR      FMTDPB+16      ; CLEAR THE STATUS WORD
1113 014530 110237 004510      MOV      R2, FMTDPB+$COMND      ; LOAD THE COMMAND INTO PDB
1114 014534 013737 001224 030310      MOV      DRIVE, DTUW      ; ACTIVE DRIVE NUMBER
1115 014542 012737 004506 030246      MOV      #FMTDPB, TRNSWT      ; TRANSFER UNDERWAY FLAG
1116 014550 013701 001224      MOV      DRIVE, R1      ; DRIVE NUMBER
1117 014554 013704 030324      MOV      RMADR, R4      ; RH/RM BASE ADDRESS
1118 014560 112761 000001 030176      MOV      #1, DRVACT(R1)      ; ACTIVE DRIVE FLAG
1119 014566 006301      ASL      R1      ; WORD INDEX
1120 014570 012761 060000 030270      MOV      #60000, TIMER(R1)      ; ONE SECOND TIMER
1121 014576 006201      ASR      R1
1122 014600 010264 000000      MOV      R2, RMCS1(R4)      ; ISSURE WRITE CHECK OR READ COMMAND
1123 014604 012604      MOV      (SP)+, R4      ; RESTORE REG 4, 1
1124 014606 012602      MOV      (SP)+, R2
1125 014610 012601      MOV      (SP)+, R1
1126 014612 000207      RTS      PC      ; EXIT
1127
1128      ; SCORE ROUTINE
1129      ; ROUTINE TO UPDATE THE TEST SCORE
1130      ; TABLEX = ADDRESS OF CURRENT SCORE BOARD
1131      ; $DEVM = LOGICAL DRIVE #
1132      ; DRIVE = PHYSICAL DRIVE NUMBER
1133      ; CMSEC = END SECTOR ADDRESS
1134      ; STARSC = START SECTOR ADDRESS
1135
1136      ; CALL
1137      ; JSR      PC, SCORE
1138      ; RET
1139
1140      SCORE:
1141 014614 010146      MOV      R1, -(SP)      ;: PUSH R1 ON STACK
1142 014616 010246      MOV      R2, -(SP)      ;: PUSH R2 ON STACK
1143 014620 010346      MOV      R3, -(SP)      ;: PUSH R3 ON STACK
1144 014622 010446      MOV      R4, -(SP)      ;: PUSH R4 ON STACK
1145 014624 023737 001362 001360      CMP      CMSEC, STARSC      ;: CORRECT START AND STOP ADDRESSES
1146 014632 003473      BLE      9$      ;: BRANCH IF NOT
1147 014634 022737 000010 001340      CMP      #10, TSTNM      ;: ON TEST 8
1148 014642 001011      BNE      2$      ;: BRANCH IF NOT (MUST BE TEST 6)
1149 014644 005737 004504      TST      OFFCOD      ;: NEGATIVE OFFSET ?
1150 014650 001403      BEQ      1$      ;: BRANCH IF NEGATIVE OFFSET
1151 014652 012703 003676      MOV      #RDP0, R3      ;: SCORE BOARD ADDRESS
1152 014656 000413      BR      4$
1153 014660 012703 003216      1$: MOV      #RDNO, R3
1154 014664 000410      BR      4$
1155 014666 005737 004504      2$: TST      OFFCOD      ;: NEGATIVE OFFSET
1156 014672 001403      BEQ      3$      ;: BRANCH, IF NEGATIVE OFFSET

```

```

1153 014674 012703 002536      MOV    #OVWPO,R3      ;SCORE BOARD ADDRESS
1154 014700 000402      BR      4$
1155 014702 012703 002056      3$: MOV    #OVWNO,R3      ;SCORE BOARD ADDRESS
1156 014706 113702 004517      4$: MOVVB  FMTDPB+STRK,R2 ;LOAD THE TRACK NUMBER
1157 014712 005702      TST     R2              ;ON TRACK 0
1158 014714 001404      BEQ     6$              ;BRANCH IF IT IS
1159 014716 062703 000020      5$: ADD     #16.,R3      ;EACH SCORE BOARD TAKES 16 BYTES
1160 014722 005302      DEC     R2              ;LOCATED ?
1161 014724 001374      BNE     5$              ;BRANCH IF NOT
1162 014726 010337 002054      6$: MOV     R3,TABLEX      ;STORE THE TABLE STARTING ADDRESS
1163 014732 010301      MOV     R3,R1          ;RE ASSIGN REGISTERS
1164 014734 013702 001270      MOV     $DEVM,R2      ;LOGICAL DRIVE #
1165 014740 013703 001360      MOV     STARSC,R3      ;START SECTOR
1166 014744 116204 004424      MOVVB  INDST(R2),R4 ;LOCATE THE STARTING POINT FOR SCORE BOARD
1167 014750 060304      ADD     R3,R4          ;UPDATE POINTER
1168 014752 022704 000017      11$: CMP    #15.,R4      ;SHOULD POINTER BE ADJUSTED ?
1169 014756 003003      BGT     7$              ;BR IF NO
1170 014760 162704 000020      SUB     #16.,R4      ;ENTRY POINT AT THE SCORE BOARD
1171 014764 000772      BR      11$
1172 014766 060401      7$: ADD     R4,R1
1173 014770 023703 001362      8$: CMP    CMSEC,R3      ;ENDING SECTOR REACHED ?
1174 014774 002412      BLT     9$              ;BRANCH IF IT IS
1175 014776 105221      INCB    (R1)+      ;INC SCORE AND POINT TO NEXT LOGICAL DRIVE
1176 015000 005204      INC     R4              ;INCREMENT LOGICAL DRIVE #
1177 015002 005203      INC     R3              ;INCREMENT SECTOR COUNT
1178 015004 022704 000017      CMP    #15.,R4      ;TIME TO RESET TABLE ?
1179 015010 002367      BGE     8$              ;BRANCH IF NOT
1180 015012 013701 002054      MOV     TABLEX,R1 ;RESET TABLE ADDRESS
1181 015016 005004      CLR     R4              ;LOGICAL DRIVE 0
1182 015020 000763      BR      8$              ;LOOPING BACK
1183 015022      9$:
      MOV     (SP)+,R4      ;;POP STACK INTO R4
      MOV     (SP)+,R3      ;;POP STACK INTO R3
      MOV     (SP)+,R2      ;;POP STACK INTO R2
      MOV     (SP)+,R1      ;;POP STACK INTO R1
1184 015032 000207      RTS     PC
1185
1186
1187      ;LABAD ROUTINE
1188      ;LOCATE THE START SECTOR AND THE TERMINATE SECTOR OF THE PREVIOUS
1189      ;OPERATION.
1190      ;      STARSC = STARTING SECTOR
1191      ;      CMSEC  = ENDING SECTOR
1192      ;
1193      ; INFORMATION FROM $RMDA(FMTDPB), $RMWC(FMTDPB), $SEC(FMTDPB)
1194      ;      $SEC(FMTDPB), $WRDM(FMTDPB)
1195      ;CALL
1196      ;      JSR     PC,LABAD
1197      ;      RET
1198
1199      LABAD:
      MOV     R0,-(SP)      ;;PUSH R0 ON STACK
      MOV     R2,-(SP)      ;;PUSH R2 ON STACK
1200 015034 010046      MOV     #FMTDPB,R0      ;DPB ADDRESS
1201 015036 010246      MOVVB  $RMDA(R0),R2 ;HARDWARE TERMINATING SECTOR
1202 015040 012700 004506      BIC     #177740,R2 ;CHOP OFF HIGH ORDER BITS IF ANY
1203 015044 116002 000026      MOVVB  $SEC(R0),STARSC ;STARTING SECTOR ADDRESS
1204 015050 042702 177740
1205 015054 116037 000010 001360

```

```

1204 015062 005702          TST      R2          ; TERMINATOR AT 0 SECTOR
1205 015064 001404          BEQ      1$          ; BRANCH IF IT IS
1206 015066 005302          DEC      R2          ; DECREMENT ONE SECTOR COUNT
1207 015070 010237 001362    MOV      R2,CMSEC    ; ENDING SECTOR ADDRESS
1208 015074 000416          BR       3$          ; EXIT
1209 015076 005760 000022    1$:      TST      $RMWC(R0) ; WORD COUNT = 0
1210 015102 001410          BEQ      2$          ; BRANCH IF IT IS
1211 015104 026060 000022 000004    CMP      $RMWC(R0), $WRDM(R0) ; WORD COUNT CHANGED AT ALL ?
1212 015112 001004          BNE      2$          ; BRANCH IF CHANGED
1213 015114 116037 000010 001362    MOV      $SEC(R0), CMSEC ; END SECTOR = START SECTOR
1214 015122 000403          BR       3$          ; EXIT
1215 015124 112737 000037 001362    2$:      MOV      #31., CMSEC ; END AT SECTOR 31
1216 015132          3$:      MOV      (SP)+, R2      ; POP STACK INTO R2
1217 015134          012600      MOV      (SP)+, R0      ; POP STACK INTO R0
1218 015136 000207          RTS      PC
1219          ; TEST 7
1220          ; SELF TEST: WRITE CYLINDERS $DEVN+17, $DEVN+17+128, $DEVN+17+128X2, ETC.
1221          ; THEN EXECUTE WRITE CHECK TO DETECT ANY DATA OR HEADER PROBLEM
1222          ; FOR THE SELECTED LOGICAL DRIVE
1223          ; $DEVN: LOGICAL DRIVE #
1224          ; DRIVE: PHYSICAL DRIVE #
1225
1226          ; *****
          TST7: SCOPE
1227 015140 000004          MOV      #1, $TIMES      ; DO 1 ITERATION
1228 015142 012737 000001 001176    JSR      PC, RMINIT ; INITIAL THE DRIVE
1229 015150 004737 030334          MOV      #-1, $SAVEFG ; SAVE THE RH/RM REG'S
1230 015154 012737 177777 030264    MOV      #-1, $SEEKFG
1231 015162 012737 177777 030266    MOV      #7, $TSTNM ; LOAD TEST NUMBER
1232 015170 012737 000007 001340    MOV      #STACK, SP ; INITIAL THE STACK
1233 015176 012706 001100          MOV      #SELF0, R2 ; CLEAR THE SELF TEST SCORE BOARDS
1234 015202 012702 004356          1$:      CLR      (R2)+ ;
1235 015206 005022          CMP      #SELF18, R2 ; ALL DONE ?
1236 015210 022702 004422          BHS      1$          ; BRANCH IF NOT
1237 015214 103374          MOV      #FMTDPB, R0 ; SET UP DPB
1238 015216 012700 004506          MOV      #-8192., $WRDM(R0) ; LOAD FULL TRACK WORD COUNT
1239 015222 012760 160000 000004    MOV      #WRTDAT, $COMND(R0) ; WRITE DATA COMMAND
1240 015230 112760 000161 000002    CLR      $SEC(R0) ; SECTOR 0, TRACK 0
1241 015236 005060 000010          MOV      DRIVE, (R0) ; LOAD PHY. DRIVE
1242 015242 113710 001224          MOV      $DEVN, R2 ; LOCATE THE STARTING CYLINDER
1243 015246 013702 001270          ADD      #17., R2
1244 015252 062702 003021          MOV      R2, $CYL(R0) ; CYLINDER ADDRESS
1245 015256 010260 000012          MOV      #BUFFER, $BUF(R0) ; RESET BUFFER ADDRESS
1246 015262 012760 043456 000006    MOV      #RM.REG, 14(R0) ; ADDRESS TO SAVE ALL RH/RM REG'S
1247 015270 012760 004526 000014    JSR      PC, FILBUF ; FILL THE BUFFER WITH WORSE CASE PATTERN
1248 015276 004737 021042          2$:      JSR      R0, RM05 ; CALL DRIVER HANDLER
1249 015302 004037 031062          FMTDPB
1250 015306 004506          BR       2$          ; BRANCH IF QUEUE FAILS
1251 015310 000774          3$:      TST      FMTDPB+16 ; ALL DONE ?
1252 015312 005737 004524          BEQ      3$          ; BRANCH IF NOT
1253 015316 001775          MOV      #FMTDPB, R0 ; REPORT IF ANY ERROR
1254 015320 012700 004506          JSR      PC, PROCES
1255 015324 004737 017452          TST      16(R0) ; ERROR FLAG SET ?
1256 015330 005760 000016          BPL      22$      ; BRANCH IF NOT
1257 015334 100010          JSR      PC, RMINIT ; INITIAL THE DRIVE

```

```

1257 015342 012737 177777 030264      MCV      #-1,SAVEFG
1258 015350 012737 177777 030266      MOV      #-1,SEEKFG
1259 015356 105260 000011      22$: INCB    $TRK(R0)      ;NEXT TRACK
1260 015362 126037 000011 001372      CMPB    $TRK(R0),TRKLMT ;ALL TRACK ARE DONE ?
1261 015370 003744      BLE      2$          ;BRANCH IF NOT
1262 015372 005060 000010      CLR     $SEC(R0)      ;RESET SECTOR AND TRACK
1263 015376 062760 000200 000012      ADD     #128.,$CYL(R0) ;ADVANCE TO NEXT ZONE
1264 015404 022760 001440 000012      CMP     #800.,$CYL(R0) ; ALL 7 ZONES ARE DONE ?
1265 015412 103333      BHIS    2$          ;BRANCH IF NOT
1266
1267      ;EXECUTE WRITE CHECK TO DETECT ANY DATA OR HEADER PROBLEM
1268
1269 015414 112760 000151 000002      MOVB    #WCKD,$COMND(R0) ;CHANGE TO WRITE CHECK COMMAND
1270 015422 012702 004356      MOV     #SELF0,R2      ;R2 POINTS TO SCORE BOARD
1271      ;CAN NOT BE DESTORIED
1272 015426 013703 001270      MOV     $DEVM,R3      ;LOCATE STARTING ADDRESS
1273 015432 062703 000021      ADD     #17.,R3
1274 015436 010360 000012      MOV     R3,$CYL(R0)    ;STARTING CYLINDER
1275 015442 005037 004504      CLR     OFFCOD        ;SET NEGATIVE OFFSET FLAG
1276 015446 004037 031062      4$: JSR     R0,RM05     ;CALL DRIVE HANDLER
1277 015452 004506      FMTDPB
1278 015454 000774      BR      4$          ;BRANCH IF QUEUE FAILS
1279 015456 005737 004524      5$: TST     FMTDPB+16   ;ALL DONE ?
1280 015462 001775      BEQ     5$          ;BRANCH IF NOT
1281 015464 012700 004506      MOV     #FMTDPB,R0
1282 015470 004737 017452      JSR     PC,PROCES
1283 015474 005760 000016      TST     16(R0)        ;ANY ERROR
1284 015500 100401      BMI     6$          ;YES,THEN DON'T INCREMENT SCORE
1285 015502 105212      INCB    (R2)          ;INCREMENT SCORE
1286 015504 004537 014244      6$: JSR     R5,OFFST    ;OFFSET
1287 015510 000415      BR      8$          ;BRANCH IF OFFSET FAILS
1288 015512 004737 014476      JSR     PC,MAKEUP     ;EXECUTE WRITE CHECK IN OFFSET MODE
1289 015516 005737 004524      7$: TST     FMTDPB+16   ;ALL DONE /
1290 015522 001775      BEQ     7$          ;BRANCH IF NOT
1291 015524 012700 004506      MOV     #FMTDPB,R0
1292 015530 004737 017452      JSR     PC,PROCES
1293 015534 005760 000016      TST     16(R0)        ;REPORT IF ANY ERROR
1294 015540 100401      BMI     8$          ;ERROR BIT SET ?
1295 015542 105212      INCB    (R2)          ;DON'T INCREMENT SCORE
1296 015544 005737 004504      8$: TST     OFFCOD     ;INCREMENT SCORE
1297 015550 001006      BNE     9$          ;POSITIVE OFFSET IS DONE ?
1298 015552 005202      INC     R2           ;BRANCH ,IF DONE
1299 015554 012737 000001 004504      MOV     #1,OFFCOD     ;INCREMENT SCORE BOARD POINTER
1300 015562 000137 015446      JMP     4$          ;SET POSITIVE OFFSET FLAG
1301 015566 105260 000011      9$: INCB    $TRK(R0)    ;POSITIVE OFFSET TEST
1302 015572 005202      INC     R2           ;INCREMENT TO NEXT TRACK
1303 015574 126037 000011 001372      CMPB    $TRK(R0),TRKLMT ;ADVANCE SCORE BOARD POINTER
1304 015602 003004      BGT     10$         ;ALL TRACKS ARE DONE ?
1305 015604 005037 004504      CLR     OFFCOD        ;BRANCH , IF ALL DONE
1306 015610 000137 015446      JMP     4$          ;RESET OFFSET DIRECTION
1307 015614 005060 000010      10$: CLR     $SEC(R0)   ;TRY THE NEXT TRACK
1308 015620 012702 004356      MOV     #SELF0,R2     ;TRACK 0, SECTOR 0
1309 015624 062760 000200 000012      ADD     #128.,$CYL(R0) ;RESET SCORE BOARD
1310 015632 022760 001440 000012      CMP     #800.,$CYL(R0) ;ADVANCE TO NEXT ZONE
1311 015640 103404      BLO     11$         ;ALL 7 ZONES ARE DONE ?
1312 015642 005037 004504      CLR     OFFCOD        ;BRANCH ,IF ALL DONE
1313 015646 000137 015446      JMP     4$          ;RESET OFFSET FLAG
                        ;TO NEXT ZONE

```

```

1314 015652 000240      11$: NOP                      ;TO NEXT TEST
1315
1316                    ;TEST10 TEST 8      READ COMPATIBLE TEST
1317                    ;CYLINDERS TESTED : $DEV+112,$DEV+112+128XN  N=1 TO 5
1318                    ;THIS TEST SELECT ONE CYLINDER FOR EACH LOGICAL DRIVE FROM
1319                    ;ZONE 1 TO ZONE 6
1320                    ;
1321                    ; $DEV = LOGICAL DRIVE #
1322                    ; DRIVE = PHYSICAL DRIVE #
1323
1324                    ;*****
1325 015654 000004      TST10: SCOPE
1326 015656 012737 000001 001176      MOV #1,$TIMES          ;;DO 1 ITERATION
1327 015664 012737 000010 001340      MOV #10,TSTNM          ;LOAD TEST NUMBER
1328 015672 012706 001100              MOV #STACK,SP          ;INITAIL THE STACK POINTER
1329 015676 012702 003216              MOV #RDNO R2           ;CLEAR THE SCORE BOARD OF READ TEST
1330 015702 005022              1$: CLR (R2)+
1331 015710 101374              CMP #RDP18+16.,R2          ;ALL DONE
1332 015712 005037 004504              BHI 1$              ;BRANCH IF NOT
1333 015716 012700 004506              CLR OFFCOD          ;SET NEGATIVE OFFSET FLAG
1334 015722 012760 160000 000004      LOOP3: MOV #FMTDPB,R0      ;SET UP DPB BLOCK
1335 015730 005060 000010              MOV #-8192.,$WRDM(R0) ;LOAD THE WORD CTR,FULL TRACK
1336 015734 112760 000171 000002      CLR $SEC(R0)          ;TRACK 0 AND SECTOR 0
1337 015742 012760 043456 000006      MOVB #RDDAT,$COMND(R0) ;LOAD THE READ DATA COMMAND
1338 015750 012760 004526 000014      MOV #BUFFER,$BUF(R0) ;RESET BUFFER ADDRESS
1339 015756 113710 001224              MOVB #RM.REG,14(R0) ;ADDRESS TO SAVE ALL RH/RM ADDRESS
1340 015762 013703 001270              MOV $DEV,R3          ;LOAD PHY. DRIVE ADDRESS
1341 015766 062703 000160              ADD #112.,R3          ;LOCATE STARTING CYL
1342 015772 010360 000012              MOV R3,$CYL(R0)       ;STARTING CYL ADDRESS
1343 015776 004037 031062              1$: JSR R0,RM05        ;CALL THE DRIVE HANDLER
1344 016002 004506              FMTDPB
1345 016004 000774              BR 1$
1346 016006 005737 004524              2$: TST FMTDPB+16      ;COMMAND DONE ?
1347 016012 001775              BEQ 2$              ;BRANCH IF NOT
1348 016014 012700 004506              MOV #FMTDPB,R0
1349 016020 004737 017452              JSR PC,PROCES      ;REPORT IF ANY ERROR
1350 016024 004737 015034              JSR PC,LABAD       ;LOCATE START AND STOP ADDRESS
1351 016030 005760 000016              TST 16(R0)          ;ANY ERROR ?
1352 016034 100011              BPL 3$              ;BRANCH,IF NONE
1353 016036 004737 030334              JSR PC,RMINIT     ;INITIAL THE SYSTEM
1354 016042 012737 177777 030264      MOV #-1,SAVEFG
1355 016050 012737 177777 030266      MOV #-1,SEEKFG
1356 016056 000414              BR 5$              ;NOT INCREMENT THE SCORE
1357 016060 004737 014614              3$: JSR PC,SCORE      ;UPDATE THE SCORE
1358 016064 005760 000022              4$: TST $RMWC(R0)      ;WORD COUNT= 0
1359 016070 001407              BEQ 5$              ;BRANCH IIF IT IS
1360 016072 116060 000026 000010      MOVB $RMDA(R0),$SEC(R0) ;UPDATE THE NEW STARTING SECTOR
1361 016100 016060 000022 000004      MOV $RMWC(R0),$WRDM(R0) ;UPDATE WORD COUNT
1362 016106 000733              BR 1$              ;CONTINUE
1363 016110 012700 004506              5$: MOV #FMTDPB,R0      ;RESET THE DPB BLOCK
1364 016114 012760 160000 000004      MOV #-8192.,$WRDM(R0) ;FULL TRACK WORD COUNT
1365 016122 105060 000010              CLRB $SEC(R0)      ;RESET TO SECTOR 0,TRACK NOT CHANGED
1366 016126 004537 014244              6$: JSR R5,OFFST     ;CALL OFFSET
1367 016132 000443              BR 10$           ;BRANCH IF OFFSET FAILS
1368 016134 004737 014476              7$: JSR PC,MAKEUP    ;EXECUTE READ DATA IN OFFSET MODE
1369 016140 005737 004524              TST FMTDPB+16      ;OFFSET READ IS DONE ?

```

```

1374 016144 001775      BEQ      7$          ;BRANCH IF NOT
1375 016146 012700 004506  MOV      #FMTPB,R0      ;REPORT IF ANY ERROR
1376 016152 004737 017452  JSR      PC,PROCES
1377 016156 004737 015034  JSR      PC,LABAD      ;LOCATE THE START AND STOP ADDRESSES
1378 016162 005760 000016  TST      16(R0)        ;ANY ERROR ?
1379 016166 100011      BPL      8$          ;BRANCH,IF NONE
1380 016170 004737 030334  JSR      PC,RMINIT     ;INITIAL THE SYSTEM
1381 016174 012737 177777 030264  MOV      #-1,SAVEFG
1382 016202 012737 177777 030266  MOV      #-1,SEEKFG
1383 016210 000414      BR       10$         ;NOT UPDATE THE SCORE
1389 016212 004737 014614 8$:     JSR      PC,SCORE    ;UPDATE THE SCORE
1390 016216 005760 000022 9$:     TST      $RMWC(R0)   ;WORD COUNT IS 0
1391 016222 001407      BEQ      10$         ;BRANCH IF IT IS
1392 016224 116060 000026 000010  MOV      $RMDA(R0),$SEC(R0) ;UPDATE SECTOR ADDRESS
1393 016232 016060 000022 000004  MOV      $RMWC(R0),$WRDM(R0) ;UPDATE WORD COUNT
1394 016240 000732      BR       6$          ;LOOPING UNTIL CURRENT TRACK IS DONE
1395 016242 062760 000200 000012 10$:   ADD      #128,$CYL(R0) ;ADJUST CYLINDER TO NEXT ZONE
1396 016250 022760 001377 000012  CMP      #767,$CYL(R0) ;ALL 6 ZONES ARE DONE ?
1397 016256 103402      BLO      1$          ;BRANCH IF ALL DONE
1398 016260 000137 015776  JMP      1$
1399
1400 016264 013703 001270 11$:   MOV      $DEV#R3      ;LOCATE STARTING CYLINDER
1401 016270 062703 000160      ADD      #112,R3
1402 016274 010360 000012      MOV      R3,$CYL(R0) ;STARTING CYLINDER
1403 016300 105260 000011      INCB     $TRK(R0) ;TO NEXT TRACK
1404 016304 126037 000011 001372  CMPB     $TRK(R0),TRKLMT ;ALL SURFACE ARE DONE ?
1405 016312 003002      BGT      12$         ;BRANCH IF ALL DONE
1406 016314 000137 015776  JMP      1$          ;LOOPING UNTIL CURRENT TRACK IS DONE
1407
1408 016320 005737 004504 12$:   TST      OFFCOD      ;OFFSET CODE = POSITIVE ?
1409 016324 001005      BNE      13$         ;IF EQUAL, THEN ALL DONE
1410 016326 012737 000001 004504  MOV      #1,OFFCOD    ;SET POSITIVE OFFSET FLAG
1411 016334 000137 015716  JMP      LOOP3
1412 016340 000240 13$:   NOP
1413
1414      ;TEST11 TEST 9
1415      ;REPORT THE TEST SCORES
1416      ;DIRECT THE OPERATOR TO CHANGE PACK AND MOUNT TO OTHER DRIVE
1417      ;
1418      ;$DEV# = LOGICAL DRIVE #, SHOULD NOT BE UPDATED BEFORE THE
1419      ;REPORT IS COMPLETED.
1420      ;
1421      ;BADSEC = 0, (DRIVES ARE COMPATIBLE)
1422      ;          = -1, (DRIVES NOT COMPATIBLE)
1423      ;
1424      ;CMTRK = TRACK NUMBER FOR CONTROLLING THE SCORE TYPING.
1425      ;
1426      ;*****
1427      TST11: SCOPE
1428      MOV      #1,$TIMES      ;DO 1 ITERATION
1429      MOV      #11,TSTNM     ;LOAD THE TEST NUMBER
1430      MOV      #STACK,SP     ;LOAD THE STACK POINTER
1431      CLR      -(SP)         ;DUMMY SCORE BOARD ADDRESSES
1432      CLR      -(SP)
1433      MOV      #SELF0+1,-(SP) ;POSITIVE OFFSET READ TEST SCORE ADDRESS
1434      MOV      #RDPO,-(SP)   ;POSITIVE OFFSET TEST SCORE
1435      MOV      #SELF0,R2     ;NEGATIVE OFFSET READ TEST SCORE

```

1434	016404	012703	003216		MOV	#RDNO,R3	;NEGATIVE OFFSET READ COMPATIBLE TEST SCORE
1435	016410	013701	001372	1\$:	MOV	TRKLM1,R1	;R1= TRACK NUMBER
1436	016414	122712	000007	2\$:	CMPB	#7,(R2)	;SELF READ TEST SCORE IS TOO LOW
1437	016420	101411			BLOS	4\$	;BRANCH IF NOT
1438	016422	012705	000020		MOV	#16.,R5	;INCREMENT READ COMPATIBLE SCORE FOR ALL 16 DIRVES
1439	016426	111304		3\$:	MOVB	(R3),R4	;ADJUST SCORE BY ADDING 6
1440	016430	062704	000006		ADD	#6,R4	
1441	016434	110423			MOVB	R4,(R3)+	;UPDATE SCORE AND POINTS TO NEXT DRIVE
1442	016436	005305			DEC	R5	;ALL DRIVES ARE UPDATED ?
1443	016440	001372			BNE	3\$	;BRANCH IF NOT
1444	016442	000402			BR	5\$	
1445	016444	062703	000020	4\$:	ADD	#16.,R3	;ADJUST POINTER OF SCORE BOARD ADDRESS
1446	016450	062702	000002	5\$:	ADD	#2,R2	;UPDATE THE SELF TEST SCORE ADDRESS
1447	016454	005301			DEC	R1	;ALL TRACKS DONE ?
1448	016456	002356			BGE	2\$	;BRANCH IF NOT
1449	016460	012603			MOV	(SP)+,R3	;GET NEXT PAIR
1450	016462	012602			MOV	(SP)+,R2	
1451	016464	001351			BNE	1\$	
1452							
1453							
1454							;SET ACCEPTANCE FLAG, INITIALIZE THE TABLE POINTERS AND TRACK NUMBER
1455	016466	005037	001342		CLR	BADSEC	;SET ACCEPTANCE FLAG
1456	016472	005037	001364		CLR	CMTRK	;START FROM TRACK 0
1457	016476	005001			CLR	R1	;START FROM LOG DRV 0
1458	016500	012702	002056		MOV	#OVWNO,R2	;SCORE BOARD BASE ADDRESS
1459	016504	012703	000001		MOV	#BIT0,R3	;BIT POSITION FOR LOG DRV 0
1460	016510	030337	002006	LOOP4:	BIT	R3,ASNLST	;IS THE LOG DRV UNDER TEST ?
1461	016514	001502			BEQ	LOOP5	;BRANCH IF NOT
1462	016516	005037	001366		CLR	NULINE	;NEW LINE INDICATOR
1463	016522	123701	001270		CMPB	\$DEVN,R1	;SELF SCORE ?
1464	016526	001434			BEQ	SPATH	;BRANCH IF IT IS
1465	016530	122712	000016	KPATH:	CMPB	#14.,(R2)	;OVERWRITE NEG OFFSET < 14 ?
1466	016534	101403			BLOS	1\$	;BRANCH IF NOT
1467	016536	004537	017152		JSR	R5,PRINT	;REPORT EXCEPTIONS
1468	016542	000001					;COLUMN POSITION ON REPORT
1469	016544	122762	000016 000460	1\$:	CMPB	#14.,OVWPO-OVWNO(R2)	;OVERWRITE POS OFFSET < 14 ?
1470	016552	101403			BLOS	2\$	;BRANCH IF NOT
1471	016554	004537	017152		JSR	R5,PRINT	;REPORT EXECPTIONS
1472	016560	000002					;COLUMN POSITION ON REPORT
1473	016562	122762	000014 001140	2\$:	CMPB	#12.,RDNO-OVWNO(R2)	;READ NEG OFFSET < 6
1474	016570	101403			BLOS	3\$	;BRANCH IF NOT
1475	016572	004537	017152		JSR	R5,PRINT	;REPORT EXCEPTIONS
1476	016576	000003					;COLUMN POSITION ON REPORT
1477	016600	122762	000014 001620	3\$:	CMPB	#12.,RDPO-OVWNO(R2)	;READ POS OFFSET < 12 ?
1478	016606	101403			BLOS	4\$	;BRANCH IF NOT
1479	016610	004537	017152		JSR	R5,PRINT	;REPORT EXECPTIONS
1480	016614	000004					;COLUMN POSITION ON REPORT
1481	016616	000441		4\$:	BR	LOOP5	;EXIT
1482							
1483	016620	122712	000016	SPATH:	CMPB	#14.,(R2)	;OVERWRITE NEG OFF < 14
1484	016624	101403			BLOS	1\$	;BRANCH IF NOT
1485	016626	004537	017152		JSR	R5,PRINT	;REPORT EXECPTION
1486	016632	000001					;COLUMN POSITION ON REPORT
1487	016634	122762	000016 000460	1\$:	CMPB	#14.,OVWPO-OVWNO(R2)	;OVERWRITE POS OFFSET < 14
1488	016642	101403			BLOS	2\$	;BRANCH IF NOT
1489	016644	004537	017152		JSR	R5,PRINT	;REPORT EXCEPTION
1490	016650	000002					;COLUMN POSITION ON REPORT

```

1491 016652 013704 001364      2$:  MOV    CMTRK,R4      ;LOCATE THE SELF TEST SCORE
1492 016656 006304              ASL     R4              ;WORD INDEX
1493 016660 122764 000006 004356  CMPB   #6,SELF0(R4)  ;SELF READ NEG OFFSET < 6 ?
1494 016666 101403              BLOS    3$              ;BRANCH IF NOT
1495 016670 004537 017152      JSR     R5,PRINT
1496 016674 000003              000003      ;REPORT EXCEPTIONS
1497 016676 013704 001364      3$:  MOV    CMTRK,R4      ;LOCATE THE SELF TEST SCORE
1498 016702 006304              ASL     R4              ;WORD INDEX
1499 016704 122764 000006 004357  CMPB   #6,SELF0+1(R4) ;SELF READ POS OFFSET < 6 ?
1500 016712 101403              BLOS    4$              ;BRANCH IF NOT
1501 016714 004537 017152      JSR     R5,PRINT
1502 016720 000004              000004      ;REPORT EXCEPTIONS
1503 016722              4$:              ;COLUMN POSITION FOR REPORT PRINTING
1504
1505 016722 005201      LOOP5:  INC     R1              ;INCREASE THE LOGICAL DRIVE #
1506 016724 000241              CLC              ;ADJUST THE BIT POSITION
1507 016726 006103              ROL     R3              ;POINTS TO NEXT DRIVE
1508 016730 005202              INC     R2              ;UPDATE SCORE BOARD BASE ADDRESS
1509 016732 022701 000017      CMP     #15.,R1      ;ALL DRIVES ARE CHECK ?
1510 016736 103264              BHIS    LOOP4          ;BRANCH IF NOT ALL DONE
1511 016740 005237 001364      LOOP6:  INC     CMTRK          ;NEXT TRACK
1512 016744 012702 002056      MOV     #OVWNO,R2      ;RESET SCORE BOARD BASE ADDRESS
1513 016750 005001              CLR     R1              ;LOGICAL DRIVE START FROM 0
1514 016752 013703 001364      MOV     CMTRK,R3      ;LOCATE THE SCORE BOARD BASE ADDRESS
1515 016756 001404              BEQ     2$              ;BRANCH IF ON TRACK 0
1516 016760 062702 000020      1$:  ADD     #16.,R2      ;FOR EACH TRACK, 16 BYTES
1517 016764 005303              DEC     R3              ;ALL TRACK DONE ?
1518 016766 001374              BNE     1$              ;BRANCH IF NOT
1519 016770 012703 000001      2$:  MOV     #BIT0,R3      ;BIT MAP POSITION FOR DRIVE 0
1520
1521      ;AT THIS POIN :
1522      :      R1      = LOGICAL DRIVE # START FROM 0
1523      :      R2      = ADDRESS OF SCORE BOARD BASE ADDRESS
1524      :      R3      = BIT MAP, INDICATE LOGICAL DRIVE 0 1
1525
1526 016774 123737 001364 001372  CMPB   CMTRK,TRKLMT    ;ALL TRACKS ARE DONE ?
1527 017002 003642              BLF     LOOP4          ;NO
1528 017004 005737 001342      TST     @ADSEC        ;WAS SCORE PRINTED ?
1529 017010 001402              BEQ     DISMNT        ;BR IF NO
1530 017012 104401 001207      TYPE    ,$CRLF        ;CR-LF
1531
1532 017016 104401 001207 011276  DISMNT: TYPE    , $CRLF      ;CR-LF
1533 017022 012777 017444      MOV     #IDLEX,@RMVEC  ;RESET THE INTERRUPT VECTOR
1534 017030 005077 011274      CLR     @RMVEC+2      ;CLEAR THE INTERRUPT VECTOR
1535 017034 104401 042272      TYPE    ,MSG12        ;UNLOAD AND DISMOUNT MESSAGE
1536 017040 104401 001274      TYPE    , $CDW2        ;SYSTEM NAME
1537 017044 013746 001224      MOV     DRIVE,-(SP)   ;PHYSICAL DRIVE NUMBER
1538 017050 104403              TYPPOS
1539 017052      001      .BYTE    1
1540 017053      000      .BYTE    0
1541 017054 104401 042310      TYPE    ,MSG13        ;MESSAGE TYPE <CR> WHEN READY
1542
1543 017060 104411      1$:  RDLIN          ;READ IN ONE LINE
1544 017062 012605      MOV     (SP)+,R5      ;LOCATE THE READ IN LINE
1545 017064 105715      TSTB    (R5)          ;CARRIAGE RETURN ?
1546 017066 001374      BNE     1$              ;BR IF NO
1547 017070 005002      DEASG:  CLR     R2          ;LOGICAL DRIVE NUMBER

```

```

1551 017072 012703 000001      MOV    #BIT0,R3      ;BIT MAP OF ASSIGNED DRIVE
1552 017076 020237 C01270      1$:    CMP    R2,$DEVM      ;IS THE LOGICAL DRIVE UNDER TEST ?
1553 017102 001404              BEQ     2$              ;BRANCH IF IT IS
1554 017104 000241              CLC                     ;SHIFT THE BIT MAP FOR
1555 017106 006103              ROL     R3              ;NEXT DRIVE
1556 017110 005202              INC     R2              ;INCRFMENT THE LOGICAL DRIVE #
1557 017112 000771              BR      1$              ;LOOPING ,UNTIL THE LOGICAL DRIVE LOCATED
1558 017114 040337 002012      2$:    BIC     R3,ASSGN2      ;CLEAR THE ASSIGNED BIT
1559 017120 001412              BEQ     XEND2          ;BRANCH IF NO MORE DRIVES
1560 017122 005237 001270      INC     $DEVM          ;INCREMENT THE LOGICAL DRIVE #
1561 017126 013702 001270      MOV     $DEVM,R2        ;UPDATE SYSTEM BLOCK ADDRESS
1562 017132 006302              ASL     R2
1563 017134 016237 004444 001272  MOV    BLKADR(R2),$CDW1      ;SYSTEM BLOCK ADDRESS
1564 017142 000137 012772      JMP     TST5              ;JUMP TO TEST 5 FOR OTHER DRIVE
1565
1566 017146 000137 023456      *ND2:  JMP     $EOP          ;END OF PASS
1567
1568
1569      ;PRINT ROUTINE
1570      ;NAME PRINT
1571      ;PRINT EXCEPTIONS FOR TEST SCORE
1572      ;PARAMETER USED
1573      :      NULINE   = 0      A NEW LINE TO PRINT
1574      :              = 1 - 4  COLUMN NUMBER TO PRINT THE EXCEP'ION MARK
1575      :
1576      :      BADSEC   = 0      FIRST EXCEPTION DETECTED
1577      :      BADSEC   = -1     NOT FIRST EXCEPTION (DON'T PRINT THE TITLE)
1578      :
1579      ;CALL
1580      :      JSR      R5,PRINT
1581      :      NUMBER   COLUMN NUMBER
1582      :      RET
1583      :
1584      :      R1        = LOGICAL DRIVE #
1585      :      $DEVM     = LOGICAL DRIVE UNDER TEST
1586      :      R3        = BIT POSITION OF LOGICAL DRIVE IN R1
1587      :
1588
1589      PRINT:
1590      017152 010146      MOV     R1,-(SP)      ;;PUSH R1 ON STACK
1591      017154 010246      MOV     R2,-(SP)      ;;PUSH R2 ON STACK
1592      017156 010346      MOV     R3,-(SP)      ;;PUSH R3 ON STACK
1593      017160 010446      MOV     R4,-(SP)      ;;PUSH R4 ON STACK
1594      017162 005737 001342  TST     BADSEC      ;FIRST EXCEPTION
1595      017166 100434      BMI     1$              ;BRANCH IF NOT,DON'T HAVE TO PRINT
1596
1597      104401 001207      TYPE     , $CRLF          ;CRLF
1598      104401 042443      TYPE     , $MSG16        ;SCORES FOR DRIVE --
1599      104401 001274      TYPE     , $CDW2          ;SYSTEM NAME
1600      017204 013746 001224  MOV     DRIVE,-(SP)      ;TYPE THE PHYSICAL DRIVE #
1601      104403
1602      017212 001      .BYTE     1
1603      017213 000      .BYTE     0
1604      104401 001207      TYPE     , $CRLF          ;CR-LF
1605      104401 001207      TYPE     , $CRLF          ;CR-LF
1606      104401 042465      TYPE     , $MSG17        ;SUB TITLE 1
1607      104401 001207      TYPE     , $CRLF          ;CR-LF

```

1604	017234	104401	042546		TYPE	,MSG18	:SUB TITLE 2
1605	017240	104401	001207		TYPE	,\$CRLF	:CR-LF
1606	017244	012737	177777	001342	MOV	#-1,BADSEC	:RESET THE ACCEPTANCE FLAG
1607	017252	012737	000001	001376	MOV	#1,FAULT	:NOT COMPATIBLE FLAG
1608	017260	005737	001366	1\$:	TST	NULINE	:NEW LINE ?
1609	017264	001042			BNE	5\$	:BRANCH IF NOT
1610	017266	104401	001207		TYPE	,\$CRLF	:CR-LF
1611	017272	013746	001364		MOV	(MTRK,-(SP)	:SAVE CMTRK FOR TYPEOUT
	017276	104405			TYPDS		:GO TYPE--DECIMAL ASCII WITH SIGN
1612	017300	104401	042635		TYPE	,TAB	:TYPE 'TAB' CHARACTER
1613	017304	123701	001270		CMPS	\$DEVN,R1	:SCORE FOR \$DEVN ITSELF ?
1614	017310	001007			BNE	2\$	:BRANCH IF NOT
1615	017312	104401	042750		TYPE	,BLNKS3	:TYPE 3 BLANKS
1616	017316	104401	042630		TYPE	,SELF	:MESSAGE 'SEL'
1617	017322	104401	042635		TYPE	,TAB	:TYPE TAB
1618	017326	000416			BR	4\$	:NEXT STEP
1619	017330	006301		2\$:	ASL	R1	:LOCATE THE SYS HISTORY FILE
1620	017332	016137	004444	017356	MOV	BLKADR(R1),3\$	:LOCATE THE SYSTEM NAME AND DRIVE #
1621	017340	006201			ASR	R1	:RESTORE DRIVE #
1622	017342	062737	000014	017356	ADD	#\$SYSNM,3\$	:LOCATE THE SYSTEM NAME AND DRIVE #
1623	017350	104401	042750		TYPE	,BLNKS3	:TYPE 3 BLANKS
1624	017354	104401			TYPE		:TYPE THE SYSTEM AND DRIVE
1625	017356	000000		3\$:	.WORD	0	:ADDRESS FOR TYPING MESSAGE
1626	017360	104401	042635		TYPE	,TAB	:TYPE TAB
1627	017364	012737	000001	001366	4\$:	MOV	#1,NULINE
1628	017372	012504			5\$:	MOV	(R5)+,R4
1629							:RETRIEVE THE DESIRED COLUMN # FROM
1630	017374	020437	001366	6\$:	CMP	R4,NULINE	:CALLING ROUTINE
1631	017400	103414			BLO	8\$	:ON THE RIGHT COLUMN ?
1632	017402	001405			BEQ	7\$	:IF LOW NOT PRINT
1633	017404	104401	042635		TYPE	,TAB	:BRANCH IF LOCATED
1634	017410	005237	001366		INC	NULINE	:ADVANCE TO NEXT COLUMN
1635	017414	000767			BR	6\$	:INCREMENT COLUMN CTR
1636	017416	104401	042747	7\$:	TYPE	,BLNKS4	:CHECK AGAIN
1637	017422	104401	042637		TYPE	,MARKX	:TYPE 4 BLANKS
1638	017426	005237	001366		INC	NULINE	:THE EXCEPTION MARK '* 0'
1639	017432			8\$:			:NEXT COLUMN
	017432	012604			MOV	(SP)+,R4	:POP STACK INTO R4
	017434	012603			MOV	(SP)+,R3	:POP STACK INTO R3
	017436	012602			MOV	(SP)+,R2	:POP STACK INTO R2
	017440	012601			MOV	(SP)+,R1	:POP STACK INTO R1
1640	017442	000205			RTS	R5	:EXIT
1641							
1642	017444	000240		IDLEX:	NOP		:RESET ALL RH/RM VECTOR FOR MOUNT AND DISMOUNT
1643	017446	000240			NOP		
1644	017450	000002			RTI		:EXIT
1645							
1646							
1647							:PROCESS THE ORDER TERMINATION
1648	017452	111037	001330	PROCES:	MOVB	(R0),UNIT	:DRIVE NUMBER FOR ANY ERROR MESSAGES
1649	017456	005760	000016		TST	\$STATUS(R0)	:SEE IF DRIVER SIGNALLED AN ERROR
1650	017462	100431			BMI	ERPROC	:BR IF ERROR
1651	017464	032760	100000	000020	BIT	#BIT15,\$RMCS1(R0)	:SEE IF 'SC' SET
1652	017472	001410			BEQ	1\$	:BR IF NOT SET
1653	017474	032760	040000	000020	BIT	#BIT14,\$RMCS1(R0)	:SEE IF 'TRE' SET
1654	017502	001021			BNE	ERPROC	:BR IF SET
1655	017504	032760	040000	000032	BIT	#BIT14,\$RMDS(R0)	:SEE IF 'ERR' SET

```

1656 017512 001015      BNE      ERPROC      ;BR IF SET
1657 017514 004737 020572 1$:      JSR      PC,CKERR      ;NO ERROR, CHECK ERROR BITS ANYWAY
1658 017520 004737 020640      JSR      PC,CKBUS      ;NO ERROR, CHECK BUS ADDR & WC
1659 017524 004737 030334      JSR      PC,RMINIT      ;INITIALIZE THE SUB SYSTEM
1660 017530 012737 177777 030264      MOV      #-1,SAVEFG      ;CLEAR THE SAVE FLAG
1661 017536 012737 177777 030266      MOV      #-1,SEEKFG      ;SET THE NOT IMP FLAG
1662 017544 000207      RTS      PC      ;RETURN
1663
1664      ;ORDER TERMINATED WITH AN ERROR - PROCESS THE ERROR
1665
1666 017546 032760 000200 000016 ERPROC: BIT      #BIT07,$STATUS(R0) ;DONE BIT SET ?
1667 017554 001402      BEQ      ERPRC1      ;BR IF ORDER DIDN'T COMPLETE NORMALLY
1668 017556 000137 017744      JMP      DONE      ;PROCESS ERROR WITH 'DONE' BIT SET
1669
1670      ;PROCESS ORDER COMPLETION WITH 'ERROR' & 'DONE NOT' BITS
1671
1672 017562 032760 010000 000016 ERPRC1: BIT      #BIT12,$STATUS(R0) ;SEE IF DRIVE WAS UNSAFE
1673 017570 001025      BNE      PUNSAF      ;BR IF YES
1674 017572 032760 004000 000016      BIT      #BIT11,$STATUS(R0) ;PARITY ERROR OCCURRED
1675 017600 001025      BNE      UCPAR      ;BR IF IT DID
1676 017602 032760 002000 000016      BIT      #BIT10,$STATUS(R0) ;FATAL PARITY ERROR?
1677 017610 001025      BNE      FALPAR      ;BR IF THERE IS ONE
1678 017612 032760 001000 000016      BIT      #BIT09,$STATUS(R0) ;TIMEOUT?
1679 017620 001025      BNE      SWTIM      ;BR IF YES
1680 017622 032760 040002 000016      BIT      #BIT14!BIT01,$STATUS(R0) ;DRIVE WENT OFFLINE ?
1681 017630 001025      BNE      OFLIN      ;BR IF IT DID
1682 017632 032760 000004 000016      BIT      #BIT2,$STATUS(R0) ;PORT REQUEST TIME OUT ?
1683 017640 001025      BNE      PRIM      ;BR IF IT DID
1684 017642 000207      RTS      PC      ;ERROR. RETURN
1685
1686      ;DRIVE IS PERSISTENTLY UNSAFE
1687
1688 017644      PUNSAF:
1689 017644 104414 037160      DISPLY      ,EM12
1690 017650 000137 017720      JMP      DUMP2
1691
1692      ;UNCORRECTABLE MASSBUS PARITY ERROR OCCURRED
1693
1694 017654      UCPAR:
1695 017654 104414 037062      DISPLY      ,EM10
1696 017660 000137 017720      JMP      DUMP2
1697
1698      ;'FATAL' MASSBUS PARITY ERROR OCCURRED
1699
1700 017664      FALPAR:
1701 017664 104414 037125      DISPLY      ,EM11
1702 017670 000137 017720      JMP      DUMP2
1703
1704      ;SOFTWARE TIMEOUT OCCURRED
1705
1706 017674      SWTIM:
1707 017674 104414 037211      DISPLY      ,EM13
1708 017700 000137 017720      JMP      DUMP2
1709
1710      ;DRIVE WENT OFFLINE
1711
1712 017704      OFLIN:

```

```

1713 017704 104414 037263
1714 017710 000137 017720
1715
1716
1717 017714
1718 017714 104414 037306
1719 017720 104401 001207
1720 017724 104401 043243
1721 017730 104401 001207
1722 017734 104401 043106
1723 017740 000177 161434
1724
1725
1726
1727 017744 032760 000030 000016
1728 017752 001402
1729 017754 000137 020530
1730 017760 032760 040000 000030
1731 017766 001402
1732 017770 000137 020310
1733 017774 032760 040000 000032
1734 020002 001002
1735 020004 000137 020500
1736 020010 032760 000400 000034
1737 020016 001402
1738 020020 000137 020340
1739 020024 032760 000020 000034
1740 020032 001402
1741 020034 000137 020360
1742 020040 032760 000200 000034
1743 020046 001402
1744 020050 000137 020370
1745 020054 032760 020300 000034
1746 020062 001402
1747 020064 000137 020410
1748 020070 032760 000010 000034
1749 020076 001402
1750 020100 000137 020430
1751 020104 032760 000040 000034
1752 020112 001402
1753 020114 000137 020520
1754 020120 032760 002000 000034
1755 020126 001402
1756 020130 000137 020440
1757 020134 032760 004000 000034
1758 020142 001402
1759 020144 000137 020450
1760 020150 032760 001000 000034
1761 020156 001405
1762 020160 032760 002000 000032
1763 020166 001401
1764 020170 000207
1765 020172 032760 010000 000034
1766 020200 001402
1767 020202 000137 020420
1768 020206 005760 000034
1769 020212 100002

```

```

DISPLY ,EM14
JMP DUMP2
;PORT REQUEST TIMEOUT ERROR

PRTIM:
DUMP2: DISPLY ,EM15
        TYPE ,SCRLF ;CR-LF
        TYPE ,XFATL
        TYPE ,SCRLF ;CR-LF
        TYPE ,HALTX
        JMP @RSTART ;JUMP TO RESTART

;PROCESS ORDER COMPLETION WITH 'ERROR' & 'DONE' BITS SET

1$:
DONE: BIT #BIT04,BIT03,STATUS(R0) ;UNSAFE OCCURRED
      BEQ .+6 ;BR IF NOT
      JMP UNSAF ;REPORT UNSAFE
      BIT #BIT14,$RMCS2(R0) ;IS 'WCE' SET?
      BEQ .+6 ;BRANCH IF NOT SET
      JMP WCKER ;WRITE CHECK ERROR
      BIT #BIT14,$RMDS(R0) ;CHECK 'ERR'
      BNE 1$ ;BR IF SET
      JMP TRFER ;PROCESS 'TRE'
      BIT #BIT08,$RMER1(R0) ;'HCRC' SET?
      BEQ .+6 ;BR IF NOT
      JMP HRCER ;PROCESS 'HCRC'
      BIT #BIT04,$RMER1(R0) ;'FMT' SET?
      BEQ .+6 ;BR IF NOT SET
      JMP CKFMT ;CHECK FORMAT ERROR
      BIT #BIT07,$RMER1(R0) ;'HCE' SET?
      BEQ .+6 ;BR IF NOT SET
      JMP CKHCE ;CHECK 'HCE' ERROR
      BIT #BIT13,$RMER1(R0) ;'OPI' SET?
      BEQ .+6 ;BR IF NOT SET
      JMP OPIER ;REPORT 'OPI'
      BIT #BIT3,$RMER1(R0) ;'PAR' SET?
      BEQ .+6 ;BR IF NOT SET
      JMP PARER ;REPORT 'PAR'
      BIT #BIT5,$RMER1(R0) ;'WCF' SET?
      BEQ .+6 ;BR IF NOT SET
      JMP WCFER ;REPORT 'WCF'
      BIT #BIT10,$RMER1(R0) ;'IAE' SET?
      BEQ .+6 ;BR IF NOT SET
      JMP IAEER ;REPORT 'IAE'
      BIT #BIT11,$RMER1(R0) ;'WLE' SET?
      BEQ .+6 ;BR IF NOT SET
      JMP WLEER ;REPORT 'WLE'
      BIT #BIT9,$RMER1(R0) ;'AOE' SET?
      BEQ 2$ ;BR IF NOT SET
      BIT #BIT10,$RMDS(R0) ;'LST' SET?
      BEQ 2$ ;BR IF NOT SET
      RTS PC ;'AOE' & 'LST' SET, EXIT
2$:
      BIT #BIT12,$RMER1(R0) ;SEE IF 'DTE' SET
      BEQ .+6 ;BR IF NOT
      JMP DTEER ;REPORT 'DTE' ERROR
      TST $RMER1(R0) ;SEE IF 'DCK' SET
      BPL .+6 ;BR IF NOT

```

```

1770 020214 000137 020300      JMP      DCKER      ;PROCESS 'DCK'
1771 020220 032760 060000 000062  BIT      #BIT14!BIT13,$RMER2(R0) ;'SKI' OR 'OCYL' SET
1772 020226 001006          BNE      3$      ;BR IF IT IS
1773 020230 032760 100000 000062  BIT      #BIT15,$RMER2(R0) ;BAD SPOT ?
1774 020236 001004          BNE      4$      ;BRANCH IF SO
1775 020240 000137 020350      JMP      DRIVER      ;REPORT ERROR
1776 020244 000137 020510 3$:      JMP      SKIER      ;REPORT DRIVE ERROR
1777 020250 104401 001207 4$:      TYPE      , $CRLF      ;CR-LF
1778 020254 104401 042207      TYPE      ,MSG11
1779 020260 104007          EMT      7
1780 020262 104010          EMT      10
1781 020264 104011          EMT      11
1782 020266 104012          EMT      12
1783 020270 104401 043106      TYPE      ,HALTX
1784 020274 000177 161100      JMP      @RSTART      ;JUMP TO RESTART
1785
1786      ;PROCESS DATA ('DCK') CHECK ERROR
1787
1788 020300      DCKER:
1789 020300 104414 037363      DISPLY      ,EM21
1790 020304 000137 020540      JMP      DUMP
1791
1792      ;WRITE CHECK ERROR PROCESSING
1793
1794 020310      WCKER:
1795 020310 032760 100000 000034  BIT      #BIT15,$RMER1(R0) ;DCK BIT SET ?
1796 020316 001004          BNE      1$      ;BRANCH IF SET
1797 020320 104414 037467      DISPLY      ,EM23
1798 020324 000137 020540      JMP      DUMP
1799 020330 104414 037414 1$:      DISPLY      ,EM22
1800 020334 000137 020540      JMP      DUMP
1801
1802      ;REPORT 'HCRC' ERROR
1803
1804 020340      HCRCER:
1805 020340 104414 037342      DISPLY      ,EM20
1806 020344 000137 020540      JMP      DUMP
1807
1808      ;REPORT DRIVE ERROR
1809
1810 020350      DRIVER:
1811 020350 104414 037757      DISPLY      ,EM30
1812 020354 000137 020540      JMP      DUMP
1813
1814      ;PROCESS FORMAT ('FER') ERROR
1815
1816 020360      CKFMT:
1817 020360 104414 037546      DISPLY      ,EM24
1818 020364 000137 020540      JMP      DUMP
1819
1820      ;PROCESS HEADER COMPARE ('HCE') ERROR
1821
1822 020370      CKHCE:
1823 020370 104414 037614      DISPLY      ,EM25
1824 020374 000137 020540      JMP      DUMP
1825
1826      ;POSSIBLE POSITIONING ERROR

```

1827	020400	104414	041120	POSER:	DISPLY	,EM51
1828	020404	000137	020540		JMP	DUMP
1829						
1830				;REPORT 'OPI' ERROR		
1831	020410			OPIER:		
1832	020410	104414	040011		DISPLY	,EM31
1833	020414	000137	020540		JMP	DUMP
1834						
1835				;REPORT 'DTE' ERROR		
1836						
1837	020420			DTEER:		
1838	020420	104414	040054		DISPLY	,EM32
1839	020424	000137	020540		JMP	DUMP
1840						
1841				;REPORT 'PAR' ERROR		
1842						
1843	020430			PARER:		
1844	020430	104414	040107		DISPLY	,EM33
1845	020434	000137	020540		JMP	DUMP
1846						
1847				;REPORT 'IAE' ERROR		
1848						
1849	020440			IAEER:		
1850	020440	104414	040226		DISPLY	,EM35
1851	020444	000137	020540		JMP	DUMP
1852						
1853				;REPORT WLE ERROR		
1854						
1855	020450			WLEER:		
1856	020450	104414	040264		DISPLY	,EM36
1857	020454	000137	020540		JMP	DUMP
1858						
1859				;REPORT FORMAT ERROR		
1860						
1861	020460			FMTER:		
1862	020460	104414	037675		DISPLY	,EM26
1863	020464	000137	020540		JMP	DUMP
1864						
1865				;REPORT HEADER COMPARE ERROR		
1866						
1867	020470	104414	037722	HCEER:	DISPLY	,EM27
1868	020474	000137	020540		JMP	DUMP
1869						
1870				;PROCESS CONTROL/INTERFACE TRANSFER ERROR		
1871						
1872	020500			TRFER:		
1873	020500	104414	040377		DISPLY	,EM40
1874	020504	000137	020540		JMP	DUMP
1875						
1876				;PROCESS 'SKI' OR 'OCYL'		
1877						
1878	020510			SKIER:		
1879	020510	104414	041062		DISPLY	,EM50
1880	020514	000137	020540		JMP	DUMP
1881						
1882				;REPORT WRITE CLOCK FAILURE		
1883						

```

1884 020520
1885 020520 104414 040164
1886 020524 000137 020540
1887
1888
1889
1890 020530
1891 020530 104414 041163
1892 020534 000137 020540
1893
1894 020540
      020540 104007
1895 020542 104010
1896 020544 104011
1897 020546 104012
1898
1899 020550 004737 030334
1900 020554 012737 177777 030264
1901 020562 012737 177777 030266
1902 020570 000207
1903
1904
1905
1906
1907 020572 032760 060000 000020
1908 020600 001012
1909 020602 032760 177400 000030
1910 020610 001006
1911 020612 005760 000034
1912 020616 001003
1913 020620 005760 000062
1914 020624 001404
1915 020626 104414 040640
1916 020632 000137 020540
1917 020636 000207
1918
1919
1920
1921 020640 005760 000022
1922 020644 001011
1923 020646 016046 000004
1924 020652 005416
1925 020654 006316
1926 020656 066016 000006
1927 020662 022660 000024
1928 020666 001404
1929 020670 104414 040446
1930 020674 000137 020540
1931 020700 000207
1932
1933
1934 020702 104401 001207
1935 020706 104401 042764
1936 020712 104401 001207
1937 020716 016001 000024
1938 020722 016046 000004
1939 020726 005416

WCFER:
      DISPLY ,EM34
      JMP DUMP

;REPORT DRIVE UNSAFE ERROR
UNSAF:
      DISPLY ,EM60
      JMP DUMP

DUMP:
      EMT 7
      EMT 10
      EMT 11
      EMT 12

      JSR PC,RMINIT ;CLEAR THE SUB-SYSTEM
      MOV #-1,SAVEFG
      MOV #-1,SEEKFG
      RTS PC

;CHECK ERROR BITS IN THE RH/RM REGISTERS
CKERR: BIT #60000,$RMCS1(R0) ;SEE IF 'TRE' OR 'MCPE'
      BNE 1$ ;YES
      BIT #177400,$RMCS2(R0) ;ERROR BITS IN CS2 /
      BNE 1$ ;YES
      TST $RMER1(R0) ;ANY ERROR IN ER1
      BNE 1$ ;YES
      TST $RMER2(R0) ;ANY ERROR IN ER2
      BEQ 2$ ;BRANCH IF NO ERROR
1$: DISPLY ,EM44
      JMP DUMP ;TYPE ALL REGISTERS
2$: RTS PC

;CHECK BUS ADDRESS REGISTER AND WORD COUNT REGISTER
CKBUS: TST $RMWC(R0) ;WORD COUNT = 0
      BNE 1$ ;NO
      MOV $WRDM(R0),-(SP) ;WORD LENGTH
      NEG (SP) ;GET THE POSITIVE NUMBER OF WORD COUNT
      ASL (SP) ;BYTE COUNT
      ADD $BUF(R0),(SP)
      CMP (SP)+,$RMBA(R0)
      BEQ 2$
1$: DISPLY ,EM41
      JMP DUMP ;TYPE ALL REGISTERS
2$: RTS PC

;ROUTINE TO DISPLAY THE SECTOR WHICH GAVE THE HARD ERROR
PRTBAD: TYPE ,$CRLF ;CR-LF
      TYPE ,MSG19
      TYPE ,$CRLF ;CR-LF
      MOV $RMBA(R0),R1 ;PUT THE END ADDRESS INTO R1
      MOV $WRDM(R0),-(SP) ;FIND THE BEGINNING OF THE SECTOR
      NEG (SP) ;GET THE POSITIVE NUMBER OF WORD COUNT

```

```

1940 020730 066016 000022      ADD    $RMWC(R0),(SP)  :SUBTRACT THE WORDS NOT TRANSFERED
1941 020734 005046              CLR    -(SP)                :MAKE THE UPPER DIVIDEND 0
1942 020736 012746 000400      MOV    #256,-(SP) :DIVIDE THE WORDS TRANSFERED BY THE SECTOR SIZE
1943 020742 004737 021764      JSR    PC,LINKDV          :DIVIDE
1944 020746 005716              TST    (SP)                :REMAINDER = 0 ?
1945 020750 001403              BEQ    1$                  :BR IF IT IS - COMPLETE SECTOR TRANSFERED
1946 020752 006316              ASL    (SP)                :CONVERT THE RESIDUAL SECTOR SIZE INTO BYTE COUNT
1947 020754 161601              SUB    (SP),R1              :SUBTRACT IT FROM THE END ADDRESS
1948 020756 000402              BR     2$                  :FINISH THE SIZING
1949 020760 162701 001000      1$: SUB    #1000,R1          :SUBTRACT FULL SECTOR SIZE FROM END ADDR
1950 020764 062706 000004      2$: ADD    #4,SP            :RESTORE THE STACK POINTER
1951 020770 012702 000007      3$: MOV    #7,R2              :R2 CONTAINS THE WORDS/LINE COUNT
1952 020774 020160 000024      4$: CMP    R1,$RMBA(R0)       :PRINTED ALL THE SECTOR ?
1953 021000 001415              BEQ    5$                  :BR IF ALL PRINTED
1954 021002 104414 042752      DISPLY  ,BLNKS1             :TYPE 1 BLANK
1955 021006 012146              MOV    (R1)+,-(SP)         :PUT THE DATA ON THE STACK
1956 021010 004737 021162      JSR    PC,LINOC1           :TYPE THE DATA
1957 021014 022711 177777      CMP    #-1,(R1)            :END OF FILE ?
1958 021020 001405              BEQ    5$                  :BRANCH IF SO
1959 021022 005302              DEC    R2                  :DECREMENT THE HORIZONTAL COUNT
1960 021024 001363              BNE    4$                  :BR IF NOT AT THE END OF THE LINE
1961 021026 104414 001207      DISPLY  ,$CRLF             :CR-LF
1962 021032 000756              BR     3$                  :RESTORE THE WORDS/LINE COUNT
1963 021034 104414 001207      5$: DISPLY  ,$CRLF             :PRINT WHAT REMAINS IN THE BUFFER
1964 021040 000207              6$: RTS    PC                :RETURN
1965
1966
1967
1968 021042 104412              FILBUF: SAVREG              :SAVE THE REGISTERS
1969 021044 016001 000006      MOV    $BUF(R0),R1          :BUFFER ADDRESS
1970 021050 016002 000004      MOV    $WRDM(R0),R2          :POSITIVE WORD COUNT
1971 021054 005402              NEG    R2                  :
1972 021056 012705 005262      MOV    #STNDAT,R5           :PATTERN ADDRESS
1973 021062 012703 000020      MOV    #20,R3              :PATTERN COUNT
1974 021066 012521              3$: MOV    (R5)+,(R1)+       :MOVE THE PATTERN INTO THE BUFFER
1975 021070 005302              DEC    R2                  :DECREMENT THE WORD COUNT
1976 021072 003407              BLE    4$                  :BR IF DONE (WORD COUNT = 0)
1977 021074 005303              DEC    R3                  :DECREMENT THE PATTERN COUNT
1978 021076 001373              BNE    3$                  :BR IF MORE PATTERN
1979 021100 012703 000020      MOV    #20,R3              :RESTORE PATTERN COUNT
1980 021104 012705 005262      MOV    #STNDAT,R5           :RESTORE THE ADDRESS
1981 021110 000766              BR     3$                  :CONTINUE DISTRIBUTING THE PATTERN
1982 021112 104413              4$: RESREG              :RESTORE THE REGISTERS
1983 021114 000207              RTS    PC                :RETURN
1984
1985
1986      .SBTTL  ERROR MESSAGE GENERATION ROUTINES
1987
1988      ;PRINT LINE 1 OF ERROR MESSAGE:
1989      ;'HH:MM:SS'
1990
1991 021116 032777 002000 160030 LINE1: BIT    #SW10,@SWR      :SWITCH 10 SET ?
1992 021124 001402              BEQ    1$                  :BR IF NOT
1993 021126 104401 001202      TYPE    ,SBELL             :RING THE BELL
1994 021132 032777 020000 160014 1$: BIT    #SW13,@SWR      :INHIBIT TYPEOUT ?
1995 021140 001403              BEQ    2$                  :BR IF NOT
1996 021142 104414 001207      DISPLY  , $CRLF             :CR-LF

```

```

1997 021146 000404
1998 021150 004737 021430
1999 021154 104414 042752
2000 021160 000207
2001
2002
2003
2004
2005
2006
2007
2008 021162 016646 000002
2009 021166 004737 022724
2010 021172 012637 021206
2011 021176 062737 000005 021206
2012 021204 104414
2013 021206 000000
2014 021210 012616
2015 021212 000207
2016
2017
2018
2019
2020
2021
2022
2023
2024 021214 016646 000002
2025 021220 004737 022674
2026 021224 004737 022300
2027 021230 012616
2028 021232 000207
2029
2030
2031
2032
2033
2034
2035
2036 021234 012737 177777 001316
2037 021242 012737 177777 001314
2038 021250 012737 021330 000004
2039 021256 005037 000006
2040 021262 005777 160014
2041 021266 005037 001316
2042 021272 005037 001314
2043 021276 013701 001306
2044 021302 012721 021526
2045 021306 012711 000300
2046 021312 012777 174575 157764
2047 021320 012777 000131 157754
2048 021326 000434
2049 021330 062706 000004
2050 021334 012737 021376 000004
2051 021342 005777 157742
2052 021346 005037 001316
2053 021352 013701 001312

;OCTAL TYPEOUT ROUTINE
;CALL:
;      MOV      NUM,-(SP)      ;PUT THE NUMBER ON THE STACK
;      JSR      PC,LINOCT
;      RETURN
LINOCT: MOV      2(SP),-(SP)    ;PUT NUMBER IN PROPER LOCATION ON STACK
;      JSR      PC,$SB20      ;CONVERT THE NUMBER TO OCTAL
;      MOV      (SP)+,1$      ;GET THE ADDRESS OF THE ASCII STRING
;      ADD      #5,1$         ;ADDRESS THE LAST 6 ASCII DIGITS
;      DISPLY   ;TYPE IT
1$:      .WORD      0          ;ADDRESS
;      MOV      (SP)+,(SP)    ;CORRECT THE STACK
;      RTS      PC           ;RETURN

;ROUTINE TO CONVERT THE INPUT NUMBER TO DECIMAL AND YPE IT WITH
;LEADING ZERO SUPPRESSION
;CALL:
;      MOV      NUM,-(SP)      ;PUT THE NUMBER ON THE STACK
;      JSR      PC,LINDEC
;      RETURN
LINDEC: MOV      2(SP),-(SP)    ;SET UP STACK FOR CONVERT
;      JSR      PC,$SB2D      ;CONVERT IT TO DECIMAL
;      JSR      PC,$SUPRS     ;TYPE IT (WITH LEADING ZEROS SUPRESSED)
;      MOV      (SP)+,(SP)    ;RESTORE STACK POINTER
;      RTS      PC

.SBTTL  GENFRAL SUPPORT SUBROUTINES

;ROUTINE TO CHECK FOR KW11-L OR KW11-P CLOCKS
CKCLK:  MOV      #-1,CLKFLG    ;CLEAR CLOCK AVAILABILITY FLAG
;      MOV      #-1,PCLOCK     ;CLEAR KW11-P CLOCK AVAILABILITY FLAG
;      MOV      #CKCLK1,ERRVEC ;SET UP VECTOR FOR CLOCK CHECK
;      CLR      @#ERRVEC+2     ;NEW PSW
;      TST      @CLKCSR        ;CHECK FOR KW11-P
;      CLR      CLKFLG        ;SET CLOCK AVAILABILITY FLAG
;      CLR      PCLOCK        ;SET KW11-P CLOCK FLAG
;      MOV      $LPVEC,R1      ;KW11-P VECTOR ADDRESS
;      MOV      #CLOCK,(R1)+   ;SET UP KW11-P VECTOR
;      MOV      #300,(R1)      ;PSW - PRI 6
;      MOV      #-1667,@CLKCSB ;LOAD COUNTER BUFFER WITH 16.67
;      MOV      #131,@CLKCSR   ;SET CLOCK - CNT UP, 10US, CONT INT
;      BR       CKCLK3
CKCLK1: ADD      #4,SP         ;RESTORE THE STACK POINTER
;      MOV      #CKCLK2,@ERRVEC ;CHANGE ERROR VECTOR TO CHECK FOR KW11-L
;      TST      @CLKS          ;LOOK FOR KW11-L
;      CLR      CLKFLG        ;SET CLOCK FLAG
;      MOV      $LLVEC,R1      ;KW11-L VECTOR ADDRESS

```

```

2054 021356 012721 021526      MOV      #CLOCK,(R1)+      ;SET UP KW11-L VECTOR
2055 021362 012711 000300      MOV      #300,(R1)      ;PSW - PRI 6
2056 021366 012777 000100      MOV      #100,@$LK$      ;SET KW11-L INTERRUPT
2057 021374 000411              BR      CKCLK3
2058 021376 062706 000004      CKCLK2: ADD      #4,SP      ;RESTORE THE STACK POINTER
2059 021402 104401 043257      TYPE      ,NEDCLK      ;'P OR L CLOCK MUST BE ON SYSTEM'
2060 021406 005737 000042      TST      42      ;UNDER MONITOR CONTROL ?
2061 021412 001400              BEQ      1$      ;BR IF NOT
2062 021414 000177 157760      1$:      JMP      @RSTART      ;JUMP TO RESTART
2063
2064 021420 012737 000006      CKCLK3: MOV      #6,@$ERRVEC      ;RESTORE THE ERROR VECTOR
2065 021426 000207              RTS      PC
2066
2067
2068      ;ROUTINE TO TYPE THE TIME
2069
2070 021430 005737 001316      $TIME:  TST      CLKFLG      ;CLOCK ON THE SYSTEM ?
2071 021434 001033              BNE      1$      ;BR IF NOT
2072 021436 104401 001207      TYPE      ,SCLF      ;CR-LF
2073 021442 013746 001344      MOV      HOUR,-(SP)      ;PUT 'HOURS' ON THE STACK
2074 021446 004737 022674      JSR      PC,$$B2D      ;CONVERT TO DECIMAL
2075 021452 004537 022210      JSR      R5,REPLZ      ;TYPE IT
2076 021456 000002              .WORD      2      ;TYPE 2 DIGITS
2077 021460 104401 042754      TYPE      ,COLON      ;':'
2078 021464 013746 001346      MOV      MINUTE,-(SP)      ;PUT 'MINUTES' ON THE STACK
2079 021470 004737 022674      JSR      PC,$$B2D      ;CONVERT TO DECIMAL
2080 021474 004537 022210      JSR      R5,REPLZ      ;TYPE IT
2081 021500 000002              .WORD      2      ;TYPE 2 DIGITS
2082 021502 104401 042754      TYPE      ,COLON      ;':'
2083 021506 013746 001350      MOV      SECOND,-(SP)      ;PUT SECONDS ON THE STACK
2084 021512 004737 022674      JSR      PC,$$B2D      ;CONVERT TO DECIMAL
2085 021516 004537 022210      JSR      R5,REPLZ      ;TYPE IT
2086 021522 000002              .WORD      2      ;TYPE 2 DIGITS
2087 021524 000207      1$:      RTS      PC
2088
2089      ;CLOCK HANDLER ROUTINE
2090
2091 021526 005337 001352      CLOCK:  DEC      SIXTEE      ;INCREMENT THE 1/60 SECOND COUNTER
2092 021532 001033              BNE      1$      ;BR IF A SECOND NOT COUNTED
2093 021534 013737 001320      MOV      HZ,SIXTEE      ;RESTORE THE VALUE
2094 021542 005237 001350      INC      SECOND      ;COUNT THE SECOND
2095 021546 022737 000074      .MP      #60.,SECOND      ;AT MAXIMUM ?
2096 021554 001022              BNE      1$      ;BR IF NOT
2097 021556 005037 001350      CLR      SECOND      ;CLEAR THE SECOND'S COUNTER
2098 021562 005237 001346      INC      MINUTE      ;COUNT THE MINUTE
2099 021566 022737 000074      CMP      #60.,MINUTE      ;AT MAXIMUM ?
2100 021574 001012              BNE      1$      ;BR IF NOT
2101 021576 005037 001346      CLR      MINUTE      ;CLEAR THE MINUTE'S COUNTER
2102 021602 005237 001344      INC      HOUR      ;COUNT THE HOURS
2103 021606 022737 001747      CMP      #999.,HOUR      ;AT MAXIMUM
2104 021614 103002              BHS      1$      ;BR IF NOT
2105 021616 005037 001344      CLR      HOUR      ;CLEAR THE HOURS
2106 021622 012746 000021      1$:      MOV      #17.,-(SP)      ;17 MS ON THE STACK
2107 021626 004737 034344      JSR      PC,RMTMR      ;DRIVER TIMER ROUTINE
2108 021632 000002      2$:      RTI
2109
2110      ;COMMAND DECODE ROUTINE

```

```

2111      :CALL:
2112      :      MOV      #-1,CFLAG      ;'CFLAG' IS NORMALLY SET BY THE TTY SERVICE
2113      :      :      :ROUTINE IN INTERRUPT MODE
2114      :      JSR      PC,KSR
2115      :      RETURN1
2116      :      RETURN2      ;SYSTEM BUSY RETURN
2117      :      :      ;RETURN AFTER KEYBOARD SERVICED
2118 021634 104412      KSR: SAVREG      ;SAVE THE REGISTERS
2119 021636 012737 000200 177776      MOV      #PR4,PS      ;SET PRIORITY TO 4
2120 021644 005037 001336      CLR      CFLAG      ;CLEAR THE 'CONTROL C' FLAG
2121 021650 004737 021430      JSR      PC,$TIME      ;TYPE THE TIME
2122 021654 005777 157302      TST      @STKB      ;CLEAR ANY GARBAGE IN THE TTY BUFFER
2123 021660 005737 001336      TST      CFLAG      ;CHECK THE CONTROL C FLAG
2124 021664 001002      BNE      7$      ;EXIT IF 'CONTROL C' ENTERED
2125 021666 000240      NOP      ;DUMP CODING FOR LATER USE 3/8/77
2126 021670 000240      NOP
2127 021672 104413      7$: RESREG      ;RESTORE R0 - R5
2128 021674 062716 000002      ADD      #2,(SP)      ;INCREMENT THE RETURN ADDRESS
2129 021700 005777 157256      TST      @STKB      ;CLEAR THE TTY BUFFER
2130 021704 052777 000100 157246      BIS      #BIT06,@STKS      ;SET TTY INTERRUPT ENABLE
2131 021712 005037 177776      CLP      PS      ;SET PRIORITY BACK TO ZERO
2132 021716 000207      RTS      PC      ;RETURN
2133
2134      ;ROUTINE TO CLEAR THE DPB FOR THE ASSIGNED DRIVE
2135      :CALL:
2136      :      MOV      #DPB,R0      ;DPB ADDRESS
2137      :      JSR      PC,CLRDPB
2138      :      RETURN
2139
2140      CLRDPB:
2141 021720 010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
2142 021722 010346      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
2143 021724 010446      MOV      R4,-(SP)      ;;PUSH R4 ON STACK
2144 021726 010546      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
2145 021730 010004      MOV      R0,R4      ;GET THE DPB ADDRESS
2146 021732 062704 000002      ADD      #2,R4      ;ADDRESS OF FIRST LOCN TO BE CLEARED
2147 021736 012703 000020      MOV      #SEMTAB-2,R3      ;NUMBER OF LOCATIONS TO CLEAR
2148 021742 005024      1$: CLR      (R4)+      ;CLEAR THE STORAGE LOCATION
2149 021744 162703 000002      SUB      #2,R3      ;DECREMENT THE BYTE COUNT
2150 021750 001374      BNE      1$      ;LOOPING BACK
2151 021752 012605      MOV      (SP)+,R5      ;;POP STACK INTO R5
2152 021754 012604      MOV      (SP)+,R4      ;;POP STACK INTO R4
2153 021756 012603      MOV      (SP)+,R3      ;;POP STACK INTO R3
2154 021760 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
2155 021762 000207      RTS      PC      ;RETURN
2156
2157 021764 104412      LINKDV: SAVREG      ;STORE R0 - R5
2158 021766 016605 000026      MOV      26(SP),R5      ;DIVISOR
2159 021772 005004      CLR      R4      ;OTHER DIVISOR WORD
2160 021774 016602 000030      MOV      30(SP),R2      ;UPPER DIVIDEND WORD
2161 022000 016603 000032      MOV      32(SP),R3      ;LOWER DIVIDEND WORD
2162 022004 005000      CLR      R0      ;CLEAR OTHER DIVIDEND REGISTERS
2163 022006 005001      CLR      R1
2164 022010 004737 022032      JSR      PC,M.DPID      ;GO TO THE DIVIDE ROUTINE
2165 022014 010166 000030      MOV      R1,30(SP)      ;REMAINDER ON THE STACK
2166 022020 010366 000032      MOV      R3,32(SP)      ;QUOTIENT ON THE STACK
2167 022024 104413      RESREG      ;RESTORE R0 - R5

```

```

2161 022026 012616      MOV      (SP)+,(SP)      ;MOVE RETURN UP THE STACK
2162 022030 000207      RTS      PC
2163
2164      :      DIVISION UTILITY SUBROUTINE
2165      :      R0-R1-R2-R3=DIVIDEND
2166      :      R4-R5=DIVISOR
2167      :      R0-R1=REMAINDER AFTER DIVISION
2168      :      R2-R3=QUOTIENT AFTER DIVISION
2169      :      ENTER WITH JSR PC,M.DPID
2170
2171 022032 012746 000040  M.DPID: MOV      #40,-(SP)      ;COUNTER FOR DIVISION CYCLES
2172 022036 010446      MOV      R4,-(SP)      ;HIGH ORDER
2173 022040 010546      MOV      R5,-(SP)      ;LOW ORDER DIVISOR TO THE STACK
2174 022042 005466 000002  NEG      2(SP)      ;FORM NEGATIVE
2175 022046 005416      NEG      @SP      ;VERSION OF THE DIVISOR
2176 022050 005666 000002  SBC      2(SP)
2177 022054 061601      ADD      @SP,R1
2178 022056 005500      ADC      R0      ;PERFORM THE INITIAL SUBTRACTION
2179 022060 066600 000002  ADD      2(SP),R0
2180 022064 103445      BCS      M.DP50      ;IF CARRY THEN OVERFLOW HAS OCCURRED
2181 022066 005046      CLR      -(SP)      ;THIS IS A LONGER LASTING CARRY BIT
2182 022070 006103  M.DP40: ROL      R3
2183 022072 006102      ROL      R2
2184 022074 006101      ROL      R1
2185 022076 006100      ROL      R0
2186 022100 005716      TST      @SP      ;TEST "CARRY" INDICATOR
2187 022102 001410      BEQ      M.DP41      ;IF NO "CARRY" THEN ADD ELSE SUBTRACT
2188 022104 005016      CLR      @SP      ;CLEAR UP FOR NEXT TIME
2189 022106 066601 000002  ADD      2(SP),R1
2190 022112 005500      ADC      R0      ;ADD -(DIVISOR)
2191 022114 005516      ADC      @SP      ;SET "CARRY"
2192 022116 066600 000004  ADD      4(SP),R0 ;<-
2193 022122 000404      BR      M.DP42
2194 022124 060501  M.DP41: ADD      R5,R1
2195 022126 005500      ADC      R0      ;ADD +(DIVISOR)
2196 022130 005516      ADC      @SP      ;SET "CARRY"
2197 022132 060400      ADD      R4,R0 ;<-
2198 022134 005516  M.DP42: ADC      @SP      ;SET "CARRY"
2199 022136 005716      TST      @SP      ;TEST THE UPDATE INDICATOR
2200 022140 001401      BEQ      .+4      ;IF ZERO FORGET IT
2201 022142 005203      INC      R3      ;NO CARRY POSSIBLE HERE
2202 022144 005366 000006  DEC      6(SP)      ;DECREMENT COUNTER
2203 022150 003347      BGT      M.DP40      ;BRANCH IF MORE TO DO
2204 022152 006003      ROR      R3
2205 022154 103404      BCS      M.DP44
2206 022156 060501      ADD      R5,R1
2207 022160 005500      ADC      R0
2208 022162 060400      ADD      R4,R0
2209 022164 000241      CLC
2210 022166 006103  M.DP44: ROL      R3
2211 022170 062706 000010  ADD      #10,SP      ;ADJUST STACK BY 4 WORDS
2212 022174 000242      CLV
2213 022176 000207      RTS      PC
2214 022200 062706 000006  M.DP50: ADD      #6,SP
2215 022204 000262      SEV
2216 022206 060207      RTS      PC
2217

```

```

2218
2219 ;ROUTINE TO REPLACE LEADING ZEROS IN A NUMERIC STRING WITH SPACES
2220 ;CALL
2221 ;:      MOV      #ADR,-(SP)      ;ADDRESS OF NUMBER (IN ASCII)
2222 ;:      JSR      R5,REPLZ
2223 ;:      .WORD    N               ;'N' IS NUMBER OF DIGITS TO BE TYPED
2224
2225 022210 010046      REPLZ: MOV      R0,-(SP)      ;SAVE R0
2226 022212 012746 000012      MOV      #10,-(SP)      ;MAXIMUM NUMBER OF DIGITS TO BE TYPED
2227 022216 162516      SUB      (R5)+,(SP)      ;SUBTRACT DIGITS TO FORM INDEX
2228 022220 016600 000006      MOV      6(SP),R0      ;ADDRESS OF NUMBER TO R0
2229 022224 122710 000060 1$: CMPB    #'0',(R0)      ;BYTE EQUAL TO ASCII '0' ?
2230 022230 001004      BNE      2$              ;BR IF NOT
2231 022232 112710 000040      MOVB    #40,(R0)      ;REPLACE THE ZERO WITH A SPACE
2232 022236 005200      INC      R0              ;INCREMENT THE BYTE ADDRESS
2233 022240 000771      BR       1$              ;GO BACK AND LOOK FOR MORE LEADING ZEROS
2234 022242 105710      2$: TSTB    (R0)          ;SEE IF ZERO BYTE TERMINATOR
2235 022244 001003      BNE      3$              ;BR IF NOT
2236 022246 005300      DEC      R0              ;BACKUP STRING POINTER
2237 022250 112710 000060      MOVB    #'0',(R0)      ;PUT A ZERO BACK IN
2238 022254 016637 000006 022270 3$: MOV      6(SP),4$      ;PUT ADDRESS IN LOCATION FOR TYPEOUT
2239 022262 062637 022270      ADD      (SP)+,4$      ;BEGINNING OF SIGNIFICANT DIGITS
2240 022266 104401      TYPE      TYPE              ;TYPE THE NUMBER
2241 022270 000000      4$: .WORD    0              ;ADDRESS OF NUMBER
2242 022272 012600      MOV      (SP)+,R0          ;RESTORE R0
2243 022274 012616      MOV      (SP)+,(SP)        ;MOVE RETURN ADDRESS
2244 022276 000205      RTS      R5              ;RETURN
2245
2246 ;TYPE NUMERICAL ASCII STRING SUPPRESS LEADING ZEROS
2247
2248 ;CALL
2249 ;:      MOV      #NUMADR,-(SP)      ;FIRST ADDRESS OF ASCII STRING
2250 ;:      JSR      PC,$SUPRS
2251
2252 022300 010046      $SUPRS: MOV      R0,-(SP)      ;SAVE R0
2253 022302 016600 000004      MOV      4(SP),R0      ;PICKUP THE POINTER
2254 022306 105710      1$: TSTB    (R0)          ;TERMINATOR ?
2255 022310 001403      BEQ      2$              ;BR IF YES
2256 022312 122720 000060      CMPB    #'0',(R0)+      ;IS THIS AN ASCII '0' ?
2257 022316 001773      BEQ      1$              ;BR IF YES
2258 022320 005300      2$: DEC      R0              ;BACKUP BY '1'
2259 022322 010037 022330      MOV      R0,3$        ;SAVE FOR TYPING
2260 022326 104414      DISPLY      0              ;GO PRINT
2261 022330 000000      3$: .WORD    0              ;ASCII POINTER GOES HERE
2262 022332 012600      MOV      (SP)+,R0          ;RESTORE R0
2263 022334 012616      MOV      (SP)+,(SP)        ;RESTORE THE STACK
2264 022336 000207      RTS      PC              ;RETURN
2265
2266 ;ROUTINE TO TYPE AT PRIORITY 4
2267
2268 022340 013746 177776      TYPRI4: MOV      @#PS,-(SP)      ;SAVE THE PRESENT STATUS
2269 022344 012737 000200 177776      MOV      #200,@#PS      ;CHANGE THE PRIORITY TO 4
2270 022352 012537 022362      MOV      (R5)+,1$        ;MESSAGE ADDRESS
2271 022356 004737 024576      JSR      PC,$TYPE      ;TYPE THE MESSAGE
2272 022362 000000      1$: .WORD    0              ;MESSAGE ADDRESS GOES HERE
2273 022364 000205      RTS      R5              ;RETURN
2274

```

```

2275 ;ROUTINE TO TYPE ERRORS
2276 ;CALL
2277 ;DISPLY
2278 ;MESADR
2279 ;RETURN
2280 ;MUST DEFINED IN 'TRAP' TABLE
2281 ;ADDRESS OF MESSAGE
2281 022366 032777 020000 156560 $DSPLY: BIT #BIT13,@SWR ;INHIBIT ERROR TIMEOUT ?
2282 022374 001002 BNE 1$ ;BR IF YES
2283 022376 000137 024576 JMP $TYPE ;TYPE THE MESSAGE
2284 022402 062716 000002 1$: ADD #2,(SP) ;INCREMENT THE RETURN
2285 022406 000002 RTI ;RETURN
2286
2287 ;THIS ROUTINE IS USED TO CHECK IF AN
2288 ;ASCII CHARACTER IS A DIGIT BETWEEN 0 AND 7.
2289 ;CALL
2290 ;MOV #ADR,R1 ;ADDRESS OF ASCII CHARACTER
2291 ;JSR R5,CK.OCT ;CHECK THE CHARACTER
2292 ;RETURN1 ;CHARACTER IS NOT BETWEEN 0-7
2293 ;RETURN2 ;CHARACTER IS IN R2 AS A
2294 ;OCTAL DIGIT
2295
2296 022410 121127 000060 CK.OCT: CMPB (R1),#0 ;LESS THAN ZERO?
2297 022414 103407 BLO 1$ ;YES -- BRANCH
2298 022416 121127 000067 CMPB (R1),#7 ;GREATER THAN SEVEN?
2299 022422 101004 BHI 1$ ;YES -- BRANCH
2300 022424 111102 MOVB (R1),R2 ;GET THE CHARACTER
2301 022426 042702 177770 BIC #^C7,R2 ;STRIP AWAY THE ASCII
2302 022432 005725 TST (R5)+ ;ADJUST FOR RETURN
2303 022434 000205 1$: RTS R5 ;RETURN
2304
2305 ;THIS ROUTINE IS USED TO CHECK AN ASCII CHARACTER
2306 ;AND DETERMINE IF IT IS A DIGIT BETWEEN 0 AND 9.
2307 ;CALL
2308 ;MOV #ADR,R1 ;ADDRESS OF ASCII CHARACTER
2309 ;JSR R5,CK.DEC ;CHECK THE CHARACTER
2310 ;RETURN1 ;NOT BETWEEN 0 AND 9
2311 ;RETURN2 ;BETWEEN 0 AND 9
2312 ;R2 = DIGIT
2313
2314 022436 121127 000060 CK.DEC: CMPB (R1),#0 ;LESS THAN ZERO?
2315 022442 103407 BLO 1$ ;YES -- BRANCH
2316 022444 121127 000071 CMPB (R1),#9 ;GREATER THAN NINE?
2317 022450 101004 BHI 1$ ;YES -- BRANCH
2318 022452 111102 MOVB (R1),R2 ;GET THE CHARACTER
2319 022454 042702 000060 BIC #0,R2 ;STRIP AWAY THE ASCII
2320 022460 005725 TST (R5)+ ;ADJUST FOR RETURN
2321 022462 000205 1$: RTS R5 ;RETURN
2322
2323 ;THIS ROUTINE WILL CHECK AN ASCII CHARACTER TO
2324 ;DETERMINE WHAT IT IS.
2325 ;CALL
2326 ;MOV #ADR,R1 ;ADDRESS OF ASCII CHARACTER
2327 ;JSR R5,CK.CHR ;CHECK CHARACTER
2328 ;RETURN ADR1 ;UNKNOWN CHARACTER
2329 ;RETURN ADR2 ;CARRIAGE RETURN * (R1)=ADR+1
2330 ;RETURN ADR3 ;COMMA * (R1)=ADR+1
2331 ;RETURN ADR4 ;PERIOD * (R1)=ADR+1

```

```
2332      :      RETURN  ADR5      :DIGIT BETWEEN 0 AND 7.
2333      :      RETURN  ADR6      :DIGIT BETWEEN 8 AND 9.
2334      :      :                :R2 = DIGIT * (R1)=ADR*
2335
2336 022464 105711      CK.CHR: TSTB   (R1)      :'CARRIAGE RETURN'?
2337 022466 001417      BEQ     3$      :YES -- BRANCH
2338 022470 121127 000054      CMPB   (R1),#',      :'COMMA'?
2339 022474 001413      BEQ     2$      :YES -- BRANCH
2340 022476 121127 000056      CMPB   (R1),#'.      :'PERIOD'?
2341 022502 001407      BEQ     1$      :YES -- BRANCH
2342 022504 004537 022436      JSR     R5,CK.DEC      :'DIGIT'?
2343 022510 000410      BR       4$      :NO -- BRANCH
2344 022512 004537 022410      JSR     R5,CK.OCT      :OCTAL ?
2345 022516 005725      TST     (R5)+      :DIGIT BETWEEN 8-9
2346 022520 005725      TST     (R5)+      :DIGIT BETWEEN 0-7
2347 022522 005725      1$: TST     (R5)+      :PERIOD
2348 022524 005725      2$: TST     (R5)+      :COMMA
2349 022526 005725      3$: TST     (R5)+      :CARRIAGE RETURN
2350 022530 005201      INC     R1      :MOVE POINTER TO NEXT CHARACTER
2351 022532 011505      4$: MOV     (R5),R5      :UNKNOWN CHARACTER
2352 022534 000205      RTS      R5      :RETURN
2353
2354      :THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL
2355      :CHARACTERS AND FORMS A DECIMAL VALUE BINARY NUMBER IN R2.
2356      :CALL
2357      :      MOV     #ADR,R1      :ADDRESS OF ASCII STRING
2358      :      MOV     #NUM,R2      :MAX. MAGNITUDE OF INPUT NUMBER
2359      :      JSR     R5,CK.DIG      :CHECK DIGITS
2360      :      RETURN  ADR1      :'CR' ONLY ENTERED -- R2=0
2361      :      RETURN  ADR2      :'PERIOD' ONLY ENTERED -- R2=0
2362      :      RETURN  ADR3      :ILLEGAL CHARACTER OR INPUT TOO LARGE -- R2 ?
2363      :      RETURN  ADR4      :'CR' -- R2 = NUMBER
2364      :      RETURN  ADR5      :'COMMA' -- R2 = NUMBER
2365      :      RETURN  ADR6      :'PERIOD' -- R2 = NUMBER
2366
2367 022536 010446      CK.DIG: MOV     R4,-(SP)      :SAVE R4
2368 022540 010346      MOV     R3,-(SP)      :SAVE R3
2369 022542 010246      MOV     R2,-(SP)      :SAVE THE MAX. SIZE ON THE STACK
2370 022544 005002      CLR     R2      :START WITH 0
2371 022546 005003      CLR     R3
2372 022550 005004      CLR     R4
2373 022552 004537 022464      JSR     R5,CK.CHR      :CHECK ONE CHARACTER
2374      022556 022652      6$      :ILLEGAL CHARACTER
2375      022560 022660      9$      :CARRIAGE RETURN
2376      022562 022657      6$      :
2377      022564 022654      7$      :
2378      022566 022572      1$      :DIGIT 0-7
2379      022570 022572      1$      :DIGIT 8-9
2380 022572 062705 000004      1$: ADD     #4,R5      :STEP RETURN POINTER PAST 'CR' & 'PERIOD' RETURNS
2381 022576 006303      2$: ASL     R3      :INPUT NUMBER *2
2382 022600 010346      MOV     R3,-(SP)      :SAVE *2
2383 022602 006303      ASL     R3      :
2384 022604 006303      ASL     R3      :
2385 022606 062603      ADD     (SP)+,R3      :(*2)+(*8) = *10
2386 022610 060203      ADD     R2,R3      :UPDATE THE INPUT NUMBER
2387 022612 004537 022464      JSR     R5,CK.CHR      :CHECK ONE CHARACTER
2388      022616 022656      8$      :ILLEGAL CHARACTER
```

```

022620 022642 5$ :CARRIAGE RETURN
022622 022640 4$ :
022624 022632 3$ :
022626 022576 2$ :DIGIT 0-7
022630 022576 2$ :DIGIT 8-9
2382 022632 105711 3$: TSTB (R1) :DOES A 'CR' FOLLOW THE 'PERIOD'
2383 022634 001010 BNE 8$ :IF NOT
2384 022636 005724 TST (R4)+ :INCREMENT THE RETURN
2385 022640 005724 4$: TST (R4)+ :INCREMENT THE RETURN
2386 022642 005724 5$: TST (R4)+ :INCREMENT THE RETURN
2387 022644 020316 CMP R3,(SP) :CHECK THE MAGNITUDE OF THE NUMBER
2388 022646 101004 BHI 9$ :BR IF ENTERED NUMBER TOO LARGE
2389 022650 000402 BR 8$ :BYPASS INCREMENT
2390 022652 005725 6$: TST (R5)+ :INCREMENT RETURN PAST INVALID RETURN
2391 022654 005725 7$: TST (R5)+ :INCREMENT RETURN
2392 022656 060405 8$: ADD R4,R5 :SETUP RETURN POINTER
2393 022660 010302 9$: MOV R3,R2 :ENTERED VALUE
2394 022662 005726 TST (SP)+ :CLEAN MAX. SIZE OFF OF STACK
2395 022664 012603 MOV (SP)+,R3 :RESTORE R3
2396 022666 012604 MOV (SP)+,R4 :RESTORE R4
2397 022670 011505 MOV (R5),R5 :GET RETURN ADDRESS
2398 022672 000205 RTS R5 :RETURN
2399
2400 :THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
2401 :UNSIGNED DECIMAL ASCII NUMBER.
2402 :CALL
2403 : MOV NUMBER,-(SP) :PUT THE NUMBER ON THE STACK
2404 : JSR PC,$SB2D :CALL
2405 : RETURN :ADDRESS OF THE 1ST ASCII CHAR IS ON THE STACK
2406
2407 :NOTE: THE PROGRAM REQUIRES THIS FORM OF '$SB2D', NOT THE VERSION ON
2408 : THE SYSMAC LIBRARY, REV C AND LATER
2409
2410 022674 016637 000002 022720 $SB2D: MOV 2(SP),1$ :SAVE THE BINARY NUMBER
2411 022702 012746 022720 MOV #1$,-(SP) :SET THE POINTER
2412 022706 004737 027004 JSR PC,$DB2D :CALL THE DOUBLE LENGTH CONVERT
2413 022712 012666 000002 MOV (SP)+,2(SP) :PICKUP THE POINTER
2414 022716 000207 RTS PC :RETURN
2415 022720 000000 000000 1$: .WORD 0,0
2416
2417 :THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
2418 :UNSIGNED OCTAL ASCII NUMBER.
2419 :CALL
2420 : MOV NUMBER,-(SP) :PUT THE NUMBER ON THE STACK
2421 : JSR PC,$SB2D :CALL
2422 : RETURN :ADDRESS OF THE 1ST ASCII CHAR IS ON THE STACK
2423
2424 :NOTE: THE PROGRAM REQUIRES THIS FORM OF '$SB2D', NOT THE VERSION ON
2425 : THE SYSMAC LIBRARY, REV C AND LATER
2426
2427 022724 016637 000002 022750 $SB2D: MOV 2(SP),1$ :SAVE THE BINARY NUMBER
2428 022732 012746 022750 MOV #1$,-(SP) :SET THE POINTER
2429 022736 004737 027200 JSR PC,$DB2D :CALL THE DOUBLE LENGTH CONVERT
2430 022742 012666 000002 MOV (SP)+,2(SP) :PICKUP THE POINTER
2431 022746 000207 RTS PC :RETURN
2432 022750 000000 000000 1$: .WORD 0,0
2433

```

```

2434      ;KEYBOARD INTERRUPT INITIALIZATION ROUTINE
2435      :CALL
2436      :      JSR      PC,$TKINT
2437      :      RETURN
2438
2439 022754 012737 023004 000060 $TKINT: MOV      # $TKSRV,TKVEC ;SETUP VECTOR
2440 022762 012737 000240 000062      MOV      #PR5,TKVEC+2 ;PRIORITY TO 5
2441 022770 005777 156166      TST      @ $TKB ;CLEAR THE BUFFER
2442 022774 012777 000100 156156      MOV      #BIT06,@ $TKS ;SET INTERRUPT ENABLE
2443 023002 000207      RTS      PC ;RETURN
2444
2445      ;KEYBOARD INTERRUPT SERVICE ROUTINE
2446      :CALL
2447      :      ENTER VIA INTERRUPT
2448
2449 023004 104410 $TKSRV: RDCHR ;READ THE KEYBOARD
2450 023006 112637 023146      MOV      (SP)+,5$ ;GET THE CHARACTER
2451 023012 023727 023146 000003      CMP      5$,#3 ;'CONTROL C' ?
2452 023020 001017      BNE      1$ ;BR IF NOT
2453 023022 104401 001207      TYPE      , $CRLF ;CR-LF
2454 023026 104401 023452      TYPE      , $CNTLC ;'^C'
2455 023032 012737 177777 001336      MOV      #-1,CFLAG ;SET THE 'CONTROL C' FLAG
2456 023040 005077 156114      CLR      @ $TKS ;CLEAR THE TTY INTERRUPT
2457 023044 104401 001207      TYPE      , $CRLF ;CR-LF
2458 023050 104401 043106      TYPE      ,HALTX ;HALT THE PROGRAM
2459 023054 000177 156320      JMP      @RSTART ;JUMP TO RESTART
2460
2461 023060 023727 001154 000176 1$: CMP      SWR,#SWREG ;SOFTWARE SWITCH REGISTER IN USE ?
2462 023066 001024      BNE      3$ ;BR IF NOT
2463 023070 023727 023146 000007      CMP      5$,#7 ;'CONTROL G' ?
2464 023076 001020      BNE      3$ ;BR IF NOT
2465 023100 104401 001207      TYPE      , $CRLF ;CR-LF
2466 023104 104401 026661      TYPE      , $CNTLG ;'^G'
2467 023110 013746 177776      MOV      PS,-(SP) ;PUT THE STATUS WORD ON THE STACK
2468 023114 012746 023130      MOV      #2$,-(SP) ;RETURN ADDRESS
2469 023120 005077 156034      CLR      @ $TKS ;CLEAR THE TTY INTERRUPT ENABLE
2470 023124 000137 026312      JMP      $GTSWR ;GET THE SWITCH REGISTER ENTRY
2471 023130 012777 000100 156022 2$: MOV      #100,@ $TKS ;ENABLE TTY KEYBOARD INTERRUPT
2472 023136 000402      BR      4$ ;EXIT
2473 023140 104401 023146 3$: TYPE      ,5$ ;ECHO THE CHARACTER
2474 023144 000002 4$: RTI ;RETURN
2475
2476 023146 000000 5$: .WORD 0 ;ENTERED CHARACTER
2477
2478      ;THIS ROUTINE WILL INPUT A STRING FROM THE TTY
2479      :CALL:
2480      :      RDLIN ;INPUT A STRING FROM THE TTY
2481      :      RETURN HERE ;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
2482      : ;TERMINATOR WILL BE A BYTE OF ALL 0'S
2483
2484 023150 010346 $RDLIN: MOV      R3,-(SP) ;SAVE R3
2485 023152 005046      CLR      -(SP) ;CLEAR THE RUBOUT KEY
2486 023154 012703 023426 1$: MOV      # $TTYIN,R3 ;GET ADDRESS
2487 023160 022703 023452 2$: CMP      # $TTYIN+20.,R3 ;BUFFER FULL?
2488 023164 101467      BLOS      8$ ;BR IF YES
2489 023166 104410      RDCHR ;GO READ ONE CHARACTER FROM THE TTY
2490 023170 112613      MOV      (SP)+,(R3) ;GET CHARACTER

```

```

2491 023172 12273 000177      CMPB    #177,(R3)      ;IS IT A RUBOUT
2492 023176 001022      BNE     4$          ;BR IF NO
2493 023200 005716      TST     (SP)        ;IS THIS THE FIRST RUBOUT?
2494 023202 0C1007      BNE     3$          ;BR IF NO
2495 023204 112737 000134 023424      MOVB    #' \,11$      ;TYPE A BACK SLASH
2496 023212 104401 023424      TYPE     ,11$
2497 023216 012716 177777      MOV     #-1,(SP)      ;SET THE RUBOUT KEY
2498 023222 005303      3$: DEC     R3          ;BACKUP BY ONE
2499 023224 020327 023426      CMP     R3,#$TTYIN    ;STACK EMPTY?
2500 023230 103445      BLO     8$          ;BR IF YES
2501 023232 111337 023424      MOVB    (R3),11$      ;SETUP TO TYPEOUT THE DELETED CHAR.
2502 023236 104401 023424      TYPE     ,11$
2503 023242 000746      BR      2$          ;GO TYPE
2504 023244 005716      4$: TST     (SP)        ;GO READ ANOTHER CHAR.
2505 023246 001406      BEQ     5$          ;RUBOUT KEY SET?
2506 023250 112737 000134 023424      MOVB    #' \,11$      ;TYPE A BACK SLASH
2507 023256 104401 023424      TYPE     ,11$
2508 023262 005016      CLR     (SP)        ;CLEAR THE RUBOUT KEY
2509 023264 122713 000025      5$: CMPB    #25,(R3)      ;IS CHARACTER A CTRL U?
2510 023270 001003      BNE     6$          ;BR IF NO
2511 023272 104401 026654      TYPE     ,SCNTLU      ;TYPE A CONTROL 'U'
2512 023276 000726      BR      1$          ;GO START OVER
2513 023300 122713 000003      6$: CMPB    #3,(R3)      ;IS CHARACTER A CTRL C ?
2514 023304 001006      BNE     7$          ;BR IF NOT
2515 023306 012737 177777 001336      MOV     #-1,CFLAG    ;SET CNTRL C FLAG
2516 023314 104401 023452      TYPE     ,SCNTLC      ;ECHO IT
2517 023320 000427      BR      10$         ;EXIT
2518 023322 122713 000012      7$: CMPB    #12,(R3)      ;IS CHARACTER A 'LF'?
2519 023326 001011      BNE     9$          ;BRANCH IF NO
2520 023330 105013      CLRB    (R3)        ;CLEAR THE CHARACTER
2521 023332 104401 001207      TYPE     ,SCRLF      ;TYPE A 'CR' & 'LF'
2522 023336 104401 023426      TYPE     ,TTYIN      ;TYPE THE INPUT STRING
2523 023342 000706      BR      2$          ;GO PICKUP ANOTHER CHACTER
2524 023344 104401 001206      8$: TYPE     ,SQUES      ;TYPE A '?'
2525 023350 000701      BR      1$          ;CLEAR THE BUFFER AND LOOP
2526 023352 111337 023424      9$: MOVB    (R3),11$      ;ECHO THE CHARACTER
2527 023356 104401 023424      TYPE     ,11$
2528 023362 122723 000015      CMPB    #15,(R3)+      ;CHECK FOR RETURN
2529 023366 001274      BNE     2$          ;LOOP IF NOT RETURN
2530 023370 105063 177777      CLRB    -1(R3)      ;CLEAR RETURN (THE 15)
2531 023374 104401 001210      TYPE     ,SLF        ;TYPE A LINE FEED
2532 023400 005726      10$: TST     (SP)+      ;CLEAN RUBOUT KEY FROM THE STACK
2533 023402 012603      MOV     (SP)+,R3      ;RESTORE R3
2534 023404 011646      MOV     (SP)-,(SP)      ;ADJUST THE STACK AND PUT ADDRESS OF THE
2535 023406 016666 000004 000002      MOV     4(SP),2(SP)      ;FIRST ASCII CHARACTER ON IT
2536 023414 012766 023426 000004      MOV     #$TTYIN,4(SP)
2537 023422 000002      RTI
2538 023424 000      11$: .BYTE    0          ;RETURN
2539 023425 000      .BYTE    0          ;STORAGE FOR ASCII CHAR. TO TYPE
2540 023426      $TTYIN: .BLKB    20.      ;TERMINATOR
2541 023452 136 103 200 $CNTLC: .ASCIIZ / ^C / <CRLF> ;RESERVE 20. BYTES FOR TTY INPUT
2542      .EVEN      ;CONTROL 'C'

```

1

.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

```

023456				\$EOP:	NOP		
023456	000240				CLR	\$TSTNM	::ZERO THE TEST NUMBER
023460	005037	001116			CLR	\$TIMES	::ZERO THE NUMBER OF ITERATIONS
023464	005037	001176			INC	\$PASS	::INCREMENT THE PASS NUMBER
023470	005237	001220			BIC	#100000,\$PASS	::DON'T ALLOW A NEG. NUMBER
023474	042737	100000	001220		DEC	(PC)+	::LOOP?
023502	005327			\$EOPCT:	.WORD	1	
023504	000001				BGT	\$DOAGN	::YES
023506	003022				MOV	(PC)+,a(PC)+	::RESTORE COUNTER
023510	012737			\$ENDCT:	.WORD	1	
023512	000001				\$EOPCT		
023514	023504			\$GET42:	MOV	a#42,R0	::GET MONITOR ADDRESS
023516	013700	000042			BEQ	\$DOAGN	::BRANCH IF NO MONITOR
023522	001414				CLR	-(SP)	::INSURE THE 'T' BIT IS CLEAR
023524	005046				MOV	#\$CLR.T,-(SP)	::SETUP FOR AN RTI OR RTT
023526	012746	023534			BR	\$RTRN	::GO DO AN RTI OR RTT TO LOAD THE PSW
023532	000426						::WITH A CLEARED 'T' BIT
023534				\$CLR.T:			
023534	013700	000042			MOV	a#42,R0	::INSURE R0 CONTAINS THE MONITORS
023540	001405				BEQ	\$DOAGN	::RETURN ADDRESS
023542	000005				RESET		::CLEAR THE WORLD
023544	004710			\$ENDAD:	JSR	PC,(R0)	::GO TO MONITOR
023546	000240				NOP		::SAVE ROOM
023550	000240				NOP		::FOR
023552	000240				NOP		::ACT11
023554				\$DOAGN:			
023554	104400				TRAP		::PUSH OLD PSW AND PC ON STACK
023556	042716	000020			BIC	#20,(SP)	::CLEAR THE 'T' BIT
023562	032777	010000	155364		BIT	#BIT12,aSWR	::RUN WITH TRACE TRAP?
023570	001005				BNE	1\$	::BR IF NO
023572	005137	023616			COM	\$TBIT	::IS IT TIME FOR TRACE TRAP
023576	100402				BMI	1\$	::BR IF NO
023600	052716	000020			BIS	#20,(SP)	::SET TRACE TRAP
023604	012746	023612		1\$:	MOV	#\$LOOP,-(SP)	::JUMP TO START OF TEST
023610	000002			\$RTRN:	RTI		::RETURN--THIS IS CHANGED TO
							::AN 'RTI' IF 'RTT' IS A LEGAL
							::INSTRUCTION
023612				\$LOOP:			
023612	000137				JMP	a(PC)+	::RETURN
023614	023620			\$RTNAD:	.WORD	FINISH	
023616	000000			\$TBIT:	.WORD	0	::'T' BIT STATE INDICATOR
2							
3	023620	005737	001376	FINISH:	TST	FAULT	::COMPATIBLE ?
4	023624	001004			BNE	1\$	::BRANCH IF NOT
5	023626	104401	001207		TYPE	,\$CRLF	::CR-LF
6	023632	104401	042402		TYPE	,\$MSG15	::MESSAGE: ALL DRIVE COMPATIBLE
7	023636	104401	001207	1\$:	TYPE	,\$CRLF	::CR-LF
8	023642	104401	043004		TYPE	,\$MSG20	

9	023646	104401	043022	TYPE	,STARS	:TYPE STARS MESSAGE
10	023652	104401	001207	TYPE	,SCRLF	:CR-LF
11	023656	000177	155516	JMP	@RSTART	:JUMP TO RESTART

.SBTTL ERROR HANDLER ROUTINE

```

*****
*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT.
*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
*AND GO TO $ERRTYP ON ERROR
*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
*SW15=1      HALT ON ERROR
*SW13=1      INHIBIT ERROR TYPEOUTS
*SW10=1      BELL ON ERROR
*SW09=1      LOOP ON ERROR
*CALL
*
*      ERROR  N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER

023662 105037 024254 $ERROR: CLRB      IBSAVE      ;;CLEAR THE ITEM BYTE SAVE LOCATION
023666 104407          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
023670 105237 001117 7$:      INCB      $ERFLG      ;;SET THE ERROR FLAG
023674 001775          BEQ      7$      ;;DON'T LET THE FLAG GO TO ZERO
023676 013777 001116 155252 MOV      $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
023704 032777 002000 155242 BIT      #BIT10,@SWR      ;;BELL ON ERROR?
023712 001402          BEQ      1$      ;;NO - SKIP
023714 104401 001202          TYPE      $BELL      ;;RING BELL
023720 005237 001126 1$:      INC      $ERTTL      ;;COUNT THE NUMBER OF ERRORS
023724 011637 001132          MOV      (SP), $ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
023730 162737 000002 001132 SUB      #2, $ERRPC
023736 117737 155170 001130 MOV      @ $ERRPC, $ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
023744 032777 001000 155202 BIT      #BIT09,@SWR      ;;SEE IF LOOP ON ERROR IS SET
023752 001060          BNE      1004$      ;;BRANCH AROUND ROUTINE IF SO
023754 122737 000177 001130 CMP      #177, $ITEMB ;;SEE IF THIS IS THE POWER FAIL CALL
023762 001454          BEQ      1004$      ;;BRANCH AROUND ROUTINE IF IT IS
023764 105737 024254          TSTB      IBSAVE      ;;SEE IF THIS IS THE 2ND ERROR CALL IN THIS ROUTINE
023770 001047          BNE      1003$      ;;BRANCH IF SO
023772 022737 177777 024252 CMP      #-1, CPSAVE      ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
024000 001445          BEQ      1004$      ;;BRANCH IF SO
024002 013746 000004          MOV      ERRVEC, -(SP) ;;SAVE CONTENTS OF ERROR VECTOR
024006 012737 024024 000004 MOV      #1000$, ERRVEC      ;;SETUP 'TRAP' RETURN ADDRESS
024014 013737 177766 024252 MOV      177766, CPSAVE ;;MOVE CPU ERROR REGISTER TO CPSAVE FOR TEST
024022 000406          BR      1001$
024024 012737 177777 024252 1000$: MOV      #-1, CPSAVE      ;;SET CPU ERROR REGISTER TIMEOUT INDICATOR
024032 012716 024040          MOV      #1001$, (SP)      ;;SETUP RETURN ADDRESS
024036 000002          RTI
024040 012637 000004          1001$: MOV      (SP)+, ERRVEC      ;;RESTORE CONTENTS OF ERROR VECTOR

024044 022737 177777 024252 1002$: CMP      #-1, CPSAVE      ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
024052 001420          BEQ      1004$      ;;BRANCH IF SO
024054 032737 000001 024252 BIT      #BIT00, CPSAVE      ;;SEE IF POWER MONITOR BIT IS SET IN CPU ERR REG
024062 001414          BEQ      1004$      ;;BRANCH IF OK
024064 042737 000001 177766 BIC      #BIT00, 177766      ;;CLEAR THE BIT FOUND SET
024072 113737 001130 024254 MOV      $ITEMB, IBSAVE      ;;MAKE IBSAVE NON-ZERO FOR DUAL ERROR CALL
024100 112737 000177 001130 MOV      #177, $ITEMB      ;;SET $ITEMB TO SPECIAL POWER FAIL POINTER
024106 000402          BR      1004$      ;;BRANCH OVER IBSAVE CLEARING

024110 105037 024254          1003$: CLRB      IBSAVE      ;;CLEAR IBSAVE SO 2ND TIME THROUGH EXITS
024114          1004$:
024114 032777 020000 155032 BIT      #BIT13,@SWR      ;;SKIP TYPEOUT IF SET
024122 001004          BNE      20$      ;;SKIP TYPEOUTS
024124 004737 024256          JSR      PC, $ERRTYP      ;;GO TO USER ERROR ROUTINE

```

024130	104401	001207		TYPE	,\$CRLF	
024134			20\$:			
024134	122737	000001	001232	CMPB	#APTENV,\$ENV	:::RUNNING IN APT MODE
024142	001007			BNE	2\$	:::NO,SKIP APT ERROR REPORT
024144	113737	001130	024156	MOVB	\$ITEMB,21\$	:::SET ITEM NUMBER AS ERROR NUMBER
024152	004737	025150		JSR	PC,\$ATY4	:::REPORT FATAL ERROR TO APT
024156	000		21\$:	.BYTE	0	
024157	000			.BYTE	0	
024160	000777		22\$:	BR	22\$	:::APT ERROR LOOP
024162	105737	024254	2\$:	TSTB	IBSAVE	:::SEE IF IBSAVE IS LOADED
024166	001005			BNE	3\$	:::BRANCH IF NOT - NO HALT ON PWR MON BIT ERROR
024170	005777	154760		TST	@SWR	:::HALT ON ERROR
024174	100002			BPL	3\$	:::SKIP IF CONTINUE
024176	000000			HALT		:::HALT ON ERROR!
024200	104407			CKSWR		:::TEST FOR CHANGE IN SOFT-SWR
024202			3\$:			
024202	032777	001000	154744	BIT	#BIT09,@SWR	:::LOOP ON ERROR SWITCH SET?
024210	001402			BEQ	4\$	:::BR IF NO
024212	013716	001124		MOV	\$LPERR,(SP)	:::FUDGE RETURN FOR LOOPING
024216	005737	001200	4\$:	TST	\$ESCAPE	:::CHECK FOR AN ESCAPE ADDRESS
024222	001402			BEQ	5\$	:::BR IF NONE
024224	013716	001200		MOV	\$ESCAPE,(SP)	:::FUDGE RETURN ADDRESS FOR ESCAPE
024230			5\$:			
024230	022737	023544	000042	CMP	#SENDAD,@#42	:::ACT-11 AUTO-ACCEPT?
024236	001001			BNE	6\$	:::BRANCH IF NO
024240	000000			HALT		:::YES
024242			6\$:			
024242	105737	024254		TSTB	IBSAVE	:::SEE IF ITEM BYTE SAVE LOCATION HAS AN ERROR CALL
024246	001210			BNE	7\$	:::BRANCH BACK TO CALL ORIGINAL ERROR
024250	000002			RTI		:::RETURN
024252	000000		CPSAVE:	.WORD	0	:::LOCATION TO SAVE CPU ERROR REG CONTENTS
024254	000000		IBSAVE:	.WORD	0	:::LOCATION TO SAVE ITEM BYTE

.SBTTL ERROR MESSAGE TYPEOUT ROUTINE

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

024256				\$ERRTYP:		
024256	104401	001207		TYPE	, \$CRLF	:: 'CARRIAGE RETURN' & 'LINE FEED'
024262	010046			MOV	RO, -(SP)	:: SAVE RO
024264	005000			CLR	RO	:: PICKUP THE ITEM INDEX
024266	153700	001130		BISB	@\$ITEMB, RO	
024272	001004			BNE	1\$	:: IF ITEM NUMBER IS ZERO, JUST
						:: TYPE THE PC OF THE ERROR
024274	013746	001132		MOV	\$ERRPC, -(SP)	:: SAVE \$ERRPC FOR TYPEOUT
						:: ERROR ADDRESS
024300	104402			TYPOC		:: GO TYPE--OCTAL ASCII(ALL DIGITS)
024302	000437			BR	6\$	:: GET OUT
024304	122700	000177		1\$: CMPB	#177, RO	:: SEE IF THIS ERROR CALL IS SPECIAL POWER FAIL CALL
024310	001006			BNE	1000\$	:: BRANCH IF NOT
024312	013737	001216	024574	MOV	\$TESTN, PFTSTN	:: GET TEST NUMBER
024320	012700	024434		MOV	#PFECH, RO	:: MOVE POWER FAIL ERROR CALL TABLE TO RO
024324	000406			BR	1001\$	:: BRANCH TO CALL ERROR
024326	005300			1000\$: DEC	RO	:: ADJUST THE INDEX SO THAT IT WILL
024330	006300			ASL	RO	:: WORK FOR THE ERROR TABLE
024332	006300			ASL	RO	
024334	006300			ASL	RO	
024336	062700	005362		ADD	#\$ERRTB, RO	:: FORM TABLE POINTER
024342	012037	024352		1001\$: MOV	(RO)+, 2\$	:: PICKUP 'ERROR MESSAGE' POINTER
024346	001404			BEQ	3\$	:: SKIP TYPEOUT IF NO POINTER
024350	104401			TYPE		:: TYPE THE 'ERROR MESSAGE'
024352	000000			2\$: .WORD	0	:: 'ERROR MESSAGE' POINTER GOES HERE
024354	104401	001207		TYPE	, \$CRLF	:: 'CARRIAGE RETURN' & 'LINE FEED'
024360	012037	024370		3\$: MOV	(RO)+, 4\$	:: PICKUP 'DATA HEADER' POINTER
024364	001404			BEQ	5\$	:: SKIP TYPEOUT IF 0
024366	104401			TYPE		:: TYPE THE 'DATA HEADER'
024370	000000			4\$: .WORD	0	:: 'DATA HEADER' POINTER GOES HERE
024372	104401	001207		TYPE	, \$CRLF	:: 'CARRIAGE RETURN' & 'LINE FEED'
024376	011000			5\$: MOV	(RO), RO	:: PICKUP 'DATA TABLE' POINTER
024400	001004			BNE	7\$	:: GO TYPE THE DATA
024402	012600			6\$: MOV	(SP)+, RO	:: RESTORE RO
024404	104401	001207		TYPE	, \$CRLF	:: 'CARRIAGE RETURN' & 'LINE FEED'
024410	000207			RTS	PC	:: RETURN
024412				7\$: MOV	@(RO)+, -(SP)	:: SAVE @(RO)+ FOR TYPEOUT
024414	013046			TYPOC		:: GO TYPE--OCTAL ASCII(ALL DIGITS)
024416	104402			TST	(RO)	:: IS THERE ANOTHER NUMBER?
024420	005710			BEQ	6\$	:: BR IF NO
024422	001770			TYPE	, 8\$	:: TYPE TWO(2) SPACES
024426	104401	024430		BR	7\$	:: LOOP
024430	040	040	000	8\$: .ASCIIZ	/ /	:: TWO(2) SPACES
				.EVEN		
024434	024444	024526	024560	PFECH:	PFECH1, PFECH2, PFECH3, PFECH4	:: WORDS DEFINING TABLES BELOW
024444	120	117	127	PFECH1:	.ASCIIZ	?POWER MONITOR BIT IN CPU ERROR REGISTER FOUND SET?
024526	124	105	123	PFECH2:	.ASCIIZ	?TESTNO ERR PC CPUERREG?
				.EVEN		
024560	024574	001132	024252	PFECH3:	.WORD	PFTSTN, \$ERRPC, CPSAVE, 0

2 C2RMTBO RM05/3/2 DR CMPT TST MACRO V04.00 4-APR-81 18:12:15 PAGE 12-1
ERROR MESSAGE TYPEOUT ROUTINE

SEQ 0095

024570 000 000 000 PFECH4: .BYTE 0,0,0,0
024574 000000 PFTSTN: .WORD 0

:::CONTAINS TEST NUMBER FOR PF B1 ERROR

.SBTTL TYPE ROUTINE

```

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

```

```

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

```

024576	105737	001173	\$TYPE:	TSTB	\$TPFLG	:: IS THERE A TERMINAL?
024602	100002			BPL	1\$	:: BR IF YES
024604	000000			HALT		:: HALT HERE IF NO TERMINAL
024606	000430			BR	3\$	:: LEAVE
024610	010046		1\$:	MOV	R0,-(SP)	:: SAVE R0
024612	017600	000002		MOV	@2(SP),R0	:: GET ADDRESS OF ASCIZ STRING
024616	122737	000001	001232	CMPB	#APTENV,\$ENV	:: RUNNING IN APT MODE
024624	001011			BNE	62\$	:: NO,GO CHECK FOR APT CONSOLE
024626	132737	000100	001233	BITB	#APTSPOOL,\$ENV	:: SPOOL MESSAGE TO APT
024634	001405			BEQ	62\$	:: NO,GO CHECK FOR CONSOLE
024636	010037	024646		MOV	R0,61\$	:: SETUP MESSAGE ADDRESS FOR APT
024642	004737	025140		JSR	PC,\$ATY3	:: SPOOL MESSAGE TO APT
024646	000000		61\$:	.WORD	0	:: MESSAGE ADDRESS
024650	132737	000040	001233	62\$:	BITB	#APTCUP,\$ENV
024656	001003			BNE	60\$	:: APT CONSOLE SUPPRESSED
024660	112046		2\$:	MOVB	(R0)+,-(SP)	:: YES,SKIP TYPE OUT
024662	001005			BNE	4\$	:: PUSH CHARACTER TO BE TYPED ONTO STACK
024664	005726			TST	(SP)+	:: BR IF IT ISN'T THE TERMINATOR
024666	012600		60\$:	MOV	(SP)+,R0	:: IF TERMINATOR POP IT OFF THE STACK
024670	062716	000002	3\$:	ADD	#2,(SP)	:: RESTORE R0
024674	000002			RTI		:: ADJUST RETURN PC
024676	122716	000011	4\$:	CMPB	#HT,(SP)	:: RETURN
024702	001430			BEQ	8\$	:: BRANCH IF <HT>
024704	122716	000200		CMPB	#CRLF,(SP)	:: BRANCH IF NOT <CRLF>
024710	001006			BNE	5\$	
024712	005726			TST	(SP)+	:: POP <CR><LF> EQUIV
024714	104401			TYPE		:: TYPE A CR AND LF
024716	001207			\$CRLF		
024720	105037	025126		CLRB	\$CHARCNT	:: CLEAR CHARACTER COUNT
024724	000755			BR	2\$	:: GET NEXT CHARACTER
024726	004737	025010	5\$:	JSR	PC,\$TYPEC	:: GO TYPE THIS CHARACTER
024732	123726	001172	6\$:	CMPB	\$FILLC,(SP)+	:: GO TYPE THIS CHARACTER
024736	001350			BNE	2\$	:: IS IT TIME FOR FILLER CHARS.?
024740	013746	001170		MOV	\$NULL,-(SP)	:: IF NO GO GET NEXT CHAR.
						:: GET # OF FILLER CHARS. NEEDED
						:: AND THE NULL CHAR.
024744	105366	000001	7\$:	DECB	1(SP)	:: DOES A NULL NEED TO BE TYPED?
024750	002770			BLT	6\$	:: BR IF NO--GO POP THE NULL OFF OF STACK
024752	004737	025010		JSR	PC,\$TYPEC	:: GO TYPE A NULL
024756	105337	025126		DECB	\$CHARCNT	:: DO NOT COUNT AS A COUNT
024762	000770			BR	7\$	:: LOOP

;HORIZONTAL TAB PROCESSOR

```

024764 112716 000040      8$:  MOVB  #' , (SP)      ;; REPLACE TAB WITH SPACE
024770 004737 025010      9$:  JSR   PC, $TYPEC      ;; TYPE A SPACE
024774 132737 000007 025126  BITB  #7, $CHARCNT      ;; BRANCH IF NOT AT
025002 001372          BNE   9$      ;; TAB STOP
025004 005726          TST   (SP)+      ;; POP SPACE OFF STACK
025006 000724          BR    2$      ;; GET NEXT CHARACTER
025010          $TYPEC:
025010 105777 154144      TSTB  @ $TKS      ;; CHAR IN KYBD BUFFER?
025014 100022          BPL   10$      ;; BR IF NOT
025016 017746 154140      MOV   @ $TKB, -(SP)      ;; GET CHAR
025022 042716 177600      BIC   #177600, (SP)      ;; STRIP EXTRANEIOUS BITS
025026 122716 000023      CMPB  # $XOFF, (SP)      ;; WAS CHAR XOFF
025032 001012          BNE   102$      ;; BR IF NOT
025034          101$:
025034 105777 154120      TSTB  @ $TKS      ;; WAIT FOR CHAR
025040 100375          BPL   101$
025042 117716 154114      MOVB  @ $TKB, (SP)      ;; GET CHAR
025046 042716 177600      BIC   #177600, (SP)      ;; STRIP IT
025052 122716 000021      CMPB  # $XON, (SP)      ;; WAS IT XON?
025056 001366          BNE   101$      ;; BR IF NOT
025060          102$:
025060 005726          TST   (SP)+      ;; FIX STACK
025062          10$:
025062 105777 154076      TSTB  @ $TPS      ;; WAIT UNTIL PRINTER IS READY
025066 100375          BPL   10$
025070 116677 000002 154070  MOVB  2(SP), @ $TPB      ;; LOAD CHAR TO BE TYPED INTO DATA REG.
025076 122766 000015 000002  CMPB  #CR, 2(SP)      ;; IS CHARACTER A CARRIAGE RETURN?
025104 001003          BNE   1$      ;; BRANCH IF NO
025106 105037 025126      CLRB  $CHARCNT      ;; YES--CLEAR CHARACTER COUNT
025112 000406          BR    $TYPEX      ;; EXIT
025114 122766 000012 000002 1$:  CMPB  #LF, 2(SP)      ;; IS CHARACTER A LINE FEED?
025122 001402          BEQ   $TYPEX      ;; BRANCH IF YES
025124 105227          INCB  (PC)+      ;; COUNT THE CHARACTER
025126 000000      $CHARCNT: .WORD 0      ;; CHARACTER COUNT STORAGE
025130 000207      $TYPEX: RTS   PC

```

.SBTTL APT COMMUNICATIONS ROUTINE

```

*****
025132 112737 000001 025376 $ATY1: MOVB #1,$FFLG ;;TO REPORT FATAL ERROR
025140 112737 000001 025374 $ATY3: MOVB #1,$MFLG ;;TO TYPE A MESSAGE
025146 000403
025150 112737 000001 025376 $ATY4: MOVB #1,$FFLG ;;TO ONLY REPORT FATAL ERROR
025156 $ATYC:
025156 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
025160 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
025162 105737 025374 TSTB $MFLG ;;SHOULD TYPE A MESSAGE?
025166 001450 BEQ 5$ ;;IF NOT: BR
025170 122737 000001 001232 CMPB #APTENV,$ENV ;;OPERATING UNDER APT?
025176 001031 BNE 3$ ;;IF NOT: BR
025200 132737 000100 001233 BITB #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
025206 001425 BEQ 3$ ;;IF NOT: BR
025210 017600 000004 MOV @4(SP),R0 ;;GET MESSAGE ADDR.
025214 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
025222 005737 001212 1$: TST $MSGTYPE ;;SEE IF DONE W/ LAST XMISSION?
025226 001375 BNE 1$ ;;IF NOT: WAIT
025230 010037 001226 MOV R0,$MSGAD ;;PUT ADDR IN MAILBOX
025234 105720 2$: TSTB (R0)+ ;;FIND END OF MESSAGE
025236 001376 BNE 2$
025240 163700 001226 SUB $MSGAD,R0 ;;SUB START OF MESSAGE
025244 006200 ASR R0 ;;GET MESSAGE LGTH IN WORDS
025246 010037 001230 MOV R0,$MSGGLT ;;PUT LENGTH IN MAILBOX
025252 012737 000004 001212 MOV #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
025260 000413 BR 5$
025262 017637 000004 025306 3$: MOV @4(SP),4$ ;;PUT MSG ADDR IN JSR LINKAGE
025270 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDRESS
025276 013746 177776 MOV 177776,-(SP) ;;PUSH 177776 ON STACK
025302 004737 024576 JSR PC,$TYPE ;;CALL TYPE MACRO
025306 000000 4$: .WORD 0
025310 5$:
025310 105737 025376 10$: TSTB $FFLG ;;SHOULD REPORT FATAL ERROR?
025314 001416 BEQ 12$ ;;IF NOT: BR
025316 005737 001232 TST $ENV ;;RUNNING UNDER APT?
025322 001413 BEQ 12$ ;;IF NOT: BR
025324 005737 001212 11$: TST $MSGTYPE ;;FINISHED LAST MESSAGE?
025330 001375 BNE 11$ ;;IF NOT: WAIT
025332 017637 000004 001214 MOV @4(SP),$FATAL ;;GET ERROR #
025340 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
025346 005237 001212 INC $MSGTYPE ;;TELL APT TO TAKE ERROR
025352 105037 025376 12$: CLRB $FFLG ;;CLEAR FATAL FLAG
025356 105037 025375 CLRB $LFLG ;;CLEAR LOG FLAG
025362 105037 025374 CLRB $MFLG ;;CLEAR MESSAGE FLAG
025366 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
025370 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
025372 000207 RTS PC ;;RETURN
025374 000 $MFLG: .BYTE 0 ;;MESSG. FLAG
025375 000 $LFLG: .BYTE 0 ;;LOG FLAG
025376 000 $FFLG: .BYTE 0 ;;FATAL FLAG
               .EVEN
000200 APTSIZE = 200
000001 APTENV = 001
000100 APTSPOOL = 100
000040 APTCSUP = 040

```

.SBTTL POWER DOWN AND UP ROUTINES

```

*****
:POWER DOWN ROUTINE
025400 012737 025552 000024 $PWRDN: MOV    $SILLUP,@#PWRVEC ;;SET FOR FAST UP
025406 012737 000340 000026      MOV    #340,@#PWRVEC+2 ;;PRIO:7
025414 010046      MOV    R0,-(SP) ;;PUSH R0 ON STACK
025416 010146      MOV    R1,-(SP) ;;PUSH R1 ON STACK
025420 010246      MOV    R2,-(SP) ;;PUSH R2 ON STACK
025422 010346      MOV    R3,-(SP) ;;PUSH R3 ON STACK
025424 010446      MOV    R4,-(SP) ;;PUSH R4 ON STACK
025426 010546      MOV    R5,-(SP) ;;PUSH R5 ON STACK
025430 017746 153520      MOV    @SWR,-(SP) ;;PUSH @SWR ON STACK
025434 010637 025556      MOV    SP,$SAVR6 ;;SAVE SP
025440 012737 025452 000024      MOV    $PWRUP,@#PWRVEC ;;SET UP VECTOR
025446 000000      HALT
025450 000776      BR      .-2 ;;HANG UP
*****

:POWER UP ROUTINE
025452 012737 025552 000024 $PWRUP: MOV    $SILLUP,@#PWRVEC ;;SET FOR FAST DOWN
025460 013706 025556      MOV    $SAVR6,SP ;;GET SP
025464 005037 025556      CLR     $SAVR6 ;;WAIT LOOP FOR THE ITY
025470 005237 025556      1$: INC    $SAVR6 ;;WAIT FOR THE INC
025474 001375      BNE     1$ ;;OF WORD
025476 012677 153452      MOV    (SP)+,@SWR ;;POP STACK INTO @SWR
025502 012605      MOV    (SP)+,R5 ;;POP STACK INTO R5
025504 012604      MOV    (SP)+,R4 ;;POP STACK INTO R4
025506 012603      MOV    (SP)+,R3 ;;POP STACK INTO R3
025510 012602      MOV    (SP)+,R2 ;;POP STACK INTO R2
025512 012601      MOV    (SP)+,R1 ;;POP STACK INTO R1
025514 012600      MOV    (SP)+,R0 ;;POP STACK INTO R0
025516 012737 025400 000024      MOV    $PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
025524 012737 000340 000026      MOV    #340,@#PWRVEC+2 ;;PRIO:7
025532 104401      TYPE     $POWER ;;REPORT THE POWER FAILURE
025534 025560      $PWRMG: .WORD $POWER ;;POWER FAIL MESSAGE POINTER
025536 042766 000020 000002      BIC    #20,2(SP) ;;CLEAR 'T' BIT
025544 005037 023616      CLR     $TBIT ;;CLEAR THE 'T' BIT FLAG
025550 000002      RTI
025552 000000      $SILLUP: HALT ;;THE POWER UP SEQUENCE WAS STARTED
025554 000776      BR      .-2 ;;BEFORE THE POWER DOWN WAS COMPLETE
025556 000000      $SAVR6: 0 ;;PUT THE SP HERE
025560 015 012 120 $POWER: .ASCIZ <15><12>'POWER'
                                .EVEN

```

.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

```

025570	017646	000000		\$TYPOS:	MOV	@(SP),-(SP)	;;PICKUP THE MODE
025574	116637	000001	026013		MOVB	1(SP), \$OFILL	;;LOAD ZERO FILL SWITCH
025602	112637	026015			MOVB	(SP)+, \$OMODE+1	;;NUMBER OF DIGITS TO TYPE
025606	062716	000002			ADD	#2, (SP)	;;ADJUST RETURN ADDRESS
025612	000406				BR	\$TYPON	
025614	112737	000001	026013	\$TYPOC:	MOVB	#1, \$OFILL	;;SET THE ZERO FILL SWITCH
025622	112737	000006	026015		MOVB	#6, \$OMODE+1	;;SET FOR SIX(6) DIGITS
025630	112737	000005	026012	\$TYPON:	MOVB	#5, \$OCNT	;;SET THE ITERATION COUNT
025636	010346				MOV	R3, -(SP)	;;SAVE R3
025640	010446				MOV	R4, -(SP)	;;SAVE R4
025642	010546				MOV	R5, -(SP)	;;SAVE R5
025644	113704	026015			MOVB	\$OMODE+1, R4	;;GET THE NUMBER OF DIGITS TO TYPE
025650	005404				NEG	R4	
025652	062704	000006			ADD	#6, R4	;;SUBTRACT IT FOR MAX. ALLOWED
025656	110437	026014			MOVB	R4, \$OMODE	;;SAVE IT FOR USE
025662	113704	026013			MOVB	\$OFILL, R4	;;GET THE ZERO FILL SWITCH
025666	016605	000012			MOV	12(SP), R5	;;PICKUP THE INPUT NUMBER
025672	005003				CLR	R3	;;CLEAR THE OUTPUT WORD
025674	006105		1\$:		ROL	R5	;;ROTATE MSB INTO 'C'
025676	000404				BR	3\$	;;GO DO MSB
025700	006105		2\$:		ROL	R5	;;FORM THIS DIGIT
025702	006105				ROL	R5	
025704	006105				ROL	R5	
025706	010503				MOV	R5, R3	
025710	006103		3\$:		ROL	R3	;;GET LSB OF THIS DIGIT
025712	105337	026014			DECB	\$OMODE	;;TYPE THIS DIGIT?
025716	100016				BPL	7\$	;;BR IF NO
025720	042703	177770			BIC	#177770, R3	;;GET RID OF JUNK
025724	001002				BNE	4\$	;;TEST FOR 0
025726	005704				TST	R4	;;SUPPRESS THIS 0?
025730	001403				BEQ	5\$	;;BR IF YES
025732	005204		4\$:		INC	R4	;;DON'T SUPPRESS ANYMORE 0'S

025734	052703	000060		BIS	#'0,R3	::MAKE THIS DIGIT ASCII
025740	052703	000040	5\$:	BIS	#',R3	::MAKE ASCII IF NOT ALREADY
025744	110337	026010		MOVB	R3,8\$	::SAVE FOR TYPING
025750	104401	026010		TYPE	,8\$	::GO TYPE THIS DIGIT
025754	105337	026012	7\$:	DECB	\$OCNT	::COUNT BY 1
025760	003347			BGT	2\$	::BR IF MORE TO DO
025762	002402			BLT	6\$	::BR IF DONE
025764	005204			INC	R4	::INSURE LAST DIGIT ISN'T A BLANK
025766	000744			BR	2\$	::GO DO THE LAST DIGIT
025770	012605		6\$:	MOV	(SP)+,R5	::RESTORE R5
025772	012604			MOV	(SP)+,R4	::RESTORE R4
025774	012603			MOV,	(SP)+,R3	::RESTORE R3
025776	016666	000002 000004		MOV	2(SP),4(SP)	::SET THE STACK FOR RETURNING
026004	012616			MOV	(SP)+,(SP)	
026006	000002			RTI		::RETURN
026010	000		8\$:	.BYTE	0	::STORAGE FOR ASCII DIGIT
026011	000			.BYTE	0	::TERMINATOR FOR TYPE ROUTINE
026012	000		\$OCNT:	.BYTE	0	::OCTAL DIGIT COUNTER
026013	000		\$OFILL:	.BYTE	0	::ZERO FILL SWITCH
026014	000000		\$OMODE:	.WORD	0	::NUMBER OF DIGITS TO TYPE

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```

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

$TYPDS:
MOV      R0,-(SP)      ;;PUSH R0 ON STACK
MOV      R1,-(SP)      ;;PUSH R1 ON STACK
MOV      R2,-(SP)      ;;PUSH R2 ON STACK
MOV      R3,-(SP)      ;;PUSH R3 ON STACK
MOV      R5,-(SP)      ;;PUSH R5 ON STACK
MOV      #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
MOV      20(SP),R5      ;;GET THE INPUT NUMBER
BPL      1$            ;;BR IF INPUT IS POS.
NEG      R5            ;;MAKE THE BINARY NUMBER POS.
MOVB     #'-,1(SP)      ;;MAKE THE ASCII NUMBER NEG.
1$:      CLR      R0      ;;ZERO THE CONSTANTS INDEX
MOV      #SDBLK,R3      ;;SETUP THE OUTPUT POINTER
MOVB     #'',(R3)+      ;;SET THE FIRST CHARACTER TO A BLANK
2$:      CLR      R2      ;;CLEAR THE BCD NUMBER
MOV      $DTBL(R0),R1    ;;GET THE CONSTANT
3$:      SUB      R1,R5    ;;FORM THIS BCD DIGIT
BLT      4$            ;;BR IF DONE
INC      R2            ;;INCREASE THE BCD DIGIT BY 1
BR       3$
4$:      ADD      R1,R5    ;;ADD BACK THE CONSTANT
TST      R2            ;;CHECK IF BCD DIGIT=0
BNE      5$            ;;CALL THROUGH IF 0
TSTB     (SP)          ;;STILL DOING LEADING 0'S?
BMI      7$            ;;BR IF YES
5$:      ASLB     (SP)      ;;MSD?
BCC      6$            ;;BR IF NO
MOVB     1(SP),-1(R3)    ;;YES--SET THE SIGN
6$:      BIS      #'0,R2    ;;MAKE THE BCD DIGIT ASCII
7$:      BIS      #' ,R2    ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
MOVB     R2,(R3)+      ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
TST      (R0)+          ;;JUST INCREMENTING
CMP      R0,#10        ;;CHECK THE TABLE INDEX
BLT      2$            ;;GO DO THE NEXT DIGIT
BGT      8$            ;;GO TO EXIT
MOV      R5,R2          ;;GET THE LSD
BR       6$            ;;GO CHANGE TO ASCII
8$:      TSTB     (SP)+    ;;WAS THE LSD THE FIRST NON-ZERO?
BPL      9$            ;;BR IF NO
MOVB     -1(SP),-2(R3)  ;;YES--SET THE SIGN FOR TYPING
9$:      CLRB     (R3)      ;;SET THE TERMINATOR
MOV      (SP)+,R5      ;;POP STACK INTO R5
MOV      (SP)+,R3      ;;POP STACK INTO R3
MOV      (SP)+,R2      ;;POP STACK INTO R2
MOV      (SP)+,R1      ;;POP STACK INTO R1

```

```

026016
026016 010046
026020 010146
026022 010246
026024 010346
026026 010546
026030 012746 020200
026034 016605 000020
026040 100004
026042 005405
026044 112766 000055 000001
026052 005000
026054 012703 026232
026060 112723 000040
026064 005002
026066 016001 026222
026072 160105
026074 002402
026076 005202
026100 000774
026102 060105
026104 005702
026106 001002
026110 105716
026112 100407
026114 106316
026116 103003
026120 116663 000001 177777
026126 052702 000060
026132 052702 000040
026136 110223
026140 005720
026142 020027 000010
026146 002746
026150 003002
026152 010502
026154 000764
026156 105726
026160 100003
026162 116663 177777 177776
026170 105013
026172 012605
026174 012603
026176 012602
026200 012601

```

026202	012600		MOV	(SP)+,R0	::POP STACK INTO R0
026204	104401	026232	TYPE	, \$DBLK	::NOW TYPE THE NUMBER
026210	016666	000002 000004	MOV	2(SP),4(SP)	::ADJUST THE STACK
026216	012616		MOV	(SP)+,(SP)	
026220	000002		RTI		::RETURN TO USER
026222	023420		\$DTBL:	10000.	
026224	001750			1000.	
026226	000144			100.	
026230	000012			10.	
026232			\$DBLK:	.BLKW 4	

.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.

026242	022737	000176	001154	\$CKSWR: CMP	#SWREG,SWR	:::1. THE SOFT-SWR SELECTED?
026250	001074			BNE	15\$	:::BRANCH IF NO
026252	105777	152702		TSTB	@\$TKS	:::CHAR THERE?
026256	100071			BPL	15\$	:::IF NO, DON'T WAIT AROUND
026260	117746	152676		MOVB	@\$TKB,-(SP)	:::SAVE THE CHAR
026264	042716	177600		BIC	#^C177,(SP)	:::STRIP-OFF THE ASCII
026270	022726	000007		CMP	#7,(SP)+	:::IS IT A CONTROL G?
026274	001062			BNE	15\$	:::NO, RETURN TO USER
026276	123727	001150	000001	CMPB	\$AUTOB,#1	:::ARE WE RUNNING IN AUTO-MODE?
026304	001456			BEQ	15\$	:::BRANCH IF YES
026306	104401	026661		TYPE	,\$CNTLG	:::ECHO THE CONTROL-G (^G)
026312	104401	026666		\$GTSWR: TYPE	,\$MSWR	:::TYPE CURRENT CONTENTS
026316	013746	000176		MOV	SWREG,-(SP)	:::SAVE SWREG FOR TYPEOUT
026322	104402			TYPOC		:::GO TYPE--OCTAL ASCII(ALL DIGITS)
026324	104401	026677		TYPE	,\$MNEW	:::PROMPT FOR NEW SWR
026330	005046			19\$: CLR	-(SP)	:::CLEAR COUNTER
026332	005046			CLR	-(SP)	:::THE NEW SWR
026334	105777	152620		7\$: TSTB	@\$TKS	:::CHAR THERE?
026340	100375			BPL	7\$	:::IF NOT TRY AGAIN
026342	117746	152614		MOVB	@\$TKB,-(SP)	:::PICK UP CHAR
026346	042716	177600		BIC	#^C177,(SP)	:::MAKE IT 7-BIT ASCII
026352	021627	000025		9\$: CMP	(SP),#25	:::IS IT A CONTROL-U?
026356	001005			BNE	10\$	:::BRANCH IF NOT
026360	104401	026654		TYPE	,\$CNTLU	:::YES, ECHO CONTROL-U (^U)
026364	062706	000006		20\$: ADD	#6,SP	:::IGNORE PREVIOUS INPUT
026370	000757			BR	19\$	:::LET'S TRY IT AGAIN
026372	021627	000015		10\$: CMP	(SP),#15	:::IS IT A <CR>?
026376	001022			BNE	16\$	:::BRANCH IF NO
026400	005766	000004		TST	4(SP)	:::YES, IS IT THE FIRST CHAR?
026404	001403			BEQ	11\$	:::BRANCH IF YES
026406	016677	000002	152540	MOV	2(SP),@SWR	:::SAVE NEW SWR
026414	062706	000006		11\$: ADD	#6,SP	:::CLEAR UP STACK
026420	104401	001207		14\$: TYPE	,\$CRLF	:::ECHO <CR> AND <LF>
026424	123727	001151	000001	CMPB	\$INTAG,#1	:::RE-ENABLE TTY KBD INTERRUPTS?
026432	001003			BNE	15\$	:::BRANCH IF NOT
026434	012777	000100	152516	MOV	#100,@\$TKS	:::RE-ENABLE TTY KBD INTERRUPTS
026442	000002			15\$: RTI		:::RETURN
026444	004737	025010		16\$: JSR	PC,\$TYPEC	:::ECHO CHAR
026450	021627	000060		CMP	(SP),#60	:::CHAR < 0?
026454	002420			BLT	18\$	:::BRANCH IF YES

026456	021627	000067		CMP	(SP),#67	::CHAR > 7?
026462	003015			BGT	18\$	::BRANCH IF YES
026464	042726	000060		BIC	#60,(SP)+	::STRIP-OFF ASCII
026470	005766	000002		TST	2(SP)	::IS THIS THE FIRST CHAR
026474	001403			BEQ	17\$	::BRANCH IF YES
026476	006316			ASL	(SP)	::NO, SHIFT PRESENT
026500	006316			ASL	(SP)	::CHAR OVER TO MAKE
026502	006316			ASL	(SP)	::ROOM FOR NEW ONE.
026504	005266	000002	17\$:	INC	2(SP)	::KEEP COUNT OF CHAR
026510	056616	177776		BIS	-2(SP),(SP)	::SET IN NEW CHAR
026514	000707			BR	7\$	::GET THE NEXT ONE
026516	104401	001206	18\$:	TYPE	\$QUES	::TYPE ?<CR><LF>
026522	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

026524	011646			*RDCHR:	MOV	(SP),-(SP)	::PUSH DOWN THE PC
026526	016666	000004	000002		MOV	4(SP),2(SP)	::SAVE THE PS
026534	105777	152420		1\$:	TSTB	@\$TKS	::WAIT FOR
026540	100375				BPL	1\$	::A CHARACTER
026542	117766	152414	000004		MOVB	@\$TKB,4(SP)	::READ THE TTY
026550	042766	177600	000004		BIC	#^C<177>,4(SP)	::GET RID OF JUNK IF ANY
026556	026627	000004	000023		CMP	4(SP),#23	::IS IT A CONTROL-S?
026564	001013				BNE	3\$	::BRANCH IF NO
026566	105777	152366		2\$:	TSTB	@\$TKS	::WAIT FOR A CHARACTER
026572	100375				BPL	2\$	::LOOP UNTIL ITS THERE
026574	117746	152362			MOVB	@\$TKB,-(SP)	::GET CHARACTER
026600	042716	177600			BIC	#^C<177>,(SP)	::MAKE IT 7-BIT ASCII
026604	022627	000021			CMP	(SP)+,#21	::IS IT A CONTROL-Q?
026610	001366				BNE	2\$	::IF NOT DISCARD IT
026612	000750				BR	1\$	::YES, RESUME
026614	026627	000004	000021	3\$:	CMP	4(SP),#\$XON	::IS IT A RANDOM XON?
026622	001744				BEQ	1\$	::BRANCH IF YES
026624	026627	000004	000140		CMP	4(SP),#140	::IS IT UPPER CASE?
026632	002407				BLT	4\$	::BRANCH IF YES
026634	026627	000004	000175		CMP	4(SP),#175	::IS IT A SPECIAL CHAR?
026642	003003				BGT	4\$	::BRANCH IF YES
026644	042766	000040	000004		BIC	#40,4(SP)	::MAKE IT UPPER CASE
026652	000002			4\$:	RTI		::GO BACK TO USER
026654	136	125	015	\$CNTLU:	.ASCIIZ	/^U/<15><12>	::CONTROL 'U'
026661	136	107	015	\$CNTLG:	.ASCIIZ	/^G/<15><12>	::CONTROL 'G'
026666	015	012	123	\$MSWR:	.ASCIIZ	<15><12>/SWR = /	
026677	040	040	116	\$MNEW:	.ASCIIZ	/ NEW = /	

.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

```

026710			\$SAVREG:		
026710	010046		MOV	R0,-(SP)	::PUSH R0 ON STACK
026712	010146		MOV	R1,-(SP)	::PUSH R1 ON STACK
026714	010246		MOV	R2,-(SP)	::PUSH R2 ON STACK
026716	010346		MOV	R3,-(SP)	::PUSH R3 ON STACK
026720	010446		MOV	R4,-(SP)	::PUSH R4 ON STACK
026722	010546		MOV	R5,-(SP)	::PUSH R5 ON STACK
026724	016646	000022	MOV	22(SP),-(SP)	::SAVE PS OF MAIN FLOW
026730	016646	000022	MOV	22(SP),-(SP)	::SAVE PC OF MAIN FLOW
026734	016646	000022	MOV	22(SP),-(SP)	::SAVE PS OF CALL
026740	016646	000022	MOV	22(SP),-(SP)	::SAVE PC OF CALL
026744	000002		RTI		

*RESTORE R0-R5

```

*CALL:
*   RESREG

```

026746			\$RESREG:		
026746	012666	000022	MOV	(SP)+,22(SP)	::RESTORE PC OF CALL
026752	012666	000022	MOV	(SP)+,22(SP)	::RESTORE PS OF CALL
026756	012666	000022	MOV	(SP)+,22(SP)	::RESTORE PC OF MAIN FLOW
026762	012666	000022	MOV	(SP)+,22(SP)	::RESTORE PS OF MAIN FLOW
026766	012605		MOV	(SP)+,R5	::POP STACK INTO R5
026770	012604		MOV	(SP)+,R4	::POP STACK INTO R4
026772	012603		MOV	(SP)+,R3	::POP STACK INTO R3
026774	012602		MOV	(SP)+,R2	::POP STACK INTO R2
026776	012601		MOV	(SP)+,R1	::POP STACK INTO R1
027000	012600		MOV	(SP)+,R0	::POP STACK INTO R0
027002	000002		RTI		

.SBTTL DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

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

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

027004	104412		\$DB2D:	SAVREG	;; SAVE REGISTERS
027006	016602	000002		MOV 2(SP), R2	;; PICKUP THE DATA POINTER
027012	012700	027164		MOV #SDECVL, R0	;; GET ADDRESS OF 'SDECVL' STRING
027016	010666	000002		MOV R0, 2(SP)	;; PUT ADDRESS OF ASCII STRING ON STACK
027022	012201			MOV (R2)+, R1	;; PICKUP THE BINARY NUMBER
027024	012202			MOV (R2)+, R2	
027026	012737	000012	027102	MOV #10, 4\$	;; SET UP TO DO 10 CONVERSIONS
027034	012704	027114		MOV \$TNPWR, R4	;; ADDRESS OF TEN POWER
027040	012705	027116		MOV \$TNPWR+2, R5	
027044	005003		1\$:	CLR R3	;; CLEAR PARTIAL
027046	161401		2\$:	SUB (R4), R1	;; SUBTRACT TEN POWER
027050	005602			SEC R2	
027052	161502			SUB (R5), R2	
027054	002402			BLT 3\$	;; BR IF TEN POWER TOO LARGE
027056	005203			INC R3	;; ADD 1 TO PARTIAL
027060	000772			BR 2\$	;; LOOP
027062	062401		3\$:	ADD (R4)+, R1	;; RESTORE SUBTRACTED VALUE
027064	005502			ADC R2	
027066	062402			ADD (R4)+, R2	
027070	022525			CMP (R5)+, (R5)+	;; MOVE TO NEXT TEN POWER
027072	052703	000060		BIS #0, R3	;; CHANGE PARTIAL TO ASCII
027076	110320			MOVB R3, (R0)+	;; SAVE IT
027100	005327			DEC (PC)+	;; DONE?
027102	000000		4\$:	.WORD 0	
027104	001357			BNE 1\$	;; BR IF NO
027106	105020			CLRB (R0)+	;; TERMINATOR
027110	104413			RESREG	;; RESTORE REGISTERS
027112	000207			RTS PC	;; RETURN
027114	145000		\$TNPWR:	145000	;; 1.0E09
027116	035632			35632	
027120	160400			160400	;; 1.0E08
027122	002765			2765	
027124	113200			113200	;; 1.0E07
027126	000230			230	
027130	041100			041100	;; 1.0E06
027132	000017			17	
027134	103240			103240	;; 1.0E05
027136	000001			1	
027140	023420			23420	;; 1.0E04
027142	000000			0	
027144	001750			1750	;; 1.0E03
027146	000000			0	
027150	000144			144	;; 1.0E02
027152	000000			0	

027154	000012	12	::1.0E01
027156	000000	0	
027160	000001	1	::1.0E00
027162	000000	0	
027164		\$DECL: .BLKB 12.	::RESERVE STORAGE FOR ASCII STRING

.SBTTL DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE

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

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

027200	104412		\$DB20:	SAVREG		::SAVE ALL REGISTERS
027202	016601	000002		MOV	2(SP), R1	:: PICKUP THE POINTER TO LOW WORD
027206	012705	027317		MOV	#SOCTVL+13., R5	:: POINTER TO DATA TABLE
027212	012704	000014		MOV	#12., R4	:: DO ELEVEN CHARACTERS
027216	012703	177770		MOV	#^C7, R3	:: MASK
027222	012100			MOV	(R1)+, R0	:: LOWER WORD
027224	012101			MOV	(R1)+, R1	:: HIGH WORD
027226	005002			CLR	R2	:: TERMINATOR
027230	110245		1\$:	MOV	R2, -(R5)	:: PUT CHARACTER IN DATA TABLE
027232	010002			MOV	R0, R2	:: GET THIS DIGIT
027234	005304			DEC	R4	:: COUNT THIS CHARACTER
027236	003007			BGT	3\$	:: BR IF NOT THE LAST DIGIT
027240	001405			BEQ	2\$	:: BR IF IT IS THE LAST DIGIT
027242	005205			INC	R5	:: ALL DIGITS DONE-ADJUST POINTER FOR FIRST
027244	010566	000002		MOV	R5, 2(SP)	:: ASCIIZ CHAR. & PUT IT ON THE STACK
027250	104413			RESREG		:: RESTORE ALL REGISTERS
027252	000207			RTS	PC	:: RETURN TO USER
027254	006203		2\$:	ASR	R3	:: POSITION THE MASK FOR THE LAST DIGIT
027256	006001		3\$:	ROR	R1	:: POSITION THE BINARY NUMBER FOR
027260	006000			ROR	R0	:: THE NEXT OCTAL DIGIT
027262	006001			ROR	R1	
027264	006000			ROR	R0	
027266	006001			ROR	R1	
027270	006000			ROR	R0	
027272	040302			BIC	R3, R2	:: MASK OUT ALL JUNK
027274	062702	000060		ADD	#^0, R2	:: MAKE THIS CHAR. ASCII
027300	000753			BR	1\$	:: GO PUT IT IN THE DATA TABLE
027302			SOCTVL:	.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=10T

```

```

027320 104407 040000 151624 1$: CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
027320 032777 040000 151624 1$: BIT        #BIT14,@SWR      ;;LOOP ON PRESENT TEST?
027330 001402 027730 040000 1$: BEQ        9$              ;;NO IF SW14=0
027332 000137 027730 040000 1$: JMP        $OVER        ;;JUMP OVER SCOPE ROUTINE
027336 000416 027730 040000 9$:          ;;
;*****START OF CODE FOR THE XOR TESTER*****
$XTSTR: BR 6$
027336 000416 027730 040000 6$:          ;;IF RUNNING ON THE 'XOR' TESTER (CHANGE
027340 013746 000004 000004 6$:          ;;THIS INSTRUCTION TO A 'NOP' (NOP=240)
027344 012737 027364 000004 6$:          ;;SAVE THE CONTENTS OF THE ERROR VECTOR
027352 005737 177060 000004 6$:          ;;SET FOR TIMEOUT
027356 012637 000004 000004 6$:          ;;TIME OUT ON XOR?
027362 000544 000004 000004 6$:          ;;RESTORE THE ERROR VECTOR
027364 022626 000004 000004 6$:          ;;GO TO THE NEXT TEST
027366 012637 000004 000004 6$:          ;;CLEAR THE STACK AFTER A TIME OUT
027372 000504 000004 000004 6$:          ;;RESTORE THE ERROR VECTOR
027374 000504 000004 000004 6$:          ;;LOOP ON THE PRESENT TEST
027374 032777 000400 151552 6$:;*****END OF CODE FOR THE XOR TESTER*****
027402 001404 000400 151552 6$:          ;;LOOP ON SPEC. TEST?
027404 127737 151544 001116 6$:          ;;BR IF NO
027412 001546 000400 151552 6$:          ;;ON THE RIGHT TEST? SWR<7:0>
027414 105737 001117 000400 6$:          ;;BR IF YES
027420 001502 000400 151552 6$:          ;;HAS AN ERROR OCCURRED?
027422 022737 177777 024252 6$:          ;;BR IF NO
027430 001455 000400 151552 6$:          ;;SEE IF TIMEOUT WAS PREVIOUSLY RECORDED
027432 013746 000004 000004 6$:          ;;KICK AROUND ROUTINE IF SO
027436 012737 027454 000004 6$:          ;;SAVE CONTENTS OF ERROR VECTOR
027444 013737 177766 024252 6$:          ;;SETUP 'TRAP' RETURN ADDRESS
027452 000406 000400 151552 6$:          ;;MOVE CPU ERROR REGISTER TO CPSAVE FOR TEST
027454 012737 177777 024252 2000$: MOV      #-1,CPSAVE      ;;SET CPU ERROR REGISTER TIMEOUT INDICATOR
027462 012716 027470 024252 2000$: MOV      #2001$,(SP)      ;;SETUP RETURN ADDRESS
027466 000002 000004 000004 2000$: RTI
027470 012637 000004 000004 2001$: MOV      (SP)+,ERRVEC      ;;RESTORE CONTENTS OF ERROR VECTOR
027474 022737 177777 024252 2002$: CMP      #-1,CPSAVE      ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
027502 001430 000001 024252 2002$: BEQ      2003$          ;;BRANCH IF SO
027504 032737 000001 024252 2002$: BIT      #BIT00,CPSAVE      ;;SEE IF THE POWER MONITOR BIT IS ON
027512 001424 000001 177766 2002$: BEQ      2003$          ;;BRANCH TO CONTINUE ROUTINE IF CLEAR
027514 042737 000001 177766 2002$: BIC      #BIT00,177766      ;;CLEAR THE BIT FOUND TO BE SET
027522 013746 001154 000000 2002$: MOV      SWR,-(SP)        ;;SAVE SWR ADDRESS
027526 017646 000000 001154 2002$: MOV      @($P),-(SP)      ;;SAVE SWR VALUE
027532 012737 000176 001154 2002$: MOV      #176,SWR        ;;GET SOFTWARE SWR ADDRESS

```

```

027540 011677 151410      MOV      (SP),@SWR      ;;GET CURRENT SWR VALLE
027544 042777 001000 151402 BIC      #BIT09,@SWR      ;;DON'T ALLOW LOOP ON ERROR ON THIS ERROR
027552 104177      EMT      177      ;;CALL SPECIAL POWER FAIL BIT ERROR CALL
027554 012676 000000      MOV      (SP)+,@(SP)      ;;RESTORE SWR TO ORIGINAL VALUE
027560 012637 001154      MOV      (SP)+,SWR      ;;RESTORE SWR ADDRESS
027564      2003$:
027564 123737 001131 001117 CMPB      $ERMAX,$ERFLG      ;;MAX. ERRORS FOR THIS TEST OCCURRED?
027572 101015      BHI      3$      ;;BR IF NO
027574 032777 001000 151352 BIT      #BIT09,@SWR      ;;LOOP ON ERROR?
027602 001404      BEQ      4$      ;;BR IF NO
027604 013737 001124 001122 7$:      MOV      $LPERR,$LPADR      ;;SET LOOP ADDRESS TO LAST SCOPE
027612 000446      BR      $OVER
027614 105037 001117      4$:      CLRB      $ERFLG      ;;ZERO THE ERROR FLAG
027620 005037 001176      CLR      $TIMES      ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
027624 000415      BR      1$      ;;ESCAPE TO THE NEXT TEST
027626 032777 004000 151320 3$:      BIT      #BIT11,@SWR      ;;INHIBIT ITERATIONS?
027634 001011      BNE      1$      ;;BR IF YES
027636 005737 001220      TST      $PASS      ;;IF FIRST PASS OF PROGRAM
027642 001406      BEQ      1$      ;;INHIBIT ITERATIONS
027644 005237 001120      INC      $ICNT      ;;INCREMENT ITERATION COUNT
027650 023737 001176 001120      CMP      $TIMES,$ICNT      ;;CHECK THE NUMBER OF ITERATIONS MADE
027656 002024      BGE      $OVER      ;;BR IF MORE ITERATION REQUIRED
027660 012737 000001 001120 1$:      MOV      #1,$ICNT      ;;REINITIALIZE THE ITERATION COUNTER
027666 013737 027744 001176      MOV      $MXCNT,$TIMES      ;;SET NUMBER OF ITERATIONS TO DO
027674 105237 001116      $SVLAD: INCB      $TSTNM      ;;COUNT TEST NUMBERS
027700 113737 001116 001216      MOVB      $TSTNM,$TESTN      ;;SET TEST NUMBER IN APT MAILBOX
027706 011637 001122      MOV      (SP),$LPADR      ;;SAVE SCOPE LOOP ADDRESS
027712 011637 001124      MOV      (SP),$LPERR      ;;SAVE ERROR LOOP ADDRESS
027716 005037 001200      CLR      $ESCAPE      ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
027722 112737 000001 001131      MOVB      #1,$ERMAX      ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
027730 013777 001116 151220 $OVER:      MOV      $TSTNM,@DISPLAY      ;;DISPLAY TEST NUMBER
027736 013716 001122      MOV      $LPADR,(SP)      ;;FUDGE RETURN ADDRESS
027742 000002      RTI      ;;FIXES PS
027744 003720      $MXCNT: 2000.      ;;MAX. NUMBER OF ITERATIONS

```

.SBTTL ROUTINE TO SIZE MEMC.Y

```

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

027746 010046          $SIZE: MOV      R0,-(SP)          ;;SAVE R0 ON THE STACK
027750 010146          MOV      R1,-(SP)          ;;SAVE R1 ON THE STACK
027752 013746 000114    MOV      @#114,-(SP)        ;;SAVE MEMORY ERROR VECTOR PS & PC
027756 013746 000116    MOV      @#116,-(SP)
027762 012737 000116 000114    MOV      #116,@#114    ;;IGNORE PARITY ERRORS WHILE SIZING
027770 012737 000002 000116    MOV      #RT1,@#116
027776 013746 000004          MOV      @#ERRVEC,-(SP) ;;SAVE PRESENT ERROR VECTOR PS & PC
030002 013746 000006          MOV      @#ERRVEC+2,-(SP)
030006 010600          MOV      SP,R0              ;;SAVE THE STACK POINTER
;;SET THE ERRVEC PS TO THE PRESENT PS
030010 104400          TRAP
030012 012637 000006          MOV      (SP)+,@#ERRVEC+2 ;;PUSH OLD PSW AND PC ON STACK
030016 012737 030036 000004    MOV      #2,@#ERRVEC    ;;SAVE THE PSW IN @#ERRVEC+2
030024 012701 020000          MOV      #20000,R1      ;;SET FOR TIMEOUT
030030 005711          1$:   TST      (R1)            ;;FIRST ADDRESS
030032 005721          TST      (R1)+                ;;TEST THIS ADDRESS
030034 000775          BR       1$                    ;;STEP TO NEXT ADDRESS
030036 162701 000002          2$:   SUB      #2,R1      ;;TRY ANOTHER
030042 010006          MOV      R0,SP                ;;DROP BACK
030044 012637 000006          MOV      (SP)+,@#ERRVEC+2 ;;RESTORE THE STACK
030050 012637 000004          MOV      (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
030054 012637 000116          MOV      (SP)+,@#116    ;;RESTORE MEMORY ERROR VECTOR
030060 012637 000114          MOV      (SP)+,@#114
030064 010137 030076          MOV      R1,$LSTAD      ;;LAST ADDRESS
030070 012601          MOV      (SP)+,R1            ;;RESTORE R1
030072 012600          MOV      (SP)+,R0            ;;RESTORE R0
030074 000207          RTS      PC
030076 000000          $LSTAD: .WORD 0              ;;CONTAINS THE LAST ADDRESS
  
```

.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.

030100	010046		\$TRAP:	MOV	R0,-(SP)	::SAVE R0
030102	016600	000002		MOV	2(SP),R0	::GET TRAP ADDRESS
030106	005740			TST	-(R0)	::BACKUP BY 2
030110	111000			MOVB	(R0),R0	::GET RIGHT BYTE OF TRAP
030112	006300			ASL	R0	::POSITION FOR INDEXING
030114	016000	030134		MOV	\$TRPAD(R0),R0	::INDEX TO TABLE
030120	000200			RTS	R0	::GO TO ROUTINE

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

030122	011646		\$TRAP2:	MOV	(SP),-(SP)	::MOVE THE PC DOWN
030124	016666	000004		MOV	4(SP),2(SP)	::MOVE THE PSW DOWN
030132	000002	000002		RTI		::RESTORE THE PSW

.SBTTL TRAP TABLE

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

			ROUTINE		

030134	030122	\$TRPAD:	.WORD \$TRAP2		
030136	024576		\$TYPE ::CALL=TYPE	TRAP+1(104401)	TTY TYPEOUT ROUTINE
030140	025614		\$TYPOC ::CALL=TYPOC	TRAP+2(104402)	TYPE OCTAL NUMBER (WITH LEADING ZEROS)
030142	025570		\$TYPOS ::CALL=TYPOS	TRAP+3(104403)	TYPE OCTAL NUMBER (NO LEADING ZEROS)
030144	025630		\$TYPON ::CALL=TYPON	TRAP+4(104404)	TYPE OCTAL NUMBER (AS PER LAST CALL)
030146	026016		\$TYPDS ::CALL=TYPDS	TRAP+5(104405)	TYPE DECIMAL NUMBER (WITH SIGN)
030150	026312		\$GTSWR ::CALL=GTSWR	TRAP+6(104406)	GET SOFT-SWR SETTING
030152	026242		\$CKSWR ::CALL=CKSWR	TRAP+7(104407)	TEST FOR CHANGE IN SOFT-SWR
030154	026524		\$RDCHR ::CALL=RDCHR	TRAP+10(104410)	TTY TYPEIN CHARACTER ROUTINE
030156	023150		\$RDLIN ::CALL=RDLIN	TRAP+11(104411)	TTY TYPEIN STRING ROUTINE
030160	026710		\$SAVREG ::CALL=SAVREG	TRAP+12(104412)	SAVE R0-R5 ROUTINE
030162	026746		\$RESREG ::CALL=RESREG	TRAP+13(104413)	RESTORE R0-R5 ROUTINE
2 030164	022366		\$DSPLY ::CALL=DISPLY	TRAP+14(104414)	ROUTINE TO TYPE ERROR MESSAGES
3	000032		\$TERM=.-\$TRPAD		

```

1      .SBTTL SINGLE/DUAL PORT RH/RM DRIVER (REV 6.5) 1981
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      ;1980 *****
8
9      ;COPYRIGHT (C) 1977,1981
10     ;DIGITAL EQUIPMENT CORP.
11     ;MAYNARD, MA 01754
12     ;AUTHOR(S): JIM LACEY/CHUCK HESS
13     ;REVISED BY: MIKE LEAVITT      11-APR-80, 27-MAR-81
14
15     ;:*****
16
17     ;STORAGE FOR RMDs, RMER1, RMER2, AND RMMR2 ON AN ERROR '2'
18     ;RMERRS = RMDs
19     ;RMERRS+2 = RMER1
20     ;RMERRS+4 = RMER2
21     ;RMERRS+6 = RMMR2
22
23     030166 000000 000000 000000 RMERRS: .WORD 0,0,0,0
24
25     ;TABLE OF DRIVE ACTIVE INDICATORS (DRVACT=8 BYTES)
26     ;DRVACT=0 IF DRIVE IS IDLE
27     ;DRVACT>0 IF DRIVE IS ACTIVE WITH A COMMAND
28     ;DRVACT<0 IF DRIVE IS ACTIVE WITH AN ERROR RECOVERY OPERATION
29
30     030176 000      DRVACT: .BYTE 0      ;DRIVE 0
31     030177 000      .BYTE 0      ;DRIVE 1
32     030200 000      .BYTE 0      ;DRIVE 2
33     030201 000      .BYTE 0      ;DRIVE 3
34     030202 000      .BYTE 0      ;DRIVE 4
35     030203 000      .BYTE 0      ;DRIVE 5
36     030204 000      .BYTE 0      ;DRIVE 6
37     030205 000      .BYTE 0      ;DRIVE 7
38
39     ;TABLE OF DRIVE STATUS INDICATORS (DRVSTA=8 BYTES)
40     ;DRVSTA=0 IF DRIVE IS OFFLINE OR NONEXISTENT
41     ;DRVSTA>0 IF DRIVE IS ONLINE
42     ;DRVSTA<0 IF DRIVE IS UNSAFE
43
44     030206 000      DRVSTA: .BYTE 0      ;DRIVE 0
45     030207 000      .BYTE 0      ;DRIVE 1
46     030210 000      .BYTE 0      ;DRIVE 2
47     030211 000      .BYTE 0      ;DRIVE 3
48     030212 000      .BYTE 0      ;DRIVE 4
49     030213 000      .BYTE 0      ;DRIVE 5
50     030214 000      .BYTE 0      ;DRIVE 6
51     030215 000      .BYTE 0      ;DRIVE 7
52
53     ;TABLE OF DRIVE TYPES (DRV Typ=8 BYTES)
54     ;DRV Typ=0 IF DRIVE IS NONEXISTENT (DRVSTA=0, ALSO)
55     ;DRV Typ=7 IF DRIVE IS RM05 *****
56     ;DRV Typ=5 IF DRIVE IS RM02 *****
57     ;DRV Typ=4 IF DRIVE IS RM03

```

```
50                                     ;DRV TYP=-1 IF NOT RM05/3/2
51
52 030216      000      DRV TYP: .BYTE 0      ;DRIVE 0
55 030217      000      .BYTE 0      ;DRIVE 1
    030220      000      .BYTE 0      ;DRIVE 2
    030221      000      .BYTE 0      ;DRIVE 3
    030222      000      .BYTE 0      ;DRIVE 4
    030223      000      .BYTE 0      ;DRIVE 5
    030224      000      .BYTE 0      ;DRIVE 6
    030225      000      .BYTE 0      ;DRIVE 7
56
57                                     ;TABLE OF DUAL PORT INITIALIZATION INDICATORS
58                                     ;DPINT=0 IF INITIALIZATION IS NOT ACTIVE ON THE DRIVE
59                                     ;DPINT<0 IF INITIALIZATION IS IN PROGRESS
60
61 030226      000      DPINT: .BYTE ^      ;DRIVE 0
64 030227      000      .BYTE 0      ;DRIVE 1
    030230      000      .BYTE 0      ;DRIVE 2
    030231      000      .BYTE 0      ;DRIVE 3
    030232      000      .BYTE 0      ;DRIVE 4
    030233      000      .BYTE 0      ;DRIVE 5
    030234      000      .BYTE 0      ;DRIVE 6
    030235      000      .BYTE 0      ;DRIVE 7
65
66                                     ;TABLE OF PENDING DUAL PORT REQUESTS
67                                     ;DPRQS=0 IF THAT A DUAL PORT REQUEST IS NOT PENDING FOR THAT DRIVE
68                                     ;DPRQS<0 IF THAT A DUAL PORT REQUEST IS PENDING FOR THAT DRIVE
69
70 030236      000      DPRQS: .BYTE 0      ;DRIVE 0
73 030237      000      .BYTE 0      ;DRIVE 1
    030240      000      .BYTE 0      ;DRIVE 2
    030241      000      .BYTE 0      ;DRIVE 3
    030242      000      .BYTE 0      ;DRIVE 4
    030243      000      .BYTE 0      ;DRIVE 5
    030244      000      .BYTE 0      ;DRIVE 6
    030245      000      .BYTE 0      ;DRIVE 7
74
75                                     ;TRANSFER WAIT FLAG (TRNSWT=1 WORD)
76                                     ;THIS IS A ONE WORD QUEUE. IT WILL CONTAIN THE ADDRESS OF
77                                     ;'DPB' OF THE I/O OPERATION.
78
79 030246      000000      TRNSWT: .WORD 0
80
81                                     ;SEARCH WAIT KEYS (SRCHWT=1 WORD)
82                                     ;THIS IS A ONE WORD QUEUE THAT WILL CONTAIN A KEY FOR EACH OF
83                                     ;THE DRIVES THAT ARE PERFORMING A SEARCH COMMAND FOR THE I/O
84                                     ;REQUEST THAT IS AT THE TOP OF THEIR REQUEST QUEUE.
85                                     ;EACH DRIVE IS ASSIGNED ONE BIT, STARTING AT BIT00 FOR DRIVE 0.
86
87 030250      000000      SRCHWT: .WORD 0
88
89                                     ;RM DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)
90                                     ;ACTDRV=0 IF DRIVER IS INACTIVE
91                                     ;ACTDRV>0 IF DRIVER IS ACTIVE
92
93 030252      000      ACTDRV: .BYTE 0
94
```

```

95      ;SOFTWARE TIMER ROUTINE ACTIVE FLAG (ACTSTR=1 BYTE)
96      ;ACTSTR=0 IF SOFTWARE TIMER ROUTINE IS INACTIVE
97      ;ACTSTR>0 IF SOFTWARE TIMER ROUTINE IS ACTIVE
98
99 030253      000      ACTSTR: .BYTE      0
100
101      ;UNLOAD FLAG (ULDFLG=8 BYTES)
102      ;ULDFLG=0 IF NO UNLOAD COMMAND
103      ;ULDFLG>0 IF UNLOAD COMMAND IN PROGRESS
104      ;ULDFLG<0 IF UNLOAD COMMAND IN WAIT QUEUE
105
106 030254      000      ULDFLG: .BYTE      0      ;DRIVE 0
107 030255      000      .BYTE      0      ;DRIVE 1
108 030256      000      .BYTE      0      ;DRIVE 2
109 030257      000      .BYTE      0      ;DRIVE 3
110 030260      000      .BYTE      0      ;DRIVE 4
111 030261      000      .BYTE      0      ;DRIVE 5
112 030262      000      .BYTE      0      ;DRIVE 6
113 030263      000      .BYTE      0      ;DRIVE 7
114
115      ;SAVE REGISTERS FLAG (SAVEFG =1 WORD)
116      ;SAVEFG <0 IF SAVE THE RH/RM REGISTERS WHEN THE
117      ;OPERATION IS COMPLETED AS PER (DPB+14).
118      ;SAVEFG=0 IF SAVE THE RH/RM REGISTERS, AS PER
119      ;(DPB+14), AFTER AN ERROR.
120
121 030264      000000      SAVEFG: .WORD      0
122
123      ;SEEK FLAG (SEEKFG=1 WORD)
124      ;SEEKFG=0 IF WHEN THE DISK ADDRESS ISN'T IN THE WINDOW
125      ;FOR A DATA TRANSFER START A SEARCH COMMAND
126      ;SEEKFG<0 IF DATA TRANSFER WILL DO IMPLIED SEEKS,
127      ;DISREGARD THE WINDOW
128
129 030266      177777      SEEKFG: .WORD     -1
130
131      ;TIMEOUT TABLE (TIMER=8 WORDS)
132      ;THIS TABLE CONTAINS THE TIME ALLOWED FOR AN OPERATION
133
134 030270      177777      TIMER:  .WORD     -1      ;DRIVE 0
135 030272      177777      .WORD     -1      ;DRIVE 1
136 030274      177777      .WORD     -1      ;DRIVE 2
137 030276      177777      .WORD     -1      ;DRIVE 3
138 030300      177777      .WORD     -1      ;DRIVE 4
139 030302      177777      .WORD     -1      ;DRIVE 5
140 030304      177777      .WORD     -1      ;DRIVE 6
141 030306      177777      .WORD     -1      ;DRIVE 7
142
143      ;DATA TRANSFER UNDERWAY INDICATOR (DTUW=1 WORD)
144      ;DTUW<0 IF NO DATA TRANSFER UNDERWAY
145      ;DTUW=+N (WHERE N=0 TO 7) IMPLIES DATA TRANSFER UNDERWAY ON DRIVE N
146
147 030310      177777      DTUW:  .WORD     -1
148
149      ;ATTENTION BITS TABLE (ATABIT=8 BYTES)
150      ;THIS TABLE CONTAINS THE CORRESPONDING BIT TO EACH DRIVES
151      ;ATTENTION BIT

```

```

144
145 030312      001      ATABIT: .BYTE 1      ;DRIVE 0
146 030313      002      .BYTE 2      ;DRIVE 1
147 030314      004      .BYTE 4      ;DRIVE 2
148 030315      010      .BYTE 10     ;DRIVE 3
149 030316      020      .BYTE 20     ;DRIVE 4
150 030317      040      .BYTE 40     ;DRIVE 5
151 030320      100      .BYTE 100    ;DRIVE 6
152 030321      200      .BYTE 200    ;DRIVE 7
153
154      ;NUMBER OF 'MASSBUS CONTROL PARITY ERRORS' (MCPE) ALLOWED BEFORE
155      ;CALLING IT FATAL (MCPEMX=1 WORD)
156
157 030322      000003    MCPEMX: .WORD 3
158
159      ;STORAGE FOR RMADR (THE FIRST ADDRESS (776700) OF THE RH/RM),
160      ;RMVEC (THE VECTOR ADDRESS (254)), AND RMVEC+2 (THE BR LEVEL (5)).
161
162 030324      176700    RMADR: .WORD 176700
163 030326      000254    0C0240    RMVEC: .WORD 254,5*32.
164
165      ;MAXIMUM SEARCH FOR I/O WINDOW IS 5 SECTORS (MXWNDW-1 WORD)
166 030332      000005    MXWNDW: .WORD 5
167
168      ;DEFINITIONS OF THE RH/RM ADDRESS INDEXES
169
170
171
172
173      000000    RMCS1 = 0      ;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 0)
174      000002    RMWC = 2      ;WORD COUNT REGISTER (NOT A DRIVE REG)
175      000004    RMBA = 4      ;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)
176      000006    RMDA = 6      ;DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 5)
177      000010    RMCS2 = 10     ;CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
178      000012    RMDS = 12     ;DRIVE STATUS REGISTER (DRIVE REG 1)
179      000014    RMER1 = 14     ;ERROR REGISTER #1 (DRIVE REG. 2)
180      000016    RMAS = 16     ;ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 4)
181      000020    RMLA = 20     ;LOOK AHEAD REGISTER (DRIVE REG. 7)
182      000022    RMDB = 22     ;DATA BUFFER REGISTER (NOT A DRIVE REG.)
183      000024    RMR1 = 24     ;MAINTAINABILITY REGISTER (DRIVE REG. 3)
184      000026    RMDT = 26     ;DRIVE TYPE REGISTER (DRIVE REG. 6)
185      000030    RMSN = 30     ;SERIAL NUMBER REGISTER (DRIVE REG. 10)
186      000032    RMOF = 32     ;OFFSET REGISTER (DRIVE REG. 11)
187      000034    RMDC = 34     ;DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
188      000036    RMHR = 36     ;DUMMY ADDRESS REGISTER (DRIVE REG. 13)
189      000040    RMR2 = 40     ;MAINTENANCE REGISTER #2
190      000042    RMER2 = 42     ;ERROR REGISTER #2 (DRIVE REG. 15)
191      000044    RMEC1 = 44     ;ECC POSITION REGISTER (DRIVE REG. 16)
192      000046    RMEC2 = 46     ;ECC PATTERN REGISTER (DRIVE R-G. 17)
193
194
195
196
197
198      .SBTTL RH/RM DRIVER INITIALIZATION CODE
199
200      ;THIS ROUTINE WILL DETERMINE WHICH RM DRIVES ARE
201      ;AVAILABLE FOR TESTING AND SET THE DRVSTA INDICATOR
202      ;TO THE PROPER STATE FOR EACH DRIVE.
203      ;NOTE: THIS ROUTINE CALLS DRVINT
204      ;
205      ;CALL
206      ;
207      ;      JSR      PL,RMINIT

```

```

208
209
210
211
212 030334 104412
213 030336 013746 177776
214 030342 012737 000240 177776
215 030350 004737 036012
216 030354 012701 030166
217 030360 012702 030266
218 030364 005021
219 030366 020102
220 030370 103775
221 030372 012702 030310
222 030376 012721 177777
223 030402 020102
224 030404 101774
225 030406 005037 030206
226 030412 005037 030210
227 030416 005037 030212
228 030422 005037 030214
229 030426 013703 030326
230 030432 012723 032762
231 030436 013713 030330
232 030442 013704 030324
233 030446 012764 005000 000010
234 030454 005001
235 030456 004037 030546
236 030462 000401
237 030464 000402
238 030466 105061 030206
239 030472 005201
240 030474 042701 177770
241 030500 001366
242 030502 012701 000007
243 030506 005037 177776
244 030512 105761 030226
245 030516 001405
246 030520 004737 035446
247 030524 105761 030226
248 030530 001375
249 030532 005301
250 030534 100366
251 030536 012637 177776
252 030542 104413
253 030544 000207
254
255
256
257
258
259
260
261
262
263
264

RETURN
NOTE: THE 'P' OR 'L' CLOCK MUST BE STARTED
RMINIT: SAVREG
MOV PS, -(SP)
MOV #<5-32>, PS
JSR PC, CLRQUE
MOV #RMERRS, R1
MOV #SLEKFG, R2
1$: CLR (R1)+
CMP R1, R2
BLO 1$
MOV #DTUM, R2
2$: MOV #1, (R1)+
CMP R1, R2
BLOS 2$
CLR DRVSTA
CLR DRVSTA+2
CLR DRVSTA+4
CLR DRVSTA+6
MOV RMVEC, R3
MOV #ISR, (R3)+
MOV RMVEC+2, (R3)
MOV RMADR, R4
MOV #CLR, RMCS2(R4)
3$: CLR R1
JSR R0, DRVINT
BR 4$
BR 5$
4$: CLRB DRVSTA(R1)
5$: INC R1
BIC #^C7, R1
BNE 3$
MOV #7, R1
CLR PS
6$: TSTB DPINT(R1)
BEQ 8$
JSR PC, SET.IE
7$: TSTB DPINT(R1)
BNE 7$
8$: DEC R1
BPL 6$
MOV (SP)+, PS
RESREG
RTS PC

:SAVE R0 - R5
:SAVE THE PRESENT PROCESSOR STATUS
:CHANGE THE PRIORITY TO 5
:CLEAR ALL REQUEST QUEUES
:FIRST ADDRESS TO BE CLEARED
:LAST ADDRESS TO BE CLEARED
:CLEAR
:ARE WE DONE?
:BR IF NO
:LAST ADDRESS
:INITIALIZE
:DONE?
:LOOP IF NO
:SET ALL DRIVES TO OFFLINE

:SETUP THE RH/RM VECTOR
:FIRST ADDRESS OF RH/RM
:MASSBUS INIT
:START WITH DRIVE 0
:INIT THE DRIVE
:'DVA' NOT SET OR PARITY ERROR
:NORMAL RETURN
:SET DRIVE STATUS TO OFFLINE
:GO TO NEXT DRIVE
:MASK OUT UNUSED BITS
:BR IF MORE DRIVES TO GO
:START WITH DRIVE 7
:CLEAR THE PROCESSOR STATUS
:WAITING FOR DRIVE TO SWITCH PORTS ?
:BR NOT WAITING
:SET INTERRUPT
:DRIVE SWITCHED PORTS ?
:BR IF NOT
:GO TO THE NEXT DRIVE
:CHECK NEXT DRIVE
:RESTORE THE PROCESSOR STATUS
:RESTORE R0 - R5
:BYE-BYE

:DRIVE INITIALIZATION ROUTINE
:THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
:AN RM05/3/2. IF IT IS, A 'READ-IN PRESET' IS ISSUED AND FMT16
:IS SET TO A '1'. THEN MOL, DPR, DRY, AND VV ARE CHECKED TO
:INSURE THEY ARE ALL ON A '1'. AND DEPENDING ON THEIR STATE,
:DRVSTA IS SET TO THE PROPER CONDITION.
:CALL
MOV #DRVNUM, R1
MOV RMADR, R4
JSR R0, DRVINT
:DRIVE NUMBER TO R1
:UNIBUS ADDRESS OF RH/RM (RMCS1)
:CALLED BY A JSR

```

```

265      :      RETURN1      :ERROR OCCURRED (PARITY)
266      :      RETURN2      :NORMAL RETURN
267      :
268
269 030546 010546      DRVINT: MOV      R5,-(SP)      :SAVE R5
270 030550 105061 030206      CLRB      DRVSTA(R1)      :START DRIVE STATUS AS OFFLINE
271 030554 105061 030216      CLRB      DRVSTYP(R1)      :CLEAR 1 DRIVE TYPE INDICATOR
272 030560 105061 030254      CLRB      ULDFLG(R1)      :CLEAR THE UNLOAD FLAG
273 030564 010164 000010      MOV      R1,RMCS2(R4)      :SELECT A DRIVE
274 030570 112764 000111 000000      MOVB      #11,RMCS1(R4)      :DO A DRIVE CLEAR COMMAND (& SEIZE DRIVE)
275 030576 032764 010000 000010      BIT      #BIT12,RMCS2(R4)      :NONEXISTENT DRIVE?
276 030604 001403      BEQ      1$      :NO
277 030606 004737 035446      JSR      PC,SET.IE      :GO SET 'IE' WITHOUT A 'TRE'
278 030612 000520      BR      4$      :LEAVE THIS ROUTINE
279
280 030614 105061 030206      1$: CLRB      DRVSTA(R1)      :SET DRIVE STATUS TO OFFLINE
281 030620 032764 004000 000000      BIT      #BIT11,RMCS1(R4)      :SEE IF DRIVE AVAILABLE
282 030626 001512      BEQ      4$      :BR IF DRIVE NOT AVAILABLE
283 030630 004037 034756      JSR      R0,RD.RM      :READ THE DRIVE TYPE REG.
284 030634 000026      RMDT      SS
285 030636 031056      SS      :ERROR RETURN ADDRESS
286 030640 012605      MOV      (SP)+,R5      :PUT DRIVE TYPE IN R5
287 030642 112761 000004 030216      MOVB      #4,DRVSTYP(R1)      :SET RM03 INDICATOR
288 030650 022705 020024      CMP      #20024,R5      :SINGLE PORT RM03 ?
289 030654 001431      BEQ      2$      :BR IF YES
290 030656 022705 024024      CMP      #24024,R5      :DUAL PORT RM03 ?
291 030662 001426      BEQ      2$      :BR IF YES
292 030664 112761 000005 030216      MOVB      #5,DRVSTYP(R1)      :SET RM02 INDICATOR
293 030672 022705 020025      CMP      #20025,R5      :SINGLE PORT RM02 ?
294 030676 001420      BEQ      2$      :BR IF SO
295 030700 022705 024025      CMP      #24025,R5      :DUAL PORT RM02 ?
296 030704 001415      BEQ      2$      :BR IF SO
297 030706 112761 000007 030216      MOVB      #7,DRVSTYP(R1)      :SET RM05 INDICATOR
298 030714 022705 020027      CMP      #20027,R5      :SINGLE PORT RM05 ?
299 030720 001407      BEQ      2$      :BR IF YES
300 030722 022705 024027      CMP      #24027,R5      :DUAL PORT RM05 ?
301 030726 001404      BEQ      2$      :BR IF YES
302 030730 112761 177777 030216      MOVB      #-1,DRVSTYP(R1)      :SET INDICATOR TO 'OTHER'
303 030736 000446      BR      4$      :EXIT
304
305 030740 012746 000121      2$: MOV      #121,-(SP)      :DO A 'READ-IN PRESET'
306 030744 004037 035136      JSR      R0,WRT.RM
307 030750 000000      RMCS1      SS
308 030752 031056      SS
309 030754 012746 010000      MOV      #BIT12,-(SP)      :SET FMT16=1
310 030760 034037 035136      JSR      R0,WRT.RM
311 030764 000032      RMOF      SS
312 030766 031056      SS
313 030770 004037 034756      JSR      R0,RD.RM      :READ RMDS
314 030774 000012      RMDS      SS
315 030776 031056      SS
316 031000 012605      MOV      (SP)+,R5      :AND SAVE IT IN R5
317 031002 100015      BPL      3$      :BR IF ATA=0
318 031004 116164 030312 000016      MOVB      ATABIT(R1),RMAS(R4)      :CLEAR ATTENTION BIT
319 031012 004037 034756      JSR      R0,RD.RM      :FIND OUT WHY ATA=1
320 031016 000014      RMER1      SS
321 031020 031056

```

```

322 031022 006126          ROL      (SP)+      ;IS IT UNSAFE?
323 031024 100004          BPL      3$        ;BR IF NOT
324 031026 112761 177777 030206  MOVB   #-1,DRVSTA(R1) ;SET UNSAFE INDICATOR
325 031034 000407          BR       4$        ;EXIT
326
327 031036 005105          3$:   COM      R5      ;CHECK MOL, DPR, DRY, AND VV
328 031040 042705 167077    BIC      #*(BIT12:BIT08:BIT07:BIT06),R5
329 031044 001003          BNE      4$        ;BR IF MOL, DPR, DRY, OR VV IS CLEAR
330 031046 112761 000001 030206  MOVB   #1,DRVSTA(R1) ;SET DRIVE STATUS TO ONLINE
331 031054 005720          4$:   TST      (R0)+   ;STEP OVER THE ERROR RETURN
332 031056 012605          5$:   MOV      (SP)+,R5 ;RESTORE R5
333 031060 000200          RTS      R0        ;EXIT
334
335          ;REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
336
337          ;CALL
338
339          ;JSR      R0,RM05      ;CALL THE RM05 DRIVER
340          ;PNTADR   ;ADDRESS OF POINTER OF DRIVES PARAMETER BLOCK
341          ;RETURN1  ;RETURN HERE IF QUEUE IS FULL
342          ;RETURN2  ;RETURN HERE IF REQUEST IS IN QUEUE OR THERE
343          ;IS AN ERROR CONDITION
344
345 031062 013746 177776  RM05:   MOV      PS,-(SP) ;SAVE THE CALLING STATUS
346 031066 013737 030330 177776  MOV      RMVEC+2,PS ;DON'T ALLOW ANY RM INTERRUPTS
347 031074 112737 000001 030252  MOVB   #1,ACTDRV ;SET 'ACTIVE DRIVER' FLAG
348 031102 104412          SAVREG ;SAVE R0 - R5
349 031104 011002          MOV      (R0),R2      ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
350 031106 005062 000016          CLR      16(R2) ;CLEAR THE STATUS/ERROR INDICATOR
351 031112 111201          MOVB   (R2),R1      ;PICKUP THE DRIVE NUMBER
352 031114 013704 030324          MOV      RMADR,R4 ;UNIBUS ADDRESS OF RMCS1
353 031120 105761 030206          TSTB   DRVSTA(R1) ;CHECK DRIVES STATUS
354 031124 003014          BGT      1$        ;BR IF ONLINE
355 031126 105761 030254          TSTB   ULDFLG(R1) ;UNLOAD COMMAND IN QUEUE?
356 031132 001036          BNE      3$        ;BR IF YES
357 031134 105761 030226          TSTB   DPINT(R1) ;TRYING TO INIT THE DRIVE
358 031140 001042          BNE      5$        ;BR IF YES
359 031142 004037 030546          JSR      R0,DRVINT ;GO INIT. THE DRIVE
360 031146 000434          BR       4$        ;ERROR RETURN
361 031150 105761 030206          TSTB   DRVSTA(R1) ;IS DRIVE STATUS ONLINE?
362 031154 003445          BLE      6$        ;BR IF NOT
363 031156 105761 030236          1$:   TSTB   DPRQS(R1) ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
364 031162 001031          BNE      5$        ;BR IF YES
365 031164 010164 000010          MOV      R1,RMCS2(R4) ;SELECT THE DRIVE
366 031170 004037 036110          JSR      R0,DRVQUE ;PUT THIS REQUEST IN QUEUE
367 031174 000460          BR       9$        ;QUEUE IS FULL
368 031176 122762 000103 000002  CMPB   #103,2(R2) ;IS THIS REQ. FOR AN UNLOAD?
369 031204 001003          BNE      2$        ;BR IF NO
370 031206 112761 177777 030254  MOVB   #-1,ULDFLG(R1) ;SET THE 'UNLOAD IN QUEUE' FLAG
371 031214 105761 030176          2$:   TSTB   DRVACT(R1) ;IS THIS DRIVE ACTIVE?
372 031220 001043          BNE      8$        ;BR IF YES
373 031222 004737 031354          JSR      PC,OPT ;CALL THE OPTIMIZER
374 031226 000440          BR       8$
375 031230 012762 120000 000016  3$:   MOV      #BIT15:BIT13,16(R2) ;SET THE 'UNLOAD IN QUEUE' ERROR FLAG
376 031236 000434          BR       8$        ;EXIT
377 031240 004737 032432          4$:   JSR      PC,C17 ;GO HANDLE THE PARITY ERROR
378 031244 000431          BR       8$

```

```

379 031246 004037 036110 5$: JSR R0,DRVQUE ;PUT REQUEST IN QUEUE
380 031252 000431 BR 9$ ;QUEUE IS FULL
381 031254 032714 000100 BIT #B1'06,(R4) ;IE BIT SET ?
382 031260 001023 BNE 8$ ;YES
383 031262 004737 035446 JSR PC,SET.IE ;SET THE INTERRUPT
384 031266 000420 BR 8$ ;RETURN
385 031270 105761 030206 6$: TSTB DRVSTA(R1) ;SEE IF DRIVE OFFLINE OR UNSAFE
386 031274 002412 BLT 7$ ;BR IF UNSAFE
387 031276 012762 140000 000016 MOV #B1'15,BIT14,16(R2) ;SET OFFLINE ERROR INDICATOR
388 031304 105761 030216 TSTB DRVTP(R1) ;SEE IF OFFLINE OR NONEXISTENT
389 031310 001007 BNE 8$ ;BR IF OFFLINE
390 031312 012762 100002 000016 MOV #B1'15,BIT01,16(R2) ;REPORT DRIVE NONEXISTENT
391 031320 000403 BR 8$ ;GO TO EXIT
392 031322 012762 110000 000016 7$: MOV #B1'15,BIT12,16(R2) ;DRIVE IS UNSAFE
393 031330 104413 8$: RESREG ;RESTORE R0 - R5
394 031332 005720 TST (R0)+ ;SETUP FOR NORMAL RETURN
395 031334 000401 BR 10$ ;FINISH UP, THEN EXIT
396 031336 104413 9$: RESREG ;RESTORE R0 - R5
397 031340 005720 10$: TST (R0)+ ;CORRECT THE RETURN ADDRESS
398 031342 105037 030252 CLRB ACTDRV ;CLEAR 'ACTIVE DRIVER' FLAG
399 031346 012637 177776 MOV (SP)+,PS ;RETURN 'PS' TO USER LEVEL
400 031352 000200 RTS R0 ;RETURN TO CALLER
401
402 ;OPTIMIZER-CALLED FOR A PARTICULAR DRIVE
403
404 :CALL
405 :
406 : MOV #DRVNUM,R1 ;DRIVE NUMBER TO R1
407 : JSR PC,OPT ;SETUP A COMMAND
408 031354 104412 OPT: SAVREG ;SAVE R0 - R5
409 031356 013746 177776 MOV PS,-(SP) ;SAVE PROC. STATUS
410 031362 146137 030312 030250 BICB ATABIT(R1),SRCHWT ;CLEAR LA SEACH FLAG
411 031370 105061 030236 CLRB DPRQS(R1) ;RESET THE PORT REQ FLAG ****
412 031374 004737 036164 JSR PC,GETREQ ;GET 'DPB' POINTER OF REQUEST
413 031400 005702 R2 ;IS THERE A REQUEST IN QUEUE?
414 031402 001466 BEQ 7$ ;NO--BR TO EXIT
415 031404 010164 000010 MOV R1,RMCS2(R4) ;LOAD THE DRIVE ADDRESS *****
416 031410 012764 000111 000000 MOV #111,RMCS1(R4) ;CLEAR THE DRIVE
417 031416 032764 004000 000000 BIT #B1'11,RMCS1(R4) ;DVA SET ?
418 031424 001442 BEQ 5$ ;TO PORT REQUEST, IF NOT
419 031426 105761 030206 1$: TSTB DRVSTA(R1) ;IS DRIVE ONLINE?
420 031432 003014 BGT 2$ ;YES
421 031434 004737 036206 JSR PC,POPOUE ;NO--REMOVE REQUEST FROM QUEUE
422 031440 012762 140000 000016 MOV #B1'15,BIT14,16(R2) ;SET OFFLINE STATUS/ERROR INDICATOR
423 031446 105761 030206 TSTB DRVSTA(R1) ;IS DRIVE UNSAFE ?
424 031452 100047 BPL 8$ ;BR TO EXIT IF NOT
425 031454 012762 110000 000016 MOV #B1'15,BIT12,16(R2) ;SET UNSAFE STATUS/ERROR INDICATOR
426 031462 000443 BR 8$ ;BR TO EXIT
427 031464 2$: CMPB #150,2(R2) ;IS THE REQUEST FOR I/O?
436 031464 122762 000150 000002 BLT 3$ ;YES
437 031472 002403 JSR PC,C14 ;CALL THE COMMAND INITIATOR
438 031474 004737 032016 BR 8$ ;BR TO EXIT
439 031500 000434 3$: TST DTUW ;DATA TRANSFER UNDERWAY?
440 031502 005737 030310 BGE 4$ ;YES--GO START A SEARCH
441 031506 002006 TST SEEKFG ;DO IMPLIED SEEKS?
442 031510 005737 030266 BPL 4$ ;NO, DO A SEARCH
443 031514 100003

```

```

444 031516 004737 031602      JSR      PC,C11      ;START A DATA TRANSFER
445 031522 000423              BR        8$
446 031524 004737 031710      4$: JSR      PC,C13      ;START A SEARCH
447 031530 000420              BR        8$      ;GO TO THE EXIT
448 031532 112761 177777 030236 5$: MOV     #1,DPRQS(R1) ;SET PORT REQUEST INDICATOR
449 031540 010103              MOV     R1,R3      ;SET UP TO ADDRESS WORDS
450 031542 006303              ASL     R3          ;CONVERT TO WORD INDEX
451 031544 012763 035230 030270 MOV     #15000.,TIMER(R3) ;START 15. SECOND TIMER
452 031552 000402              BR        7$      ;EXIT
453 031554 004737 032432      6$: JSR      PC,C17      ;PROCESS THE PARITY ERROR
454 031560 032714 000100      7$: BIT     #B106,(R4) ;SEE IF 'IE' ALREADY SET
455 031564 001002              BNE     8$          ;BR IF SET
456 031566 004737 035446      JSR      PC,SET.IE    ;SET 'IE' WITHOUT A 'TRE'
457 031572 012637 177776      8$: MOV     (SP)+,PS    ;RESTORE PROC. STATUS
458 031576 104413              RESREG ;RESTORE RC - R5
459 031600 000207              RTS     PC
460
461      ;COMMAND INITIATOR
462
463      ;CALL
464      MOV     #DRYNUM,R1      ;DRIVE NUMBER
465      MOV     #DPB,R2        ;ADDRESS OF DPB
466      JSR     PC,C1?         ;C1?= C11,C13, OR C14
467      ;WHERE:
468      ;C11=DATA TRANSFER
469      ;C13=SEARCH REQUESTED BY DATA XFER
470      ;C14=NOT DATA TRANSFER
471
472 031602 004737 036206      11$: JSR     PC,PCPQUE ;REMOVE REQUEST FROM 'DRIVES WAIT' QUEUE
473 031606 010237 030246      MOV     R2,TRNSWT ;PUT REQ. IN TRANSFER WAIT QUEUE
474 031612 010203              MOV     R2,R3      ;DPB ADDRESS TO R3
475 031614 013704 030324      MOV     RMADR,R4    ;RMCS1 ADDRESS
476 031620 010164 000010      MOV     R1,RMCS2(R4) ;SELECT DRIVE
477 031624 062703 000004      ADD     #4,R3      ;DESIRED WORD COUNT
478 031630 062704 000002      ADD     #2,R4      ;RMWC ADDRESS
479 031634 012324              MOV     (R3)+,(R4)+ ;LOAD WORD COUNT
480 031636 012324              MOV     (R3)+,(R4)+ ;LOAD BUFFER ADDRESS
481 031640 012346              MOV     (R3)+,-(SP) ;LOAD SECTOR AND TRACK
482 031642 004037 035136      JSR     R0,WRT.RM ;CALL THE LOAD(WRITE) ROUTINE
483 031646 000006              RMDA     C17        ;INDEX OF REGISTER TO LOAD
484 031650 032432              C17      ;ERROR RETURN ADDRESS
485 031652 012346              MOV     (R3)+,-(SP) ;LOAD CYLINDER ADDRESS
486 031654 004037 035136      JSR     R0,WRT.RM
487 031660 000034              RMDC     C17
488 031662 032432              C17
489 031664 016246 000002      MOV     2(R2),-(SP) ;LOAD 'COMMAND+GO', 'A17&A16', AND 'PSEL'
490 031670 004037 035136      JSR     R0,WRT.RM
491 031674 000000              RMCS1
492 031676 032432              C17
493 031700 010137 030310      MOV     R1,DTUW      ;SET 'DATA TRANSFER UNDERWAY'
494 031704 000137 032374      JMP     C15
495
496 031710 013704 030324      13$: MOV     RMADR,R4    ;RMCS1 ADDRESS
497 031714 010164 000010      MOV     R1,RMCS2(R4) ;SELECT DRIVE
498 031720 016246 000012      MOV     12(R2),-(SP) ;DESIRED CYLINDER ADDRESS
499 031724 004037 035136      JSR     R0,WRT.RM
500 031730 000034              RMDC

```

```

501 031732 032432 C17
502 031734 116203 000010 MOV 10(R2),R3 ;PICKUP SECTOR ADDRESS
503 031740 163703 030332 SUB MXWINDW,R3 ;BACKUP BY MAX. SEARCH FOR I/O WINDOW
504 031744 002002 BGE 18
505 031746 062703 000040 ADD #32,R3
506 031752 010346 'S: MOV R3,-(SP) ;COMBINE THE ADJUSTED SECTOR WITH
507 031754 116266 000011 0000C1 MOV 11(R2),1(SP) ;THE DESIRED TRACK
508 031762 004037 035136 JSR R0,WRT.RM ;LOAD DESIRED TRACK & SECTOR
509 031766 000006 RMDA
510 031770 032432 C17
511 031772 012746 000131 MOV #131,-(SP) ;START A SEARCH
512 031776 004037 035136 JSR R0,WRT.RM
513 032002 000000 RMCS1
514 032004 032432 C17
515 032006 156137 030312 030250 BISB ATAB11(R1),SRCHWT ;SET 'SEARCH WAIT' KEY
516 032014 000567 BR C15
517
518 032016 013704 030324 C14: MOV RMADR,R4 ;RMCS1 ADDRESS
519 032022 010164 000010 MOV R1,RMCS2(R4) ;SELECT DRIVE
520 032026 116203 000002 MOV 2(R2),R3 ;PICKUP THE REQUESTED COMMAND
521 032032 122703 000131 CMPB #131,R3 ;IS IT A SEARCH COMMAND?
522 032036 001007 BNE 18 ;BR IF NO
523 032040 016246 000010 MOV 10(R2),-(SP) ;LOAD DESIRED TRACK & SECTOR
524 032044 004037 035136 JSR R0,WRT.RM
525 032050 000006 RMDA
526 032052 032432 C17
527 032054 000403 BR 28
528 032056 122703 000105 'S: CMPB #105,R3 ;GO LOAD CYLINDER
529 032062 001007 BNE 38 ;IS IT A SEEK COMMAND
530 032064 016246 000012 28: MOV 12(R2),-(SP) ;BR IF NO
531 032070 004037 035136 JSR R0,WRT.RM ;LOAD DESIRED CYLINDER
532 032074 000034 RMDA
533 032076 032432 C17
534 032100 000546 BR C16
535 032102 122703 000115 38: CMPB #115,R3 ;IS IT AN 'OFFSET' COMMAND?
536 032106 001013 BNE 48 ;BR IF NO
537 032110 004037 034756 .SR R0,RD.RM ;MERGE THE OFFSET VALUE INTO RMOF
538 032114 000032 RMOF ;BUT DON'T CHANGE THE UPPER
539 032116 032432 C17
540 032120 116216 000001 MOV 1(R2),1(SP) ;BYTE WHEN LOADING THE
541 032124 004037 035136 JSR R0,WRT.RM ;REGISTER (RMOF)
542 032130 000032 RMOF
543 032132 032432 C17
544 032134 000530 BR C16
545 032136 122703 000107 48: CMPB #107,R3 ;GO START THE COMMAND
546 032142 001525 BEQ C16 ;IS IT A 'RECALIBRATE' COMMAND?
547 032144 122703 000117 CMPB #117,R3 ;BR IF YES
548 032150 001522 BEQ C16 ;IS IT A RETURN TO CENTER?
549 032152 122703 000103 CMPB #103,R3 ;BR IF YES
550 032156 001016 BNE 58 ;IS IT AN 'UNLOAD' COMMAND?
551 032160 112761 000001 030176 MOV #1,DRVACT(R1) ;BR IF NO
552 032166 105061 030206 CLRB DRVSTA(R1) ;SET THE DRIVE ACTIVE INDICATOR
553 032172 112761 000001 030254 MOV #1,ULDFLG(R1) ;PUT DRIVE STATUS TO OFFLINE
554 032200 010346 MOV R3,-(SP) ;SET 'UNLOAD IN PROGRESS' FLAG
555 032202 004037 035136 JSR R0,WRT.RM ;START THE 'UNLOAD' COMMAND
556 032206 000000 RMCS1
557 032210 032432 C17

```

```

558 032212 000207      RTS      PC      ;RETURN TO USER
559 032214 122703 000143 68:     CMPB   #143,R3    ;IS IT A 'SET FORMAT' COMMAND?
560 032220 001014      BNE      68      ;BR IF NO
561 032222 004037 034756      JSR      R0,RD,RM    ;READ THE OFFSET REGISTER
562 032226 000032      RMOF      C17
563 032230 032432      C17
564 032232 116266 000001 000001  MOVB   1(R2),1(SP)    ;COMBINE 'FMT16','ECI', AND 'HCI'
565 032240 004037 035136      JSR      R0,WRT,RM    ;LOAD 'FMT16','ECI', AND/OR 'HCI'
566 032244 000032      RMOF      C17
567 032246 032432      C17
568 032250 000436      BR
569 032252 122703 000141 68:     CMPB   #141,R3    ;IS IT A 'GET REGISTER' COMMAND?
570 032256 001023      BNE      108     ;BR IF NO
571 032260 016203 000006 78:     MOV     6(R2),R3    ;POINTS TO 1ST ADDRESS OF WHERE
572                                ;TO PUT THE REGISTER(S)
573 032264 116237 000010 032302  MOVB   10(R2),98    ;INIT. THE INDEX FOR THE FIRST REG.
574 032272 116205 000011      MOVB   11(R2),R5    ;INDEX OF LAST REG. TO MOVE
575 032276 004037 034756 88:     JSR      R0,RD,RM    ;READ RH/RM REGISTER
576 032302 000000 68:     RMC S1    ;INDEX OF REG. TO READ
577 032304 032432      C17
578 032306 012623      MOV     (SP)+,(P3)+    ;GET THE CONTENTS OF RH/RM REG.
579 032310 023705 032302      CMP      98,R5      ;LAST REG. BEEN READ?
580 032314 001414      BEQ      128     ;GET OUT IF YES
581 032316 062737 000002 032302  ADD     #2,98    ;INCREASE THE INDEX BY 2
582 032324 000764      BR
583 032326 122703 000145 108:    CMPB   #145,R3    ;IS IT A 'SELECT DRIVE' COMMAND?
584 032332 001405      BEQ      128     ;BR IF YES
585 032334 010346 118:     MOV     R3,-(SP)    ;LOAD THE COMMAND
586 032336 004037 035136      JSR      R0,WRT,RM
587 032342 000000      RMC S1
588 032344 032432      C17
589 032346 004737 036206 128:    JSR      PC,POPQUE    ;REMOVE REQ. FROM QUEUE
590 032352 052762 000200 000016  BIS     #BIT7,16(R2)    ;SET THE 'DONE' BIT
591 032360 005737 030264      TST     SAVEFG    ;SAVE THE RH/RM REGISTERS?
592 032364 100002      BPL      138     ;BR IF NO
593 032366 004737 035330      JSR      PC,SVRW70    ;YES--GO SAVE THE REGISTERS
594 032372 000207 138:     RTS      PC      ;RETURN TO USER
595
596 032374 006301 015:     ASL      R1
597 032376 012761 023420 030270  MOV     #10000.,TIMER(R1)    ;START 10. SECOND TIMER
598 032404 006201      ASR      R1
599 032406 112761 000001 030176  MOVB   #1,DRVACT(R1)    ;SET THE DRIVE ACTIVE
600 032414 000207      RTS      PC      ;RETURN TO THE USER
601
602 032416 010346 016:     MOV     R3,-(SP)    ;LOAD THE COMMAND
603 032420 004037 035136      JSR      R0,WRT,RM
604 032424 000000      RMC S1
605 032426 032432      C17
606 032430 000761      BR      C15
607
608 032432 032764 010000 000010 17:     BIT     #BIT12,RMC S2(R4)    ;DRIVE NON-EXISTENT ?
612 032440 005702 18:     TST     R2      ;ANYTHING IN QUEUE ?
616 032442 001001      BNE      28      ;BR IF QUEUE IS THERE
617 032444 000207      RTS      PC      ;OTHERWISE EXIT
618 032446 012762 104000 000016 28:     MOV     #BIT15,BIT11,16(R2)    ;SET 'PARITY' ERROR INDICATOR
622
623 032454 012746 000111 17B:    MOV     #111,-(SP)    ;DO A 'DRIVE CLEAR'

```

Address	Hex	Label	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	
---------	-----	-------	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	--

```

690
691 032762 112737 000001 030252 ISR:  MOVB  #*,ACTDRV      ;SET 'ACTIVE DRIVER' FLAG
692 032770 104412          SAVREG      ;SAVE R0 - R5
693 032772 013704 030324      MOV  RMADR,R4      ;ADDRESS OF RMCS1
694 032776 013701 030310      MOV  DTUW,R1      ;GET 'DATA TRANSFER UNDERWAY' INDICATOR
695 033002 002402          BLT  1$          ;BR IF NO DATA TRANSFER UNDERWAY
696 033004 004737 033024      JSR  PC,TD      ;CALL TRANSFER DONE
697 033010 004737 033174      JSR  PC,SC      ;CALL SPECIAL CONDITIONS
698 033014 104413          2$:  RESREG      ;RESTORE R0 - R5
699 033016 105037 030252      CLRB  ACTDRV      ;CLEAR 'ACTIVE DRIVER' FLAG
700 033022 000002          RTI              ;RETURN
701
702          ;TRANSFER DONE ROUTINE
703
704 033024 105061 030176      TD:  CLRB  DRVACT(R1)      ;SET DRIVE ACTIVE INDICATOR TO IDLE
705 033030 012737 177777 030310      MOV  #-1,DTUW      ;NO DATA TRANSFERS UNDERWAY
706 033036 006301          ASL  R1
707 033040 012761 177777 030270      MOV  #-1,TIMER(R1) ;CANCEL TIMEOUT
708 033046 006201          ASR  R1
709 033050 013702 030246      MOV  TRNSWT,R2      ;GET 'DPB' ADDRESS FROM THE
710 033054 005037 030246      CLR  TRNSWT      ;TRANSFER WAIT QUEUE--CLEAR QUEUE
711 033060 052762 000200 000016      BIS  #BIT07,16(R2) ;SET DONE
712 033066 010164 000010      MOV  R1,RMCS2(R4) ;SELECT THE DRIVE
713 033072 004037 034756      JSR  R0,RD.RM      ;TRANSFER ERROR(TRE=1)?
714 033076 000000          RMCS1
715 033100 032432          C17
716 033102 006126          ROL  (SP)+          ;IS TRE=1 ?
717 033104 100417          BMI  3$          ;BR IF YES
718 033106 005737 030264      TST  SAVEFG      ;SAVE THE RH/RM REGISTERS?
719 033112 100002          BPL  1$          ;BR IF NO
720 033114 004737 035330      JSR  PC,SVRH70      ;YES--SAVE THE REGISTERS
721 033120 004737 036164      JSR  PC,GETREQ      ;GET DPB POINT=R
722 033124 005702          TST  R2          ;ENTRY FOR DRIVE ?
723 033126 001403          BEQ  2$          ;BR IF NOT
724 033130 004737 031354      JSR  PC,OPT      ;CALL OPTIMIZER
725 033134 000207          RTS  PC          ;RETURN
726 033136 012714 000113      2$:  MOV  #113,(R4) ;RELEASE THE DRIVE
727 033142 000207          RTS  PC          ;RETURN
728
729 033144 052762 100100 000016 3$:  BIS  #BIT15:BIT06,16(R2) ;SET DATA ERROR FLAG
730 033152 004737 036070      JSR  PC,EMPTYQ      ;EMPTY THE 'DRIVE'S WAIT' QUEUE
731 033156 004737 035330      JSR  PC,SVRH70      ;SAVE THE PH/RM REGISTERS
732 033162 012714 040111      MOV  #40111,(R4) ;ISSUE A 'DRIVE CLEAR'
733 033166 012714 000113      MOV  #113,(R4) ;ISSUE A RELEASE TO THE DRIVE
734 033172 000207          RTS  PC          ;RETURN
735
736          ;SPECIAL CONDITION ROUTINE
737
738 033174 116403 000016      SC:  MOVB  RMAS(R4),R3      ;READ 'RMAS'
739 033200 001014          BNE  2$          ;BR IF ANY 'ATA' BITS SET
740 033202 004037 034756      JSR  R0,RD.RM      ;READ CONTROL AND STATUS REGISTER
741 033206 000000          RMCS1
742 033210 032532          C18
743 033212 106126          ROLB (SP)+          ;IS 'IE'=1?
744 033214 100405          BMI  1$          ;YES, NO DRIVES TO CHECK
745 033216 004037 036254      JSR  R0,ES.SAV      ;SAVE THE ADDRESS IN 'ESCAPE'
746 033222 104001          EMT  1          ;REPORT AN ILLEGAL INTERRUPT

```

773	033224	004737	035446		JSR	PC,SET.IE	:SET INTERRUPT ENABLE
774	033230	000207		1\$:	RTS	PC	:RETURN
775	033232	005046		2\$:	CLR	-(SP)	:PROCESS ALL DRIVES THAT HAVE
776	033234	110316			MOVB	R3,(SP)	:AN 'ATA'=1
777	033236	012703	000001		MOV	#1,R3	
778	033242	005001			CLR	R1	
779							
780	033244	030316		SC3:	BIT	R3,(SP)	:ATA=1?
781	033246	001005			BNE	SC5	:YES
782							
783	033250	005201		SC4:	INC	R1	:MOVE TO THE NEXT DRIVE
784	033252	106303			ASLB	R3	
785	033254	001373			BNE	SC3	:BR IF MORE TO CHECK?
786	033256	005726			TST	(SP)+	:CLEAN OFF THE STACK
787	033260	000207			R*5	PC	:RETURN TO USER
788							
789	033262	105761	030226	SC5:	TSTB	DPINT(R1)	:INITIALIZING THE DRIVE ?
790	033266	001402			BEQ	1\$	:BR IF NOT
791	033270	000137	034206		JMP	SC13	:PROCESS THE DRIVE
792	033274	105761	030236	1\$:	TSTB	DPROS(R1)	:PORT REQUEST OUTSTANDING ?
793	033300	001402			BEQ	2\$	:BR IF NOT
794	033302	000137	034206		JMP	SC13	:START THE OUTSTANDING COMMAND
795	033306	105761	030206	2\$:	TSTB	DRVSTA(R1)	:CHECK THE DRIVE STATUS
796	033312	003023			BGT	4\$	:BR IF ONLINE
797	033314	105761	030254		TSTB	ULDFLG(R1)	:UNLOAD IN PROGRESS?
798	033320	003420			BLE	4\$	:BR IF NOT
799	033322	004737	036164		JSR	PC,GETREQ	:GET DPB POINTER
800	033326	004737	035330		JSR	PC,SVRH70	:SAVE THE RH/RM REGISTERS
801	033332	004737	034136		JSR	PC,SC12	:SAVE RMD5, RMER1, RMER2, AND RMMR2
802							:ALSO DO A DRIVE INIT (DRVINT)
803	033336	105761	030206		TSTB	DRVSTA(R1)	:DID DRIVE COME ONLINE?
804	033342	003414			BLE	5\$	:NO
805	033344	032737	040000	030166	BIT	#BIT14,RMERRS	:WAS THERE AN ERROR?
806	033352	001000			BNE	3\$	:BR IF ERROR
810	033354	013705	030170	3\$:	MOV	RMERRS+2,R5	:YES -- PICKUP RMER1 AND
811	033360	000504			BR	SC6A	:GO PROCESS THE ERROR
812	033362	105761	030176	4\$:	TSTB	DRVACT(R1)	:DRIVE ACTIVE WITH COMMAND OR ERROR RECOVERY ?
813	033366	001033			BNE	SC6	:BR IF EITHER
814	033370	004737	034136		JSR	PC,SC12	:SAVE RMD5, RMER1, RMER2, AND RMMR2
815							:ALSO DO A DRVINT
816	033374	105761	030226	5\$:	TSTB	DPINT(R1)	:TRYING TO INIT THE DRIVE ?
817	033400	001323			BNE	SC4	:BR IF YES, CHECK ON MORE DRIVES
818	033402	105761	030206		TSTB	DRVSTA(R1)	:CHECK ON DRIVE'S STATUS
819	033406	100412			BMI	6\$	:BR IF UNSAFE
820	033410	032737	020000	030172	BIT	#BIT13,RMERRS+4	:ADDRESS PLUG CHANGED ?
821	033416	001013			BNE	7\$	:BR IF YES
825	033420	012746	000111		MOV	#111, -(SP)	:DRIVE CLEAR
826	033424	004037	035136		JSR	R0,WRT.RM	:WRITE THE COMMAND INTO RMCS1
827	033430	000000			RMCS1		:REGISTER INDEX
828	033432	033776			SC8		:PARITY EXIT ADDRESS
829	033434	011605		6\$:	MOV	(SP),R5	:PICKUP (RMAS) BEFORE THE ERROR CALL
830	033436	004037	036254		JSR	R0,ES.SAV	:SAVE THE ADDRESS IN '\$ESCAPE'
	033442	104002			EMI	2	:REPORT THE UNEXPECTED ATTENTION
831	033444	000701			BR	SC4	:GO CHECK FOR MORE ATA'S
832	033446			7\$:			
	033446	004037	036254		JSR	R0,ES.SAV	:SAVE THE ADDRESS IN '\$ESCAPE'
	033452	104005			EMI	5	:REPORT THE ADDRESS PLUG CHANGE

```

833 033454 000675      BR      SC4      ;CHECK FOR MORE DRIVES
834
835 033456 006301      SC6:  ASL      R1      ;SETUP TO ADDRESS WORDS
836 033460 012761 177777 030270      MOV      R1      ;STOP THE TIMER
837 033466 006201      ASR      R1      ;RESTORE THE DRIVE ADDRESS
838 033470 004737 036164      JSR      PC,GETREQ ;GET THE DPB POINTER FROM THE QUEUE
839 033474 010164 000010      MOV      R1,RMCS2(R4) ;SELECT DRIVE
840 033500 000137 034026      JMP      SC11 ;PROCESS THE SEARCH
841 033504 004037 034756      JSR      R0,RD.RM ;READ THE RM'S STATUS REG.
842 033510 000012      RMD5
843 033512 033776      SC8
844 033514 011605      MOV      (SP),R5 ;AND PUT IT IN R5
845 033516 006126      ROL      (SP)+ ;WAS THERE AN ERROR?
846 033520 100407      BMI      1$ ;BR IF ERROR
847 033522 105761 030176      TSTB     DRVACT(R1) ;CHECK DRIVE'S STATE
848 033526 003137      BGT      SC11 ;BR IF DRIVE ACTIVE WITH ORDER
849 033530 052762 100210 000016      BIS      #BIT15!BIT07!BIT03,16(R2) ;INFORM USER OF ERROR RECOVER COMPLETION
850 033536 000470      BR      SC7
851 033540 004037 034756      1$: JSR      R0,RD.RM ;READ ERROR REGISTER #1
852 033544 000014      RMER1
853 033546 033776      SC8
854 033550 012605      MOV      (SP)+,R5 ;AND SAVE IT IN R5
855 033552 004737 035330      JSR      PC,SVRH70 ;SAVE RH/RM REGISTERS
856 033556 012746 000111      MOV      #111,-(SP) ;ISSUE A DRIVE CLEAR
857 033562 004037 035136      JSR      R0,WRT.RM
858 033566 000000      RMCS1
859 033570 033776      SC8
860
861 033572 006105      SC6A: ROL      R5 ;WAS 'UNSAFE' CONDITION =1?
862 033574 100406      BMI      1$ ;BR IF YES
863 033576 005702      TST      R2 ;ANYTHING IN QUEUE ?
864 033600 001447      BEQ      SC7 ;BR IF NOT
865 033602 052762 100240 000016      BIS      #BIT15!BIT07!BIT05,16(R2) ;INFORM USER OF ERROR
866 033610 000443      BR      SC7
867 033612 004037 034756      1$: JSR      R0,RD.RM ;READ DRIVE STATUS REG. #1
868 033616 000012      RMD5
869 033620 033776      SC8
870 033622 011605      MOV      (SP),R5 ;SAVE RMD5 IN R5
871 033624 006126      ROL      (SP)+ ;'ERR'=1?
872 033626 100011      BPL      2$ ;BR IF NO--UNSAFE CLEARED
873 033630 112761 177777 030206      MOVB     #-1,DRVSTA(R1) ;DRIVE IS UNSAFE
874 033636 004737 035330      JSR      PC,SVRH70 ;SAVE RH/RM REGISTERS
875 033642 052762 110000 000016      BIS      #BIT15!BIT12,16(R2) ;INFORM USER OF UNSAFE ERROR
876 033650 000423      BR      SC7
877 033652 032705 010000      2$: BIT      #BIT12,R5 ;'MOL' = 1 ?
878 033656 001015      BNE      3$ ;BR IF YES
879 033660 112761 177777 030176      MOVB     #-1,DRVACT(R1) ;ACTIVE ERROR RECOVER
880 033666 112761 000001 030206      MOVB     #1,DRVSTA(R1) ;ONLINE
881 033674 006301      ASL      R1
882 033676 012761 035230 030270      MOV      #15000.,TIMER(R1) ;START 15. SECOND TIMER
883 033704 006201      ASR      R1
884 033706 000137 033250      JMP      SC4
885 033712 052762 100220 000016      3$: BIS      #BIT15!BIT07!BIT04,16(R2) ;INFORM USER OF ERROR
886
887 033720 105061 030176      SC7: CLRB     DRVACT(R1) ;DRIVE IS IDLE
888 033724 004737 036206      JSR      PC,POPQUE ;REMOVE THE QUEUE
889 033730 105761 030254      TSTB     ULDFLG(R1) ;UNLOAD IN PROGRESS OR QUEUE?

```

```

893 033734 003002      BGT 1$ ;BR IF NOT
894 033736 105061 030254 CLR  ULDFLG(R1) ;CLEAR UNLOAD FLAG
895 033742 116164 030312 000016 1$: MOV  ATABIT(R1),RMAS(R4) ;CLEAR ATTENTION BIT
896 033750 105761 030206      TST  DRVSTA(R1) ;IS THE DRIVE UNSAFE ?
897 033754 100406      BMI  2$ ;BR IF IT IS
901 033756 012746 000111      MOV  #111, -(SP) ;DRIVE CLEAR COMMAND
902 033762 004037 035136      JSR  R0, WRT.RM ;WRITE THE COMMAND INTO RPCS1
903 033766 000000      RMCS1 ;REGISTER INDEX
904 033770 033776      SCB  ;PARITY EXIT ADDRESS
905 033772 000137 033250      JMP  SC4 ;CHECK FOR MORE DRIVES
906
907 033776 105761 030176      SCB: TST  DRVACT(R1) ;IS DRIVE IDLE?
908 034002 001405      BEQ  1$ ;YES
909 034004 004737 036164      JSR  PC, GETREQ ;GET DPB POINTER
910 034010 004737 032432      JSR  PC, C17 ;PROCESS THE PARITY ERROR
911 034014 000402      BR   2$ ;CONTINUE
912 034016
916 034016 004737 032454      JSR  PC, C17B ;PROCESS THE UNCORRECTABLE PARITY ERROR
917 034022 000137 033250      JMP  SC4 ;CHECK MORE DRIVES
918
919 034026 105761 030254      SC11: TST  ULDFLG(R1) ;'UNLOAD IN PROGRESS'?
920 034032 003402      BLE  1$ ;BR IF NO
921 034034 105061 030254      CLR  ULDFLG(R1) ;CLEAR UNLOAD FLAG
922 034040 105061 030176      CLR  DRVACT(R1) ;SET DRIVE IDLE
923 034044 136137 030312 030250      BIT  ATABIT(R1), SRCHWT ;DOING A SEARCH OPERATION FOR
924                                     ;AN I/O COMMAND?
925 034052 001012      BNE  2$ ;BR IF YES
926 034054 004737 036206      JSR  PC, POPQUE ;REMOVE REQUEST FROM QUEUE
927 034060 052762 000200 000016      BIS  #BIT07, 16(R2) ;SET 'DONE' BIT
928 034066 005737 030264      TST  SAVEFG ;SAVE THE REGISTERS?
929 034072 100002      BPL  2$ ;BR IF NO
930 034074 004737 035330      JSR  PC, SVRH70 ;YES--SAVE ALL OF THE RH/RM REG'S
931 034100 116164 030312 000016 2$: MOV  ATABIT(R1), RMAS(R4) ;CLEAR ATTENTION BIT
932 034106 146137 030312 030250      BICB ATABIT(R1), SRCHWT ;CLEAR IMPLIED SEEK SET
933 034114 006301      ASL  R1 ;WORD INDEX
934 034116 012761 177777 030270      MOV  #-1, TIMER(R1) ;STOP CLOCK
935 034124 006201      ASR  R1 ;RESTORE R1
936 034126 004737 031354      JSR  PC, OPT ;START A REQUEST
937 034132 000137 033250      JMP  SC4 ;CHECK FOR MORE DRIVES
938
939 034136 010164 000010      SC12: MOV  R1, RMCS2(R4) ;SELECT DRIVE
940 034142 016437 000012 030166      MOV  RMD5(R4), RMERRS ;SAVE THE FOUR REGISTERS THAT
941 034150 016437 000014 030170      MOV  RMER1(R4), RMERRS+2 ;WILL TELL US SOMETHING
942 034156 016437 000042 030172      MOV  RMER2(R4), RMERRS+4
943 034164 016437 000040 030174      MOV  RMER2(R4), RMERRS+6
944 034172 004037 030546      JSR  R0, DRVINT ;INIT. THE STATE OF THE DRIVE
945 034176 000401      BR   1$ ;TAKE ERROR EXIT
946 034200 000207      RTS  PC ;RETURN
947 034202 005726      1$: TST  (SP)+ ;POP PC OFF OF THE STACK
948 034204 000674      BR   SC8 ;PROCESS THE PARITY ERROR
949
950 034206 006301      SC13: ASL  R1 ;SETUP TO ADDRESS WORDS
951 034210 012761 177777 030270      MOV  #-1, TIMER(R1) ;STOP THE TIMER
952 034216 006201      ASR  R1 ;
953 034220 010164 000010      MOV  R1, RMCS2(R4) ;SELECT THE DRIVE
954 034224 116164 030312 000016      MOV  ATABIT(R1), RMAS(R4) ;CLEAR THE ATTENTION BIT
955 034232 105761 030226      1$: TST  DPINT(R1) ;INITIALIZING THE DRIVE ?

```

```

956 034236 001424 BEQ 2$ ;BR IF NOT
957 034240 105061 030226 CLRB DPINT(R1) ;CLEAR THE INIT INDICATOR
958 034244 004037 030546 JSR R0,DRVINT ;GO INIT THE DRIVE
959 034250 000240 NOP ;DUMMY PARITY ERROR RETURN
960 034252 105761 030206 TSTB DRVSTA(R1) ;DRIVE ONLINE ?
961 034256 003014 BGT 2$ ;BR IF YES -- START ORDER
962 034260 005702 TST R2 ;QUEUE ENTRY FOR THE DRIVE
963 034262 001426 BEQ 3$ ;BR IF NOT
964 034264 004737 036164 JSR PC,GETREQ ;GET DPB ADDRESS
965 034270 052762 140000 000016 BIS #BIT15:BIT14,16(R2) ;INFORM USER THAT DRIVE OFFLINE
966 034276 004737 035330 JSR PC,SVRH70 ;SAVE THE REGISTERS
970 034302 004737 036206 JSR PC,POPQUE ;REMOVE THE QUEUE
971 034306 000414 BR 3$
972 034310 032764 004000 000000 2$: BIT #BIT11,RMC1(R4) ;DVA SET ?
973 034316 001006 BNE 4$ ;SET THEN CALL OPT
974 034320 006301 ASL R1
975 034322 012761 035230 030270 MOV #15000.,TIMER(R1) ;START 15. SECOND TIMER
976 034330 006201 ASR R1
977 034332 000402 BR 3$
978 034334 004737 031354 4$: JSR PC,OPT ;START THE PENDING REQUEST
979 034340 000137 033250 3$: JMP SC4 ;PROCESS OTHER DRIVES
980
981 ;RM TIMER ROUTINE
982 ;CALL
983 ;
984 ; MOV #TIME,-(SP) ;ELAPSED TIME IN MILLISECONDS ON THE STACK
985 ; JSR PC,RMTMR ;CALL RM05 TIME ROUTINE
986 034344 005737 030252 RMTMR: TST ACTDRV ;CHECK 'ACTDRV & ACTSTR'
987 034350 001027 BNE 4$ ;IF NON ZERO EXIT
988 034352 112737 000001 030253 MOVB #1,ACTSTR ;SET 'ACTSTR'
989 034360 104412 SAVREG ;SAVE R0 - R5
990 034362 005001 CLR R1 ;START WITH DRIVE 0
991 034364 005003 CLR R3
992 034366 005763 030270 1$: TST TIMER(R3) ;IS THE TIMER RUNNING?
993 034372 002406 BLT 2$ ;BR IF NO
994 034374 166663 000002 030270 SUB 2(SP),TIMER(R3) ;COUNT THE INTERVAL
995 034402 003002 BGT 2$ ;BR IF NO SOFTWARE TIMEOUT
996 034404 004737 034434 JSR PC,STO ;CALL SOFTWARE TIMEOUT ROUTINE
997 034410 005201 2$: INC R1 ;MOVE TO NEXT DRIVE
998 034412 005723 TST (R3)+
999 034414 022701 000010 CMP #8.,R1 ;OUT OF DRIVES?
1000 034420 003362 BGT 1$ ;BR IF NO
1001 034422 104413 3$: RESREG ;RESTORE R0 - R5
1002 034424 105037 030253 CLRB ACTSTR ;ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
1003 034430 012616 4$: MOV (SP)+,(SP) ;ADJUST THE STACK
1004 034432 000207 RTS PC ;RETURN
1005
1006 ;SOFTWARE TIMEOUT ROUTINE
1007 ;
1008 ;NOTE: THIS ROUTINE MUST BE ENTERED AT PRIORITY 6
1009 ;OR GREATER
1010 ;
1011 ;CALL: STO
1012 ; MOV #DRVNUM,R1 ;DRIVE NUMBER
1013 ; JSR PC,STO ;CALL
1014 ;
1015 ; RETURN

```

```

1016 034434 010116          STO:  MOV    R1,-(SP)      :SAVE R1
1017 034436 010246          MOV    R2,-(SP)      :SAVE R2
1018 034440 010346          MOV    R3,-(SP)      :SAVE R3
1019 034442 010446          MOV    R4,-(SP)      :SAVE R4
1020 034444 013704 030324    MOV    RMADR,R4      :GET ADDRESS OF 'RMCS1'
1021 034450 010164 000010    MOV    R1,RMCS2(R4)   :SELECT THE DRIVE
1022 034454 004037 034756    JSR    R0,RD.RM      :READ 'DRIVE STATUS REG'
1023 034460 000012          RMD5
1027 034462 034744          ST09
1028 034464 105726          TSTB   (SP)+         :IS 'DRY'=1?
1029 034466 100436          BMI     ST02          :BR IF YES
1030 034470 105761 030226    ST01:  TSTB   DPINT(R1)   :TRYING TO INITIALIZE THE DRIVE ?
1031 034474 001033          BNE     ST02          :BR IF YES
1032 034476 105761 030236    TSTB   DPRQS(R1)   :OUTSTANDING PORT REQUEST OR THE DRIVE ?
1033 034502 001030          BNE     ST02          :BR IF YES
1034 034504 013702 030246    MOV     TRNSWT,R2     :PICKUP TRANSFER WAIT QUEUE
1035 034510 020137 030310    CMP     R1,DTUW      :TRANSFER UNDERWAY ON THIS DRIVE?
1036 034514 001404          BEQ     1$           :BR IF YES
1037 034516 000137 034744    JMP     ST09          :IF NOT DON'T BOTHER DRIVES
1038 034522 004737 036164    JSR     PC,GETREQ    :GET DPB ADDRESS
1039 034526 052762 101000 000016 1$:  BIS     #BIT15!BIT09,16(R2) :SET THE ERROR FLAGS
1040 034534 004737 035330    JSR     PC,SVRH70     :SAVE RH/RM REGISTERS
1044 034540 105061 030176    CLRB   DRVACT(R1)   :DRIVE IS IDLE
1045 034544 105061 030254    CLRB   ULDFLG(R1)   :CLEAR THE UNLOAD FLAG
1046 034550 005037 030246    CLR     TRNSWT      :CLEAR DPB ADDRESS
1047 034554 012737 177777 030310    MOV     #-1,DTUW      :CLEAR THE TRANSFER DRIVE #
1048 034562 000470          BR      ST09          :DON'T BOTHER OTHER DRIVES
1049
1050 034564 116405 000016    ST02:  MOVB   RMAS(R4),R5   :READ ATTENTION REG
1051 034570 136105 030312    BITB   ATABIT(R1),R5   :IS ATTENTION FOR THIS DRIVE UP ?
1052 034574 001007          BNE     ST03          :YES
1053 034576 105761 030226    TSTB   DPINT(R1)   :TRYING TO INITIALIZE THE DRIVE ?
1054 034602 001021          BNE     ST06          :BR IF YES - DRIVE NOT ONLINE
1055 034604 105761 030236    TSTB   DPRQS(R1)   :OUTSTANDING PORT REQUEST FOR THE DRIVE ?
1056 034610 001035          BNE     ST07          :BR IF YES - NO RESPONSE TO REQUEST
1057 034612 000454          BR      ST09          :OTHER WISE EXIT
1058
1059 034614 105761 030226    ST03:  TSTB   DPINT(R1)   :INITIALIZING THE DRIVE ?
1060 034620 001003          BNE     1$           :BR IF INIT PENDING
1061 034622 105761 030236    TSTB   DPRQS(R1)   :PORT REQUEST PENDING ?
1062 034626 001446          BEQ     ST09          :BR IF NOT
1063 034630 012763 177777 030270 1$:  MOV     #-1,TIMER(R3) :STOP THE TIMER
1064 034636 000442          BR      ST09          :EXIT
1065
1066 034640 004737 032532    ST05:  JSR     PC,C18      :GO HANDLE THE PARITY ERROR
1067 034644 000437          BR      ST09
1068
1069 034646 105061 030226    ST06:  CLRB   DPINT(R1)   :CLEAR THE INITIALIZE INDICATOR
1070 034652 105061 030206    CLRB   DRVSTA(R1)   :SET UNIT OFFLINE
1071 034656 012763 177777 030270    MOV     #-1,TIMER(R3) :STOP THE TIMER
1072 034664 004737 036164    JSR     PC,GETREQ    :GET THE DPB ADDRESS
1073 034670 005702          TST     R2           :REQUEST IN QUEUE ?
1074 034672 001424          BEQ     ST09          :BR IF NOT
1075 034674 052762 140000 000016    BIS     #BIT15!BIT14,16(R2) :INFORM THE USER DRIVE NOT AVAILABLE
1076 034702 000414          BR      ST08          :FINISH
1077
1078 034704 012763 177777 030270 ST07:  MOV     #-1,TIMER(R3) :STOP THE TIMER

```

```

1079 034712 105061 030236          CLR8    DPRQS(R1)      ;CLEAR PORT REQUEST INDICATOR
1080 034716 004737 036164          JSR      PC.GETREQ    ;GET DPB ADDRESS
1081 034722 005702                   TST      R2              ;QUEUE ENTRY FOR DRIVE ?
1082 034724 001407                   BEQ      ST09            ;BR IF NONE
1083 034726 012762 100004 000016   ST08:  MOV     #BIT15!BIT2,16(R2) ;INFORM USER OF PORT REQUEST ERROR
1084 034734 004737 036070          JSR      PC.EMPTYQ    ;CLEAR THE QUEUE FOR THE DRIVE
1085 034740 004737 035330          JSR      PC.SVRH70    ;SAVE THE REGISTERS
1086 034744 012604          ST09:  MOV     (SP)+,R4      ;RESTORE R4
1087 034746 012603          MOV     (SP)+,R3      ;RESTORE R3
1088 034750 012602          MOV     (SP)+,R2      ;RESTORE R2
1089 034752 012601          MOV     (SP)+,R1      ;RESTORE R1
1090 034754 000207          RTS      PC              ;RETURN
1091
1092          ;ROUTINE TO READ A RH/RM REGISTER
1093          ;CALL
1094          ;JSR      R0,RD.RM      ;GO READ A REGISTER
1095          ;INDEX      ;REG. INDEX FROM BASE
1096          ;ERRADR    ;ERROR ADDRESS--PROCESS ERROR STARTING
1097          ;          ;AT THIS ADDRESS
1098          ;          ;CONTENTS OF REG. IS ON THE STACK
1099          ;RETURN
1100
1101 034756 013737 030322 035124   RD.RM:  MOV     MCPEMX,RD.RM2 ;MAX. RETRYS ALLOWED
1102 034764 011646          MOV     (SP),-(SP)    ;SAVE R0 FOR RETURN
1103 034766 013737 030324 035002   MOV     RMADR,RD.ADR    ;FORM THE DESIRED ADDRESS
1104 034774 062037 035002          ADD     (R0)+,RD.ADR ;USING THE BASE AND THE INDEX
1105 035000 013727          RD.RM1: MOV     @ (PC)+,(PC)+ ;READ THE DESIRED REGISTER OF THE RM DRIVE
1106 035002 000000          RD.ADR:  .WORD    0      ;ADDRESS IS FORMED HERE
1107 035004 000000          RD.WRD:  .WORD    0      ;REG. CONTENTS PUT HERE
1108 035006 013766 035004 000002   MOV     RD.WRD,2(SP)    ;RETURN IT TO THE USER
1109 035014 013746 030324          MOV     RMADR,-(SP)   ;PUT THE ADDRESS ON THE STACK
1110 035020 062716 000010          ADD     #RMCS2,(SP)    ;FORM THE ADDRESS OF RMCS2
1111 035024 032736 010000          BIT     #BIT12,@(SP)+ ;CHECK THE 'NED' BIT
1112 035030 001037          BNE      RD.RM3        ;BR IF DRIVE NON-EXISTENT
1113 035032 017746 173266          MOV     @RMADR,-(SP)   ;READ RMCS1
1114 035036 032716 020000          BIT     #BIT13,(SP)    ;DID MCPE SET?
1115 035042 001002          BNE      1$            ;BR IF YES
1116 035044 022620          CMP      (SP)+,(R0)+ ;ADJUST FOR RETURN
1117 035046 000432          BR       RD.RM4        ;EXIT
1118 035050          1$:
1119 035050 004037 036254          JSR      R0,ES.SAV    ;SAVE THE ADDRESS IN '$ESCAPE'
1120 035054 104003          EMT      3              ;REPORT 'MCPE' ERROR
1121 035056 005737 030310          TST      DTUW        ;DATA TRANSFER UNDERWAY?
1122 035062 100405          BMI      2$            ;NO
1123 035064 032716 040000          BIT     #BIT14,(SP)    ;'TRE' = 1 ?
1124 035070 001402          BEQ      2$            ;NO
1125 035072 005726          TST      (SP)+        ;YES--CLEAN OFF THE STACK AND
1126 035074 000415          BR       RD.RM3        ;TAKE THE FATAL ERROR EXIT
1127 035076 052716 040000          2$:  BIS      #BIT14,(SP)    ;CLEAR 'MCPE' BY SENDING A '1' TO 'TRE'
1128 035102 000316          SWAB     (SP)          ;POSITION BEFORE WRITING
1129 035104 013737 030324 035120   MOV     RMADR,3$      ;FORM ADDRESS OF HIGH BYTE
1130 035112 005237 035120          INC      3$
1131 035116 112637          MOV8     (SP)+,@(PC)+ ;WRITE THE HIGH BYTE OF RMCS1
1132 035120 000000          3$:  .WORD    0      ;ADDRESS STORAGE
1133 035122 005327          DEC      (PC)+    ;EXCEEDED MAX. RETRYS
1134 035124 000003          RD.RM2: .WORD    3
1135 035126 002324          BGE      RD.RM1        ;BR IF NO

```

```

1134 035130 011000 RD.RM3: MOV (R0),R0 ;FATAL ERROR EXIT
1135 035132 012616 MOV (SP)+,(SP)
1136 035134 000200 RD.RM4: RTS R0
1137
1138 ;ROUTINE TO WRITE A REGISTER
1139 :CALL
1140 :CALL
1141 :CALL MOV DATA,-(SP) ;DATA TO BE LOADED ON THE STACK
1142 :CALL JSR R0,WRT.RM ;CALL THE ROUTINE TO LOAD(WRITE) THE REG.
1143 :CALL INDEX ;INDEX OF THE REGISTER TO BE LOADED
1144 :CALL ERRADR ;ADDRESS TO RETURN TO ON AN ERROR
1145 :CALL RETURN ;ERROR FREE RETURN
1146
1147 035136 013737 030322 035314 WRT.RM: MOV MCPMX,WRT.R2 ;MAX RETRYS ALLOWED
1148 035144 016637 000002 035224 MOV 2(SP),WRT.WD ;SAVE THE WORD TO WRITE
1149 035152 012616 MOV (SP)+,(SP) ;ADJUST THE STACK
1150 035154 012037 035226 MOV (R0)+,WRT.AD ;GET INDEX OF REGISTER TO BE WRITTEN
1151 035160 001015 BNE 1$ ;BR IF NOT RMCS1
1152 035162 122737 000150 035224 CMPSB #150,WRT.WD ;IS THE COMMAND FOR DATA TRANSFERS?
1153 035170 002411 BLT 1$ ;YES--DON'T GET THE OLD A16 & A17, & PSEL
1154 035172 004037 034756 JSR R0,RD.RM ;NO---COMBINE A16&A17, & PSEL WITH
1155 035176 000000 RMCS1 ;THE COMMAND BEFORE SENDING IT TO
1156 035200 035320 WRT.R3 ;THE RH/RM
1157 035202 000316 SWAB (SP)
1158 035204 042716 177770 BIC #^C7,(SP)
1159 035210 112637 035225 MOVB (SP)+,WRT.WD+1
1160 035214 063737 030324 035226 1$: ADD RMADR,WRT.AD ;FORM THE ADDRESS OF THE DISK REG.
1161 035222 012737 WRT.R1: MOV (PC)+,a(PC)+ ;LOAD THE DESIRED REG.
1162 035224 000000 WRT.WD: .WORD 0 ;WORD TO WRITE GOES HERE
1163 035226 000000 WRT.AD: .WORD 0 ;ADDRESS IS FORMED HERE
1164 035230 013746 030324 MOV RMADR,-(SP) ;PUT THE ADDRESS ON THE STACK
1165 035234 062716 000010 ADD #RMCS2,(SP) ;FORM THE ADDRESS OF RMCS2
1166 035240 032736 010000 BIT #BIT12,a(SP)+ ;CHECK THE 'NED' BIT
1167 035244 001025 BNE WRT.R3 ;BR IF DRIVE NON-EXISTENT
1168 035246 004037 034756 JSR R0,RD.RM ;CHECK FOR PARITY ERROR ON WRITE
1169 035252 000014 RMER1
1170 035254 035320 WRT.R3
1171 035256 032726 000010 BIT #BIT03,(SP)+
1172 035262 001420 BEQ WRT.R4 ;BR IF 'PAR=0'
1173 035264 016037 177776 035276 MOV -2(R0),1$ ;PICKUP THE INDEX
1174 035272 004037 034756 JSR R0,RD.RM ;READ THE REG.
1175 035276 000000 1$: .WORD 0 ;REG. INDEX
1176 035300 035320 WRT.R3 ;RETURN TO THIS ADDRESS ON ERROR
1177 035302 004037 036254 JSR R0,ES.SAV ;SAVE THE ADDRESS IN '$ESCAPE'
1178 035306 104004 EMT 4 ;REPORT THE PARITY ON WRITE ERROR
1179 035310 005726 TST (SP)+ ;CLEAR OFF THE STACK
1180 035312 005327 DEC (PC)+ ;DECREMENT THE ERROR COUNT
1181 035314 000003 WRT.R2: .WORD 3 ;RETRY COUNTER
1182 035316 002341 BGE WRT.R1 ;TRY AGAIN IF NOT FINISHED
1183 035320 011000 WRT.R3: MOV (R0),R0 ;TAKE THE 'PARITY ON WRITE' ERROR EXIT
1184 035322 000401 BR WRT.R5 ;EXIT
1185 035324 005720 WRT.R4: TST (R0)+ ;ADJUST FOR ERROR FREE EXIT
1186 035326 000200 WRT.R5: RTS R0
1187
1188 ;ROUTINE TO SAVE THE RH/RM REGISTERS AS PER DPB+14
1189 :CALL

```

```

1190      :      MOV      #DPBNUM,R2      :DPB POINTER TO R2
1191      :      JSR      PC,SVRH70        :SAVE THE DRIVES REG'S
1192
1193 035330 104412      SVRH70: SAVREG      :SAVE R0 - R5
1194 035332 005702      TST      R2        :QUEUE ENTRY FOR THE DRIVE ?
1195 035334 001442      BEQ      6$        :BR IF NONE
1196 035336 013704 030324      MOV      RMADR,R4
1197 035342 111264 000010      MOV      (R2),RMCS2(R4) :SELECT DRIVE
1198 035346 016203 000014      MOV      14(R2),R3      :GET THE ERROR TABLE POINTER
1199 035352 001433      BEQ      6$        :EXIT IF NO ADDRESS
1200 035354 005037 035410      CLR      3$        :COUNTER & POINTER
1201 035360 023727 035410 000022 1$:      CMP      3$,#RMD8      :REACHED THE BUFFER REGISTER ?
1202 035366 001006      BNE      2$        :BR IF NOT
1203 035370 032764 000200 000010      BIT      #BIT07,RMCS2(R4) :'OR' SET ?
1204 035376 001002      BNE      2$        :BR IF SET
1205 035400 005023      CLR      (R3)+      :STORE RMD8 AS ZEROES
1206 035402 000405      BR      4$        :CONTINUE
1207 035404 004037 034756      2$:      JSR      R0,RD.RM      :READ THE SELECTED REGISTER
1208 035410 000000      3$:      .WORD      0        :REGISTER INDEX
1209 035412 035436      5$:      .FILL      5$        :ERROR RETURN ADDRESS
1210 035414 012623      MOV      (SP)+,(R3)+      :STORE THE REGISTER CONTENTS
1211 035416 023727 035410 000046 4$:      CMP      3$,#RMEC2      :REACHED THE END ?
1212 035424 001406      BEQ      6$        :BR IF YES
1213 035426 062737 000002 035410      ADD      #2,3$      :INCREMENT THE REGISTER INDEX
1214 035434 000751      BR      1$        :CONTINUE READING THE REGISTERS
1215 035436 004737 032432      5$:      JSR      PC,C17      :PROCESS THE UNCORRECTABLE PARITY ERROR
1226 035442 104413      6$:      RESREG      :RESTORE R0 - R5
1228 035444 000207      RTS      PC        :RETURN
1229
1230      :ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A 'TRE'
1231      :CALL
1232      :      MOV      #DRVNUM,R1      :DRIVE NUMBER TO R1
1233      :      JSR      PC,SET.IE      :SET 'IE'
1234      :
1235
1236 035446 010446      SET.IE: MOV      R4,-(SP)      :SAVE R4
1237 035450 013704 030324      MOV      RMADR,R4      :PICKUP ADDRESS OF RMCS1
1238 035454 010164 000010      MOV      R1,RMCS2(R4)      :SELECT DRIVE
1239 035460 011446      MOV      (R4),-(SP)      :READ RMCS1
1240 035462 052716 040000      BIS      #BIT14,(SP)      :SET THE 'TRE' BIT OF THE WORD READ
1241 035466 000316      SWAB      (SP)      :ADJUST FOR DAT0
1242 035470 112714 000100      MOV      #BIT06,(R4)      :SET 'IE'
1243 035474 032764 010000 000010      BIT      #BIT12,RMCS2(R4) :IS 'NED'=1?
1244 035502 001002      BNE      1$        :YES--CLEAR 'TRE'
1245 035504 005726      TST      (SP)+      :CLEAN OFF THE STACK
1246 035506 000402      BR      2$        :
1247 035510 112664 000001      1$:      MOV      (SP)+,1(R4)      :CLEAR 'TRE'
1248 035514 012604      2$:      MOV      (SP)+,R4      :RESTORE R4
1249 035516 000207      RTS      PC        :RETURN TO CALLER
1250
1251      :QUEUE COUNT
1252
1253 035520 000      QCNT: .BYTE      0        :DRIVE 0
1254 035521 000      .BYTE      0        :DRIVE 1
1255 035522 000      .BYTE      0        :DRIVE 2
1256 035523 000      .BYTE      0        :DRIVE 3
1257 035524 000      .BYTE      0        :DRIVE 4

```

```

035525      000                .BYTE      0                ;DRIVE 5
035526      000                .BYTE      0                ;DRIVE 6
035527      000                .BYTE      0                ;DRIVE 7

1257
1258
1259      .      ;QUEUE INPUT POINTERS
1260 035530 035612      QINPT: .WORD      QDRV0                ;DRIVE 0
1263 035532 035632      .WORD      QDRV1                ;DRIVE 1
      035534 035652      .WORD      QDRV2                ;DRIVE 2
      035536 035672      .WORD      QDRV3                ;DRIVE 3
      035540 035712      .WORD      QDRV4                ;DRIVE 4
      035542 035732      .WORD      QDRV5                ;DRIVE 5
      035544 035752      .WORD      QDRV6                ;DRIVE 6
      035546 035772      .WORD      QDRV7                ;DRIVE 7

1264
1265      .      ;QUEUE OUTPUT POINTERS
1266
1267 035550 035612      QOUTPT: .WORD      QDRV0                ;DRIVE 0
1270 035552 035632      .WORD      QDRV1                ;DRIVE 1
      035554 035652      .WORD      QDRV2                ;DRIVE 2
      035556 035672      .WORD      QDRV3                ;DRIVE 3
      035560 035712      .WORD      QDRV4                ;DRIVE 4
      035562 035732      .WORD      QDRV5                ;DRIVE 5
      035564 035752      .WORD      QDRV6                ;DRIVE 6
      035566 035772      .WORD      QDRV7                ;DRIVE 7

1271
1272 035570 035612      QSTART: .WORD      QDRV0                ;DRIVE 0 START ADDRESS
1273 035572 035632      QSTOP:  .WORD      QDRV1                ;DRIVE 0 STOP ADDRESS & DRIVE 1 START ADDRESS
1274 035574 035652      .WORD      QDRV2                ;STOP DRIVE 1--START DRIVE 2
1275 035576 035672      .WORD      QDRV3                ;STOP DRIVE 2--START DRIVE 3
1276 035600 035712      .WORD      QDRV4                ;STOP DRIVE 3--START DRIVE 4
1277 035602 035732      .WORD      QDRV5                ;STOP DRIVE 4--START DRIVE 5
1278 035604 035752      .WORD      QDRV6                ;STOP DRIVE 5--START DRIVE 6
1279 035606 035772      .WORD      QDRV7                ;STOP DRIVE 6--START DRIVE 7
1280 035610 036012      .WORD      QTERM                ;STOP DRIVE 7
1281
1282      .      ;DRIVE REQUEST QUEUES
1283
1286 035612      QDRV0:  .BLKW      10
      035632      QDRV1:  .BLKW      10
      035652      QDRV2:  .BLKW      10
      035672      QDRV3:  .BLKW      10
      035712      QDRV4:  .BLKW      10
      035732      QDRV5:  .BLKW      10
      035752      QDRV6:  .BLKW      10
      035772      QDRV7:  .BLKW      10
1287      036012      QTERM=.
1288
1289      .      ;ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
1290      .
1291      .      ;CALL
1292      .      JSR      PC,CLRQUE
1293
1294 036012 104412      CLRQUE: SAVREG                ;SAVE R0 - R5
1295 036014 012702      MOV      #QCNT,R2                ;ZERO THE QUEUE COUNTS
1296 036020 005022      CLR      (R2)+                ;DRIVES 0 & 1
1297 036022 005022      CLR      (R2)+                ;DRIVES 2 & 3
  
```

```

1298 036024 005022          CLR      (R2)+      ;DRIVES 4 & 5
1299 036026 005022          CLR      (R2)+      ;DRIVES 6 & 7
1300 036030 012703 000010    MOV      #8,R3      ;MOVE THE STARTING
1301 036034 012701 035570    MOV      #QSTART,R1 ;ADDRESS OF THE QUEUE INTO
1302 036040 012122          1$: MOV      (R1)+,(R2)+ ;THE QUEUE INPUT POINTER
1303 036042 005303          DEC      R3
1304 036044 001375          BNE      1$
1305 036046 012703 000010    MOV      #8,R3      ;MOVE THE STARTING ADDRESS
1306 036052 012701 035570    MOV      #QSTART,R1 ;OF THE QUEUE INTO THE
1307 036056 012122          2$: MOV      (R1)+,(R2)+ ;QUEUE OUTPUT POINTER
1308 036060 005303          DEC      R3
1309 036062 001375          BNE      2$
1310 036064 104413          RESREG
1311 036066 000207          RTS      PC      ;RESTORE R0 - R5
1312
1313          ;EMPTY THE QUEUE SPECIFIED BY R1
1314
1315          ;CALL
1316          ;
1317          ;      MOV      DRVNUM,R1      ;DRIVE NUMBER TO R1
1318          ;      JSR      PC,EMPTYQ
1319 036070 105061 035520    EMPTYQ: CLRB    QCNT(R1)      ;CLEAR NUMBER OF ITEMS IN QUEUE
1320 036074 006301          ASL      R1
1321 036076 016161 035530 035550    MOV      QINPT(R1),QOUTPT(R1) ;SET OUTPUT QUEUE POINTER INPUT POINTER
1322 036104 006201          ASR      R1
1323 036106 000207          RTS      PC
1324
1325          ;ROUTINE TO PUT A REQUEST IN QUEUE
1326
1327          ;CALL
1328          ;
1329          ;      MOV      #DRVNUM,R1      ;DRIVE NUMBER
1330          ;      MOV      #DPB,R2      ;ADDRESS OF PARAMETER BLOCK
1331          ;      JSR      R0,DRVQUE      ;GO PUT REQUEST IN QUEUE
1332          ;      RETURN1      ;RETURN HERE IF QUEUE IS FULL
1333          ;      RETURN2      ;RETURN HERE IF REQUEST IS IN QUEUE
1334 036110 122761 000010 035520    DRVQUE: CMPB    #10,QCNT(R1) ;IS QUEUE FULL?
1335 036116 001421          BEQ      2$      ;BR IF YES-TAKE RETURN1
1336 036120 105261 035520          INCB    QCNT(R1)      ;INCREMENT QUEUE COUNT
1337 036124 006301          ASL      R1
1338 036126 010271 035530          MOV      R2,QINPT(R1) ;PUT THIS REQUEST IN QUEUE
1339 036132 062761 000002 035530          ADD      #2,QINPT(R1) ;UPDATE THE QUEUE POINTER
1340 036140 026161 035530 035572    CMP      QINPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER
1341 036146 001003          BNE      1$      ;BR IF NO
1342 036150 016161 035570 035530          MOV      QSTART(R1),QINPT(R1) ;YES--RESET POINTER
1343 036156 006201          1$: ASR      R1
1344 036160 005720          TST      (R0)+      ;TAKE RETURN 2
1345 036162 000200          2$: RTS      R0      ;RETURN TO USER
1346
1347          ;ROUTINE TO GET THE 'DPB' ADDRESS OF NEXT REQUEST IN QUEUE
1348
1349          ;CALL
1350          ;
1351          ;      MOV      #DRVNUM,R1      ;DRIVE NUMBER TO R1
1352          ;      JSR      PC,GETREQ      ;GO GET THE REQUEST
1353          ;      RETURN      ;R2='DPB' ADDRESS OF THE REQUEST
1354          ;      ;R2=0 IF NO REQUEST IN QUEUE

```

1355	036164	005002		GETREQ:	CLR	R2	
1356	036166	105761	035520		TSTB	QCNT(R1)	;IS THERE ANY REQUEST IN QUEUE?
1357	036172	001404			BEQ	2\$	;NO
1358	036174	006301		1\$:	ASL	R1	
1359	036176	017102	035550		MOV	@QOUTPT(R1),R2	;PICKUP 'DPB' POINTER FOR THIS DRIVE
1360	036202	006201			ASR	R1	
1361	036204	000207		2\$:	RTS	PC	;RETURN TO USER
1362							
1363							;ROUTINE TO 'POP' THE REQUEST FROM QUEUE
1364							
1365							
1366					CALL		
1367					MOV	#DRVNUM,R1	;DRIVE NUMBER TO R1
1368					JSR	PC,POPQUE	;CALL TO REMOVE REQUEST
1369					RETURN		;R2=ADDRESS OF DPB REMOVED
1370	036206	105361	035520	POPQUE:	DECB	QCNT(R1)	;DECREMENT QUEUE COUNT
1371	036212	006301			ASL	R1	
1372	036214	017102	035550		MOV	@QOUTPT(R1),R2	;GET THE 'DPB' POINTER
1373	036220	005071	035550		CLR	@QOUTPT(R1)	;REMOVE DPB ADDRESS FROM THE QUEUE
1374	036224	062761	000002 035550		ADD	#2,QOUTPT(R1)	;UPDATE THE QUEUE POINTER
1375	036232	026161	035550 035572		CMP	QOUTPT(R1),QSTOP(R1)	;TIME TO RESET THE POINTER?
1376	036240	001003			BNE	1\$	;NO--BR TO EXIT
1377	036242	016161	035570 035550		MOV	QSTART(R1),QOUTPT(R1)	;YES--RESET THE POINTER
1378	036250	006201		1\$:	ASR	R1	
1379	036252	000207			RTS	PC	;RETURN TO USER
1380							
1382							;ROUTINE TO SAVE THE CONTENTS OF '\$ESCAPE' WHEN THE DRIVER
1383							;REPORTS AN ERROR DIRECTLY.
1384							
1385					CALL		
1386					JSR	RO,ES.SAV	
1387					ERROR	N	;:THE ERROR CALL
1388					RETURN		;THE RETURN IS PAST THE ERROR CALL
1389							
1390	036254	012037	036270	ES.SAV:	MOV	(RO)+,1\$	;GET THE ERROR CALL
1391	036260	013746	001200		MOV	\$ESCAPE,-(SP)	;SAVE THE ADDRESS IN '\$ESCAPE'
1392	036264	005037	001200		CLR	\$ESCAPE	;CLEAR THE ESCAPE RETURN
1393	036270	000000		1\$:	.WORD	0	;THE ERROR CALL IS MOVED HERE
1394	036272	012637	001200		MOV	(SP)+,\$ESCAPE	;RESTORE THE ESCAPE ADDRESS
1395	036276	000200			RTS	RO	;RETURN

```

1      .SBTTL  BUSADR - GET BUS ADDRESS AND VECTOR ADDRESS
2
3      ;THIS ROUTINE IS USED TO INSURE THE BUS ADDRESS
4      ;OF THE RH/RM IS SETUP FOR THE PROPER ADDRESS.
5      ;IT WILL ALSO READ THE ADDRESS FROM THE TTY IF
6      ;REQUIRED.
7      ;NOTE: THIS ROUTINE DESTROYS R0-R4
8      ;CALL
9
10     :
11     :      JSR      PC,BUSADR
12     :      RETURN
13
14     036300 104412 001276 BUSADR: SAVREG      ;SAVE ALL REG
15     036302 012700 036450 1$:  MOV      #SRMADR,R0  ;FIRST ADDRESS
16     036306 104401 036450      TYPE      ,MRMCS1    ;'RMCS1='
17     036312 011046      MOV      (R0),-(SP)  ;PRESENT RMCS1 ADDRESS
18     036314 104402      TYPOC      ;TYPE IT
19     036316 104401 042752      TYPE      ,BLNKS1    ;TYPE 1 BLANK
20     036322 104411      RDLIN      ;GET THE ENTRY
21     036324 012601      MOV      (SP)+,R1      ;ADDRESS OF ASCII TEXT
22     036326 004537 036466      JSR      R5,CK.NUM  ;ENTER AND STORE THE NEW ADDRESS
23     036332 000763      BR        1$          ;ERROR EXIT
24     036334 012700 001300 2$:  MOV      #SRMVEC,R0  ;VECTOR ADDRESS
25     036340 104401 036457      TYPE      ,MRMVEC    ;'RMVEC='
26     036344 011046      MOV      (R0),-(SP)  ;PRESENT RH/RM VECTOR ADDRESS ON THE STACK
27     036346 104402      TYPOC      ;TYPE IT
28     036350 104401 042752      TYPE      ,BLNKS1    ;TYPE 1 BLANK
29     036354 104411      RDLIN      ;READ THE ENTRY
30     036356 012601      MOV      (SP)+,R1      ;ASCII TEXT ADDRESS
31     036360 004537 036466      JSR      R5,CK.NUM  ;ENTER AND STORE NEW ADDRESS
32     036364 000763      BR        2$          ;ERROR EXIT
33     036366 012700 001276 3$:  MOV      #SRMADR,R0  ;FIRST ADDRESS OF NEW PARAMETERS
34     036372 012701 030324      MOV      #RMADR,R1   ;FIRST ADDRESS OF WHERE TO PUT THEM
35     036376 013705 000004      MOV      @ERRVEC,R5  ;SAVE ERROR VECTOR
36     036402 012737 036426 000004      MOV      #4$,@ERRVEC ;LOAD NEW VECTOR ADDRESS
37     036410 005777 142662      TST      @SRMADR    ;LEGAL I/O ADDRESS ?
38     036414 010537 000004      MOV      R5,@ERRVEC  ;YES,IF NOT TRAP TO TIMEOUT
39     036420 012021      MOV      (R0)+,(R1)+      ;LOAD THE BUS ADDRESS
40     036422 012021      MOV      (R0)+,(R1)+      ;LOAD VECTOR ADDRESS
41     036424 000407      BR        6$          ;COMMON EXIT
42     036426 012716 036434 4$:  MOV      #5$, (SP)  ;SET RETURN ADDRESS
43     036432 000002      RTI          ;RETRUN FROM TIME OUT TRAP
44     036434 104006      5$:
45     036436 010537 000004      EMT      6
46     036442 000717      MOV      R5,@ERRVEC      ;RESTORE THE TIME OUT TRAP
47     036444 104413      BR        1$          ;TRY AGAIN
48     036446 000207      RESREG      ;RESTORE ALL REG
49     036450 122 115 103 MRMCS1: .ASCII @RMCS1=@
50     036457 122 115 126 MRMVEC: .ASCII @RMVEC=@
51
52     .SBTTL  CK.NUM - CHECK NUMBER (OCTAL)
53     ;THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL CHARACTERS
54     ;AND FORMS AN OCTAL NUMBER IN R2
55     ;CALL:
56     :      MOV      #ADR,R1      ;ADDRESS OF ASCII STRING

```

57			:	JSR	R5,CK.NUM	:R5 CHANGED
58			:	RET		:ERROR EXIT
59			:	RET		:NORMAL EXIT
60	036466	010246	CK.NUM:	MOV	R2,-(SP)	:SAVE R2
61	036470	010346		MOV	R3,-(SP)	:SAVE R3
62	036472	010446		MOV	R4,-(SP)	:SAVE R4
63	036474	012703	000006	MOV	#6,R3	:MAX OCTAL DIGITS IN THE NUMBER
64	036500	005002		CLR	R2	:FINAL OCTAL VALUE
65	036502	112104	1\$:	MOVB	(R1)+,R4	:GET CURRENT POINTED BYTE
66	036504	001424		BEQ	3\$	:BRANCH,IF TERMINATOR DETECTED
67	036506	120427	000060	CMPB	R4,#'0	:SMALLER THAN ASCII-0 ?
68	036512	103425		BLO	5\$	:YES,ERROR EXIT
69	036514	120427	000067	CMPB	R4,#'7	:LARGER THAN ASCII-7 ?
70	036520	101022		BHI	5\$	:YES,ERROR EXIT
71	036522	006302		ASL	R2	:SHIFT LEFT
72	036524	103420		BCS	5\$	:
73	036526	006302		ASL	R2	:ONE
74	036530	103416		BCS	5\$	:
75	036532	006302		ASL	R2	:OCTAL DIGIT
76	036534	103414		BCS	5\$	:ERROR IF CARRY BIT SET
77	036536	042704	177770	BIC	#177770,R4	:CHOP OFF HIGHER BITS
78	036542	060402		ADD	R4,R2	:APPENDING CURRENT DIGIT TO NUMBER
79	036544	005303		DEC	R3	:DECREMENT BYTE COUNT
80	036546	001401		BEQ	2\$	:BRANCH,IF LAST BYTE
81	036550	000754		BR	1\$	:LOOPING BACK
82	036552	112104	2\$:	MOVB	(R1)+,R4	:CHECK TERMINATOR
83	036554	001004		BNE	5\$	:ERROR EXIT
84	036556	005702	3\$:	TST	R2	:FINAL VALUE = 0
85	036560	001402		BEQ	5\$	:YES, TAKE ERROR EXIT
86	036562	010210		MOV	R2,(R0)	:REPLACE THE ORIGINAL VALUE
87	036564	005725	4\$:	TST	(R5)+	:ADJUST FOR NORMAL RETURN
88	036566	012604	5\$:	MOV	(SP)+,R4	:RESTORE R4
89	036570	012603		MOV	(SP)+,R3	:RESTORE R3
90	036572	012602		MOV	(SP)+,R2	:RESTORE R2
91	036574	000205		RTS	R5	:EXIT

```
1          .SBTTL  ERROR MESSAGES
2
3 036576    122    110    040  EM1:   .ASCIIZ  /RH INTERRUPT OCCURRED (RMAS=0)/
4 036635    125    116    105  EM2:   .ASCIIZ  /UNEXPECTED ATTENTION OCCURRED/
5 036673    115    101    123  EM3:   .ASCIIZ  /MASSBUS PARITY ERROR (MCPE=1)/
6 036731    115    101    123  EM4:   .ASCIIZ  /MASSBUS PARITY ERROR (PAR=1)/
7 036766    101    104    104  EM5:   .ASCIIZ  /ADDRESS PLUG CHANGE BIT SET/
8 037022    122    110    040  EM6:   .ASCIIZ  /RH DIDN'T RESPOND TO ADDRESSING/
9 037062    125    116    103  EM10:  .ASCIIZ  /UNCORRECTABLE MASSBUS PARITY ERROR/
10 037125    106    101    124  EM11:  .ASCIIZ  /FATAL MASSBUS PARITY ERROR/
11 037160    120    105    122  EM12:  .ASCIIZ  /PERSISTENT DEVICE UNSAFE/
12 037211    117    120    105  EM13:  .ASCIIZ  /OPERATION NOT COMPLETED WITHIN TIME LIMIT/
13 037263    104    122    111  EM14:  .ASCIIZ  /DRIVE WENT OFFLINE/
14 037306    116    117    040  EM15:  .ASCIIZ  /NO RESPONSE TO PORT REQUEST/
15 037342    110    105    101  EM20:  .ASCIIZ  /HEADER CRC ERROR/
16 037363    104    101    124  EM21:  .ASCIIZ  /DATA CHECK ('DCK') ERROR/
17 037414    127    122    111  EM22:  .ASCIIZ  /WRITE CHECK ERROR - DATA CHECK ('DCK') SET/
18 037467    127    122    111  EM23:  .ASCIIZ  /WRITE CHECK ERROR - DATA CHECK ('DCK') NOT SET/
19 037546    110    105    101  EM24:  .ASCIIZ  /HEADER READ ERROR - 'FMT' BIT DROPPED/
20 037614    110    105    101  EM25:  .ASCIIZ  /HEADER READ ERROR - HEADER COMPARE ('HCE') ERROR/
21 037675    106    117    122  EM26:  .ASCIIZ  /FORMAT ERROR ('FER')/
22 037722    110    105    101  EM27:  .ASCIIZ  /HEADER COMPARE ('HCE') ERROR/
23 037757    115    111    123  EM30:  .ASCIIZ  /MISCELLANEOUS DRIVE ERROR/
24 040011    117    120    105  EM31:  .ASCIIZ  /OPERATION INCOMPLETE ('OPI') ERROR/
25 040054    104    122    111  EM32:  .ASCIIZ  /DRIVE TIMING ('DTE') ERROR/
26 040107    120    101    122  EM33:  .ASCIIZ  /PARITY ('PAR') ERROR AFTER OPERATION STARTED/
27 040164    127    122    111  EM34:  .ASCIIZ  /WRITE CLOCK FAILURE ('WCF') ERROR/
28 040226    111    116    126  EM35:  .ASCIIZ  /INVALID ADDRESS ('IAE') ERROR/
29 040264    127    122    111  EM36:  .ASCIIZ  /WRITE LOCK ('WLE') ERROR/
30 040315    104    101    124  EM37:  .ASCIIZ  /DATA CHECK ('DCK') SET DURING WRITE CHECK COMMAND/
31 040377    122    110    040  EM40:  .ASCIIZ  /RH CONTROLLER OR UNIBUS TRANSFER ERROR/
32 040446    102    125    123  EM41:  .ASCIIZ  /BUS ADDRESS OR WORD COUNT INCORRECT/
33 040512    104    101    124  EM42:  .ASCIIZ  /DATA COMPARE ERRORS - NO OTHER ERROR(S) DETECTED/
34 040573    103    101    116  EM43:  .ASCIIZ  /CAN'T MATCH DATA READ WITH A PATTERN/
35 040640    105    122    122  EM44:  .ASCIIZ  /ERROR BIT(S) SET, BUT NO ERROR SIGNALLED BY THE RH/
36 040722    105    103    103  EM45:  .ASCIIZ  /ECC LOGIC FAILURE - POSITION REGISTER VALUE NOT VALID/
37 041010    102    125    123  EM46:  .ASCIIZ  /BUS ADDRESS AND WORD COUNT NOT CONSISTENT/
38 041062    123    105    105  EM50:  .ASCIIZ  /SEEK INCOMPLETE ('SKI') ERROR/
39 041120    120    122    117  EM51:  .ASCIIZ  /PROGRAM DETECTED POSITIONING ERROR/
40 041163    104    122    111  EM60:  .ASCIIZ  /DRIVE UNSAFE ERROR/
41 041206    040    040    000  EM61:  .ASCIIZ  / /
42          .EVEN
```

[illegible]

```
1 041776      123      125      102  MSG1:  .ASCIIZ  /SUBSYS /
2 042006      104      122      111  MSG2:  .ASCIIZ  /DRIVE(S): /
3 042021      052      052      040  MSG3:  .ASCIIZ  /** STARTING PASS 1 /
4 042045      200      115      117  MSG4:  .ASCIIZ  <CRLF>/MOUNT PACK ON DRIVE /
5 042073      101      116      104  MSG8:  .ASCIIZ  /AND LOAD. /
6 042106      124      131      120  MSG9:  .ASCIIZ  /TYPE <CR> WHEN DRIVE READY./
7 042142      200      104      122  MSG10: .ASCIIZ  <CRLF>/DRIVE IS NOT READY, TEST IS ABORTED/
8 042207      200      120      101  MSG11: .ASCIIZ  <CRLF>/PACK IS NOT ACCEPTABLE, CHANGE PACK AND TRY AGAIN/
9 042272      125      116      114  MSG12: .ASCIIZ  /UNLOAD DRIVE /
10 042310      040      101      116  MSG13: .ASCII  / AND REMOVE PACK/<CRLF>
11 042331      124      131      120  .ASCIIZ  /TYPE <CR> WHEN DONE./
12 042356      052      052      040  MSG14: .ASCIIZ  /** STARTING PASS 2 /
13 042402      052      052      040  MSG15: .ASCIIZ  /** ALL DRIVES ARE COMPATIBLE **/<CRLF>
14 042443      123      103      117  MSG16: .ASCIIZ  /SCORES FOR DRIVE /
15 042465      040      040      040  MSG17: .ASCIIZ  / TRACK DRIVE OVRWRT OVRWRT READ READ/
16 042546      040      040      040  MSG18: .ASCIIZ  / NUMBR READ OFST- OFST+ OFST- OFST+/
17 042630      123      105      114  SELF:  .ASCIIZ  /SELF/
18 042635      011      000      TAB:  .ASCIIZ  <HT>
19 042637      052      040      060  MARKX: .ASCIIZ  /* 0/<HT>
20 042644      042      000      QUOTM: .ASCIIZ  /'/
21 042646      054      040      000  COMMA: .ASCIIZ  /./
22 042651      127      111      114  MSG5:  .ASCIIZ  /WILL TEST /
23 042664      117      116      040  MSG6:  .ASCIIZ  /ON /
24 042670      200      111      123  MSG7:  .ASCIIZ  <CRLF>/IS THERE ANOTHER SUB-SYSTEM (Y OR N) ? /
25 042741      131      200      000  Y:  .ASCIIZ  /Y/<CRLF>
26 042744      116      200      000  N:  .ASCIIZ  /N/<CRLF>
27 042747      040      BLNKS4: .ASCII  / /
28 042750      040      BLNKS3: .ASCII  / /
29 042751      040      BLNKS2: .ASCII  / /
30 042752      040      000      BLNKS1: .ASCIIZ  / /
31 042754      072      000      COLON: .ASCIIZ  /:/
32 042756      040      057      040  SLASH: .ASCIIZ  @ / @
33 042762      077      000      QUES:  .ASCIIZ  /?/
34 042764      040      102      101  MSG19: .ASCIIZ  / BAD SPOT FILE /
35 043004      124      105      123  MSG20: .ASCIIZ  /TEST COMPLETE/
36 043022      200      052      052  STARS: .ASCIIZ  <CRLF>/*****/<CRLF>
37 043044      104      122      111  MSG21: .ASCII  /DRIVE NOT ON-LINE OR NOT ASSIGNED/<CRLF>
41 043106      007      120      122  HALTX: .ASCIIZ  <07>/PROGRAM HALT/<CRLF>
42 043125      104      122      111  HALT1: .ASCIIZ  /DRIVE FAILED ON BASIC READ & WRITE TEST/<CRLF>
43 043176      104      122      111  HALT2: .ASCIIZ  /DRIVE FAILED ON WRITE TEST (TEST 4)/<CRLF>
44 043243      106      101      124  XFATL: .ASCIIZ  /FATAL ERROR/
45 043257      200      040      047  NEDCLK: .ASCIIZ  <CRLF>/ 'L' OR 'P' CLOCK REQUIRED ON SYSTEM/<CRLF>
46 043326      040      077      104  QDRIV: .ASCIIZ  / ?DRIVE/
47 043336      040      11      123  LODEV: .ASCIIZ  / IS LOAD DEVICE/
48 043356      040      077      103  NOTST: .ASCIIZ  @ ?CANNOT SELECT RM03/2'S AND RM05'S TOGETHER (NOT COMPATIBLE)@<CRLF>
49
50      .EVEN
51
52 043456      BUFFER:
53
54      000200      .END      200
```

ABASE = 176700	AJSWR = 000000	CKSWR = 104407	DRIV4 = 004732	EM43 = 040573
ACDW1 = 000000	AJECT1 = 000254	CK.CHR = 022464	DRIV5 = 004754	EM44 = 040640
ACDW2 = 000000	AJECT2 = 000000	CK.DEC = 022436	DRIV6 = 004776	EM45 = 040722
ACK = 000123	BADSEC = 001342	CK.DIG = 022536	DRIV7 = 005020	EM46 = 041010
ACPUOP = 000000	BADTMO = 005502	CK.NUM = 036466	DRVACT = 030176	EM5 = 036766
ACTDRV = 030252	BIT0 = 000001	CK.OCT = 022410	DRVCLR = 000111	EM50 = 041062
ACTSTR = 030253	BIT00 = 000001	CLKFLG = 001316	DRIVER = 020350	EM51 = 041120
ADDW0 = 000000	BIT01 = 000002	CLOCK = 021526	DRVINT = 030546	EM6 = 037022
ADDW1 = 000000	BIT02 = 000004	CLRDPB = 021720	DRVQUE = 036110	EM60 = 041163
ADDW10 = 000000	BIT03 = 000010	CLRQUE = 036012	DRVSTA = 030206	EM61 = 041206
ADDW11 = 000000	BIT04 = 000020	CMCNT = 001354	DRVSTP = 030216	ENDX1 = 011004
ADDW12 = 000000	BIT05 = 000040	CMCYL = 001356	DSWR = 177570	ERPRC1 = 017562
ADDW13 = 000000	BIT06 = 000100	CMSEC = 001362	DTEER = 020420	ERPROC = 017546
ADDW14 = 000000	BIT07 = 000200	CMTRK = 001364	DTUW = 030310	ERROR = 104000
ADDW15 = 000000	BIT08 = 000400	COLON = 042754	DTYP = 001402	ERRVEC = 000004
ADDW2 = 000000	BIT09 = 001000	COMMA = 042646	DT1 = 041634	ES.SAV = 036254
ADDW3 = 000000	BIT1 = 000002	CPSAVE = 024252	DT10 = 041720	FALPAR = 017664
ADDW4 = 000000	BIT10 = 002000	CR = 000015	DT11 = 041734	FAULT = 001376
ADDW5 = 000000	BIT11 = 004000	CRLF = 000200	DT12 = 041750	FILBUF = 021042
ADDW6 = 000000	BIT12 = 010000	CYLIMT = 001374	DT2 = 041640	FINISH = 023620
ADDW7 = 000000	BIT13 = 020000	CYLNDR = 004616	DT3 = 041656	FMTDPB = 004506
ADDW8 = 000000	BIT14 = 040000	DCKER = 020300	DT4 = 041666	FMTER = 020460
ADDW9 = 000000	BIT15 = 100000	DDISP = 177570	DT6 = 041700	GENDPB = 004576
ADEVCT = 000000	BIT2 = 000004	DEASG = 017070	DT7 = 041704	GETREG = 000141
ADEVN = 000000	BIT3 = 000010	DF1 = 041760	DUMP = 020540	GETREQ = 036164
AENV = 000000	BIT4 = 000020	DF2 = 041761	DUMP2 = 017720	GTSWR = 104406
AENVN = 000000	BIT5 = 000040	DF3 = 041767	EMPTYQ = 036070	HALTX = 043106
AFATAL = 000000	BIT6 = 000100	DF4 = 041772	EMTVEC = 000030	HALT1 = 043125
AMADR1 = 000000	BIT7 = 000200	DH1 = 041212	EM1 = 036576	HALT2 = 043176
AMADR2 = 000000	BIT8 = 000400	DH10 = 041460	EM10 = 037062	HCEER = 020470
AMADR3 = 000000	BIT9 = 001000	DH11 = 041531	EM11 = 037125	HCR CER = 020340
AMADR4 = 000000	BLKADR = 004444	DH12 = 041602	EM12 = 037160	HOOR = 001344
AMAMS1 = 000000	BLNKS1 = 042752	DH2 = 041221	EM13 = 037211	HT = 000011
AMAMS2 = 000000	BLNKS2 = 042751	DH3 = 041303	EM14 = 037263	HZ = 001320
AMAMS3 = 000000	BLNKS3 = 042750	DH4 = 041334	EM15 = 037306	IAEER = 020440
AMAMS4 = 000000	BLNKS4 = 042747	DH6 = 041376	EM2 = 036635	IBSAVE = 024254
AMSGAD = 000000	BPTVEC = 000014	DH7 = 041407	EM20 = 037342	IDLEX = 017444
AMSGLG = 000000	BUFFER = 043456	DISMNT = 017016	EM21 = 037363	INDST = 004424
AMSGTY = 000000	BUSADR = 036300	DISPLA = 001156	EM22 = 037414	IOTVEC = 000020
AMTYP1 = 000000	CFLAG = 001336	DISPLY = 104414	EM23 = 037467	ISR = 032762
AMTYP2 = 000000	CHGADR = 001334	DISPRE = 000174	EM24 = 037546	KPATH = 016530
AMTYP3 = 000000	CI1 = 031602	DONE = 017744	EM25 = 037614	KSR = 021634
AMTYP4 = 000000	CI3 = 031710	DPINT = 030226	EM26 = 037675	LABAD = 015034
APASS = 000000	CI4 = 032016	DPRQS = 030236	EM27 = 037722	LF = 000012
APRIOR = 000000	CI5 = 032374	DRIVE = 001224	EM3 = 036673	LINDEC = 021214
APTCU = 000040	CI6 = 032416	DRIV0 = 004622	EM30 = 037757	LINE1 = 021116
APTENV = 000001	CI7 = 032432	DRIV1 = 004644	EM31 = 040011	LINKDV = 021764
APTSIZ = 000200	CI7B = 032454	DRIV10 = 005042	EM32 = 040054	LINOCT = 021162
APTSPO = 000100	CI8 = 032532	DRIV11 = 005064	EM33 = 040107	LODEV = 043336
ASNLST = 002006	CKBUS = 020640	DRIV12 = 005106	EM34 = 040164	LOG0 = 001406
ASSGN1 = 002010	CKCLK = 021234	DRIV13 = 005130	EM35 = 040226	LOG1 = 001426
ASSGN2 = 002012	CKCLK1 = 021330	DRIV14 = 005152	EM36 = 040264	LOG10 = 001646
ASWREG = 000000	CKCLK2 = 021376	DRIV15 = 005174	EM37 = 040315	LOG11 = 001666
ATABIT = 030312	CKCLK3 = 021420	DRIV16 = 005216	EM4 = 036731	LOG12 = 001706
ATESTN = 000000	CKERR = 020572	DRIV17 = 005240	EM40 = 040377	LOG13 = 001726
ATIN = 001326	CKFMT = 020360	DRIV2 = 004666	EM41 = 040446	LOG14 = 001746
ATNIT = 000000	CKHLE = 020370	DRIV3 = 004710	EM42 = 040512	LOG15 = 001766

LOG2	001446	N	042744	P1RQ	= 177772	RDN6	003356	RMSN	= 000030
LOG3	001466	NEDCLK	043257	P1RQVE	= 000240	RDN7	003376	RMTMR	034344
LOG4	001506	NOTST	043356	POPOUE	036206	RDN8	003416	RMVEC	030326
LOG5	001526	MULINE	001366	POSER	020400	RDN9	003436	RMWC	= 000002
LOG6	001546	OFFCOD	004504	PRINT	017152	RDP0	003676	RM.REG	004526
LOG7	001566	OFFSET=	000115	PROCES	017452	RDP1	003716	RMO5	031062
LOG8	001606	OFFST	014244	PRTBAD	020702	RDP10	004136	RNOP	= 000101
LOG9	001626	OFLIN	017704	PRTIM	017714	RDP11	004156	RSTART	001400
LOOP1	013630	OPIER	020410	PRO	= 000000	RDP12	004176	RTC	= 000117
LOOP2	013646	OPT	031354	PR1	= 000040	RDP13	004216	R6	= 0000006
LOOP3	015716	OVWNO	002056	PR2	= 000100	RDP14	004236	R7	= 0000007
LOOP4	016510	OVWN1	002076	PR3	= 000140	RDP15	004256	SAVEFG	030264
LOOP5	016722	OVWN10	002316	PR4	= 000200	RDP16	004276	SAVREG=	104412
LOOP6	016740	OVWN11	002336	PR5	= 000240	RDP17	004316	SC	033174
LISTAD	001332	OVWN12	002356	PR6	= 000300	RDP18	004336	SCOPE	= 000004
MAKEUP	014476	OVWN13	002376	PR7	= 000340	RDP2	003736	SCORE	014614
MARKX	042637	OVWN14	002416	PS	= 177776	RDP3	003756	SC11	034026
MCPEMY	030322	OVWN15	002436	PSEUDO	005322	RDP4	003776	SC12	034136
MESG1	041776	OVWN16	002456	PSW	= 177776	RDP5	004016	SC13	034206
MESG10	042142	OVWN17	002476	PUNSAF	017644	RDP6	004036	SC3	033244
MESG11	042207	OVWN18	002516	PWRVEC	000024	RDP7	004056	SC4	033250
MESG12	042272	OVWN2	002116	QCNT	035520	RDP8	004076	SC5	033262
MESG13	042310	OVWN3	002136	QDRIV	043326	RDP9	004116	SC6	033456
MESG14	042356	OVWN4	002156	QDRV0	035612	RD.ADR	035002	SC6A	033572
MESG15	042402	OVWN5	002176	QDRV1	035632	RD.RM	034756	SC7	033720
MESG16	042443	OVWN6	002216	QDRV2	035652	RD.RM1	035000	SC8	033776
MESG17	042465	OVWN7	002236	QDRV3	035672	RD.RM2	035124	SEARCH=	000131
MESG18	042546	OVWN8	002256	QDRV4	035712	RD.RM3	035130	SECLMT	001370
MESG19	042764	OVWN9	002276	QDRV5	035732	RD.RM4	035134	SECOND	001350
MESG2	042006	OVWP0	002536	QDRV6	035752	RD.WRD	035004	SEEK	= 000105
MESG20	043004	OVWP1	002556	QDRV7	035772	READIN	000121	SEEKFG	030266
MESG21	043044	OVWP10	002776	QINPT	035530	RECAL	= 000107	SELDRV=	000145
MESG3	042021	OVWP11	003016	QOUTPT	035550	RELE	= 000113	SELF	042630
MESG4	042045	OVWP12	003036	QSTART	035570	REPLZ	022210	SELF0	004356
MESG5	042651	OVWP13	003056	QSTOP	035572	RESREG=	104413	SELF1	004360
MESG6	042664	OVWP14	003076	QTERM	= 036012	RESVEC=	000010	SELF10	004402
MESG7	042670	OVWP15	003116	QUES	042762	RMADR	030324	SELF11	004404
MESG8	042073	OVWP16	003136	QUOTM	042644	RMAS	= 000016	SELF12	004406
MESG9	042106	OVWP17	003156	RDCHR	= 104410	RMB	= 000004	SELF13	004410
MINUTE	001346	OVWP18	003176	RDDAT	= 000171	RMCS1	= 000000	SELF14	004412
MOD00	007136	OVWP2	002576	RDHD	= 000173	RMCS2	= 000010	SELF15	004414
MOD11	007444	OVWP3	002616	RDLIN	= 104411	RMDA	= 000006	SELF16	004416
MOD12	007624	OVWP4	002636	RDN0	003216	RMDB	= 000022	SELF17	004420
MOD21	007162	OVWP5	002656	RDN1	003236	RMDC	= 000034	SELF18	004422
MOD22	007430	OVWP6	002676	RDN10	003456	RMD5	= 000012	SELF2	004362
MOD23	007364	OVWP7	002716	RDN11	003476	RMDT	= 000026	SELF3	004364
MOD30	007456	OVWP8	002736	RDN12	003516	RMEC1	= 000044	SELF4	004366
MONTR	007106	OVWP9	002756	RDN13	003536	RMEC2	= 000046	SELF5	004370
MRMCS1	036450	PACK	001324	RDN14	003556	RMERRS	030166	SELF6	004372
MRMVEC	036457	PARER	020430	RDN15	003576	RMER1	= 000014	SELF7	004374
MXWINDW	030332	PCLOCK	001314	RDN16	003616	RMER2	= 000042	SELF8	004376
M.DP10	022032	PFECH	024434	RDN17	003636	RMHR	= 000036	SELF9	004400
M.DP40	022070	PFECH1	024444	RDN18	003656	RMINIT	030334	SETFMT=	000143
M.DP41	022124	PFECH2	024526	RDN2	003256	RMLA	000020	SETVEC	007074
M.DP42	022134	PFECH3	024560	RDN3	003276	RMMR1	= 000024	SET.IE	035446
M.DP44	022166	PFECH4	024570	RDN4	003316	RMMR2	= 000040	SIXTEE	001352
M.DP50	022200	PFTSTN	024574	RDN5	003336	RMOF	= 000032	SIZMEM	007030

SKIER	020510	TAB.XV=	001114	\$AUTOB	001150	\$HD	= 000000	\$RMADR	001276
SLASH	042756	TBITVE=	000014	\$BASE	001266	\$HIBTS	001100	\$RMAS	= 000036
SPATH	016620	TD	033024	\$BDADR	001136	\$ICNT	001120	\$RMBA	= 000024
SPOTX	010636	TIMER	030270	\$BDDAT	001142	\$ILLUP	025552	\$RMCS1=	000020
SRCHWT	030250	TKVEC	= 000060	\$BELL	001202	\$INTAG	001151	\$RMCS2=	000030
STACK	= 001100	TPVEC	= 000064	\$BUF	= 000006	\$ITEMB	001130	\$RMDA	= 000026
STARS	043022	TRAPVE=	000034	\$CDW1	001272	\$LF	001210	\$RMDB	= 000042
STARSC	001360	TRFER	020500	\$CDW2	001274	\$LFLG	025375	\$RMDC	= 000054
START	005566	TRKLMT	001372	\$CHARC	025126	\$LKCSB	001304	\$RMD5	= 000032
START1	005604	TRNSWT	030246	\$CKSWR	025242	\$LKCSR	001302	\$RMDT	= 000046
START2	005622	TRTVEC=	000014	\$CLR.T	023534	\$LKS	001310	\$RMEC1=	000064
START3	005634	TSTNM	001340	\$CMTAG	001114	\$LLVEC	001312	\$RMEC2=	000066
STATIN	001322	TST1	011020	\$CM3	= 000000	\$LOOP	023612	\$RMER1=	000034
STKLMT=	177774	TST10	015654	\$CM4	= 000001	\$LPADR	001122	\$RMER2=	000062
STNDAT	005262	TST11	016342	\$CNTLC	023452	\$LPERR	001124	\$RMHR	= 000056
STO	034434	TST2	011374	\$CNTLG	026661	\$LPVEC	001306	\$RMLA	= 000040
STO1	034470	TST3	011650	\$CNTLU	026654	\$LSTAD	030076	\$RMMR1=	000044
STO2	034564	TST4	012274	\$COMND=	000002	\$MADR1	001244	\$RMMR2=	000060
STO3	034614	TST5	012772	\$CPUOP	001240	\$MADR2	001250	\$RMOF	= 000052
STO5	034640	TST6	013344	\$CRLF	001207	\$MADR3	001254	\$RMSN	= 000050
STO6	034646	TST7	015140	\$CYL	= 000012	\$MADR4	001260	\$RMVEC	001300
STO7	034704	TYPDS	= 104405	\$DBLK	026232	\$MAIL	001212	\$RMWC	= 000022
STO8	034734	TYPE	= 104401	\$DB2D	027004	\$MAMS1	001242	\$RTNAD	023614
STO9	034744	TYPOC	= 104402	\$DB20	027200	\$MAMS2	001246	\$RTRN	023610
SVRH70	035330	TYPON	= 104404	\$DECVL	027164	\$MAMS3	001252	\$SAVRE	026710
SWR	001154	TYPOS	= 104403	\$DEVCT	001222	\$MAMS4	001256	\$SAVR6	025556
SWREG	000176	TYPRI4	022340	\$DEVM	001270	\$MBADR	001102	\$SB2D	022674
SWTIM	017674	UCPAR	017654	\$DOAGN	023554	\$MFLG	025374	\$SB20	022724
SWO	= 000001	ULDFLG	030254	\$DSPLY	022366	\$MNEW	026677	\$SCOPE	027320
SWOU	= 000001	UNIT	001330	\$DTBL	026222	\$MSGAD	001226	\$SEC	= 000010
SW01	= 000002	UNSAF	020530	\$EMTAB=	000022	\$MSGLG	001230	\$SETUP=	000137
SW02	= 000004	WCFER	020520	\$ENDAD	023544	\$MSGTY	001212	\$SIZE	027746
SW03	= 000010	WCKD	= 000151	\$ENDCT	023512	\$MSWR	026666	\$STUP	= 177777
SW04	= 000020	WCKER	020310	\$ENV	001232	\$MTYP1	001243	\$SUPRS	022300
SW05	= 000040	WCKHD	= 000153	\$ENVM	001233	\$MTYP2	001247	\$SVLAD	027674
SW06	= 000100	WLEER	020450	\$EOP	023456	\$MTYP3	001253	\$SVPC	= 000214
SW07	= 000200	WRTDAT=	000161	\$EOPCT	023504	\$MTYP4	001257	\$SWR	= 177600
SW08	= 000400	WRTHD	= 000163	\$ERFLG	001117	\$MXCNT	027744	\$SWREG	001234
SW09	= 001000	WRT.AD	035226	\$ERMAX	001131	\$NULL	001170	\$SWRMK=	000000
SW1	= 000002	WRT.RM	035136	\$ERROR	023662	\$NWTST=	000000	\$SYSNM=	000014
SW10	= 002000	WRT.R1	035222	\$ERRPC	001132	\$OCNT	026012	\$STATUS=	000016
SW11	= 004000	WRT.R2	035314	\$ERRTB	005362	\$OCTVL	027302	\$TBIT	023616
SW12	= 010000	WRT.R3	035320	\$ERRTY	024256	\$OMODE	026014	\$TERM	= 000032
SW13	= 020000	WRT.R4	035324	\$ERTTL	001126	\$OVER	027730	\$TESTN	001216
SW14	= 040000	WRT.R5	035326	\$ESCAP	001200	\$PASS	001220	\$TIME	021430
SW15	= 100000	WRT.WD	035224	\$ETABL	001232	\$PASTM	001106	\$TIMES	001176
SW2	= 000004	XEND2	017146	\$ETEND	001276	\$PHYDR=	000015	\$TKB	001162
SW3	= 000010	XFATL	043243	\$FATAL	001214	\$POWER	025560	\$TKINT	022754
SW4	= 000020	XPASS1	010054	\$FFLG	025376	\$PSEL	= 000003	\$TKS	001160
SW5	= 000040	XPASS2	012460	\$FILLC	001172	\$PWRDN	025400	\$TKSRV	023004
SW6	= 000100	XXDP	001404	\$FILLS	001171	\$PWRMG	025534	\$TMPO	001174
SW7	= 000200	Y	042741	\$FMT	= 000001	\$PWRUP	025452	\$TN	= 000012
SW8	= 000400	\$APTHD	001100	\$GAP	= 000016	\$QUES	001206	\$TNPWR	027114
SW9	= 001000	\$ATYC	025156	\$GDADR	001134	\$RDCHR	026524	\$TPB	001166
SYSADR	002014	\$ATY1	025132	\$GDDAT	001140	\$RDLIN	023150	\$TPFLG	001173
TAB	042635	\$ATY3	025140	\$GET42	023516	\$RDSZ	= 000001	\$TPS	001164
TABEX	002054	\$ATY4	025150	\$GTSWR	026312	\$RESRE	026746	\$TRAP	030100

\$TRAP2	030122	\$TTYIN	023426	\$TYPOC	025614	\$USWR	001236	\$XON	= 000021
\$TRK	= 000011	\$TYPDS	026016	\$TYPON	025630	\$VECT1	001262	\$XTSTR	027336
\$TRP	= 000015	\$TYPE	024576	\$TYPOS	025570	\$VECT2	001264	\$SGET4	= 000001
\$TRPAD	030134	\$TYPEC	025010	\$UNIT	001224	\$WRDM	= 000004	\$OFILL	026013
\$TSTM	001104	\$TYPEX	025130	\$UNITM	001110	\$XOFF	= 000023	.\$x	= 001100
\$TSTM	001116								

. ABS. 043456 000
000000 001

ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 60416 WORDS (236 PAGES)
DYNAMIC MEMORY AVAILABLE FOR 70 PAGES
CZRMTB.BIN,CZRMTB/C=CZRMTB.DOC,CZRMTB,SYSMAC/M

\$SET4	10-1	10-1#	10-1#												
\$OFILL	16-1	16-1#	16-1*	16-1*											
\$LOCAT	11-1	22-1													
\$APTHD	5-11	5-11#													
\$ASTAT	14-1	14-1													
\$ATY1	14-1#														
\$ATY3	13-1	14-1#													
\$ATY4	11-1	14-1#													
\$ATYC	14-1	14-1#													
\$AUTOB	6-0#	9-36*	18-1	18-1	18-1										
\$BASE	6-0#														
\$3DADR	6-0#														
\$BDDAT	6-0#														
\$BELL	6-0#	9-C93	11-1	11-1	11-1										
\$BUF	7-0#	9-351*	9-563*	9-624*	9-630	9-912*	9-917	9-956*	9-985*	9-:64	9-<44*	9-=36*	9-C26	9-C69	
\$CDW1	6-0#	9-184*	9-188	9-303*	9-307	9-484	9-486	9-558	9-718*	9-746*	9-750	9-810	9-812	9-906	
	9-?63*														
\$CDW2	6-0#	9-140*	9-147	9-178	9-223	9-253	9-268	9-271*	9-272	9-304*	9-315	9-320	9-487	9-489*	
	9-497	9-502	9-516	9-700	9-747*	9-759	9-764	9-813	9-815*	9-818	9-830	9-835	9-?39	9-?95	
\$CHARC	13-1	13-1#	13-1*	13-1*	13-1*										
\$CKSWR	18-1#	24-1	24-1												
\$CLR.F	10-1	10-1#													
\$CM3	6-0	6-0#													
\$CM4	6-0	6-0	6-0#	6-0#											
\$CMTAG	6-0#	9-31	9-31	9-31	9-31	9-31	9-31	9-31							
\$CNTLC	9-H54	9-I16	9-I41#												
\$CNTLG	9-H66	18-1	18-1#												
\$CNTLU	9-I11	18-1	18-1#												
\$COMND	7-0	7-0	7-0	7-0	7-0	7-0	7-0	7-0	7-0	7-0	7-0	7-0	7-0	7-0	
	7-0	7-0	7-0#	9-357*	9-566*	9-577*	9-587*	9-623*	9-910*	9-955*	9-:46	9-:48*	9-:70*	9-:79*	
	9-:13*	9-<38*	9-<69*	9-=35*											
\$CPUOP	6-0#														
\$CRLF	6-0#	9-56	9-144	9-274	9-310	9-318	9-327	9-369	9-397	9-463	9-492	9-500	9-509	9-538	
	9-542	9-592	9-682	9-698	9-754	9-762	9-771	9-793	9-825	9-833	9-842	9-864	9-868	9-?30	
	9-?35	9-?93	9-200	9-201	9-203	9-205	9-210	9-A19	9-A21	9-A77	9-C34	9-C36	9-C61	9-C63	
	9-C96	9-D72	9-H53	9-H57	9-H65	9-I21	10-5	10-7	10-10	11-1	11-1	11-1	12-1	12-1	
\$CYL	12-1	12-1	13-1	13-1	13-1	18-1									
	7-0#	9-354*	9-561*	9-646*	9-666*	9-670*	9-671	9-911*	9-942*	9-943	9-953*	9-:13*	9-:14	9-:17*	
	9-:61	9-<43*	9-<63*	9-<64	9-<74*	9-=09*	9-=10	9-=41*	9-=95*	9-=96	9->02*				
\$DB2D	9-H12	20-1#													
\$DB2O	9-H29	21-1#													
\$DBLK	17-1	17-1	17-1#												
\$DECVL	20-1	20-1#													
\$DEVCT	6-0#														
\$DEVMM	6-0#	9-302*	9-560	9-614	9-627	9-708	9-720*	9-745*	9-911	9-914	9-953	9-:17	9-:64	9-<41	
	9-<72	9-=39	9->00	9->63	9-?52	9-?60*	9-?61	9-213							
\$DOAGN	10-1	10-1	10-1	10-1#											
\$DSPLY	9-F81#	24-2													
\$DTBL	17-1	17-1#													
\$EMTAB	7-0	7-0	7-0	7-0	7-0	7-0	7-0	7-0	7-0	7-0	7-0	7-0	7-0	7-0	
	7-0	7-0	7-0#	9-E45											
\$ENDAD	5-8	9-36	10-1#	11-1											
\$ENDCT	9-31	10-1#													
\$ENV	6-0#	9-36	11-1	13-1	14-1	14-1									
\$ENVMM	6-0#	9-31	13-1	13-1	14-1										
\$EOP	9-?66	10-1#													

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

[illegible]

CROSS REFERENCE	TABLE (REF)	VOL-05
\$TYPOC	16-1#	24-1
\$TYPON	16-1	16-1#
\$TYPOS	16-1#	24-1
\$UNIT	6-0#	7-0
\$UNITM	5-11#	
\$USWR	6-0#	
\$VECT1	6-0#	
\$VECT2	6-0#	
\$WRDM	7-0#	
	9-93*	9-353*
	9-C23	9-562*
		9-626*
		9-908*
		9-976*
		9-983*
		9-:11*
		9-:63
		9-<11
		9-<37*
		9--33*
		9--65*
		9--68*
\$XOFF	13-1	13-1
\$XON	13-1	13-1
\$XTSTR	22-1#	18-1
.SASTA	14-1	14-1
.SX	5-11	5-11#
ABASE	4-108#	6-0
ACDW1	6-0	6-0
ACDW2	6-0	6-0
ACK	4-96#	
ACPUOP	6-0	6-0
ACTDRV	25-93#	25-347*
ACTSTR	25-99#	25-398*
		25-691*
		25-699*
		25-986
ADDW0	6-0	
ADDW1	6-0	
ADDW10	6-0	
ADDW11	6-0	
ADDW12	6-0	
ADDW13	6-0	
ADDW14	6-0	
ADDW15	6-0	
ADDW2	6-0	
ADDW3	6-0	
ADDW4	6-0	
ADDW5	6-0	
ADDW6	6-0	
ADDW7	6-0	
ADDW8	6-0	
ADDW9	6-0	
ADEVCT	6-0	6-0
ADEVM	6-0	6-0
AENV	6-0	6-0
AENVM	6-0	6-0
AFATAL	6-0	6-0
AMADR1	6-0	6-0
AMADR2	6-0	6-0
AMADR3	6-0	6-0
AMADR4	6-0	6-0
AMAMS1	6-0	6-0
AMAMS2	6-0	6-0
AMAMS3	6-0	6-0
AMAMS4	6-0	6-0
AMSGAD	6-0	6-0
AMSGLG	6-0	6-0
AMSGTY	6-0	6-0
AMTYP1	6-0	6-0
AMTYP2	6-0	6-0
AMTYP3	6-0	6-0

CROSS REFERENCE	TABLE	CREF	VOL	PAGE
AMTYP4	6-0	6-0		
APASS	6-0	6-0		
APRIOR	6-0			
APTCSU	13-1	14-1#		
APTENV	11-1	13-1	14-1	14-1#
APTSIZ	9-31	14-1#		
APTSPO	13-1	14-1	14-1#	
ASNLST	7-0#	9-75	9-135	9-210
ASSGN1	7-0#	9-225*	9-286*	9-298
ASSGN2	7-0#	9-226*	9-741	9-?58*
ASWREG	6-0	6-0		
ATABIT	9-158	9-228	9-238	25-145#
AESTN	6-0	6-0		
ATTN	7-0#	28-12	28-13	
AUNIT	6-0	6-0		
AUSWR	6-0	6-0		
AVECT1	4-109#	6-0	6-0	
AVECT2	6-0	6-0		
BADSEC	7-0#	9->55*	9-^28	9-?90
BADTMO	9-3#	9-33		9-a06*
BIT0	4-58#	9->59	9-?19	9-?51
BIT00	4-58	4-58#	11-1	11-1
BIT01	4-58	4-58#	9-a80	25-390
BIT02	4-58	4-58#		25-655
BIT03	4-58	4-58#	9-A27	25-849
BIT04	4-58	4-58#	9-A27	25-:71
BIT05	4-58	4-58#	25-865	9-A39
BIT06	4-58	4-58#	9-E30	25-885
BIT07	4-58	4-58#	9-A42	25-328
BIT08	4-58	4-58#	9-a66	25-381
BIT09	4-58	4-58#	9-A36	25-454
BIT10	4-58#	9-437	9-a78	25-734
BIT11	4-58#	9-a76	9-A54	25-849
BIT12	4-58#	9-a74	9-A57	25-865
BIT13	4-58#	9-a72	9-A65	25-885
BIT14	4-58#	9-A45	9-A71	25-927
BIT15	4-58#	9-a53	9-a55	25-965
BIT16	4-58#	9-a51	9-A73	25-972
BIT17	4-58#	9-a51	9-A95	25-985
BIT18	4-58#	9-a82	25-875	25-992
BIT19	4-58#	9-A48	25-885	25-999
BLKADR	7-0#	9-108	9-222	9-252
BLNKS1	9-149	9-259	9-325	9-329
BLNKS2	29-29#			
BLNKS3	9-a15	9-a23	29-28#	
BLNKS4	9-a36	29-?7#		
BPTVEC	4-58#			
BUFFER	7-0	9-351	9-377	9-380
BUSADR	9-191	26-13#		

CROSS REFERENCE	7-0#	9-85*	9-E20*	9-E23	9-H55*	9-I15*	9-283									
CHLAG	7-0#	9-16*	9-20*	9-24*	9-176	9-242										
CHGADR	7-0#	9-16*	9-20*	9-24*	9-176	9-242										
CI1	25-444	25-472#														
CI3	25-446	25-496#														
CI4	25-438	25-518#														
CI5	25-494	25-516	25-596#	25-606												
CI6	25-534	25-544	25-546	25-548	25-602#											
CI7	25-377	25-453	25-484	25-488	25-492	25-501	25-510	25-514	25-526	25-533	25-539	25-543	25-557	25-563		
	25-567	25-577	25-588	25-605	25-608#	25-715	25-910	25- 15								
CI7B	25-623#	25-916														
CI8	25-626	25-640#	25-769	25-:66												
CK.CHR	9-G36#	9-G73	9-G81													
CK.DEC	9-G14#	9-G42														
CK.DIG	9-G67#															
CK.NUM	26-21	26-30	26-60#													
CK.OCT	9-156	9-F96#	9-G44													
CKBUS	9-258	9-C21#														
CKCLK	9-100	9-D36#														
CKCLK1	9-D38	9-D49#														
CKCLK2	9-D50	9-D58#														
CKCLK3	9-D48	9-D57	9-D64#													
CKFRR	9-257	9-C07#														
CKFMT	9-A41	9-B16#														
CKHCE	9-A44	9-B22#														
CKSWR	11-1	11-1	22-1	24-1#												
CLKFLG	7-0#	9-D36*	9-D41*	9-D52*	9-D70											
CLOCK	9-D44	9-D54	9-D91#													
CLRDPB	9-109	9-E40#														
CLRQUE	25-215	25-679	25- 94 #													
CMCNT	7-0#	9-641*	9-672*	9-675*	9-678											
CMCYL	7-0#	9-642*	9-646	9-671*	9-674*	9-676										
CMSEC	7-0#	9-643*	9-668*	9-:41	9-:73	9- 07 *	9- 13 *	9- 15 *								
CMTRK	7-0#	9- 56 *	9- 91	9- 97	9-?11*	9-?14	9-?26	9-211								
COLON	9-D77	9-D82	29-31#													
COMMA	29-21#															
CPSAVE	11-1	11-1	11-1	11-1	11-1	11-1#	11-1*	11-1*	12-1	22-1	22-1	22-1	22-1*	22-1*		
CR	4-58#	13-1	13-1													
CRLF	4-58#	9-6	9-36	9-36	9-50	9-61	9-67	9-68	9-141	13-1						

CROSS REFERENCE TABLE (CREF V01-U3)															SEQ 0154
DISMNT	9-?29	9-?35#													
DISPLA	6-0#	9-31*	9-31*	11-1*	22-1*										
DISPLY	9-289	9-295	9-A01	9-A07	9-A13	9-A18	9-A89	9-A97	9-A99	9-B05	9-B11	9-B17	9-B23	9-B27	
	9-B32	9-B38	9-B44	9-B50	9-B56	9-B62	9-B67	9-B73	9-B79	9-B85	9-B91	9-C15	9-C29	9-C54	
	9-C61	9-C63	9-C96	9-C99	9-D12	9-F60	24-2#								
DISPRE	5-1#	9-31													
DONE	9-268	9-A27#													
DPINT	25-61#	25-244	25-247	25-357	25-789	25-816	25-955	25-957*	25-:30	25-:53	25-:59	25-:69*			
DPRQS	25-70#	25-363	25-411*	25-448*	25-628*	25-645	25-663*	25-792	25-:32	25-:55	25-:61	25-:79*			
DRIVO	7-0	7-0#													
DRIV1	7-0	7-0#													
DRIV10	7-0	7-0#													
DRIV11	7-0	7-0#													
DRIV12	7-0	7-0#													
DRIV13	7-0	7-0#													
DRIV14	7-0	7-0#													
DRIV15	7-0	7-0#													
DRIV16	7-0	7-0#													
DRIV17	7-0	7-0#													
DRIV2	7-0	7-0#													
DRIV3	7-0	7-0#													
DRIV4	7-0	7-0#													
DRIV5	7-0	7-0#													
DRIV6	7-0	7-0#													
DRIV7	7-0	7-0#													
DRIVE	7-0#	9-143*	9-158*	9-186	9-206	9-228	9-238*	9-317*	9-321	9-338	9-350	9-499*	9-503	9-526	
	9-559	9-622	9-701	9-761*	9-781	9-832*	9-852	9-907	9-:47	9-:59	9-:14	9-:16	9-:40	9-:38	
	9-740	9-796	28-13	28-14	28-15	28-17									
DRVACT	9-:18*	25-30#	25-371	25-551*	25-599*	25-630*	25-643	25-662*	25-704*	25-812	25-847	25-879*	25-887*	25-907	
	25-922*	25-:44*													
DRVCLR	4-91#														
DRVER	9-A75	9-B10#													
DRVINT	25-235	25-269#	25-359	25-944	25-958										
DRVQUE	25-366	25-379	25-:34#												
DRVSTA	9-339	9-527	9-782	9-853	25-40#	25-225*	25-226*	25-227*	25-228*	25-238*	25-270*	25-280*	25-324*	25-330*	
	25-353	25-361	25-385	25-419	25-423	25-552*	25-683*	25-795	25-803	25-818	25-873*	25-880*	25-896	25-960	
	25-:70*														
DRVTyp	9-341	9-343	9-345	9-529	9-533	9-536	9-784	9-786	9-788	9-855	9-859	9-862	25-52#	25-271*	
	25-287*	25-292*	25-297*	25-302*	25-388	25-684*									
DSWR	4-58#	6-0	9-31												
DT1	8-6	28-12#													
DT10	8-57	28-18#													
DT11	8-64	28-19#													
DT12	8-71	28-20#													
DT2	8-13	8-34	28-13#												
DT3	8-20	28-14#													
DT4	8-27	28-15#													
DT6	8-41	28-16#													
DT7	8-50	28-17#													
DTEER	9-A67	9-B37#													
DTUW	9-:14*	25-139#	25-221	25-440	25-493*	25-637*	25-648	25-664	25-666*	25-677*	25-694	25-705*	25-:35	25-:47*	
	25-:19														
DTYP	7-0#	9-343*	9-531	9-786*	9-857										
DUMP	9-A90	9-A98	9-B00	9-B06	9-B12	9-B18	9-B24	9-B28	9-B33	9-B39	9-B45	9-B51	9-B57	9-B63	
	9-B68	9-B74	9-B80	9-B86	9-B92	9-B94#	9-C16	9-C30							
DUMP2	9-290	9-296	9-A02	9-A08	9-A14	9-A19#									
EM1	8-4	27-3#													

CROSS REFERENCE	TABLE (CREF VOL-05)																
EM10	9-a95	27-9#															
EM11	9-A01	27-10#															
EM12	9-a89	27-11#															
EM13	9-A07	27-12#															
EM14	9-A13	27-13#															
EM15	9-A18	27-14#															
EM2	8-11	27-4#															
EM20	9-B05	27-15#															
EM21	9-A89	27-16#															
EM22	9-A99	27-17#															
EM23	9-A97	27-18#															
EM24	9-B17	27-19#															
EM25	9-B23	27-20#															
EM26	9-B62	27-21#															
EM27	9-B67	27-22#															
EM3	8-18	27-5#															
EM30	9-B11	27-23#															
EM31	9-B32	27-24#															
EM32	9-B38	27-25#															
EM33	9-B44	27-26#															
EM34	9-B85	27-27#															
EM35	9-B50	27-28#															
EM36	9-B56	27-29#															
EM37	27-30#																
EM4	8-25	27-6#															
EM40	9-B73	27-31#															
EM41	9-C29	27-32#															
EM42	27-33#																
EM43	27-34#																
EM44	9-C15	27-35#															
EM45	27-36#																
EM46	27-37#																
EM5	8-32	27-7#															
EM50	9-B79	27-38#															
EM51	9-B27	27-39#															
EM6	8-39	27-8#															
EM60	9-B91	27-40#															
EM61	27-41#																
EMPTYQ	25-627	25-682	25-735	25-:84	25-=19#												
EMIVEC	4-58#	9-31*	9-31*														
ENDX1	9-300	9-371	9-463#	9-795													
ERPRC1	9-a67	9-a72#															
ERPROC	9-a50	9-a54	9-a56	9-a66#													

[illegible]

NAME	9-E93	9-E98#	9-F10#	9-F14#	9-E71#	9-05#	9-<88	9-=72	9-495	9-757	9-828	29-1#	9-869	29-7#	9-828	29-1#
M.DP42	9-E93	9-E98#	9-F10#	9-F14#	9-E71#	9-05#	9-<88	9-=72	9-495	9-757	9-828	29-1#	9-869	29-7#	9-828	29-1#
M.DP44	9-E93	9-E98#	9-F10#	9-F14#	9-E71#	9-05#	9-<88	9-=72	9-495	9-757	9-828	29-1#	9-869	29-7#	9-828	29-1#
M.DP50	9-E80	9-F14#	9-E71#	9-05#	9-<88	9-=72	9-495	9-757	9-828	29-1#	9-869	29-7#	9-828	29-1#	9-828	29-1#
M.DPID	9-E57	9-E71#	9-05#	9-<88	9-=72	9-495	9-757	9-828	29-1#	9-869	29-7#	9-828	29-1#	9-828	29-1#	9-828
MAKEUP	9-989	9-05#	9-<88	9-=72	9-495	9-757	9-828	29-1#	9-869	29-7#	9-828	29-1#	9-828	29-1#	9-828	29-1#
MARKX	9-237	29-19#	25-:01	25-:47	9-313	9-495	9-757	9-828	29-1#	9-869	29-7#	9-828	29-1#	9-828	29-1#	9-828
MCPEMX	25-157#	25-:01	25-:47	9-313	9-495	9-757	9-828	29-1#	9-869	29-7#	9-828	29-1#	9-828	29-1#	9-828	29-1#
MSG1	9-145	9-266	9-313	9-495	9-757	9-828	29-1#	9-869	29-7#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828
MSG10	9-370	9-543	9-794	9-869	29-7#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828
MSG11	9-396	9-A78	29-8#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828
MSG12	9-699	9-738	29-9#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828
MSG13	9-705	9-744	29-10#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828
MSG14	9-755	9-826	29-12#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828
MSG15	10-6	29-13#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#
MSG16	9-794	29-14#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#
MSG17	9-202	29-15#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#
MSG18	9-204	29-16#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#
MSG19	9-335	29-34#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#
MSG2	9-150	9-247	29-2#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828
MSG20	10-8	29-35#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#	9-828	29-1#
MSG21	9-464	29-37#	9-828													

(CROSS REFERENCE TABLE) (CREV V01-05)					
OVW14	7-0#				
OVW15	7-0#				
OVW16	7-0#				
OVW17	7-0#				
OVW18	7-0#				
OVW2	7-0#				
OVW3	7-0#				
OVW4	7-0#				
OVW5	7-0#				
OVW6	7-0#				
OVW7	7-0#				
OVW8	7-0#				
OVW9	7-0#				
OVWP0	7-0#	9->53	9->69	9->87	
OVWP1	7-0#				
OVWP10	7-0#				
OVWP11	7-0#				
OVWP12	7-0#				
OVWP13	7-0#				
OVWP14	7-0#				
OVWP15	7-0#				
OVWP16	7-0#				
OVWP17	7-0#				
OVWP18	7-0#				
OVWP2	7-0#				
OVWP3	7-0#				
OVWP4	7-0#				
OVWP5	7-0#				
OVWP6	7-0#				
OVWP7	7-0#				
OVWP8	7-0#				
OVWP9	7-0#				
PACK	7-0#	9-619*	9-645		
PAPER	9-A50	9-B43#			
PCLOCK	7-0#	9-D37*	9-D42*		
PFECH	12-1	12-1#			
PFECH1	12-1	12-1#			
PFECH2	12-1	12-1#			
PFECH3	12-1	12-1#			
PFECH4	12-1	12-1#			
PFTSTN	12-1	12-1#	12-1*		
PIRQ	4-58#				
PIRQUE	4-58#				
POPQUE	25-421	25-472	25-589	25-891	25-926 25-970 25--70#
POSER	9-B27#				
PRO	4-58#				
PR1	4-58#				
PR2	4-58#				
PR3	4-58#				
PR4	4-58#	9-E19			
PR5	4-58#	9-H40			
PR6	4-58#	9-34			
PR7	4-58#				
PRINT	9->67	9->71	9->75	9->79	9->85 9->89 9->95 9-?01 9-?89#
PROCFS	9-574	9-584	9-655	9-932	9-964 9-994 9-<53 9-<82 9-<92 9- 48 9- 76 9-a48#
PRTBAD	9-390	9-401	9-C34#		
PRTJM	9-a83	9-A17#			

SEQ 0159

[illegible]

CROSS REFERENCE TABLE (CREF VOL-03)															SEP 0160	
RDP1	7-0#															
RDP10	7-0#															
RDP11	7-0#															
RDP12	7-0#															
RDP13	7-0#															
RDP14	7-0#															
RDP15	7-0#															
RDP16	7-0#															
RDP17	7-0#															
RDP18	7-0#	9--29														
RDP2	7-0#															
RDP3	7-0#															
RDP4	7-0#															
RDP5	7-0#															
RDP6	7-0#															
RDP7	7-0#															
RDP8	7-0#															
RDP9	7-0#															
READIN	4-95#															
RECAL	4-90#															
RELSE	4-92#															
REPLZ	9-D75	9-D80	9-D85	9-F25#												
RESREG	9-C82	9-E27	9-E60	20-1	21-1	24-1#	25-252	25-393	25-396	25-458	25-686	25-698	25-:01	25-<26		
	25-=10	26-46														
RESVEC	4-58#	9-31	9-31*	9-31*												
RM.REG	7-0	7-0	7-0#	9-352	9-564	9-625	9-913	9-957	9-986	9-<45	9-=37	28-17	28-17	28-17		
	28-17	28-18	28-18	28-18	28-18	28-18	28-19	28-19	28-19	28-19	28-19	28-20	28-20	28-20		
RMO5	9-359	9-568	9-578	9-649	9-926	9-958	9-:49	9-:71	9-<47	9-<76	9- 42	25-345#				
RMADR	9-73*	9-305*	9-521*	9-748*	9-823*	9-:58	9-:17	25-162#	25-232	25-352	25-475	25-496	25-518	25-693		
	25-:20	25-:03	25-:09	25-:13	25-:27	25-:60	25-:64	25-:96	25-<37	26-33	28-16					
RMA5	25-180#	25-318*	25-765	25-895*	25-931*	25-954*	25-:50									
RMB5A	9-:64*	25-175#														
RMCS1	9-:22*	25-173#	25-274*	25-281	25-307	25-416*	25-417	25-491	25-513	25-556	25-576	25-587	25-604	25-625		
	25-714	25-768	25-827	25-858	25-903	25-972	25-:55									
RMCS2	9-:60*	25-177#	25-253*	25-273*	25-275	25-365*	25-415*	25-476*	25-497*	25-519*	25-608	25-653	25-669	25-680*		
	25-712*	25-839*	25-939*	25-953*	25-:21*	25-:10	25-:65	25-:97*	25-<03	25-<38*	25-<43					
RMDA	9-:62*	25-176#	25-483	25-509	25-525											
RMDB	25-182#	25-<01														
RMDC	9-:61*	25-187#	25-487	25-500	25-532											
RMD5	25-178#	25-314	25-842	25-868	25-940	25-:23										
RMDT	25-184#	25-284														
RMEC1	25-191#															
RMEC2	25-192#	25-<11														
RMER1	25-179#	25-320	25-852	25-941	25-:69											
RMER2	25-190#	25-942														
RMERRS	25-23#	25-216	25-805	25-810	25-820	25-940*	25-941*	25-942*	25-943*	28-13	28-13	28-13	28-13			
RMHR	25-188#															
RMINIT	9-335	9-523	9-778	9-849	9-935	9-968	9-998	9-<27	9-<56	9-=52	9--80	9-559	9-899	25-212#		
RMLA	25-181#															
RMMR1	25-183#															
RMMR2	25-189#	25-943														
RMOF	25-186#	25-311	25-538	25-542	25-562	25-566										
RMSN	25-185#															
RMTMR	9-E07	25-986#														
RMVEC	9-74*	9-306*	9-308*	9-309*	9-490*	9-491*	9-522*	9-696*	9-697*	9-749*	9-751*	9-752*	9-816*	9-817*		
	9-824*	9-?36*	9-?37*	25-163#	25-229	25-231	25-346									
RMLC	9-:63*	25-174#														

[illegible]

TD	25-696	25-704#												25-994#	25-:63#
TIMER	9-:20*	25-130#	25-451*	25-597*	25-661*	25-707*	25-836*	25-882*	25-934*	25-951*	25-975*	25-992	25-994#	25-:63#	
	25-:71*	25-:78*													
TKVEC	4-58#	9-H39*	9-H40*												
TPVEC	4-58#														
TRAPVE	4-58#	9-31*	9-31*												
TRFER	9-A35	9-B72#													
TRKLMT	7-0#	9-344*	9-347*	9-355	9-589	9-663	9-787*	9-790*	9-939	9-:19	9-<60	9-=03	9->04	9->35	
	9-?26														
TRNSWT	9-:15*	25-79#	25-473*	25-631	25-636*	25-647	25-667*	25-678*	25-709	25-710*	25-:34	25-:46*			
TRTVEC	4-58#														
TST1	9-403	9-481#	9-544	9-729											
TST10	9-=24#														
TST11	9->26#														
TST2	9-554#														
TST3	9-591	9-610#													
TST4	9-681	9-693#													
TST5	9-807#	9-870	9-?64												
TST6	9-896#														
TST7	9-:26	9-<26#													
TSTNM	7-0#	9-482*	9-555*	9-611*	9-694*	9-808*	9-897*	9-:09	9-:43	9-<30*	9-=25*	9->27*			
TYPDS	9-a11	24-1#													
TYPE	9-6	9-36	9-50	9-56	9-61	9-67	9-68	9-144	9-145	9-146	9-147	9-148	9-149	9-150	
	9-165	9-167	9-246	9-247	9-259	9-265	9-266	9-267	9-268	9-269	9-274	9-275	9-280	9-282	
	9-310	9-311	9-312	9-313	9-314	9-315	9-316	9-318	9-319	9-320	9-325	9-326	9-327	9-328	
	9-329	9-369	9-370	9-396	9-397	9-463	9-464	9-492	9-493	9-494	9-495	9-496	9-497	9-498	
	9-500	9-501	9-502	9-57	9-508	9-509	9-510	9-538	9-539	9-542	9-543	9-592	9-593	9-594	
	9-682	9-683	9-684	9-698	9-699	9-700	9-705	9-754	9-755	9-756	9-757	9-758	9-759	9-760	
	9-762	9-763	9-764	9-769	9-770	9-771	9-772	9-793	9-794	9-825	9-826	9-827	9-828	9-829	
	9-830	9-831	9-833	9-834	9-835	9-840	9-841	9-842	9-843	9-864	9-865	9-868	9-869	9-?30	
	9-?35	9-?38	9-?39	9-?44	9-?93	9-?94	9-?95	9-a00	9-a01	9-a02	9-a03	9-a04	9-a05	9-a10	
	9-a12	9-a15	9-a16	9-a17	9-a23	9-a24	9-a26	9-a33	9-a36	9-a37	9-A19	9-A20	9-A21	9-A22	
	9-A77	9-A78	9-A83	9-C34	9-C35	9-C36	9-C93	9-D59	9-D72	9-D77	9-D82	9-F40	9-H53	9-H54	
	9-H57	9-H58	9-H65	9-H66	9-H73	9-H96	9-I02	9-I07	9-I11	9-I16	9-I21	9-I22	9-I24	9-I27	
	9-I31	10-5	10-6	10-7	10-8	10-9	10-10	11-1	11-1	12-1	12-1	12-1	12-1	12-1	
	12-1	12-1	13-1	15-1	16-1	17-1	18-1	18-1	18-1	18-1	18-1	18-1	18-1	24-1#	26-15
	26-18	26-24	26-27												
TYPOC	9-8	12-1	12-1	18-1	24-1#	26-17	26-26								
TYPON	24-1#														
TYPOS	9-53	9-64	9-166	9-256	9-322	9-504	9-702	9-766	9-837	9-?41	9-?97	24-1#			
TYPRI4	9-F68#														
UCPAR	9-a75	9-a94#													
ULDFLG	25-106#	25-272*	25-355	25-370*	25-553*	25-629*	25-668*	25-797	25-892	25-894*	25-919	25-921*	25-:45*		
UNIT	7-0#	9-a48*													
UNSAF	9-A29	9-B90#													
WCFER	9-A53	9-B84#													
WCHKX	25-721	25-741													
WCKD	4-101#	9-577	9-955	9-:08	9-<69										
WCKER	9-A32	9-A94#													
WCKHD	4-102#														
WLEER	9-A59	9-B55#													
WRT.AD	25-:50*	25-:60*	25-:63#	28-15											
WRT.R1	25-:61#	25-:81													
WRT.R2	25-:47*	25-:80#													
WRT.R3	25-:56	25-:67	25-:70	25-:76	25-:82#										
WRT.R4	25-:72	25-:84#													
WRT.R5	25-:83	25-:85#													

SSCMRE	5-250#													
SSCTM	5-250#	6-0												
SSSCA	4-58#													
SSNEWT	4-58#	9-481	9-554	9-610	9-693	9-807	9-896	9-<26	9-=24	9->26				
SSSET	24-1	24-1	24-1	24-1	24-1	24-1	24-1	24-1	24-1	24-1	24-1	24-1#	24-2	
SSSETM	9-31	9-31#												
SSSETU	9-31	9-31#												
SSSKIP	4-58#													
.SACT1	4-47#	5-8												
.SAPTB	4-50#	6-0	6-0#											
.SAPTH	4-50#	5-11												
.SAPTY	4-50#	14-1												
.SCATC	4-48#	5-1												
.SCMTA	4-48#	5-250												
.SDB2D	4-49#	20-1												
.SDB2O	4-49#	21-1												
.SEOP	4-50#	10-1												
.SERRO	4-48#	11-1												
.SERRT	4-48#	12-1												
.SPOWE	4-50#	15-1												
.SREAD	4-47#	18-1												
.SSAVE	4-49#	19-1												
.SSCOP	4-50#	22-1												
.SSIZE	4-49#	23-1												
.STRAP	4-49#	24-1												
.STYPD	4-48#	17-1												
.STYPE	4-47#	13-1												
.STYPO	4-48#	16-1												
.EQUAT	4-47#	4-58												
.HEADE	4-47#	4-54												
.SETUP	4-47#	4-111												
.SWRHI	4-47#	4-55												
.SWRLO	4-47#	4-55#	4-56											
CKCHR	4-8#	9-673	9-681											
CKDIG	4-18#													
CKNUM	4-31#													
COMMEN	4-58#													
ENDCOM	4-58#													
ERRCAL	24-4#	25-772	25-830	25-832	25-;18	25-;77								
ERROR	4-58#	9-A79	9-A80	9-A81	9-A82	9-B94	9-B95	9-B96	9-B97	22-1	26-43			
ESCAPE	4-58#													
GETPRI	4-58#	10-1	23-1											
GETSWR	4-58#	9-36	9-36#											
MORETA	5-14#	6-0												
MULT	4-58#													
NEWTST	4-58#	9-481	9-554	9-610	9-693	9-807	9-896	9-<26	9-=24	9->26				
POP	4-58#	9-;83	9-<16	9-239	9-E47	14-1	14-1	15-1	15-1	17-1	19-1			
PUSH	4-58#	9-;40	9-;99	9-?89	9-E40	14-1	14-1	14-1	15-1	15-1	17-1	19-1		
REPORT	4-58#													
SETPRI	4-58#													
SETTRA	24-1	24-1	24-1	24-1	24-1	24-1	24-1	24-1	24-1	24-1	24-1	24-1#	24-2	
SETUP	4-58#	9-31												
SKIP	4-58#													
SLASH	4-58#													
STARS	4-58#	5-8	5-11	5-11	5-11	6-0	6-0	6-0	9-481	9-554	9-610	9-693	9-807	9-896
	9-<26	9-=24	9->26	10-1	11-1	12-1	13-1	14-1	15-1	15-1	16-1	17-1	18-1	18-1

	18-1	19-1	20-1	21-1	22-1	23-1	24-1	25-15
SWRSU	4-58#	9-31	9-31#					
TRMTRP	24-1#	24-3						
TYPBIN	4-58#							
TYPDEC	4-58#	9-211						
TYPNAM	4-58#	9-36						
TYPNUM	4-58#							
TYPPCS	4-58#	9-166						
TYP OCT	4-58#	12-1	12-1	18-1				
TYP TXT	4-58#	9-6	9-50	9-61	9-67	9-68		