

RM05/3/2

RM05/3/2 FCTNL TST1 AH-F922A-MC
CZRMMAO FICHE 1 OF 2

JUN 1980
COPYRIGHT © 1980
MADE IN USA



The main body of the document is a large, dense grid of data. Each row and column contains small, illegible text or numbers, likely representing a detailed technical or financial record. The grid is organized into approximately 15 columns and 25 rows, with some cells containing more complex sub-tables or diagrams. The overall appearance is that of a highly structured data table.

RM05/3/2

RM05/3/2 FCTNL TST1 AH-F922A-MC
CZRMMAO FICHE 2 OF 2

JUN 1980
COPYRIGHT © 1980
MADE IN USA



The main body of the document is a large, dense grid of data. Each cell in the grid contains a small, structured table or set of data points. The data is organized into columns and rows, with some cells containing numerical values, some containing text, and some containing graphical elements like small bar charts or histograms. The overall appearance is that of a complex data matrix or a series of related sub-tables. The text is very small and difficult to read in detail, but the structure is consistent across the entire page.

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

.REM \

IDENTIFICATION

PRODUCT CODE: AC-F921A-MC
PRODUCT NAME: CZRMMAO RM05/3/2 FCTNL TST 1
DATE CREATED: APRIL 1980
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 TO THE PURCHASER UNDER A LICENSE FOR USE ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH INCLUSION OF DIGITAL'S COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY OTHERWISE BE PROVIDED IN WRITING BY DIGITAL.

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

COPYRIGHT (C) 1980 DIGITAL EQUIPMENT CORPORATION

CONTENTS

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

- 1. INTRODUCTION
 - 1. ABSTRACT
 - 2. UNIT UNDER TEST
- 2. OPERATING REQUIREMENTS
 - 1. HARDWARE REQUIREMENTS
 - 2. MEDIA REQUIREMENTS
 - 3. PREREQUISITE DIAGNOSTIC PROGRAMS
- 3. OPERATING PROCEDURE
 - 1. LOADING
 - 2. SWITCH OPTIONS
 - 3. STARTING
 - 4. HALTING
 - 5. RESTARTING
- 4. OPERATOR INTERFACE
 - 1. PROGRAM I.D.
 - 2. CONSOLE DIALOGUE
 - 3. PROGRESS REPORTS
 - 4. PERFORMANCE REPORTS
 - 5. PROGRAM HALTS
 - 6. ERROR REPORTS
- 5. ENVIRONMENTAL SUPPORT
 - 1. PROCESSOR COMPATIBILITY
 - 2. DUAL PORT CONFIGURATIONS
 - 3. MEMORY PARITY HARDWARE
 - 4. MEMORY MANAGEMENT HARDWARE
 - 5. ACT, APT COMPATIBILITY
 - 6. XXDP COMPATIBILITY
 - 7. OPERATING SYSTEM COMPATIBILITY
- 6. TEST DESCRIPTION

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

1.0 INTRODUCTION

1.1 ABSTRACT

THE RM05/3/2 SUBSYSTEM FUNCTIONAL TEST IS A STAND ALONE PROGRAM WHICH USES FUNCTIONAL MEANS TO VERIFY THE OPERABILITY OF THE RM05/3/2 DISK SUBSYSTEM. IN PARTICULAR, THE PROGRAM SERVES THE FOLLOWING PURPOSES:

TO EXPLICITLY ESTABLISH CONFIDENCE IN THE BASIC OPERATIONS OF THE DISK DRIVE, INCLUDING MECHANICAL POSITIONING AND DATA TRANSFER OPERATIONS:

TO IMPLICITLY ESTABLISH CONFIDENCE IN THE DRIVE/ADAPTER ELECTRICAL INTERFACE;

TO VERIFY THE FUNCTIONALITY OF THE RM05/3/2 SUBSYSTEM, INCLUDING THE MASSBUS CONTROLLER, MASSBUS ADAPTER AND THE DISK DRIVE.

THE TEST IS COMPRISED OF 3 PARTS, WHICH WOULD NORMALLY BE RUN IN SEQUENCE, STARTING WITH PART 1. BRIEFLY, PART 1 TESTS HOUSEKEEPING AND MECHANICAL POSITIONING OPERATIONS; PART 2 TESTS WRITE, READ AND WRITE CHECK OPERATIONS USING HEADER AND DATA; PART 3 TESTS WRITE, READ AND WRITE CHECK OPERATIONS USING DATA.

1.2 UNIT UNDER TEST

THE UNIT UNDER TEST (UUT) IS THE RM05/3/2 DISK SUBSYSTEM WHICH CONSISTS OF THE RHXX MASSBUS CONTROLLER, THE RM05/3/2 MASSBUS ADAPTER, AND THE STORAGE MODULE DISK DRIVE. NOTE THAT A DISK PACK IS REQUIRED FOR TESTING AND IS CONSIDERED AN INTEGRAL OF THE STORAGE MODULE DISK DRIVE.

2.0 OPERATING REQUIREMENTS

2.1 HARDWARE REQUIREMENTS

THE FOLLOWING MINIMUM HARDWARE CONFIGURATION, ASSUMED TO BE OPERATIONAL, IS REQUIRED TO LOAD AND EXECUTE THE RM05/3/2 SUBSYSTEM FUNCTIONAL TEST:

PDP-11 PROCESSOR
8K MEMORY
KW11-L OR KW11-P CLOCK
PROGRAM LOADING DEVICE
TERMINAL
RH11 OR RH70 CONTROLLER
1 TO 8 DISK DRIVES (ANY COMBINATION OF RM05'S, RM03'S OR RM02'S)

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

2.2 MEDIA REQUIREMENTS

EACH UNIT BEING TESTED MUST BE LOADED WITH A SCRATCH DISK PACK BEFORE TESTING BEGINS ON THAT UNIT. THE DISK MAY BE FORMATTED OR UNFORMATTED.

2.3 PREREQUISITE DIAGNOSTIC PROGRAMS

RM05/3/2 DISKLESS DIAGNOSTIC, PART 1 & 2

3.0 OPERATING PROCEDURE

3.1 LOADING

THE PROGRAM MAY BE LOADED BY EITHER OF THE FOLLOWING MEDIA:

- .PAPER TAPE, USING THE STANDARD PAPER TAPE LOADING PROCEDURE.
- .XXDP MEDIA, USING THE APPROPRIATE LOADING DEVICE.

3.2 SWITCH OPTIONS

THE FOLLOWING SWITCH OPTIONS ARE PROVIDED TO ENHANCE THE UTILITY OF THE PROGRAM.

SW15	HALT ON ERROR
SW14	LOOP ON TEST (CURRENTLY BEING EXECUTED)
SW13	INHIBIT ERROR TYPEOUTS
SW12	UNUSED
SW11	INHIBIT TEST ITERATIONS
SW10	BELL ON ERROR
SW09	LOOP ON ERROR
SW08	LOOP ON TEST IN SW07-00

THE LOW ORDER 8 SWITCHES (SW07-SW00), ARE USED IN CONJUNCTION WITH SW08 TO SPECIFY THE OCTAL NUMBER OF THE TEST WHICH THE PROGRAM WILL LOOP ON.

3.3 STARTING

THE PROGRAM STARTS AT LOCATION 200, WHICH PROVIDES WORST CASE TEST CONDITIONS IF RUNNING IN AN AUTOMATIC ENVIRONMENT. IF RUNNING IN A STAND-ALONE ENVIRONMENT, THE PROGRAM USES CONSOLE DIALOGUE TO ALLOW THE OPERATOR TO CONTROL TEST CONDITIONS.

3.4 HALTING

THE PROGRAM CAN BE HALTED BY TYPING CONTROL C FROM THE CONSOLE.

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

3.5 RESTARTING

THE PROGRAM CAN BE RESTARTED AT ADDRESS 200.

4.0 OPERATOR INTERFACE

4.1 PROGRAM ID

THE PROGRAM TYPES ITS TITLE AND MAINDEC NUMBER THE FIRST TIME IT IS STARTED AFTER BEING LOADED. PROGRAM IDENTIFICATION DOES NOT OCCUR IF THE PROGRAM IS RESTARTED.

4.2 CONSOLE DIALOGUE

WHEN THE PROGRAM IS RUNNING IN STAND ALONE MODE, IT ENTERS A CONSOLE DIALOGUE SEQUENCE AFTER TYPING THE PROGRAM I.D..

THE FIRST QUESTION TYPED OUT IS: "TYPE HELP TEXT (Y/N) ?". IF THE OPERATOR RESPONDS WITH A "Y", THE PROGRAM WILL TYPE A BRIEF HELP MESSAGE WHICH WILL LIST SWITCH OPTIONS, ETC.

NOTE: THE FIRST QUESTION IS ONLY ASKED ON THE INITIAL PROGRAM START AND NOT ON SUBSEQUENT START-UP'S.

THE SECOND QUESTION TYPED IS, "CHANGE ADDRESSES (Y/N) ?". IF THE UNIBUS ADDRESS OF THE RH/RM IS NON STANDARD, THE OPERATOR SHOULD RESPOND WITH A "Y", THEN ANSWER SUBSEQUENT QUESTIONS TO SPECIFY THE UNIBUS ADDRESS, VECTOR ADDRESS AND INTERRUPT PRIORITY. IF THE OPERATOR RESPONSE IS A "N", THE PROGRAM SKIPS TO THE THIRD QUESTION.

THE THIRD QUESTION TYPED IS, "TYPE 'A' TO TEST ALL DRIVES, OR TYPE DRIVE NUMBER(S) AND TERMINATE INPUT WITH A CARRIAGE RETURN." THEN, 'DRIVE(S):' IS TYPED AND WAITS FOR THE OPERATOR TO TYPE AN 'A', TO TEST ALL POSSIBLE DRIVES OR THE NUMBER(S) OF THE DRIVE(S) HE WANTS TESTED AND TERMINATE HIS INPUT WITH A "CARRIAGE RETURN".

NOTE: THE LONG VERSION OF THE THIRD QUESTION IS ONLY TYPED ON THE INITIAL PROGRAM START. ON ALL SUBSEQUENT STARTS, ONLY THE 'DRIVE(S):' PROMPT IS TYPED.

IF THE PROGRAM IS RESTARTED, THE FIRST QUESTION TYPED IS, "USE SAME DEVICES (Y/N) ?". IF THE OPERATOR TYPES "Y", THE TEST IS RESTARTED USING THE SAME DEVICES AS THE LAST TIME, OTHERWISE, THE TEST RESTARTS THE DIALOGUE AS IF THE PROGRAM WERE STARTED FOR THE FIRST TIME.

4.3 PROGRESS REPORTS

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

AN END OF PASS REPORT OCCURS EACH TIME THE PROGRAM IS EXECUTED FOR ALL DEVICES IN THE TEST QUE. THE END OF PASS REPORT INCLUDES A MESSAGE AND AN ERROR SUMMARY.

4.4 PERFORMANCE REPORT

NO PERFORMANCE REPORTS ARE GIVEN DURING THE EXECUTION OF THE PROGRAM.

4.5 PROGRAM HALTS

THERE ARE NO SCHEDULED HALTS DURING THE EXECUTION OF THE PROGRAM. PROCESSOR HALTS ARE DUE TO THE TRAP CATCHER.

4.6 ERROR REPORTS

THE FIRST LINE OF THE ERROR REPORT CONTAINS THE NUMBER OF THE UNIT BEING TESTED, THE TEST NUMBER, THE ERROR NUMBER AND THE VALUE OF THE PROGRAM COUNTER WHERE THE ERROR WAS CALLED. THIS LINE IS FOLLOWED BY THE ERROR MESSAGE: ONE OR MORE LINES OF TEXT WHICH GIVE A BRIEF, YET COMPREHENSIVE DESCRIPTION OF THE ERROR. THE ERROR MESSAGE IS NORMALLY FOLLOWED BY ONE OR MORE PAIRS OF LINES CONTAINING DATA HEADERS AND DATA PERTINENT TO THE ERROR, INCLUDING EXPECTED AND ACTUAL TEST RESULTS.

5.0 ENVIRONMENTAL SUPPORT

5.1 PROCESSOR COMPATIBILITY

THE RM05/3/2 SUBSYSTEM FUNCTIONAL TEST IS EXECUTABLE ON ANY PDP-11 PROCESSOR, PROVIDING PREVIOUSLY MENTIONED HARDWARE REQUIREMENTS ARE MET, AND PROVIDING THAT DATA THROUGHPUT ON THE SYSTEM IS SUFFICIENT TO SUSTAIN DATA TRANSFER OPERATIONS.

5.2 DUAL PORT CONFIGURATIONS

THE RM05/3/2 SUBSYSTEM FUNCTIONAL TEST DOES NOT SPECIFICALLY TEST DUAL PORT LOGIC IN THE RM05/3/2 ADAPTER BUT IS EXECUTABLE ON RM05/3/2 SUBSYSTEMS HAVING THE DUAL PORT OPTION PROVIDING THE DUAL PORT SWITCH IS SET TO THE APPROPRIATE PORT (A OR B).

5.3 MEMORY PARITY HARDWARE

MEMORY PARITY HARDWARE IS NOT USED DURING THE EXECUTION OF THE RM05/3/2 SUBSYSTEM FUNCTIONAL TEST.

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

5.4 MEMORY MANAGEMENT HARDWARE

MEMORY MANAGEMENT HARDWARE IS NOT USED DURING THE RM05/3/2 SUBSYSTEM FUNCTIONAL TEST. CAPABILITIES OF THE MASSBUS CONTROLLER.

5.5 ACT11, APT11 COMPATIBILITY

THE RM05/3/2 SUBSYSTEM FUNCTIONAL TEST IS COMPATIBLE WITH ACT11 AND APT11 IN BOTH DUMP AND AUTOMATIC MODES. FURTHER, THE PROGRAM WILL EXECUTE A QUICK PASS DURING THE FIRST PASS IN SUPPORT OF QUICK VERIFY MODE.

5.6 XXDP COMPATIBILITY

THE RM05/3/2 SUBSYSTEM FUNCTIONAL TEST IS COMPATIBLE WITH XXDP IN DUMP AND CHAIN MODES, AND PROVIDES MEDIA PROTECTION IN THE CASE WHERE THE RM05/3/2 IS THE XXDP LOADING DEVICE.

5.7 OPERATING SYSTEM COMPATIBILITY

THE PROGRAM IS NOT COMPATIBLE WITH ANY SOFTWARE OPERATING SYSTEM.

6.0 TEST DESCRIPTION

CONTROLLER ACCESS TEST

PURPOSE:

TO VERIFY THAT THE UNIBUS ADDRESS OF THE SUBSYSTEM IS CORRECT, AS DEFINED AT LOCATION \$BASE.

PROCEDURE:

THE TEST TRIES TO ACCESS ALL MASSBUS CONTROLLER REGISTERS USING THE \$BASE ADDRESS. REGISTER CONTENTS ARE IGNORED DURING THE TEST, AND THE TEST FAILS IF A BUS TIMEOUT OCCURS FOR ANY REGISTER TRANSFER.

IF THE TEST FAILS AND THE PROGRAM IS RUNNING IN A STAND ALONE ENVIRONMENT, I.E., LOCATION 42 IS 0, THE PROGRAM WILL JUMP TO LOCATION 204 WHICH ALLOWS THE OPERATOR TO CHANGE THE \$BASE ADDRESS VIA CONSOLE DIALOGUE. OTHERWISE, THE PROGRAM ESCAPES TO

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

THE END OF PASS HANDLER.

DEVICE AVAILABLE TEST

PURPOSE:

TO VERIFY THAT THE UNIT UNDER TEST IS AVAILABLE FOR TESTING,
AND IS NOT LOCKED OR PROGRAMMED TO THE ALTERNATE PORT.

PROCEDURE:

THIS TEST SELECTS THE DEVICE AND READS CONTROL STATUS
REGISTERS 1 AND 2 TO VERIFY THAT THE SELECTED DEVICE IS AVAILABLE
FOR TESTING, AS INDICATED BY DVA STATUS, BIT 11 OF RMCS1 AND NED
STATUS, BIT 12 OF RMCS2. THE RESULTS OF THE TEST CAN VARY AS
FOLLOWS:

.NONEXISTANT DEVICE - THE DEVICE IS NONEXISTENT OR IS LOCKED
ON THE ALTERNATE PORT AND IS THEREFORE NOT AVAILABLE FOR TESTING;

.DEVICE NOT AVAILABLE - THE DEVICE EXISTS BUT IS SEIZED BY
THE ALTERNATE PORT AND IS NOT AVAILABLE FOR TESTING;

.BUS TIMEOUT - THE MASSBUS CONTROLLER FAILED TO DETECT A
NONEXISTENT DEVICE;

.DEVICE AVAILABLE - THE DEVICE IS AVAILABLE FOR TESTING.

THE PROGRAM JUMPS TO THE SUBPASS HANDLER WHICH SELECTS THE
NEXT DRIVE TO BE TESTED IF THE DEVICE IS NOT AVAILABLE.

DRIVE TYPE TEST

PURPOSE:

TO VERIFY THAT THE UNIT UNDER TEST IS AN RM05, RM03 OR RM02
SINGLE PORT OR DUAL PORT SUBSYSTEM.

PROCEDURE:

THIS TEST READS THE DRIVE TYPE REGISTER, RMDT, OF THE
SELECTED DEVICE AND VERIFIES THAT THE DEVICE IS A SINGLE PORT OR
DUAL PORT RM05/3/2 SUBSYSTEM. IF THE SELECTED DEVICE IS NOT AN
RM05/3/2, THE PROGRAM JUMPS TO THE SUBPASS HANDLER WHICH WILL
SELECT THE NEXT DEVICE FOR TESTING.

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

UNIBUS INITIALIZE TEST

PURPOSE:

TO VERIFY THAT ALL APPLICABLE SUBSYSTEM REGISTERS ARE INITIALIZED BY THE RESET INSTRUCTION.

PROCEDURE:

NONZERO VALUES ARE WRITTEN IN EACH APPLICABLE REGISTER. A RESET INSTRUCTION IS EXECUTED, AND THE REGISTERS ARE TESTED TO INSURE THEY WERE INITIALIZED. THIS TEST IS DONE ONCE BECAUSE OF APT COMPATIBILITY REQUIREMENTS.

THE FOLLOWING REGISTERS ARE PRESET BEFORE THE INITIALIZE OCCURS:

RMCS1 - 003577

RMBA - 777776

RMCS2 - 021037

RMER1 - 777777

RMER2 - 777777

RMMR - 040001

IN ADDITION, THE DATA BUFFER IS USED TO FORCE DLT, TRE, SC AND OR TO A ONE AND TO FORCE IR TO A ZERO.

CONTROLLER CLEAR TEST

PURPOSE:

TO VERIFY THAT APPLICABLE SUBSYSTEM REGISTERS ARE INITIALIZED BY A "CONTROLLER CLEAR" OPERATION.

PROCEDURE:

LIKE THE UNIBUS INITIALIZE TEST, THIS TEST WRITES NONZERO VALUES IN THOSE REGISTERS WHICH ARE INITIALIZED BY CONTROLLER CLEAR. THE SUBSYSTEM IS THEN CLEARED USING A CONTROLLER CLEAR, I.E., BIT 5 OF CONTROL STATUS REGISTER 2 (RMCS2), AND EACH REGISTER IS READ TO INSURE IT WAS CLEARED.

ERROR CLEAR TEST

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

PURPOSE:

TO VERIFY THAT ALL APPLICABLE RM MASSBUS CONTROLLER STATUS AND ERROR CONDITIONS ARE INITIALIZED BY AN ERROR CLEAR OPERATION.

PROCEDURE:

AN 'RH ERROR CLEAR' OPERATION, I.E., WRITING A ONE IN TRE, BIT 14 OF RMCS1 WILL CLEAR THE FOLLOWING STATUS BITS:

.TRE, BIT 14 OF RMCS1
.MCPE, BIT 13 OF RMCS1 - READ ONLY
.DLT, BIT 15 OF RMCS2 - READ ONLY
.WCE, BIT 14 OF RMCS2 - READ ONLY
.UPE, BIT 13 OF RMCS2
.NED, BIT 12 OF RMCS2 - READ ONLY
.PGE, BIT 10 OF RMCS2 - READ ONLY
.MXF, BIT 09 OF RMCS2
.MDPE, BIT 08 OF RMCS2 - READ ONLY

THE TEST SETS UPE AND MXF STATUS BITS, THEN SETS TRE (ERROR CLEAR) AND VERIFIES THAT ALL THE ABOVE STATUS BITS ARE CLEARED.

DRIVE STATUS TEST

PURPOSE:

TO VERIFY THAT THE STORAGE MODULE DISK DRIVE IS IN A STATE THAT PERMITS FURTHER TESTING.

PROCEDURE:

THIS TEST INITIALIZES THE MASSBUS AND EXAMINES STATUS OF THE SELECTED DEVICE FOR THE FOLLOWING CONDITIONS:

.MOL, BIT 12 OF RMDS =1, INDICATING THAT UNIT READY IS ASSERTED BY THE DRIVE;
.WRL, BIT 11 OF RMDS =0, INDICATING THAT THE DRIVE IS NOT IN A WRITE PROTECT STATE;
.DVC, BIT 07 OF RMER3 =0, INDICATING DRIVE FAULT IS UNASSERTED BY THE DRIVE;

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

.UNS, BIT 14 OF RMER1 SHOULD EQUAL DVC, OTHERWISE AC POWER IS LOW OR A FAILURE HAS OCCURRED WITH UNSAFE STATUS.

PRIMARY/SECONDARY ERROR TEST

PURPOSE:

TO VERIFY THAT THE RM05/3/2 CAN EXECUTE A COMMAND WITHOUT INCURRING UNEXPECTED ERRORS.

PROCEDURE:

THE TEST EXECUTES A PACK ACKNOWLEDGE COMMAND AND MAKES SURE THAT GO RESETS AND THAT THERE ARE NO PARITY ERRORS, ETC. VOLUME VALID STATUS IS IGNORED.

DIAGNOSTIC MODE TEST

PURPOSE:

TO VERIFY THAT MAINTENANCE HARDWARE IS OPERATIONAL.

PROCEDURE:

THE TEST THAT DIAGNOSTIC MODE CAN BE SET AND RESET, THEN VERIFIES THAT "MOL, PIP, WRL, SKI, AND DVC" CAN BE CONTROLLED USING MAINTENANCE REGISTER 1.

PACK ACKNOWLEDGE TEST

PURPOSE:

TO VERIFY THAT VOLUME VALID CAN BE SET BY A PACK ACKNOWLEDGE COMMAND, LENDING CREDENCE TO THE EXECUTION OF THE COMMAND AND TO THE STABILITY OF THE UNIT READY SIGNAL FROM THE DRIVE.

PROCEDURE:

A PACK ACKNOWLEDGE COMMAND IS ISSUED TO THE SELECTED DEVICE AND VOLUME VALID STATUS, BIT 10 OF RMDS, IS CHECKED FOR ONE.

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

RECALIBRATE TEST

PURPOSE:

THE PRIMARY PURPOSE IS TO ASCERTAIN THAT THE DRIVE WILL EXECUTE A RECALIBRATE OPERATION TO THE EXTENT THAT "PIP" AND "SKI" STATUS BECOME UNASSERTED AT THE COMPLETION OF THE RECALIBRATE. THE SECONDARY PURPOSE IS TO PUT THE DRIVE IN A KNOWN STATE SO THAT FURTHER TESTS CAN CHECK FOR UNEXPECTED STATE CHANGES IN THE DRIVE.

PROCEDURE:

THE RECALIBRATE TEST PRESETS THE DISK ADDRESS REGISTER, RMDA, AND THE DESIRED CYLINDER REGISTER, RMDC, TO ZERO, THEN EXECUTES A RECALIBRATE COMMAND. THE TEST VERIFIES THE FOLLOWING CONDITIONS:

.SKI=0, "SEEK ERROR" IS INACTIVE;

.PIP=0, "ON CYLINDER" IS ACTIVE, AS INDICATED BY "POSITIONING IN PROGRESS" BEING INACTIVE;

.IAE=0, NO "INVALID ADDRESS ERROR" DURING RECALIBRATE;

.OPI=0, NO "OPERATION INCOMPLETE ERROR", INDICATING THAT THAT THE DRIVE WAS READY AT THE START OF THE COMMAND AND THAT ON CYLINDER STATUS WENT INACTIVE WHEN THE DRIVE RECEIVED THE COMMAND;

.ATA=1, ATTENTION IS SET BY RECALIBRATE.

ABORT RECALIBRATE TEST

PURPOSE:

TO VERIFY THAT THE RM05/3/2 INHIBITS A RECALIBRATE WHEN AN ABORT CONDITION EXISTS AT THE START OF THE COMMAND.

PROCEDURE:

THE TEST SETS AN ERROR IN THE ERROR REGISTER AND ISSUES A RECALIBRATE COMMAND, VERIFYING THAT "PIP" REMAINS INACTIVE.

IVC RECALIBRATE TEST

PURPOSE:

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

TO VERIFY THAT INVALID COMMAND STATUS, BIT 12 OF RMER2, SETS WHEN VOLUME VALID IS INACTIVE DURING A RECALIBRATE COMMAND.

PROCEDURE:

THE PROGRAM SETS AND RESETS DIAGNOSTIC MODE WHICH CAUSES VOLUME VALID, BIT 6 OF RMDS TO RESET. THE PROGRAM THEN EXECUTES A RECALIBRATE COMMAND AND VERIFIES THAT "IVC" STATUS SETS AND THAT "PIP" REMAINS INACTIVE.

IAE RECALIBRATE TEST

PURPOSE:

TO VERIFY THAT INVALID ADDRESS ERROR DOES NOT SET DURING A RECALIBRATE COMMAND .

PROCEDURE:

THE TEST PRESETS THE DISK ADDRESS (RMDA) AND THE DESIRED CYLINDER ADDRESS (RMDC) TO ILLEGAL VALUES AND ISSUES A RECALIBRATE COMMAND, VERIFYING THAT "IAE" DOES NOT SET DURING THE COMMAND.

RECALIBRATE AT OFFSET

PURPOSE:

TO VERIFY THAT OFFSET MODE DOES NOT AFFECT RECALIBRATE.

PROCEDURE:

THE TEST EXECUTES AN OFFSET COMMAND, THEN EXECUTES A RECALIBRATE COMMAND AND VERIFIES THAT THERE ARE NO ERRORS DURING RECALIBRATE.

DRIVE CLEAR TEST

PURPOSE:

TO VERIFY THAT ALL APPLICABLE RM05/3/2 MASSBUS ADAPTER ERROR AND STATUS CONDITIONS ARE INITIALIZED BY A "DRIVE CLEAR" COMMAND.

PROCEDURE:

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

THIS TEST WRITES ONES IN THOSE MASSBUS ADAPTER BITS WHICH ARE INITIALIZED BY DRIVE CLEAR, THEN ISSUES THE DRIVE CLEAR COMMAND AND VERIFIES THAT EACH BIT IS CLEARED. ADDITIONALLY, REGISTERS WHICH ARE NOT AFFECTED BY "DRIVE CLEAR" ARE ALSO PRESET AND VERIFIED.

THE FOLLOWING ITEMS ARE PRESET:

.RMER1, ERROR REGISTER 1 IS SET TO ALL ONES;

.DMD, DIAGNOSTIC MODE IS SET;

.RMER2, ERROR REGISTER 3, IS SET TO ALL ONES.

FOLLOWING THE DRIVE CLEAR COMMAND, THE FOLLOWING ITEMS ARE CHECKED:

.RMCS1, IS CHECKED FOR DVA=1, F0-F4 AND GO=0, ALL OTHER BITS ARE DONT CARES;

.RMDS, IS CHECKED FOR 0, EXCEPT FOR MOL,DPR,DRY, AND VV WHICH SHOULD BE ONE, AND PGM WHICH IS A DONT CARE.

.RMER1, IS CHECKED FOR 0;

.RMAS, IS CHECKED TO INSURE THE APPROPRIATE ATA BIT IS 0;

.RMMR, IS CHECKED FOR 0, EXCEPT FOR WORD CLOCK, LAST SECTOR, AND LAST SECTOR AND TRACK WHICH ARE DONT CARES;

.RMMR2 IS CHECKED FOR 0, EXCEPT FOR THE TEST BIT WHICH IS ONE, AND REQA, REQB WHICH ARE DONT CARES;

.RMEC2 IS CHECKED FOR 0;

.RMER3 IS CHECKED FOR 0;

NOP TEST

PURPOSE:

TO VERIFY THE EXECUTION OF "NOP" COMMAND.

PROCEDURE:

A NOP COMMAND IS EXECUTED ON THE SELECTED DEVICE AND STATUS IS CHECKED TO VERIFY THAT THERE WERE NO ERRORS OR UNEXPECTED CHANGES IN STATUS.

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

OFFSET TEST

PURPOSE:

TO VERIFY THE EXECUTION OF "OFFSET" COMMAND.

PROCEDURE:

THE OFFSET COMMAND IS EXECUTED AND THE PROGRAM CHECKS THAT OFFSET STATUS, BIT 0 OF RMDS IS SET AND THAT ATTENTION, BIT 15 OF RMDS IS ALSO SET. ADDITIONALLY, CONTROLLER, ADAPTER, AND DRIVE STATUS IS CHECKED FOR UNEXPECTED CHANGES.

GO/ATA TEST

PURPOSE:

TO VERIFY THAT "ATA" WILL RESET WITH "GO" PROVIDING COMPOSITE ERROR IS INACTIVE.

PROCEDURE:

ATTENTION, BIT 15 OF RMDS, IS SET USING AN OFFSET COMMAND AND RESET USING A NOP COMMAND.

WRITE ATA TEST

PURPOSE:

TO VERIFY THAT ATTENTION CAN BE RESET BY WRITING THE ATTENTION SUMMARY REGISTER, RMAS.

PROCEDURE:

ATTENTION IS SET USING AN OFFSET COMMAND AFTER WHICH THE PROGRAM WRITES A 0 IN THE ATTENTION SUMMARY REGISTER, VERIFYING THAT ATTENTION REMAINS SET. FOLLOWING THAT, THE PROGRAM WRITES A 1 IN RMAS AND VERIFIES THAT ATTENTION RESETS.

ERROR/ATA TEST

PURPOSE:

TO VERIFY THAT "GO" DOES NOT RESET "ATA" WHEN THERE IS A COMPOSITE ERROR.

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

PROCEDURE:

"ATA" IS SET WITH AN OFFSET COMMAND AFTER WHICH ONE OF THE ERROR BITS IS SET. THE PROGRAM THEN ISSUES A NOP COMMAND AND VERIFIES THAT THE ATTENTION BIT REMAINS SET.

PROGRAM INTERRUPT TEST

PURPOSE:

TO VERIFY THAT THE SUBSYSTEM WILL GENERATE A PROGRAM INTERRUPT WHEN PROCESSOR PRIORITY IS LESS THAN CONTROLLER PRIORITY AND INTERRUPT IS ENABLED.

PROCEDURE:

ATTENTION IS SET USING AN OFFSET COMMAND. WITH INTERRUPT ENABLED AND PROCESSOR PRIORITY SET BELOW CONTROLLER PRIORITY, THE PROGRAM VERIFIES THAT A PROGRAM INTERRUPT OCCURS.

INHIBIT INTERRUPT TEST

PURPOSE:

TO VERIFY THAT A PROGRAM INTERRUPT DOES NOT OCCUR WHEN (1) PROCESSOR AND CONTROLLER PRIORITY ARE THE SAME AND INTERRUPT IS ENABLED OR (2) WHEN PROCESSOR PRIORITY IS LESS THAN CONTROLLER AND INTERRUPTS ARE NOT ENABLED.

PROCEDURE:

ATTENTION IS SET USING AN OFFSET COMMAND WITH THE PRIORITY OF THE PROCESSOR SET EQUAL TO THE PRIORITY OF THE CONTROLLER. INTERRUPT IS ENABLED AND THE TEST VERIFIES THAT A PROGRAM INTERRUPT DOES NOT OCCUR. INTERRUPTS ARE DISABLED AND PROCESSOR PRIORITY IS LOWERED AND THE TEST VERIFIES THAT A PROGRAM INTERRUPT DOES NOT OCCUR.

RETURN TO CENTERLINE TEST

PURPOSE:

TO VERIFY THE EXECUTION OF "RETURN TO CENTERLINE" COMMAND.

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

PROCEDURE:

THIS TEST ISSUES AN RTC COMMAND AND VERIFIES THAT OFFSET STATUS, BIT 0 OF RMDS IS RESET AND THAT ATTENTION, BIT 15 OF RMDS IS SET. UNEXPECTED STATUS OR ERROR CONDITIONS ARE ALSO VERIFIED.

READ IN PRESET TEST

PURPOSE:

TO VERIFY THE EXECUTION OF "READ IN PRESET" COMMAND.

PROCEDURE:

THIS TEST LOADS NON ZERO VALUES IN THOSE ADAPTER REGISTERS WHICH ARE INITIALIZED BY THE READ IN PRESET COMMAND, THEN EXECUTES THE COMMAND AND VERIFIES THE CONTENTS OF EACH REGISTER. THE FOLLOWING REGISTERS ARE CHECKED:

.RMDA, THE DISK ADDRESS REGISTER, IS LOADED WITH ALL ONES AND VERIFIED TO BE ZERO AFTER THE RIP COMMAND;

.RMDC, THE DESIRED CYLINDER REGISTER, IS LOADED WITH ALL ONES (001777) AND VERIFIED TO BE ZERO AFTER THE RIP COMMAND;

.RMOF, THE OFFSET REGISTER, IS PRESET (TO 016200) AND VERIFIED TO BE ZERO AFTER THE TEST;

RMDC CLEAR OFFSET TEST

PURPOSE:

TO VERIFY THAT WRITING THE DESIRED CYLINDER REGISTER (RMDC) WILL CLEAR OFFSET MODE.

PROCEDURE:

OFFSET MODE IS SET USING THE OFFSET COMMAND, THEN RESET BY WRITING RMDC.

ILLEGAL FUNCTION TEST

PURPOSE:

TO VERIFY THAT THE RM05/3/2 SUBSYSTEM DETECTS ALL ILLEGAL

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

FUNCTIONS.

PROCEDURE:

EACH ILLEGAL FUNCTION CODE IS EXECUTED WITH THE PROGRAM VERIFYING THAT "ILLEGAL FUNCTION" STATUS, BIT 0 OF RMER1 IS SET FOR EACH CODE. THE SUBSYSTEM IS INITIALIZED PRIOR TO EACH ILLEGAL FUNCTION TEST.

INVALID COMMAND TEST

PURPOSE:

TO VERIFY IVC ERROR DETECTION.

PROCEDURE:

THE TEST RESETS VOLUME VALID USING MAINTENANCE UNIT READY, THEN EXECUTES A NOP COMMAND AND VERIFIES THAT IVC IS SET. THE PROCESS IS REPEATED FOR EACH FUNCTION CODE, WITH IVC BEING CHECKED ACCORDING TO THE FUNCTION.

INVALID ADDRESS ERROR TEST

PURPOSE:

TO VERIFY IAE ERROR DETECTION.

PROCEDURE:

THE TEST EXECUTES EACH FUNCTION CODE WITH RMDA, AND RMDC SET TO ILLEGAL ADDRESSES, AND VERIFIES IAE ACCORDING TO THE FUNCTION.

WRITE LOCK ERROR TEST

PURPOSE:

TO VERIFY WLE ERROR DETECTION.

PROCEDURE:

THE TEST SIMULATES WRITE PROTECT USING MAINTENANCE WRITE PROTECT AND VERIFIES WLE ACCORDING TO THE FUNCTION BEING EXECUTED. EACH FUNCTION CODE IS TESTED.

913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969

ERROR ABORT TESTS

PURPOSE:

TO TEST COMMAND EXECUTION DURING AN ABORT CONDITION.

PROCEDURE:

EACH FUNCTION CODE IS EXECUTED UNDER A SIMULATED UNSAFE CONDITION AND THE TEST VERIFIES THAT GO IS RESET.

RMR TEST

PURPOSE:

TO VERIFY THAT RMR ERROR IS DETECTED.

PROCEDURE:

THE TEST EXECUTES A NOP COMMAND WITH DEBUG CLOCK ENABLED. WITH GO SET, THE TEST WRITES RMCSI, VERIFYING THAT RMR ERROR SETS.

PARITY ERROR TEST

PURPOSE:

TO VERIFY THAT PARITY ERRORS ARE DETECTED.

PROCEDURE:

WITH PAT SET TO CAUSE BAD PARITY ON THE CONTROL BUS, THE TEST WRITES A SHIFTING ONE BIT PATTERN TO RMDA AND VERIFIES THAT AN ERROR IS DETECTED BY THE RM05/3/2 FOR EACH PATTERN.

ILLEGAL REGISTER TEST

PURPOSE:

TO VERIFY THE DETECTION OF ILLEGAL REGISTER ADDRESSES BY THE SUSBSYSTEM.

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

PROCEDURE:

EACH REGISTER ADDRESS IS ACCESSED AND ILLEGAL REGISTER STATUS, BIT 1 OF RMER1 IS CHECKED ACCORDING TO THE ADDRESS USED. NOTE THAT THE EXECUTION OF THIS TEST IS DEPENDENT ON THE REGISTER ADDRESS JUMPER IN THE RH CONTROLLER BECAUSE THE RANGE OF ADDRESSES WHICH THE CONTROLLER WILL RESPOND TO IS LIMITED BY THE WAY THE JUMPER IS CUT.

SEEK TESTS

PURPOSE:

THE PURPOSE OF EACH OF THE FOLLOWING SEEK TESTS IS TO VERIFY THE EXECUTION OF SEEK OPERATIONS BY THE RM05/3/2 SUBSYSTEM USING A SET OF ADDRESSES THAT TEST THE ADAPTER/DEVICE INTERFACE AND ELECTROMECHANICAL HEAD POSITIONING HARDWARE.

PROCEDURE:

EACH TEST WILL RECALIBRATE THE DRIVE IF "PIP" OR "SKI" IS ACTIVE. FOLLOWING THAT, THE TEST EXECUTES A SEEK TO A CYLINDER ADDRESS OR SERIES OF ADDRESSES. AT THE COMPLETION OF EACH SEEK OPERATION, SUBSYSTEM STATUS IS STORED AND THE TEST CHECKS FOR PRIMARY ERRORS WHICH PRECLUDE FURTHER ERROR CHECKING. IF THERE ARE NO PRIMARY ERRORS THE TEST CHECKS FOR OPERATIONAL ERRORS AND THEN SECONDARY ERRORS.

SEEK TO LAST CYLINDER TEST

THIS TEST SEEKS TO THE LAST CYLINDER, I.E., CYLINDER 822.

SEEK TO FIRST CYLINDER TEST

THIS TEST SEEKS TO THE FIRST CYLINDER, I.E. CYLINDER 0.

SEEK PRIME CYLINDERS TEST

THIS TEST SEEKS FORWARD TO EACH PRIME CYLINDER ADDRESS, I.E. CYLINDERS 1,2,4,8,....,512.

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

SEEK ZERO DIFFERENCE TEST

THIS TEST EXECUTES SUCCESSIVE SEEKS TO CYLINDER 0.

SEEK MAXIMUM DIFFERENCE FORWARD TEST

THIS TEST SEEKS TO CYLINDER 0 FOLLOWED BY A SEEK TO THE LAST CYLINDER.

SEEK ADJACENT FORWARD TEST

THIS TEST SEEKS TO CYLINDER 0, FOLLOWED BY A SEEK TO THE ADJACENT FORWARD CYLINDER, I.E., CYLINDER 1.

SEEK ADJACENT REVERSE TEST

THIS TEST SEEKS TO CYLINDER 1, FOLLOWED BY A SEEK TO THE ADJACENT REVERSE CYLINDER, I.E., CYLINDER 0.

SEEK TO INVALID SECTOR TEST

WITH THE OFFSET REGISTER SET FOR 18 BIT FORMAT, THE PROGRAM SEEKS TO CYLINDER 0, TRACK 0, FOR EACH INVALID SECTOR ADDRESS AND VERIFIES THAT INVALID ADDRESS ERROR SETS FOR EACH SECTOR.

SEEK TO INVALID TRACK TEST

THE TEST SEEKS TO EACH INVALID TRACK ADDRESS WITH CYLINDER AND SECTOR ADDRESS SET TO 0 AND VERIFIES THAT INVALID ADDRESS ERROR SETS FOR EACH TRACK ADDRESS.

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

SEEK TO INVALID CYLINDER TEST

THE PROGRAM SEEKS TO EACH INVALID CYLINDER ADDRESS WITH THE SECTOR AND TRACK ADDRESS SET TO 0 AND VERIFIES THAT INVALID ADDRESS ERROR SETS FOR EACH CYLINDER ADDRESS.

IVC SEEK TEST

PURPOSE:

TO VERIFY THAT INVALID COMMAND STATUS, BIT 12 OF RMER2 SETS WHEN VOLUME VALID IS INACTIVE DURING A SEEK COMMAND.

PROCEDURE:

THE TEST RESETS VOLUME VALID BY SETTING AND RESETTING DIAGNOSTIC MODE, THEN EXECUTES A SEEK COMMAND AND VERIFIES THAT "IVC" STATUS IS SET.

ABORT SEEK TEST

PURPOSE:

TO VERIFY THAT THE RM05/3/2 INHIBITS A SEEK WHEN AN ABORT CONDITION EXISTS AT THE START OF THE COMMAND.

PROCEDURE:

THE TEST SETS AN ERROR IN THE ERROR REGISTER AND ISSUES A SEEK COMMAND, VERIFYING THAT "PIP" REMAINS INACTIVE.

SEEK AT OFFSET

PURPOSE:

TO VERIFY THAT OFFSET MODE DOES NOT CAUSE ERRORS DURING SEEK.

PROCEDURE:

THE TEST EXECUTES AN OFFSET COMMAND THEN EXECUTES A SEEK COMMAND, VERIFYING THE RESULTS OF THE SEEK.

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

LOOK AHEAD TEST

PURPOSE:

TO INSURE THAT THE SECTOR COUNT WHICH ORIGINATES AT THE DRIVE AND IS VISIBLE THROUGH THE LOOK AHEAD REGISTER (RMLA) IS OPERATIONAL.

PROCEDURE:

WITH THE OFFSET REGISTER SET FOR 18 BIT FORMAT, THE PROGRAM SAMPLES THE LOOK AHEAD REGISTER AND COLLECTS EACH DIFFERENT SAMPLE UNTIL THE VALUE OF THE FIRST SAMPLE IS DETECTED OR UNTIL 31. SAMPLES ARE TAKEN. THE COLLECTION IS THEN TESTED TO DETERMINE THAT THE SECTOR COUNT INCREMENTS CORRECTLY THROUGH THE ENTIRE RANGE OF VALID SECTORS. THE SAME PROCEDURE IS REPEATED FOR 16 BIT FORMAT, WHERE THE LIMIT ON THE NUMBER OF SAMPLES IS 33.

SEARCH EACH SECTOR ON CYLINDER

PURPOSE:

TO VERIFY THE EXECUTION OF SEARCH OPERATIONS WITH NO HEAD MOTION USING THE SECTOR PULSE FOR SECTOR COMPARE.

PROCEDURE:

THE TEST INTIALIZES AND RECALIBRATES THE DRIVE IF "PIP" OR "SKI" IS ACTIVE THEN SEEKS TO CYLINDER 0. THE TEST THEN DOES A SEARCH TO EACH SECTOR AND VERIFIES THAT THE SEARCH COMPLETES WITHOUT ERROR. THIS TEST IS DONE ONCE IN 18 BIT FORMAT AND ONCE IN 16 BIT FORMAT.

SEARCH OFF CYLINDER

PURPOSE:

TO VERIFY THE EXECUTION OF A SEARCH COMMAND WITH IMPLIED HEAD MOTION.

PROCEDURE:

THE DRIVE IS RECALIBRATED AND INITIALIZED IF THE DRIVE IS

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

OFF CYLINDER OR HAS A SEEK ERROR. THE PROGRAM THEN EXECUTES AND EXPLICIT SEEK TO CYLINDER 410., FOLLOWED BY A SEARCH TO CYLINDER 411., TRACK 0, SECTOR 0. THE FOLLOWING IS SEARCHED FOR BY AN EXPLICIT SEEK TO CYLINDER 409., FOLLOWED BY A SEARCH TO CYLINDER 412., TRACK 0, SECTOR 1, ETC., UNTIL ALL THE SECTORS HAVE BEEN SEARCHED. THE TEST IS DONE FOR 18 BIT FORMAT, WHERE THE LAST SECTOR SEARCHED IS SECTOR 29. THE TEST IS REPEATED FOR 16 BIT FORMAT, WHERE THE LAST SECTOR SEARCHED IS 31.

SEARCH INVALID SECTOR

PURPOSE:

TO VERIFY THE DETECTION OF AN INVALID SECTOR ADDRESS DURING A SEARCH OPERATION.

PROCEDURE:

THE DRIVE IS RECALIBRATED AND INITIALIZED IF IT IS OFF CYLINDER OR HAS A SEEK ERROR. WITH THE OFFSET REGISTER SET FOR 18 BIT FORMAT, THE PROGRAM EXECUTES A SEARCH TO EACH INVALID SECTOR, I.E., SECTORS 30 AND 31 AND VERIFIES THAT INVALID ADDRESS ERROR SETS FOR EACH SECTOR.

SEARCH INVALID TRACK

PURPOSE:

TO VERIFY THE DETECTION OF AN INVALID TRACK ADDRESS DURING A SEARCH OPERATION.

PROCEDURE:

THE DRIVE IS RECALIBRATED AND INITIALIZED IF IT IS OFF CYLINDER OR HAS A SEEK ERROR. THE PROGRAM EXECUTES A SEARCH TO EACH INVALID TRACK ADDRESS WITH THE CYLINDER ADDRESS 0, AND SECTOR ADDRESS 0 AND VERIFIES THAT INVALID ADDRESS ERROR SETS FOR EACH TRACK.

SEARCH INVALID CYLINDER

PURPOSE:

TO VERIFY THE DETECTION OF AN INVALID CYLINDER ADDRESS DURING A SEARCH OPERATION.

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

PROCEDURE:

THE DRIVE IS RECALIBRATED AND INITIALIZED IF IT IS OFF CYLINDER OR HAS A SEEK ERROR. THE PROGRAM THEN EXECUTES A SEARCH TO EACH INVALID CYLINDER ADDRESS AND VERIFIES THAT INVALID ADDRESS ERROR SETS FOR EACH CYLINDER.

IVC SEARCH TEST

PURPOSE:

TO VERIFY THAT INVALID COMMAND STATUS SETS WHEN VOLUME VALID IS INACTIVE DURING A SEARCH COMMAND.

PROCEDURE:

VOLUME VALID IS RESET BY SETTING AND RESETTING DIAGNOSTIC MODE, AFTER WHICH THE TEST EXECUTES A SEARCH COMMAND, VERIFYING THAT "IVC" STATUS SETS.

ABORT SEARCH TEST

PURPOSE:

TO VERIFY THAT THE RM05/3/2 INHIBITS A SEARCH WHEN AN ABORT CONDITION EXISTS AT THE START OF THE COMMAND.

PROCEDURE:

THE TEST SETS AN ERROR IN THE ERROR REGISTER AND ISSUES A SEARCH COMMAND, VERIFYING THAT "PIP" REMAINS INACTIVE.

SEARCH AT OFFSET TEST

PURPOSE:

TO VERIFY THAT OFFSET MODE DOES NOT CAUSE SEARCH ERRORS.

PROCEDURE:

THE TEST EXECUTES AN OFFSET COMMAND, THEN EXECUTES A SEARCH COMMAND, VERIFYING THAT THERE ARE NO ERRORS DURING THE SEARCH.

1312
1313
1314
1315

1
478
479

:PROGRAM REVISION #001

.TITLE CZRMAO RM05/3/2 FCTNL TST 1

.*COPYRIGHT (C) 1980
.*DIGITAL EQUIPMENT CORP.
.*MAYNARD, MASS. 01754

.*PROGRAM BY MIKE LEAVITT

.*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
.*PACKAGE (MAINDEC-11-DZQAC-C4), 1980.

480

.SBTTL OPERATIONAL SWITCH SETTINGS

481

SWITCH	USE
15	HALT ON ERROR
14	LOOP ON TEST
13	INHIBIT ERROR TYPEOUTS
12	UNUSED
11	INHIBIT ITERATIONS
10	BELL ON ERROR
9	LOOP ON ERROR
8	LOOP ON TEST IN SWR<7:0>
7	TN128
6	TN64
5	TN32
4	TN16
3	TN8
2	TN4
1	TN2
0	TN1

482
483

.SBTTL BASIC DEFINITIONS

001100
104000
000004

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

000011
000012
000015
000200
177776
177776
177774
177772
177570
177570

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

000000
000001
000002
000003

.*GENERAL PURPOSE REGISTER DEFINITIONS
R0 = %0 ;:GENERAL REGISTER
R1 = %1 ;:GENERAL REGISTER
R2 = %2 ;:GENERAL REGISTER
R3 = %3 ;:GENERAL REGISTER

```
000004 R4 = %4 ;;GENERAL REGISTER
000005 R5 = %5 ;;GENERAL REGISTER
000006 R6 = %6 ;;GENERAL REGISTER
000007 R7 = %7 ;;GENERAL REGISTER
000006 SP = %6 ;;STACK POINTER
000007 PC = %7 ;;PROGRAM COUNTER
```

;*PRIORITY LEVEL DEFINITIONS

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

;*SWITCH REGISTER SWITCH DEFINITIONS

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

;*DATA BIT DEFINITIONS (BIT00 TO BIT15)

```
100000 BIT15 = 100000
040000 BIT14 = 40000
020000 BIT13 = 20000
010000 BIT12 = 10000
004000 BIT11 = 4000
002000 BIT10 = 2000
001000 BIT09 = 1000
000400 BIT08 = 400
000200 BIT07 = 200
000100 BIT06 = 100
000040 BIT05 = 40
```

```

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

```

;*BASIC "CPU" TRAP VECTOR ADDRESSES
000004 ERRVEC = 4 ;;TIME OUT AND OTHER ERRORS
000010 RESVEC = 10 ;;RESERVED AND ILLEGAL INSTRUCTIONS
000014 TBITVEC = 14 ;;"T" BIT
000014 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
    
```

.SBTTL RM REGISTER BIT DEFINITIONS

;*RMCS1 CONTROL STATUS REGISTER

```

484 004000 DVA = BIT11 ;DEVICE AVAILABLE-READ ONLY
485 000040 F4 = BIT05 ;FUNCTION CODE
486 000020 F3 = BIT04 ;FUNCTION CODE
487 000010 F2 = BIT03 ;FUNCTION CODE
488 000004 F1 = BIT02 ;FUNCTION CODE
489 000002 F0 = BIT01 ;FUNCTION CODE
490 000001 GO = BIT00 ;GO BIT
491 000077 FNCMSK = 000077 ;FUNCTION CODE MASK
    
```

;*FUNCTION CODES (BITS 01-05 OF RMCS1)

```

498 000000 NOP = 000000 ;NOP COMMAND
499 000002 ILF02 = 000002 ;ILLEGAL COMMAND
500 000004 SEEK = 000004 ;SEEK COMMAND
501 000006 RECAL = 000006 ;RECALIBRATE COMMAND
502 000010 DRVCLR = 000010 ;DRIVE CLEAR COMMAND
503 000012 RELEASE = 000012 ;RELEASE COMMAND
504 000014 OFFSET = 000014 ;OFFSET COMMAND
505 000016 RTC = 000016 ;RETURN TO CENTERLINE COMMAND
506 000020 RIP = 000020 ;READ IN PRESET COMMAND
507 000022 PAKACK = 000022 ;PACK ACKNOWLEDGE COMMAND
508 000022 PACACK = PAKACK
509 000024 ILF24 = 000024 ;ILLEGAL COMMAND
510 000026 ILF26 = 000026 ;ILLEGAL COMMAND
511
    
```

```

512      000030      SEARCH = 000030      ;SEARCH COMMAND
515      000030      ILF30  = 000030      ;ILLEGAL COMMAND
          000032      ILF32  = 000032      ;ILLEGAL COMMAND
          000034      ILF34  = 000034      ;ILLEGAL COMMAND
          000036      ILF36  = 000036      ;ILLEGAL COMMAND
          000040      ILF40  = 000040      ;ILLEGAL COMMAND
          000042      ILF42  = 000042      ;ILLEGAL COMMAND
          000044      ILF44  = 000044      ;ILLEGAL COMMAND
          000046      ILF46  = 000046      ;ILLEGAL COMMAND
516      000050      WCD     = 000050      ;WRITE CHECK DATA COMMAND
517      000052      WCH     = 000052      ;WRITE CHECK HEADER AND DATA
518      000054      ILF54  = 000054      ;ILLEGAL COMMAND
519      000056      ILF56  = 000056      ;ILLEGAL COMMAND
520      000060      WD      = 000060      ;WRITE DATA COMMAND
521      000062      WH      = 000062      ;WRITE HEADER AND DATA COMMAND
522      000064      ILF64  = 000064      ;ILLEGAL COMMAND
523      000066      ILF66  = 000066      ;ILLEGAL COMMAND
524      000070      RD      = 000070      ;READ DATA COMMAND
525      000072      RH      = 000072      ;READ HEADER AND DATA COMMAND
526      000074      ILF74  = 000074      ;ILLEGAL COMMAND
527      000076      ILF76  = 000076      ;ILLEGAL COMMAND
528
529      ;*RMDA DISK ADDRESS REGISTER
530
531      ;TRACK ADDRESS DEFINITIONS
532      010000      TA16   = BIT12      ;TRACK ADDRESS 16.
533      004000      TA8    = BIT11      ;TRACK ADDRESS 8.
534      002000      TA4    = BIT10      ;TRACK ADDRESS 4
535      001000      TA2    = BIT09      ;TRACK ADDRESS 2
536      000400      TA1    = BIT08      ;TRACK ADDRESS 1
537
538      ;SECTOR ADDRESS DEFINITIONS
539      000020      SA16   = BIT04      ;SECTOR ADDRESS 16.
540      000010      SA8    = BIT03      ;SECTOR ADDRESS 8.
541      000004      SA4    = BIT02      ;SECTOR ADDRESS 4
542      000002      SA2    = BIT01      ;SECTOR ADDRESS 2
543      000001      SA1    = BIT00      ;SECTOR ADDRESS 1
544
545      ;TRACK & SECTOR MASKS
546      177400      TADMSK = 177400     ;TRACK ADDRESS MASK
547      000377      SADMSK = 000377     ;SECTOR ADDRESS MASK
548
549      ;*RMDS DRIVE STATUS REGISTER
550
551      100000      ATA     = BIT15      ;ATTENTION ACTIVE
552      040000      ERR     = BIT14      ;COMPOSITE ERROR
553      020000      PIP     = BIT13      ;POSITIONING IN PROGRESS
554      010000      MOL     = BIT12      ;MEDIUM ON LINE
555      004000      WRL     = BIT11      ;WRITE LOCK
556      002000      LBT     = BIT10      ;LAST BLOCK TRANSFERRED
557      001000      PGM     = BIT09      ;PROGRAMMABLE
558      000400      DPR     = BIT08      ;DRIVE PRESENT
559      000200      DRY     = BIT07      ;DRIVE READY
560      000100      VV      = BIT06      ;VOLUME VALID
561      000001      OM      = BIT00      ;OFFSET MODE ACTIVE
562
563      ;*RMER! ERROR REGISTER #1

```



```

564
565      100000      DCK      = BIT15      ;DATA CHECK ERROR
566      040000      UNS      = BIT14      ;DRIVE UNSAFE
567      020000      OPI      = BIT13      ;OPERATION INCOMPLETE
568      010000      DTE      = BIT12      ;DRIVE TIMING ERROR
569      004000      WLE      = BIT11      ;WRITE LOCK ERROR
570      002000      IAE      = BIT10      ;INVALID ADDRESS ERROR
571      001000      AOE      = BIT09      ;ADDRESS OVERFLOW ERROR
572      000400      HCRC     = BIT08      ;HEADER CRC ERROR
573      000200      HCE      = BIT07      ;HEADER COMPARE ERROR
574      000100      ECH      = BIT06      ;ECC "HARD" ERROR
575      000040      WCF      = BIT05      ;WRITE CLOCK FAILURE
576      000020      FER      = BIT04      ;FORMAT ERROR
577      000010      PAR      = BIT03      ;PARITY ERROR
578      000004      RMR      = BIT02      ;REGISTER MODIFICATION REFUSED
579      000002      ILR      = BIT01      ;ILLEGAL REGISTER
580      000001      ILF      = BIT00      ;ILLEGAL FUNCTION
581
582      115760      NDTMSK   = DCK!DTE!WLE!AOE!HCRC!HCE!ECH!WCF!FER
583      ;"NDTMSK" IS USED TO MASK ERROR REGISTER 1 DURING NON - DATA
584      ;COMMANDS, I.E., HOUSEKEEPING AND POSITIONING COMMANDS
585
586      ;*RMAS ATTENTION SUMMARY REGISTER
587
588      000377      ATNMSK   = 377      ;MASK FOR ATTENTION BITS
589
590      ;*RMLA LOOK AHEAD REGISTER
591
592      002000      SC4      = BIT10      ;SECTOR COUNT = 16
593      001000      SC3      = BIT09      ;SECTOR COUNT = 8
594      000400      SC2      = BIT08      ;SECTOR COUNT = 4
595      000200      SC1      = BIT07      ;SECTOR COUNT = 2
596      000100      SC0      = BIT06      ;SECTOR COUNT = 1
597
598      003700      SCTMSK   = 003700    ;SECTOR COUNT MASK
599
600      ;*RMMR1 MAINTENANCE REGISTER #1
601
602      ;WRITE ONLY BITS
603      100000      DBCK     = BIT15      ;DEBUG CLOCK
604      040000      DBEN     = BIT14      ;DEBUG CLOCK ENABLE
605      020000      DEBL     = BIT13      ;DIAGNOSTIC END OF BLOCK
606      010000      DTO      = BIT12      ;DIAGNOSTIC TIMEOUT
607      004000      MCLK     = BIT11      ;MAINTENANCE CLOCK
608      002000      MRD      = BIT10      ;READ DATA
609      001000      MUR      = BIT09      ;UNIT READY
610      000400      MOC      = BIT08      ;ON CYLINDER
611      000200      MSER     = BIT07      ;SEEK ERROR
612      000100      MDF      = BIT06      ;DRIVE FAULT
613      000040      MS       = BIT05      ;SECTOR PULSE
614      000010      MWP      = BIT03      ;WRITE PROTECT
615      000004      MI       = BIT02      ;INDEX PULSE
616      000002      MSC      = BIT01      ;SECTOR COMPARE
617      000001      DMD      = BIT00      ;DIAGNOSTIC MODE
618
619      ;READ ONLY BITS
620      100000      OCC      = BIT15      ;OCCUPIED

```

621	040000	RG	= BIT14	;RUN AND GO
622	020000	EBL	= BIT13	;END OF BLOCK
623	010000	REX	= BIT12	;EXCEPTION
624	004000	ESRC	= BIT11	;ENABLE SEARCH
625	002000	PLFS	= BIT10	;LOOKING FOR SYNC
626	001000	ECRC	= BIT09	;ENABLE CRC OUT
627	000400	PDA	= BIT08	;DATA AREA
628	000200	PHA	= BIT07	;HEADER AREA
629	000100	CONT	= BIT06	;CONTINUE
630	000040	WC	= BIT05	;WORD CLOCK
631	000020	EECC	= BIT04	;ENABLE ECC OUT
632	000010	MWD	= BIT03	;WRITE DATA BIT
633	000004	LS	= BIT02	;LAST SECTOR
634	000002	LST	= BIT01	;LAST SECTOR AND TRACK
635	000001	DMD	= BIT00	;DIAGNOSTIC MODE
636				
637		;*RMDT DRIVE TYPE REGISTER		
638				
639	100000	NSA	= BIT15	;NOT SECTOR ADDRESSED = 0
640	040000	TAP	= BIT14	;TAPE DRIVE = 0
641	020000	MOH	= BIT13	;MOVING HEAD = 1
642	004000	DRQ	= BIT11	;DRIVE REQUEST REQUIRED
643				
644	020024	SNGPRT	= 020024	;SINGLE PORT DRIVE TYPE (RM02)
645	024024	DULPRT	= 024024	;DUAL PORT DRIVE TYPE (RM02)
646				
647		;*RMOF OFFSET REGISTER		
648				
649	010000	FMT16	= BIT12	;16 BIT WORD FORMAT
650	004000	ECI	= BIT11	;ECC INHIBIT
651	002000	HCI	= BIT10	;HEADER COMPARE INHIBIT
652	000200	OFD	= BIT07	;OFFSET FORWARD
653				
654		;*RMDC DESIRED CYLINDER ADDRESS REGISTER		
655				
656	001777	CYLMSK	= 001777	;MASK FOR CYLINDER ADDRESS
657				
658		;*RMMR2 MAINTENANCE REGISTER #2		
659				
660		;READ ONLY BITS		
661	100000	RQA	= BIT15	;PORT A REQUEST
662	040000	RQB	= BIT14	;PORT B REQUEST
663	020000	TAG	= BIT13	;TAG CONTROL
664	010000	TST	= BIT12	;COMMAND SEQUENCE TEST BIT
665	004000	CC	= BIT11	;CONTROL OR CYLINDER TAG
666	002000	CH	= BIT10	;CONTROL OR HEAD TAG
669	001000	BB09	= BIT09	;TAG BUS
	000400	BB08	= BIT08	;TAG BUS
	000200	BB07	= BIT07	;TAG BUS
	000100	BB06	= BIT06	;TAG BUS
	000040	BB05	= BIT05	;TAG BUS
	000020	BB04	= BIT04	;TAG BUS
	000010	BB03	= BIT03	;TAG BUS
	000004	BB02	= BIT02	;TAG BUS
	000002	BB01	= BIT01	;TAG BUS
	000001	BB00	= BIT00	;TAG BUS

670

```

671          ;*RMER2 ERROR REGISTER 2
672
673          100000      BSE      = BIT15      ;BAD SECTOR ERROR
674          040000      SKI      = BIT14      ;SEEK INCOMPLETE
675          020000      OPE      = BIT13      ;OPERATOR PLUG ERROR
676          010000      IVC      = BIT12      ;INVALID COMMAND ERROR
677          004000      LSC      = BIT11      ;LOSS OF SYSTEM CLOCK
678          002000      LBC      = BIT10      ;LOSS OF BIT CLOCK
679          000200      DVC      = BIT07      ;DEVICE CHECK
680          000010      DPE      = BIT03      ;DATA PARITY ERROR
681
682          .SBTTL  PROGRAM MNEMONICS
683
684          100000      MSE      = BIT15      ;MANUFACTURING DETECTED SECTOR ERROR
685          040000      USE      = BIT14      ;USER DETECTED SECTOR ERROR
686
687          .SBTTL  RM REGISTER INDEX VALUES
688
689          000000      RMCS1    = 00          ;CONTROL STATUS REGISTER #1
690          000006      RMDA     = 06          ;DISK ADDRESS REGISTER
691          000012      RMDS     = 12          ;DRIVE STATUS REGISTER
692          000014      RMER1    = 14          ;ERROR REGISTER #1
693          000016      RMAS     = 16          ;ATTENTION SUMMARY REGISTER
694          000020      RMLA     = 20          ;LOOK AHEAD REGISTER
695          000024      RMMR1    = 24          ;MAINTENANCE REGISTER
696          000026      RMDT     = 26          ;DRIVE TYPE REGISTER
697          000030      RMSN     = 30          ;SERIAL NUMBER REGISTER
698          000032      RMOF     = 32          ;OFFSET REGISTER
699          000034      RMDC     = 34          ;DESIRED CYLINDER REGISTER
700          000036      RMHR     = 36          ;HOLDING REGISTER
701          000040      RMMR2    = 40          ;MAINTENANCE REGISTER #2
702          000042      RMER2    = 42          ;ERROR REGISTER #2
703          000044      RMEC1    = 44          ;ECC POSITION REGISTER
704          000046      RMEC2    = 46          ;ECC PATTERN REGISTER
705          000050      ILRG50    = 50          ;ILLEGAL REGISTER 50
706          000052      ILRG52    = 52          ;ILLEGAL REGISTER 52
707          000054      ILRG54    = 54          ;ILLEGAL REGISTER 54
708          000056      ILRG56    = 56          ;ILLEGAL REGISTER 56
709          000060      ILRG60    = 60          ;ILLEGAL REGISTER 60
710          000062      ILRG62    = 62          ;ILLEGAL REGISTER 62
711          000064      ILRG64    = 64          ;ILLEGAL REGISTER 64
712          000066      ILRG66    = 66          ;ILLEGAL REGISTER 66
713          000070      ILRG70    = 70          ;ILLEGAL REGISTER 70
714          000072      ILRG72    = 72          ;ILLEGAL REGISTER 72
715          000074      ILRG74    = 74          ;ILLEGAL REGISTER 74
716          000076      ILRG76    = 76          ;ILLEGAL REGISTER 76
717
718          000077      IDXMSK    = 77          ;MASK FOR REGISTER INDEX NUMBER
719
720          .SBTTL  RM CONTROLLER REGISTER BIT DEFINITIONS
721
722          ;*RMCS1 CONTROL STATUS REGISTER #1
723
724          100000      SC        = BIT15      ;SPECIAL CONDITION-READ ONLY
725          040000      TRE        = BIT14      ;TRANSFER ERROR
726          020000      MCPE       = BIT13      ;MASSBUS CONTROL BUS PARITY ERROR-READ ONLY
727          002000      PSEL       = BIT10      ;PORT B SELECT
  
```

```

719      001000      A17      = BIT09      ;ADDRESS EXTENSION
720      000400      A16      = BIT08      ;ADDRESS EXTENSION
721      000200      RDY      = BIT07      ;READY-READ ONLY
722      000100      IE       = BIT06      ;INTERRUPT ENABLE
723
724      ;*RMCS2 RH CONTROL STATUS REGISTER #2
725
726      100000      DLT       = BIT15      ;DATA LATE-READ ONLY
727      040000      WCE       = BIT14      ;WRITE CHECK ERROR-READ ONLY
728      020000      UPE       = BIT13      ;UNIBUS PARITY ERROR
729      010000      NED       = BIT12      ;NONEXISTANT DRIVE-READ ONLY
730      004000      NEM       = BIT11      ;NONEXISTANT MEMORY-READ ONLY
731      002000      PGE       = BIT10      ;PROGRAM ERROR-READ ONLY
732      001000      MXF       = BIT09      ;MISSED TRANSFER
733      000400      MDPE      = BIT08      ;MASSBUS DATA BUS PARITY ERROR-READ ONLY
734      000200      OR        = BIT07      ;OUTPUT READY-READ ONLY
735      000100      IR        = BIT06      ;INPUT READY-READ ONLY
736      000040      CLR       = BIT05      ;CONTROLLER CLEAR
737      000020      PAT       = BIT04      ;PARITY TEST
738      000010      BAI       = BIT03      ;UNIBUS ADDRESS INCREMENT INHIBIT
741      000004      U2        = BIT02      ;UNIT SELECT
          000002      U1        = BIT01      ;UNIT SELECT
          000001      U0        = BIT00      ;UNIT SELECT

742
743      ;UNIT SELECT MASK
744
745      000007      UNTMSK    = 7          ;UNIT SELECT MASK
746
747      ;*RMCS3 RH70 CONTROL STATUS REGISTER #3
748
749      100000      APE       = BIT15      ;ADDRESS PARITY ERROR
750      040000      DPEHI     = BIT14      ;DATA PARITY ERROR HIGH WORD
751      020000      DPELO     = BIT13      ;DATA PARITY ERROR LOW WORD
752      010000      WCEHI     = BIT12      ;WRITE CHECK ERROR HIGH WORD
753      004000      WCELO     = BIT11      ;WRITE CHECK ERROR LOW WORD
754      002000      DBL       = BIT10      ;DOUBLE WORD TRANSFER
755      000100      IE       = BIT06      ;INTERRUPT ENABLE
756      000010      IPCK3     = BIT03      ;INVERT PARITY CHECK
757      000004      IPCK2     = BIT02      ;INVERT PARITY CHECK
758      000002      IPCK1     = BIT01      ;INVERT PARITY CHECK
759      000001      IPCK0     = BIT00      ;INVERT PARITY CHECK
760
761      .SBTTL  RH11/RH70 CONTROLLER REGISTER INDEX VALUES
762
763      000000      RMCS1     = 00          ;CONTROL, STATUS REGISTER #1
764      000002      RMWC      = 02          ;WORD COUNT REGISTER
765      000004      RMBA      = 04          ;BUS ADDRESS REGISTER
766      000010      RMCS2     = 10          ;CONTROL, STATUS REGISTER #2
767      000022      RMDB      = 22          ;DATA BUFFER
768      000050      RMBAE     = 50          ;BUS ADDRESS EXTENSION
769      000052      RMCS3     = 52          ;CONTROL, STATUS REGISTER #3
770
771      176700      ABASE     = 176700      ;UNIBUS ADDRESS
772      120254      AVECT1    = 120254      ;UNIBUS VECTOR ADDRESS AND PRIORITY
773
775
776      .SBTTL  TRAP CATCHER

```

```

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

.SBTTL STARTING ADDRESS(ES)
000200 000137 005412  JMP @#START          ;;JUMP TO STARTING ADDRESS OF PROGRAM
777
778

.SBTTL ACT11 HOOKS
;*****
;HOOKS REQUIRED BY ACT11
000204          $SVPC=.          ;SAVE PC
000046 000046          .=46
042326          $ENDAD          ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
000052 000052          .=52
000000          .WORD 0          ;;2)SET LOC.52 TO ZERO
000204          .=$SVPC          ;; RESTORE PC

779
780          .=1100
781          .SBTTL APT PARAMETER BLOCK

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

001100          $APTHD:
001100 000000          $HIBTS: .WORD 0          ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
001102 001222          $MBADR: .WORD $MAIL      ;;ADDRESS OF APT MAILBOX (BITS 0-15)
001104 000001          $TSTM:  .WORD 1          ;;RUN TIM OF LONGEST TEST
001106 000002          $PASTM: .WORD 2          ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
001110 000002          $UNITM: .WORD 2          ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDED UNIT
001112 000042          .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
782          TAGADR=.

```

0

.SBTTL COMMON TAGS

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

001114	001114				
001114	000000		\$CMTAG: .WORD 0	::START OF COMMON TAGS	
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		\$TMP1: .WORD 0	::USER DEFINED	
001200	000000		\$TMP2: .WORD 0	::USER DEFINED	
001202	000000		\$TMP3: .WORD 0	::USER DEFINED	
001204	000000		\$TMP4: .WORD 0	::USER DEFINED	
001206	000000		\$TIMES: 0	::MAX. NUMBER OF ITERATIONS	
001210	000000		\$ESCAPE: 0	::ESCAPE ON ERROR ADDRESS	
001212	207	377	\$BELL: .ASCIZ <207><377><377>	::CODE FOR BELL	
001216	077		\$QUES: .ASCII /?/	::QUESTION MARK	
001217	015		\$CRLF: .ASCII <15>	::CARRIAGE RETURN	
001220	012	000	\$LF: .ASCIZ <12>	::LINE FEED	

 .SBTTL APT MAILBOX-ETABLE

001222			.EVEN		
001222	000000		\$MAIL: .WORD	::APT MAILBOX	
001224	000000		\$MSGTY: .WORD	AMSGTY	::MESSAGE TYPE CODE
001226	000000		\$FATAL: .WORD	AFATAL	::FATAL ERROR NUMBER
			\$TESTN: .WORD	ATESTN	::TEST NUMBER

001230	000000	\$PASS: .WORD	APASS	::PASS COUNT
001232	000000	\$DEVCT: .WORD	ADEVCT	::DEVICE COUNT
001234	000000	\$UNIT: .WORD	AUNIT	::I/O UNIT NUMBER
001236	000000	\$MSGAD: .WORD	AMSGAD	::MESSAGE ADDRESS
001240	000000	\$MSGLG: .WORD	AMSGLG	::MESSAGE LENGTH
001242		\$ETABLE:		::APT ENVIRONMENT TABLE
001242	000	\$ENV: .BYTE	AENV	::ENVIRONMENT BYTE
001243	000	\$ENVM: .BYTE	AENVM	::ENVIRONMENT MODE BITS
001244	000000	\$SWREG: .WORD	ASWREG	::APT SWITCH REGISTER
001246	000000	\$USWR: .WORD	AUSWR	::USER SWITCHES
001250	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
001252	000	\$MAMS1: .BYTE	AMAMS1	::HIGH ADDRESS,M.S. BYTE
001253	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
001254	000000	\$MADR1: .WORD	AMADR1	::HIGH ADDRESS,BLK#1
		*		MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
001256	000	\$MAMS2: .BYTE	AMAMS2	::HIGH ADDRESS,M.S. BYTE
001257	000	\$MTYP2: .BYTE	AMTYP2	::MEM.TYPE,BLK#2
001260	000000	\$MADR2: .WORD	AMADR2	::MEM.LAST ADDRESS,BLK#2
001262	000	\$MAMS3: .BYTE	AMAMS3	::HIGH ADDRESS,M.S.BYTE
001263	000	\$MTYP3: .BYTE	AMTYP3	::MEM.TYPE,BLK#3
001264	000000	\$MADR3: .WORD	AMADR3	::MEM.LAST ADDRESS,BLK#3
001266	000	\$MAMS4: .BYTE	AMAMS4	::HIGH ADDRESS,M.S.BYTE
001267	000	\$MTYP4: .BYTE	AMTYP4	::MEM.TYPE,BLK#4
001270	000000	\$MADR4: .WORD	AMADR4	::MEM.LAST ADDRESS,BLK#4
001272	120254	\$VECT1: .WORD	AVECT1	::INTERRUPT VECTOR#1,BUS PRIORITY#1
001274	000000	\$VECT2: .WORD	AVECT2	::INTERRUPT VECTOR#2BUS PRIORITY#2
001276	176700	\$BASE: .WORD	ABASE	::BASE ADDRESS OF EQUIPMENT UNDER TEST
001300	000000	\$DEVN: .WORD	ADEVN	::DEVICE MAP
001302	000000	\$CDW1: .WORD	ACDW1	::CONTROLLER DESCRIPTION WORD#1
001304	000000	\$CDW2: .WORD	ACDW2	::CONTROLLER DESCRIPTION WORD#2
001306	000000	\$DDW0: .WORD	ADDW0	::DEVICE DESCRIPTOR WORD#0
001310	000000	\$DDW1: .WORD	ADDW1	::DEVICE DESCRIPTOR WORD#1
001312	000000	\$DDW2: .WORD	ADDW2	::DEVICE DESCRIPTOR WORD#2
001314	000000	\$DDW3: .WORD	ADDW3	::DEVICE DESCRIPTOR WORD#3
001316	000000	\$DDW4: .WORD	ADDW4	::DEVICE DESCRIPTOR WORD#4
001320	000000	\$DDW5: .WORD	ADDW5	::DEVICE DESCRIPTOR WORD#5
001322	000000	\$DDW6: .WORD	ADDW6	::DEVICE DESCRIPTOR WORD#6
001324	000000	\$DDW7: .WORD	ADDW7	::DEVICE DESCRIPTOR WORD#7
001326		\$ETEND:		
		.MEXIT		

0

.SBTTL USER DEFINED TAGS

001326 000000
001330 000000

001332 000
001333 000

CTLFG: .WORD 0 ;CONTAINS CONTROL-C FLAG
XXDP: .WORD 0 ;THE LOW BYTE CONTAINS THE DRIVE NUMBER FROM WHICH
;THE PROGRAM WAS LOADED. THE HIGH BYTE CONTAINS THE
;'XXDP' DEVICE CODE FOR THE RM05/3/2.
LSTRK: .BYTE 0 ;LO BYTE = 0
;HI BYTE, CONTAINS LAST TRACK ADDRESS OF UNIT
;UNDER TEST. RM02/3 = 4., RM05 = 18.

;THE REGISTER INPUT BUFFER IS USED FOR
;STORING DRIVE STATUS

001334

GETBUF:

001334 000000
001336 000000
001340 000000
001342 000000
001344 000000
001346 000000
001350 000000
001352 000000
001354 000000
001356 000000
001360 000000
001362 000000
001364 000000
001366 000000
001370 000000
001372 000000
001374 000000
001376 000000
001400 000000
001402 000000
001404 000000
001406 000000

;REGISTER INPUT BUFFER
RMCS1I: .WORD 0 ;CONTROL, STATUS REGISTER #1
RMWCI: .WORD 0 ;WORD COUNT REGISTER
RMBAI: .WORD 0 ;BUS ADDRESS REGISTER
RMDAI: .WORD 0 ;DISK ADDRESS REGISTER
RMCS2I: .WORD 0 ;CONTROL, STATUS REGISTER #2
RMDSI: .WORD 0 ;DRIVE STATUS REGISTER
RMER1I: .WORD 0 ;ERROR REGISTER #1
RMAI: .WORD 0 ;ATTENTION SUMMARY REGISTER
RMLAI: .WORD 0 ;LOOK AHEAD REGISTER
RMDBI: .WORD 0 ;DATA BUFFER
RMMR1I: .WORD 0 ;MAINTENANCE REGISTER #1
RMDTI: .WORD 0 ;DRIVE TYPE REGISTER
RMSNI: .WORD 0 ;SERIAL NUMBER REGISTER
RMOFI: .WORD 0 ;OFFSET REGISTER
RMDCI: .WORD 0 ;DESIRED CYLINDER REGISTER
RMHRI: .WORD 0 ;HOLDING REGISTER
RMMR2I: .WORD 0 ;MAINTENANCE REGISTER #2
RMER2I: .WORD 0 ;ERROR REGISTER #2
RMEC1I: .WORD 0 ;ECC POSITION REGISTER
RMEC2I: .WORD 0 ;ECC PATTERN REGISTER
RMBAEI: .WORD 0 ;BUS ADDRESS EXTENSION REGISTER
RMCS3I: .WORD 0 ;CONTROL, STATUS REGISTER #3

;THE REGISTER OUTPUT BUFFER IS USED FOR
;ASSEMBLING DATA GOING TO REGISTER

001410

PUTBUF:

001410 000000
001412 000000
001414 000000
001416 000000
001420 000000
001422 000000
001424 000000
001426 000000
001430 000000
001432 000000
001434 000000
001436 000000

;REGISTER OUTPUT BUFFER
RMCS1O: .WORD 0 ;CONTROL, STATUS REGISTER #1
RMWCO: .WORD 0 ;WORD COUNT REGISTER
RMBAO: .WORD 0 ;BUS ADDRESS REGISTER
RMDAO: .WORD 0 ;DISK ADDRESS REGISTER
RMCS2O: .WORD 0 ;CONTROL, STATUS REGISTER #2
RMDSO: .WORD 0 ;DRIVE STATUS REGISTER
RMER1O: .WORD 0 ;ERROR REGISTER #1
RMAO: .WORD 0 ;ATTENTION SUMMARY REGISTER
RMLAO: .WORD 0 ;LOOK AHEAD REGISTER
RMDBO: .WORD 0 ;DATA BUFFER
RMMR1O: .WORD 0 ;MAINTENANCE REGISTER #1
RMDTO: .WORD 0 ;DRIVE TYPE REGISTER

001440 000000
 001442 000000
 001444 000000
 001446 000000
 001450 000000
 001452 000000
 001454 000000
 001456 000000
 001460 000000
 001462 000000

RMSNO: .WORD 0 ;SERIAL NUMBER REGISTER
 RMOFO: .WORD 0 ;OFFSET REGISTER
 RMDCO: .WORD 0 ;DESIRED CYLINDER REGISTER
 RMHRO: .WORD 0 ;HOLDING REGISTER
 RMMR20: .WORD 0 ;MAINTENANCE REGISTER #2
 RMER20: .WORD 0 ;ERROR REGISTER #2
 RMEC10: .WORD 0 ;ECC POSITION REGISTER
 RMEC20: .WORD 0 ;ECC PATTERN REGISTER
 RMBAE0: .WORD 0 ;BUS ADDRESS EXTENSION REGISTER
 RMCS30: .WORD 0 ;CONTROL, STATUS REGISTER #3

;EACH WORD OF THE TEST QUE CONTAINS THE DEVICE NUMBER IN
 ;THE LOW BYTE AND THE ATTENTION BIT IN THE HIGH BYTE. THE
 ;FIRST WORD CONTAINS THE ADDRESS OF THE DEVICE UNDER TEST
 ;IN THE TABLE. A ZERO WORD IS A BLANK AND REPRESENTS THE
 ;END OF THE QUE.

001464 000000
 001506 000000

TSTQUE: .WORD 0 ;CONTAINS DEVICE POINTER
 .BLKW 8. ;TEST QUE FOR DEVICES UNDER TEST
 .WORD 0 ;TABLE TERMINATOR GOES HERE WHEN
 ;ALL 8. DEVICES ARE UNDER TEST.

001510 000000
 001512 000000

CLKADR: .WORD 0 ;UNIBUS ADDRESS OF KW11 CLOCK
 CLKVCT: .WORD 0 ;VECTOR ADDRESS OF KW11 CLOCK

001514

;THE GET INDEX TABLE CONTAINS A BYTE LIST OF REGISTERS WHICH
 ;ARE READ BY THE GET SUBROUTINE. THE LIST IS TERMINATED BY
 ;A NEGATIVE BYTE.
 GETINX: .BLKB 23. ;GET INDEX TABLE

001543

;THE PUT INDEX TABLE ICONTAINS A BYTE LIST OF REGISTERS WHICH
 ;ARE WRITTEN BY THE PUT SUBROUTINE. THE LIST IS TERMINATED BY
 ;A NEGATIVE BYTE.
 PUTINX: .BLKB 23. ;PUT INDEX TABLE

;PUT TAGS HERE

0

.SBTTL ERROR POINTER TABLE

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

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

001572

\$ERRTB:

1
2
3
4

;ERROR 1 WRONG UNIT SELECTED

001572 067604
001574 073676
001576 074022
001600 074112

EMT1
EHT1
EDT1
EFT1

5
6
7

;ERROR 2 DEVICE WENT UNAVAILABLE

001602 067610
001604 073676
001606 074022
001610 074112

EMT2
EHT1
EDT1
EFT1

8
9
10

;ERROR 3 DEVICE WENT NONEXISTENT

001612 067616
001614 073676
001616 074022
001620 074112

EMT3
EHT1
EDT1
EFT1

11
12
13

;ERROR 4 CONTROLLER NOT READY

001622 067624
001624 073676
001626 074022
001630 074112

EMT4
EHT1
EDT1
EFT1

14
15
16

;ERROR 5 DRIVE NOT READY AND GO NOT RESET

001632 067632
001634 073676
001636 074022
001640 074112

EMT5
EHT1
EDT1
EFT1

17

Line	Address	Value	Error Code	Description
18			:ERROR 6	UNEXPECTED VALUE FOR "ATA" STATUS
19	001642	067640	EMT6	
	001644	073676	EHT1	
	001646	074022	EDT1	
	001650	074112	EFT1	
20			:ERROR 7	BUS TIMEOUT TRYING TO READ OR WRITE REGISTER
21				
22	001652	067646	EMT7	
	001654	000000	0	
	001656	000000	0	
	001660	000000	0	
23			:ERROR 10	DRIVE NOT READY BUT GO IS RESET
24				
25	001662	067654	EMT10	
	001664	073676	EHT1	
	001666	074022	EDT1	
	001670	074112	EFT1	
26			:ERROR 11	GO NOT RESET BUT DRIVE IS READY
27				
28	001672	067660	EMT11	
	001674	073676	EHT1	
	001676	074022	EDT1	
	001700	074112	EFT1	
29			:ERROR 12	INCORRECT FUNCTION CODE
30				
31	001702	067664	EMT12	
	001704	073676	EHT1	
	001706	074022	EDT1	
	001710	074112	EFT1	
32			:ERROR 13	PARITY ERROR READING REMOTE REGISTERS
33				
34	001712	067672	EMT13	
	001714	073676	EHT1	
	001716	074022	EDT1	
	001720	074112	EFT1	
35			:ERROR 14	TRANSFER ERROR IS INCORRECT
36				
37	001722	067704	EMT14	
	001724	073676	EHT1	
	001726	074022	EDT1	
	001730	074112	EFT1	
38			:ERROR 15	INCORRECT WORD COUNT
39				

40	001732	067712	EMT15	
	001734	073676	EHT1	
	001736	074022	EDT1	
	001740	074112	EFT1	
41				
42				:ERROR 16 INCORRECT BUS ADDRESS
43	001742	067720	EMT16	
	001744	073676	EHT1	
	001746	074022	EDT1	
	001750	074112	EFT1	
44				
45				:ERROR 17 INCORRECT LBT STATUS
46	001752	067730	EMT17	
	001754	073676	EHT1	
	001756	074022	EDT1	
	001760	074112	EFT1	
47				
48				:ERROR 20 INCORRECT AOE
49	001762	067740	EMT20	
	001764	073676	EHT1	
	001766	074022	EDT1	
	001770	074112	EFT1	
50				
51				:ERROR 21 INCORRECT DISK ADDRESS
52	001772	067750	EMT21	
	001774	073676	EHT1	
	001776	074022	EDT1	
	002000	074112	EFT1	
53				
54				:ERROR 22 INCORRECT CYLINDER ADDRESS
55	002002	067760	EMT22	
	002004	073676	EHT1	
	002006	074022	EDT1	
	002010	074112	EFT1	
56				
57				:ERROR 23 INCORRECT WLE STATUS
58	002012	067770	EMT23	
	002014	073676	EHT1	
	002016	074022	EDT1	
	002020	074112	EFT1	
59				
60				:ERROR 24 INCORRECT UPE STATUS
61				

002022	070000	EMT24	
002024	073676	EHT1	
002026	074022	EDT1	
002030	074112	EFT1	
62			
63			
64		;ERROR 25	INCORRECT WCF STATUS
002032	070010	EMT25	
002034	073676	EHT1	
002036	074022	EDT1	
002040	074112	EFT1	
65			
66			
67		;ERROR 26	INCORRECT WCE STATUS
002042	070020	EMT26	
002044	073676	EHT1	
002046	074022	EDT1	
002050	074112	EFT1	
68			
69			
70		;ERROR 27	INCORRECT MDPE STATUS
002052	070030	EMT27	
002054	073676	EHT1	
002056	074022	EDT1	
002060	074112	EFT1	
71			
72			
73		;ERROR 30	INCORRECT DCK STATUS
002062	070040	EMT30	
002064	073676	EHT1	
002066	074022	EDT1	
002070	074112	EFT1	
74			
75			
76		;ERROR 31	INCORRECT ECH STATUS
002072	070050	EMT31	
002074	073676	EHT1	
002076	074022	EDT1	
002100	074112	EFT1	
77			
78			
79		;ERROR 32	DLT SHOULD NOT BE SET
002102	070060	EMT32	
002104	073676	EHT1	
002106	074022	EDT1	
002110	074112	EFT1	
80			
81			
82		;ERROR 33	MXF SHOULD NOT BE SET
002112	070070	EMT33	

	002114	073676	EHT1	
	002116	074022	EDT1	
	002120	074112	EFT1	
83				
84				
85				;ERROR 34 DTE SHOULD NOT BE SET
	002122	070100	EMT34	
	002124	073676	EHT1	
	002126	074022	EDT1	
	002130	074112	EFT1	
86				
87				
88				;ERROR 35 INCORRECT HCRC STATUS
	002132	070110	EMT35	
	002134	073676	EHT1	
	002136	074022	EDT1	
	002140	074112	EFT1	
89				
90				
91				;ERROR 36 INCORRECT HCE STATUS
	002142	070120	EMT36	
	002144	073676	EHT1	
	002146	074022	EDT1	
	002150	074112	EFT1	
92				
93				
94				;ERROR 37 INCORRECT FER STATUS
	002152	070130	EMT37	
	002154	073676	EHT1	
	002156	074022	EDT1	
	002160	074112	EFT1	
95				
96				
97				;ERROR 40 DPE SHOULD NOT BE SET (NOT A DATA COMMAND)
	002162	070140	EMT40	
	002164	073676	EHT1	
	002166	074022	EDT1	
	002170	074112	EFT1	
98				
99				
100				;ERROR 41 LOST 'MOL' DURING PACK ACKNOWLEDGE
	002172	070146	EMT41	
	002174	073676	EHT1	
	002176	074022	EDT1	
	002200	074112	EFT1	
101				
102				
103				;ERROR 42 UNSAFE ERROR DURING PACK ACKNOWLEDGE
	002202	070156	EMT42	
	002204	073676	EHT1	

	002206 074022		EDT1	
	002210 074112		EFT1	
104				
105		;ERROR 43		"OPI" ERROR DURING PACK ACKNOWLEDGE
106				
	002212 070170		EMT43	
	002214 073676		EHT1	
	002216 074022		EDT1	
	002220 074112		EFT1	
107				
108		;ERROR 44		"RMR" ERROR DURING PACK ACKNOWLEDGE
109				
	002222 070200		EMT44	
	002224 073676		EHT1	
	002226 074022		EDT1	
	002230 074112		EFT1	
110				
111		;ERROR 45		"ILR" ERROR DURING PACK ACKNOWLEDGE
112				
	002232 070210		EMT45	
	002234 073676		EHT1	
	002236 074022		EDT1	
	002240 074112		EFT1	
113				
114		;ERROR 46		"ILF" ERROR DURING PACK ACKNOWLEDGE
115				
	002242 070220		EMT46	
	002244 073676		EHT1	
	002246 074022		EDT1	
	002250 074112		EFT1	
116				
117		;ERROR 47		COMPOSITE ERROR STATUS IS INCORRECT
118				
	002252 070230		EMT47	
	002254 073676		EHT1	
	002256 074022		EDT1	
	002260 074112		EFT1	
119				
120		;ERROR 50		PARITY ERROR WRITING REMOTE REGISTERS
121				
	002262 070236		EMT50	
	002264 073676		EHT1	
	002266 074022		EDT1	
	002270 074112		EFT1	
122				
123		;ERROR 51		INCORRECT IAE STATUS DURING SEEK COMMAND
124				
	002272 070246		EMT51	
	002274 073676		EHT1	
	002276 074022		EDT1	

	002300	074112	EFT1	
125				
126			:ERROR 52	OPI ERROR DURING SEEK - MEDIUM IS NOT ON LINE
127				
	002302	070260	EMT52	
	002304	073676	EHT1	
	002306	074022	EDT1	
	002310	074112	EFT1	
128				
129			:ERROR 53	OPI ERROR DURING SEEK - MEDIUM IS ON LINE, ASSUME
130			:	ON CYLINDER LATCH DIDN'T RESET
131				
	002312	070276	EMT53	
	002314	073676	EHT1	
	002316	074022	EDT1	
	002320	074112	EFT1	
132				
133			:ERROR 54	SEEK INCOMPLETE ERROR DURING SEEK COMMAND
134				
	002322	070314	EMT54	
	002324	073676	EHT1	
	002326	074022	EDT1	
	002330	074112	EFT1	
135				
136			:ERROR 55	DEVICE CHECK DURING SEEK COMMAND
137				
	002332	070324	EMT55	
	002334	073676	EHT1	
	002336	074022	EDT1	
	002340	074112	EFT1	
138				
139			:ERROR 56	PIP IS STILL SET AFTER SEEK - SKI IS RESET
140				
	002342	070336	EMT56	
	002344	073676	EHT1	
	002346	074022	EDT1	
	002350	074112	EFT1	
141				
142			:ERROR 57	ATA DID NOT SET DURING SEEK COMMAND
143				
	002352	070354	EMT57	
	002354	073676	EHT1	
	002356	074022	EDT1	
	002360	074112	EFT1	
144				
145			:ERROR 60	IVC ERROR DURING SEEK COMMAND - LOST
146			:	VOLUME VALID
147				
	002362	070364	EMT60	
	002364	073676	EHT1	

	002366	074022		EDT1	
	002370	074112		EFT1	
148					
149			:ERROR	61	ERRONEOUS IVC ERROR DURING SEEK COMMAND -
150			:		VOLUME VALID IS STIL SET
151					
	002372	070402		EMT61	
	002374	073676		EHT1	
	002376	074022		EDT1	
	002400	074112		EFT1	
152					
153			:ERROR	62	MOL IS ZERO, BUT OPI WAS NOT
154			:		REPORTED DURING SEEK COMMAND
155					
	002402	070422		EMT62	
	002404	073676		EHT1	
	002406	074022		EDT1	
	002410	074112		EFT1	
156					
157			:ERROR	63	UNUSED
158					
	002412	000000		0	
	002414	000000		0	
	002416	000000		0	
	002420	000000		0	
159					
160			:ERROR	64	DRIVE DID NOT DETECT "IVC" ERROR DURING SEEK
161					
	002422	070440		EMT64	
	002424	073676		EHT1	
	002426	074022		EDT1	
	002430	074112		EFT1	
162					
163			:ERROR	65	DRIVE EXECUTED A SEEK WITH ERROR SET
164					
	002432	070460		EMT65	
	002434	073676		EHT1	
	002436	074022		EDT1	
	002440	074112		EFT1	
165					
166			:ERROR	66	UNEXPECTED ERROR SET IN RMER1
167					
	002442	070500		EMT66	
	002444	073676		EHT1	
	002446	074022		EDT1	
	002450	074112		EFT1	
168					
169			:ERROR	67	UNEXPECTED ERROR SET IN RMER2
170					
	002452	070512		EMT67	

	002454	073676	EHT1	
	002456	074022	EDT1	
	002460	074112	EFT1	
171				
172				:ERROR 70 ERRONEOUS "IAE" ERROR DURING RECALIBRATE
173				
	002462	070524	EMT70	
	002464	073676	EHT1	
	002466	074022	EDT1	
	002470	074112	EFT1	
174				
175				:ERROR 71 "ILF" ERROR DURING RECALIBRATE
176				
	002472	070534	EMT71	
	002474	073676	EHT1	
	002476	074022	EDT1	
	002500	074112	EFT1	
177				
178				:ERROR 72 "OPI" ERROR DURING RECALIBRATE DUE TO "MOL" = 0
179				
	002502	070544	EMT72	
	002504	073676	EHT1	
	002506	074022	EDT1	
	002510	074112	EFT1	
180				
181				:ERROR 73 "OPI" ERROR DURING RECALIBRATE BECAUSE ON
182				: CYLINDER DIDNT DROP
183				
	002512	070562	EMT73	
	002514	073676	EHT1	
	002516	074022	EDT1	
	002520	074112	EFT1	
184				
185				:ERROR 74 "IVC" ERROR DURING RECALIBRATE - "VV" = 0
186				
	002522	070600	EMT74	
	002524	073676	EHT1	
	002526	074022	EDT1	
	002530	074112	EFT1	
187				
188				:ERROR 75 ERRONEOUS "IVC" ERROR DURING RECALIBRATE - "VV" = 1
189				
	002532	070610	EMT75	
	002534	073676	EHT1	
	002536	074022	EDT1	
	002540	074112	EFT1	
190				
191				:ERROR 76 "SKI" ERROR DURING RECALIBRATE
192				
	002542	070630	EMT76	

	002544 073676	EHT1	
	002546 074022	EDT1	
	002550 074112	EFT1	
193			
194			
195			;ERROR 77 'DVC' OCCURRED DURING RECALIBRATE
	002552 070640	EMT77	
	002554 073676	EHT1	
	002556 074022	EDT1	
	002560 074112	EFT1	
196			
197			
198			;ERROR 100 LOST 'MOL' DURING RECALIBRATE - 'OPI' = 0
	002562 070652	EMT100	
	002564 073676	EHT1	
	002566 074022	EDT1	
	002570 074112	EFT1	
199			
200			
201			;ERROR 101 LOST 'VV' DURING RECALIBRATE - 'IVC' = 0
	002572 070670	EMT101	
	002574 073676	EHT1	
	002576 074022	EDT1	
	002600 074112	EFT1	
202			
203			
204			;ERROR 102 'ATA' DID NOT SET DURING RECALIBRATE
	002602 070706	EMT102	
	002604 073676	EHT1	
	002606 074022	EDT1	
	002610 074112	EFT1	
205			
206			
207			;ERROR 103 'OM' DID NOT RESET DURING RECALIBRATE
	002612 070716	EMT103	
	002614 073676	EHT1	
	002616 074022	EDT1	
	002620 074112	EFT1	
208			
209			
210			;ERROR 104 'PIP' IS STIL SET AFTER RECALIBRATE
	002622 070730	EMT104	
	002624 073676	EHT1	
	002626 074022	EDT1	
	002630 074112	EFT1	
211			
212			
213			;ERROR 105 UNEXPECTED 'ILR' ERROR DURING RECALIBRATE
	002632 070746	EMT105	
	002634 073676	EHT1	

002636	074022	EDT1	
002640	074112	EFT1	
214			
215		:ERROR	106 UNEXPECTED 'RMR' ERROR DURING RECALIBRATE
216			
002642	070756	EMT106	
002644	073676	EHT1	
002646	074022	EDT1	
002650	074112	EFT1	
217			
218		:ERROR	107 'UNS' ERROR DURING RECALIBRATE - AC POWER IS LOW
219			
002652	070766	EMT107	
002654	073676	EHT1	
002656	074022	EDT1	
002660	074112	EFT1	
220			
221		:ERROR	110 CANNOT ACCESS MASSBUS CONTROLLER VIA UNIBUS
222			
002662	071006	EMT110	
002664	073720	EHT110	
002666	074040	EDT110	
002670	074130	EFT110	
223			
224		:ERROR	111 NONEXISTENT DEVICE
225			
002672	071020	EMT111	
002674	073724	EHT111	
002676	074042	EDT111	
002700	074132	EFT111	
226			
227		:ERROR	112 DEVICE NOT AVAILABLE
228			
002702	071026	EMT112	
002704	073724	EHT111	
002706	074042	EDT111	
002710	074132	EFT111	
229			
230		:ERROR	113 BUS TIMEOUT-NED STATUS FAILURE
231			
002712	071034	EMT113	
002714	000000	0	
002716	000000	0	
002720	000000	0	
232			
233		:ERROR	114 DEVICE NOT AN RM05/3/2
234			
002722	071050	EMT114	
002724	073730	EHT114	
002726	074044	EDT114	

002730	074134	EFT114
235		
236		:ERROR 115 RMCS1 NOT INITIALIZED BY UNIBUS
237		
002732	071056	EMT115
002734	073676	EHT1
002736	074022	EDT1
002740	074112	EFT1
238		
239		:ERROR 116 RMBA NOT INITIALIZED BY UNIBUS
240		
002742	071066	EMT116
002744	073676	EHT1
002746	074022	EDT1
002750	074112	EFT1
241		
242		:ERROR 117 RMCS2 NOT INITIALIZED BY UNIBUS
243		
002752	071076	EMT117
002754	073676	EHT1
002756	074022	EDT1
002760	074112	EFT1
244		
245		:ERROR 120 RMER1 NOT INITIALIZED BY UNIBUS
246		
002762	071106	EMT120
002764	073676	EHT1
002766	074022	EDT1
002770	074112	EFT1
247		
248		:ERROR 121 RMAS NOT INITIALIZED BY UNIBUS
249		
002772	071116	EMT121
002774	073676	EHT1
002776	074022	EDT1
003000	074112	EFT1
250		
251		:ERROR 122 RMMR1 NOT INITIALIZED BY UNIBUS
252		
003002	071126	EMT122
003004	073676	EHT1
003006	074022	EDT1
003010	074112	EFT1
253		
254		:ERROR 123 RMDS NOT INITIALIZED BY UNIBUS
255		
003012	071136	EMT123
003014	073676	EHT1
003016	074022	EDT1
003020	074112	EFT1

256			
257			
258		;ERROR 124	RMEC2 NOT INITIALIZED BY UNIBUS
	003022	071146	EMT124
	003024	073676	EHT1
	003026	074022	EDT1
	003030	074112	EFT1
259			
260		;ERROR 125	RMMR2 NOT INITIALIZED BY UNIBUS
261			
	003032	071156	EMT125
	003034	073676	EHT1
	003036	074022	EDT1
	003040	074112	EFT1
262			
263		;ERROR 126	RMCS1 NOT CLEARED BY CONTROLLER CLEAR
264			
	003042	071166	EMT126
	003044	073676	EHT1
	003046	074022	EDT1
	003050	074112	EFT1
265			
266		;ERROR 127	RMBA NOT CLEARED BY CONTROLLER CLEAR
267			
	003052	071200	EMT127
	003054	073676	EHT1
	003056	074022	EDT1
	003060	074112	EFT1
268			
269		;ERROR 130	RMCS2 NOT CLEARED BY CONTROLLER CLEAR
270			
	003062	071212	EMT130
	003064	073676	EHT1
	003066	074022	EDT1
	003070	074112	EFT1
271			
272		;ERROR 131	RMER1 NOT CLEARED BY CONTROLLER CLEAR
273			
	003072	071224	EMT131
	003074	073676	EHT1
	003076	074022	EDT1
	003100	074112	EFT1
274			
275		;ERROR 132	RMAS NOT CLEARED BY CONTROLLER CLEAR
276			
	003102	071236	EMT132
	003104	073676	EHT1
	003106	074022	EDT1
	003110	074112	EFT1

277				
278				
279				
	003112	071250		
	003114	073676		
	003116	074022		
	003120	074112		
			EMT133	
			EHT1	
			EDT1	
			EFT1	
280				
281				
282				
	003122	071262		
	003124	073676		
	003126	074022		
	003130	074112		
			EMT134	
			EHT1	
			EDT1	
			EFT1	
283				
284				
285				
	003132	071274		
	003134	073676		
	003136	074022		
	003140	074112		
			EMT135	
			EHT1	
			EDT1	
			EFT1	
286				
287				
288				
	003142	071306		
	003144	073676		
	003146	074022		
	003150	074112		
			EMT136	
			EHT1	
			EDT1	
			EFT1	
289				
290				
291				
	003152	071320		
	003154	073676		
	003156	074022		
	003160	074112		
			EMT137	
			EHT1	
			EDT1	
			EFT1	
292				
293				
294				
	003162	071330		
	003164	073676		
	003166	074022		
	003170	074112		
			EMT140	
			EHT1	
			EDT1	
			EFT1	
295				
296				
297				
	003172	071340		
	003174	073676		
	003176	074022		
	003200	074112		
			EMT141	
			EHT1	
			EDT1	
			EFT1	

298

299			
300		;ERROR 142	RMDS NOT CLEARED BY DRIVE CLEAR
	003202	071350	EMT142
	003204	073676	EHT1
	003206	074022	EDT1
	003210	074112	EFT1
301			
302		;ERROR 143	RMER1 NOT CLEARED BY DRIVE CLEAR
303			
	003212	071360	EMT143
	003214	073676	EHT1
	003216	074022	EDT1
	003220	074112	EFT1
304			
305		;ERROR 144	RMAS NOT CLEARED BY DRIVE CLEAR
306			
	003222	071370	EMT144
	003224	073676	EHT1
	003226	074022	EDT1
	003230	074112	EFT1
307			
308		;ERROR 145	RMMR1 NOT CLEARED BY DRIVE CLEAR
309			
	003232	071400	EMT145
	003234	073676	EHT1
	003236	074022	EDT1
	003240	074112	EFT1
310			
311		;ERROR 146	RMMR2 NOT CLEARED BY DRIVE CLEAR
312			
	003242	071410	EMT146
	003244	073676	EHT1
	003246	074022	EDT1
	003250	074112	EFT1
313			
314		;ERROR 147	RMER2 NOT CLEARED BY DRIVE CLEAR
315			
	003252	071420	EMT147
	003254	073676	EHT1
	003256	074022	EDT1
	003260	074112	EFT1
316			
317		;ERROR 150	RMEC2 NOT CLEARED BY DRIVE CLEAR
318			
	003262	071430	EMT150
	003264	073676	EHT1
	003266	074022	EDT1
	003270	074112	EFT1
319			
320		;ERROR 151	MEDIUM NOT ON LINE

321	003272 071440	EMT151	
	003274 073676	EHT1	
	003276 074022	EDT1	
	003300 074112	EFT1	
322			
323			;ERROR 152 DRIVE FAULT
324	003302 071452	EMT152	
	003304 073676	EHT1	
	003306 074022	EDT1	
	003310 074112	EFT1	
325			
326			;ERROR 153 UNSAFE SHOULD BE SET BECAUSE DVC IS SET
327	003312 071464	EMT153	
	003314 073676	EHT1	
	003316 074022	EDT1	
	003320 074112	EFT1	
328			
329			;ERROR 154 UNSAFE SHOULD NOT BE SET, AC IS LOW
330	003322 071502	EMT154	
	003324 073676	EHT1	
	003326 074022	EDT1	
	003330 074112	EFT1	
331			
332			;ERROR 155 VOLUME VALID NOT SET BY PACK ACK
333	003332 071520	EMT155	
	003334 073676	EHT1	
	003336 074022	EDT1	
	003340 074112	EFT1	
334			
335			;ERROR 156 OFFSET MODE NOT SET BY OFFSET COMMAND
336	003342 071532	EMT156	
	003344 073676	EHT1	
	003346 074022	EDT1	
	003350 074112	EFT1	
337			
338			;ERROR 157 OFFSET MODE NOT RESET BY RTC COMMAND
339	003352 071544	EMT157	
	003354 073676	EHT1	
	003356 074022	EDT1	
	003360 074112	EFT1	
340			
341			;ERROR 160 RMOF NOT RESET BY RIP COMMAND
342			

003362	071556	EMT160
003364	073676	EHT1
003366	074022	EDT1
003370	074112	EFT1
343		
344	;ERROR 161	RMDA NOT RESET BY RIP COMMAND
345		
003372	071566	EMT161
003374	073676	EHT1
003376	074022	EDT1
003400	074112	EFT1
346		
347	;ERROR 162	RMDC NOT RESET BY RIP COMMAND
348		
003402	071600	EMT162
003404	073676	EHT1
003406	074022	EDT1
003410	074112	EFT1
349		
350	;ERROR 163	DATA WAS ECC CORRECTED BUT DOES NOT COMPARE WITH
351	;	WRITE BUFFER
352		
003412	073472	EMT336
003414	073760	EHT336
003416	074056	EDT336
003420	074146	EFT336
353		
354	;ERROR 164	OPI SHOULD NOT BE SET
355		
003422	071622	EMT164
003424	073676	EHT1
003426	074022	EDT1
003430	074112	EFT1
356		
357	;ERROR 165	IVC SHOULD NOT BE SET
358		
003432	071630	EMT165
003434	073676	EHT1
003436	074022	EDT1
003440	074112	EFT1
359		
360	;ERROR 166	IAE SHOULD NOT BE SET
361		
003442	071636	EMT166
003444	073676	EHT1
003446	074022	EDT1
003450	074112	EFT1
362		
363	;ERROR 167	NEM SHOULD NOT BE SET
364		

	003452 071644	EMT167	
	003454 073676	EHT1	
	003456 074022	EDT1	
	003460 074112	EFT1	
365			
366			
367			
	003462 000000	0	
	003464 000000	0	
	003466 000000	0	
	003470 000000	0	
368			
369			
370			
	003472 071662	EMT171	
	003474 073676	EHT1	
	003476 074022	EDT1	
	003500 074112	EFT1	
371			
372			
373			
	003502 071672	EMT172	
	003504 073676	EHT1	
	003506 074022	EDT1	
	003510 074112	EFT1	
374			
375			
376			
	003512 071702	EMT173	
	003514 073676	EHT1	
	003516 074022	EDT1	
	003520 074112	EFT1	
377			
378			
379			
	003522 071712	EMT174	
	003524 073676	EHT1	
	003526 074022	EDT1	
	003530 074112	EFT1	
380			
381			
382			
	003532 071724	EMT175	
	003534 073676	EHT1	
	003536 074022	EDT1	
	003540 074112	EFT1	
383			
384			
385			
	003542 071732	EMT176	

;ERROR 170 UNUSED

;ERROR 171 "ATA" NOT SET DURING RETURN TO CENTERLINE

;ERROR 172 "ATA" NOT SET BY OFFSET COMMAND

;ERROR 173 RMER2 NOT INITIALIZED BY UNIBUS INIT

;ERROR 174 RMER2 NOT INITIALIZED BY CONTROLLER CLEAR

;ERROR 175 SELECTED DEVICE IS IN WRITE PROTECT

;ERROR 176 CANNOT SET DIAGNOSTIC MODE

	003544	073676		EHT1	
	003546	074022		EDT1	
	003550	074112		EFT1	
386					
387			;ERROR	177	INCORRECT 'MOL' STATUS DURING DIAGNOSTIC MODE
388					
	003552	071740		EMT177	
	003554	073676		EHT1	
	003556	074022		EDT1	
	003560	074112		EFT1	
389					
390			;ERROR	200	INCORRECT 'PIP' STATUS DURING DIAGNOSTIC MODE
391					
	003562	071752		EMT200	
	003564	073676		EHT1	
	003566	074022		EDT1	
	003570	074112		EFT1	
392					
393			;ERROR	201	INCORRECT 'WRL' STATUS DURING DIAGNOSTIC MODE
394					
	003572	071764		EMT201	
	003574	073676		EHT1	
	003576	074022		EDT1	
	003600	074112		EFT1	
395					
396			;ERROR	202	INCORRECT 'SKI' STATUS DURING DIAGNOSTIC MODE
397					
	003602	071776		EMT202	
	003604	073676		EHT1	
	003606	074022		EDT1	
	003610	074112		EFT1	
398					
399			;ERROR	203	INCORRECT 'DVC' STATUS DURING DIAGNOSTIC MODE
400					
	003612	072010		EMT203	
	003614	073676		EHT1	
	003616	074022		EDT1	
	003620	074112		EFT1	
401					
402			;ERROR	204	'VV' WAS NOT RESET BY MAINTENANCE UNIT READY
403					
	003622	072022		EMT204	
	003624	073676		EHT1	
	003626	074022		EDT1	
	003630	074112		EFT1	
404					
405			;ERROR	205	SELECTED DEVICE HAS A PERSISTENT 'SKI' ERROR
406					
	003632	072040		EMT205	
	003634	073676		EHT1	

	003636 074022	EDT1	
	003640 074112	EFT1	
407			
408			
409			:ERROR 206 'LBC' DID NOT SET DURING DIAGNOSTIC MODE
	003642 072050	EMT206	
	003644 073676	EHT1	
	003646 074022	EDT1	
	003650 074112	EFT1	
410			
411			
412			:ERROR 207 UNEXPECTED LOSS OF 'MOL' - MEDIUM IS OFF LINE
	003652 072060	EMT207	
	003654 073676	EHT1	
	003656 074022	EDT1	
	003660 074112	EFT1	
413			
414			
415			:ERROR 210 UNEXPECTED LOSS OF VOLUME VALID - 'VV' = 0
	003662 072072	EMT210	
	003664 073676	EHT1	
	003666 074022	EDT1	
	003670 074112	EFT1	
416			
417			
418			:ERROR 211 UNEXPECTED MECHANICAL MOTION - 'PIP' = 1
	003672 072100	EMT211	
	003674 073676	EHT1	
	003676 074022	EDT1	
	003700 074112	EFT1	
419			
420			
421			:ERROR 212 UNEXPECTED DEVICE FAULT - 'DVC' = 1
	003702 072114	EMT212	
	003704 073676	EHT1	
	003706 074022	EDT1	
	003710 074112	EFT1	
422			
423			
424			:ERROR 213 UNEXPECTED SEEK INCOMPLETE ERROR - 'SKI' = 1
	003712 072130	EMT213	
	003714 073676	EHT1	
	003716 074022	EDT1	
	003720 074112	EFT1	
425			
426			
427			:ERROR 214 DRIVE EXECUTED A RECALIBRATE WITH ERROR SET
	003722 072140	EMT214	
	003724 073710	EHT2	
	003726 074032	EDT2	

003730	074122	EFT2	
428			
429		:ERROR 215	DRIVE DID NOT DETECT "IVC" ERROR DURING RECALIBRATE
430			
003732	072160	EMT215	
003734	073710	EHT2	
003736	074032	EDT2	
003740	074122	EFT2	
431			
432		:ERROR 216	INCORRECT "IVC" STATUS
433			
003742	072172	EMT216	
003744	073676	EHT1	
003746	074022	EDT1	
003750	074112	EFT1	
434			
435		:ERROR 217	INCORRECT "IAE" STATUS
436			
003752	072202	EMT217	
003754	073676	EHT1	
003756	074022	EDT1	
003760	074112	EFT1	
437			
438		:ERROR 220	INCORRECT "WLE" STATUS
439			
003762	072212	EMT220	
003764	073676	EHT1	
003766	074022	EDT1	
003770	074112	EFT1	
440			
441		:ERROR 221	INCORRECT "OPI" STATUS
442			
003772	072222	EMT221	
003774	073676	EHT1	
003776	074022	EDT1	
004000	074112	EFT1	
443			
444		:ERROR 222	RM DID NOT DETECT RMR ERROR
445			
004002	072232	EMT222	
004004	073676	EHT1	
004006	074022	EDT1	
004010	074112	EFT1	
446			
447		:ERROR 223	RM DID NOT DETECT PARITY ERROR ON MASSBUS CONTROL BUS
448			
004012	072242	EMT223	
004014	073734	EHT223	
004016	074046	EDT223	
004020	074136	EFT223	

449				
450			;ERROR	224 UNUSED
451				
	004022	000000		0
	004024	000000		0
	004026	000000		0
	004030	000000		0
452				
453			;ERROR	225 UNUSED
454				
	004032	000000		0
	004034	000000		0
	004036	000000		0
	004040	000000		0
455				
456			;ERROR	226 UNUSED
457				
	004042	000000		0
	004044	000000		0
	004046	000000		0
	004050	000000		0
458				
459			;ERROR	227 UNUSED
460				
	004052	000000		0
	004054	000000		0
	004056	000000		0
	004060	000000		0
461				
462			;ERROR	230 UNUSED
463				
	004062	000000		0
	004064	000000		0
	004066	000000		0
	004070	000000		0
464				
465			;ERROR	231 UNUSED
466				
	004072	000000		0
	004074	000000		0
	004076	000000		0
	004100	000000		0
467				
468			;ERROR	232 UNUSED
469				
	004102	000000		0
	004104	000000		0
	004106	000000		0
	004110	000000		0

470				
471			;ERROR	233 UNUSED
472				
	004112	000000		0
	004114	000000		0
	004116	000000		0
	004120	000000		0
473				
474			;ERROR	234 UNUSED
475				
	004122	000000		0
	004124	000000		0
	004126	000000		0
	004130	000000		0
476				
477			;ERROR	235 UNUSED
478				
	004132	000000		0
	004134	000000		0
	004136	000000		0
	004140	000000		0
479				
480			;ERROR	236 UNUSED
481				
	004142	000000		0
	004144	000000		0
	004146	000000		0
	004150	000000		0
482				
483			;ERROR	237 UNUSED
484				
	004152	000000		0
	004154	000000		0
	004156	000000		0
	004160	000000		0
485				
486			;ERROR	240 UNUSED
487				
	004162	000000		0
	004164	000000		0
	004166	000000		0
	004170	000000		0
488				
489			;ERROR	241 UNUSED
490				
	004172	000000		0
	004174	000000		0
	004176	000000		0
	004200	000000		0

491

492				
493			;ERROR 242	UNUSED
	004202	000000	0	
	004204	000000	0	
	004206	000000	0	
	004210	000000	0	
494				
495			;ERROR 243	UNUSED
496				
	004212	000000	0	
	004214	000000	0	
	004216	000000	0	
	004220	000000	0	
497				
498			;ERROR 244	UNUSED
499				
	004222	000000	0	
	004224	000000	0	
	004226	000000	0	
	004230	000000	0	
500				
501			;ERROR 245	UNUSED
502				
	004232	000000	0	
	004234	000000	0	
	004236	000000	0	
	004240	000000	0	
503				
504			;ERROR 246	"ATA" NOT RESET BY GO WHEN "ERR" = 0
505				
	004242	072316	EMT246	
	004244	073676	EHT1	
	004246	074022	EDT1	
	004250	074112	EFT1	
506				
507			;ERROR 247	"ATA" NOT RESET BY WRITING RMAS
508				
	004252	072326	EMT247	
	004254	073676	EHT1	
	004256	074022	EDT1	
	004260	074112	EFT1	
509				
510			;ERROR 250	"ATA" WAS RESET BY GO WHEN "ERR" = 1
511				
	004262	072340	EMT250	
	004264	073676	EHT1	
	004266	074022	EDT1	
	004270	074112	EFT1	
512				
513			;ERROR 251	PROGRAM INTERRUPT WAS NOT GENERATED

514	004272 072354	EMT251	
	004274 073710	EHT2	
	004276 074032	EDT2	
	004300 074122	EFT2	
515			
516		;ERROR 252	PROGRAM INTERRUPT SHOULD NOT HAVE BEEN GENERATED
517			
	004302 072362	EMT252	
	004304 073710	EHT2	
	004306 074032	EDT2	
	004310 074122	EFT2	
518			
519		;ERROR 253	OFFSET MODE WAS NOT RESET BY WRITING RMDC
520			
	004312 072370	EMT253	
	004314 073676	EHT1	
	004316 074022	EDT1	
	004320 074112	EFT1	
521			
522		;ERROR 254	INCORRECT "ILF" STATUS
523			
	004322 072406	EMT254	
	004324 073676	EHT1	
	004326 074022	EDT1	
	004330 074112	EFT1	
524			
525		;ERROR 255	INCORRECT "ATA" STATUS
526			
	004332 072416	EMT255	
	004334 073676	EHT1	
	004336 074022	EDT1	
	004340 074112	EFT1	
527			
528		;ERROR 256	INCORRECT "ILR" STATUS
529			
	004342 072426	EMT256	
	004344 073746	EHT256	
	004346 074046	EDT223	
	004350 074136	EFT223	
530			
531		;ERROR 257	INVALID IAE STATUS DURING SEARCH COMMAND
532			
	004352 072436	EMT257	
	004354 073676	EHT1	
	004356 074022	EDT1	
	004360 074112	EFT1	
533			
534		;ERROR 260	"IVC" WAS NOT DETECTED DURING SEARCH COMMAND
535			

	004362	072450		EMT260
	004364	073676		EHT1
	004366	074022		EDT1
	004370	074112		EFT1
536				
537			;ERROR	261 DRIVE EXECUTED SEARCH WITH ERROR SET
538				
	004372	072462		EMT261
	004374	073676		EHT1
	004376	074022		EDT1
	004400	074112		EFT1
539				
540			;ERROR	262 "LBC" ERROR NOT SET DURING DIAGNOSTIC MODE
541				
	004402	072502		EMT262
	004404	073676		EHT1
	004406	074022		EDT1
	004410	074112		EFT1
542				
543			;ERROR	263 "SKI" ERROR DURING SEARCH COMMAND
544				
	004412	072512		EMT263
	004414	073676		EHT1
	004416	074022		EDT1
	004420	074112		EFT1
545				
546			;ERROR	264 "IVC" ERROR DURING SEARCH - LOST VOLUME VALID
547				
	004422	072522		EMT264
	004424	073676		EHT1
	004426	074022		EDT1
	004430	074112		EFT1
548				
549			;ERROR	265 ERRONEOUS IVC ERROR DURING SEARCH - VOLUME IS VALID
550				
	004432	072542		EMT265
	004434	073676		EHT1
	004436	074022		EDT1
	004440	074112		EFT1
551				
552			;ERROR	266 DEVICE FAULT (DVC) DURING SEARCH
553				
	004442	072562		EMT266
	004444	073676		EHT1
	004446	074022		EDT1
	004450	074112		EFT1
554				
555			;ERROR	267 SKI SHOULD HAVE BEEN SET BECAUSE CYLINDER
556			;	ADDRESS IS TOO LARGE
557				

004452	072574	EMT267
004454	073676	EHT1
004456	074022	EDT1
004460	074112	EFT1
558		
559	:ERROR 270	OPI ERROR DURING SEARCH BECAUSE MOL = 0
560		
004462	072612	EMT270
004464	073676	EHT1
004466	074022	EDT1
004470	074112	EFT1
561		
562	:ERROR 271	OPI ERROR DURING SEARCH BECAUSE ON CYLINDER
563	:	DIDN'T DROP
564		
004472	072626	EMT271
004474	073676	EHT1
004476	074022	EDT1
004500	074112	EFT1
565		
566	:ERROR 272	LOST MOL DURING SEARCH, OPI IS NOT SET
567		
004502	072644	EMT272
004504	073676	EHT1
004506	074022	EDT1
004510	074112	EFT1
568		
569	:ERROR 273	PIP STIL SET AFTER SEARCH
570		
004512	072662	EMT273
004514	073676	EHT1
004516	074022	EDT1
004520	074112	EFT1
571		
572	:ERROR 274	PARITY ERROR OCCURRED WHILE WRITING REMOTE
573	:	REGISTERS BUT MXF DID NOT SET
574		
004522	072700	EMT274
004524	073676	EHT1
004526	074022	EDT1
004530	074112	EFT1
575		
576	:ERROR 275	MXF ERROR - COMPOSITE ERROR OCCURRED BEFORE DATA
577	:	COMMAND STARTED
578		
004532	072716	EMT275
004534	073676	EHT1
004536	074022	EDT1
004540	074112	EFT1

579

580			:ERROR 276	"OPI" ERROR DURING DATA TRANSFER BECAUSE "MOL" WAS
581			:	ZERO
582	004542	072730		EMT276
	004544	073676		EHT1
	004546	074022		EDT1
	004550	074112		EFT1
583				
584			:ERROR 277	"OPI" ERROR DURING DATA TRANSFER BECAUSE 1) ON
585			:	CYLINDER DIDN'T DROP OR 2) SEARCH TIMED OUT OR
586			:	3) RUN TIMED OUT
587	004552	072744		EMT277
	004554	073676		EHT1
	004556	074022		EDT1
	004560	074112		EFT1
588				
589			:ERROR 300	"IVC" ERROR DURING DATA TRANSFER BECAUSE VOLUME
590			:	WAS NOT VALID
591	004562	072762		EMT300
	004564	073676		EHT1
	004566	074022		EDT1
	004570	074112		EFT1
592				
593			:ERROR 301	ERRONEOUS "IVC" ERROR DURING DATA TRANSFER - VOLUME
594			:	IS VALID
595	004572	073002		EMT301
	004574	073676		EHT1
	004576	074022		EDT1
	004600	074112		EFT1
596				
597			:ERROR 302	"ILR" ERROR DURING DATA TRANSFER
598	004602	073024		EMT302
	004604	073676		EHT1
	004606	074022		EDT1
	004610	074112		EFT1
599				
600			:ERROR 303	"ILF" ERROR DURING DATA TRANSFER
601	004612	073036		EMT303
	004614	073676		EHT1
	004616	074022		EDT1
	004620	074112		EFT1
602				
603			:ERROR 304	"RMR" ERROR DURING DATA TRANSFER
604	004622	073050		EMT304
	004624	073676		EHT1

	004626	074022		EDT1
	004630	074112		EFT1
605				
606			:ERROR 305	INCORRECT "IAE" STATUS DURING DATA TRANSFER
607				
	004632	073062		EMT305
	004634	073676		EHT1
	004636	074022		EDT1
	004640	074112		EFT1
608				
609			:ERROR 306	"SKI" ERROR DURING DATA TRANSFER
610				
	004642	073074		EMT306
	004644	073676		EHT1
	004646	074022		EDT1
	004650	074112		EFT1
611				
612			:ERROR 307	DRIVE DID NOT DETECT SKI ERROR DUE TO CYLINDER
613				
	004652	073104		EMT307
	004654	073676		EHT1
	004656	074022		EDT1
	004660	074112		EFT1
614				
615			:ERROR 310	DEVICE FAULT DURING DATA TRANSFER
616				
	004662	073124		EMT310
	004664	073676		EHT1
	004666	074022		EDT1
	004670	074112		EFT1
617				
618			:ERROR 311	LOSS OF BIT CLOCK DURING DATA TRANSFER
619				
	004672	073136		EMT311
	004674	073676		EHT1
	004676	074022		EDT1
	004700	074112		EFT1
620				
621			:ERROR 312	LOSS OF SYSTEM CLOCK DURING DATA TRANSFER
622				
	004702	073150		EMT312
	004704	073676		EHT1
	004706	074022		EDT1
	004710	074112		EFT1
623				
624			:ERROR 313	UNSAFE ERROR DURING DATA TRANSFER (DVC = 0)
625				
	004712	073162		EMT313
	004714	073676		EHT1
	004716	074022		EDT1

Line	Code	Address	Module	Description
		004720 074112	EFT1	
626				
627	:ERROR	314		DRIVE TIMING ERROR DURING DATA TRANSFER
628				
		004722 073202	EMT314	
		004724 073676	EHT1	
		004726 074022	EDT1	
		004730 074112	EFT1	
629				
630	:ERROR	315		WRITE LOCK ERROR
631				
		004732 073214	EMT315	
		004734 073676	EHT1	
		004736 074022	EDT1	
		004740 074112	EFT1	
632				
633	:ERROR	316		ERRONEOUS WRITE LOCK ERROR
634				
		004742 073226	EMT316	
		004744 073676	EHT1	
		004746 074022	EDT1	
		004750 074112	EFT1	
635				
636	:ERROR	317		HEADER CRC ERROR DURING DATA TRANSFER
637				
		004752 073240	EMT317	
		004754 073676	EHT1	
		004756 074022	EDT1	
		004760 074112	EFT1	
638				
639	:ERROR	320		FORMAT ERROR DURING DATA TRANSFER
640				
		004762 073250	EMT320	
		004764 073676	EHT1	
		004766 074022	EDT1	
		004770 074112	EFT1	
641				
642	:ERROR	321		HEADER COMPARE ERROR DURING DATA TRANSFER
643				
		004772 073260	EMT321	
		004774 073676	EHT1	
		004776 074022	EDT1	
		005000 074112	EFT1	
644				
645	:ERROR	322		HEADER ERRORS SHOULD NOT BE SET
646				
		005002 073270	EMT322	
		005004 073676	EHT1	
		005006 074022	EDT1	
		005010 074112	EFT1	

647				
648				
649				
	005012	073276		
	005014	073676		
	005016	074022		
	005020	074112		
650				
651				
652				
	005022	073306		
	005024	073676		
	005026	074022		
	005030	074112		
653				
654				
655				
	005032	073320		
	005034	073676		
	005036	074022		
	005040	074112		
656				
657				
658				
	005042	073332		
	005044	073676		
	005046	074022		
	005050	074112		
659				
660				
661				
	005052	073350		
	005054	073676		
	005056	074022		
	005060	074112		
662				
663				
664				
	005062	073362		
	005064	073676		
	005066	074022		
	005070	074112		
665				
666				
667				
	005072	073372		
	005074	073676		
	005076	074022		
	005100	074112		

;ERROR 323 DATA CHECK ERROR DURING DATA TRANSFER

EMT323
EHT1
EDT1
EFT1

;ERROR 324 CORRECTABLE DATA CHECK ERROR DURING DATA TRANSFER

EMT324
EHT1
EDT1
EFT1

;ERROR 325 UNCORRECTABLE DATA CHECK ERROR DURING DATA TRANSFER

EMT325
EHT1
EDT1
EFT1

;ERROR 326 DATA PARITY ERROR DURING READ COMMAND

EMT326
EHT1
EDT1
EFT1

;ERROR 327 OFFSET MODE NOT RESET BY WRITE COMMAND

EMT327
EHT1
EDT1
EFT1

;ERROR 330 DATA PARITY ERROR DURING WRITE COMMAND

EMT330
EHT1
EDT1
EFT1

;ERROR 331 WRITE CLOCK FAILURE DURING WRITE COMMAND

EMT331
EHT1
EDT1
EFT1

668				
669				
670				
	005102	073404		
	005104	073676		
	005106	074022		
	005110	074112		
			EMT332	
			EHT1	
			EDT1	
			EFT1	
671				
672				
673				
	005112	073416		
	005114	073676		
	005116	074022		
	005120	074112		
			EMT333	
			EHT1	
			EDT1	
			EFT1	
674				
675				
676				
	005122	073434		
	005124	073676		
	005126	074022		
	005130	074112		
			EMT334	
			EHT1	
			EDT1	
			EFT1	
677				
678				
679				
	005132	073452		
	005134	073676		
	005136	074022		
	005140	074112		
			EMT335	
			EHT1	
			EDT1	
			EFT1	
680				
681				
682				
	005142	073472		
	005144	073760		
	005146	074056		
	005150	074146		
			EMT336	
			EHT336	
			EDT336	
			EFT336	
683				
684				
685				
	005152	073502		
	005154	073772		
	005156	074066		
	005160	074156		
			EMT337	
			EHT337	
			EDT337	
			EFT337	
686				
687				
688				
	005162	073512		
	005164	073760		
	005166	074056		
	005170	074146		
			EMT340	
			EHT336	
			EDT336	
			EFT336	

689

690			:ERROR 341	INCORRECT DATA DURING WRITE CHECK ERROR
691	005172	073524		EMT341
	005174	073760		EHT336
	005176	074056		EDT336
	005200	074146		EFT336
692				
693			:ERROR 342	"IVC" ERROR NOT DETECTED DURING DATA TRANSFER
694	005202	073532		EMT342
	005204	073676		EHT1
	005206	074022		EDT1
	005210	074112		EFT1
695				
696			:ERROR 343	"FER" NOT DETECTED DURING DATA TRANSFER
697	005212	073544		EMT343
	005214	073676		EHT1
	005216	074022		EDT1
	005220	074112		EFT1
698				
699			:ERROR 344	"HCE" NOT DETECTED DURING DATA TRANSFER
700	005222	073556		EMT344
	005224	074004		EHT344
	005226	074076		EDT344
	005230	074166		EFT344
701				
702			:ERROR 345	"BSE" NOT DETECTED DURING DATA TRANSFER
703	005232	073570		EMT345
	005234	073676		EHT1
	005236	074022		EDT1
	005240	074112		EFT1
704				
705			:ERROR 346	HEADER ERROR WAS DETECTED W/ HCI SET
706	005242	073600		EMT346
	005244	073676		EHT1
	005246	074022		EDT1
	005250	074112		EFT1
707				
708			:ERROR 347	DATA TRANSFER NOT ABORTED W/ COMP ERROR SET
709	005252	073614		EMT347
	005254	073676		EHT1
	005256	074022		EDT1
	005260	074112		EFT1
710				
711			:ERROR 350	LOST VOLUME VALID DURING SEARCH - "IVC" = 0

712 005262 073626 EMT350
 005264 073676 EHT1
 005266 074022 EDT1
 005270 074112 EFT1

713
714 ;ERROR 351 'ATA' DID NOT SET DURING SEARCH
715 005272 073644 EMT351
 005274 073676 EHT1
 005276 074022 EDT1
 005300 074112 EFT1

716
717 ;ERROR 352 PROGRAM TIMEOUT WHILE TESTING RMLA
718 005302 073654 EMT352
 005304 000000 0
 005306 000000 0
 005310 000000 0

719
720 ;ERROR 353 LOOK AHEAD TEST FAILS
721 005312 073660 EMT353
 005314 074016 EHT353
 005316 074110 EDT353
 005320 074176 EFT353

722
723 ;ERROR 354 BSE SHOULD NOT BE SET
724 005322 073670 EMT354
 005324 073676 EHT1
 005326 074022 EDT1
 005330 074112 EFT1

725
726 ;PUT ERROR TABLE HERE
727

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

```
.SBTTL ERROR TABLE USAGE
;THE ERROR TABLE ABOVE CONSISTS OF FOUR WORD ENTRIES FOR EACH ERROR
;NUMBER, I.E.,
:
:   EMT - ERROR MESSAGE TABLE ADDRESS
:   EHT - ERROR HEADER TABLE ADDRESS
:   EDT - ERROR DATA TABLE ADDRESS
:   EFT - ERROR FORMAT TABLE ADDRESS
:
;THE EMT ENTRY IS THE ADDRESS OF THE TABLE OF ERROR MESSAGE STRINGS
;FOR THE PARTICULAR ERROR. EACH ERROR MESSAGE TABLE LISTS THE ADDRESS
;OF ONE OR MORE ERROR MESSAGE STRINGS WHICH ARE TO BE FORMATTED AND
;TYPED BY THE ERROR TYPE SUBROUTINE. IF THE EMT ENTRY IS ZERO, THERE IS
;NO MESSAGE TO BE TYPED FOR THE ERROR.
:
;SIMILARLY, THE EHT, EDT, AND EFT ENTRIES ARE ADDRESSES OF TABLES
;OF HEADER, DATA AND FORMAT INFORMATION FOR A GIVEN ERROR. EACH ENTRY
;IN THE ERROR HEADER TABLE MAY OR MAY NOT HAVE AN ASSOCIATED LINE OF
;DATA, HOWEVER, EACH DATA LINE MUST HAVE AN ASSOCIATED FORMAT AND
;HEADER. THAT IS, A HEADER LINE MAY BE PRINTED WITHOUT ANY DATA,
;BUT A DATA LINE IS NOT PRINTED WITHOUT A HEADER, AND EACH DATA LINE
;MUST ALSO HAVE A FORMAT.
:
;IN SUMMARY,
:   EACH NONZERO ENTRY IS THE ADDRESS OF A TABLE,
:   EACH TABLE IS A LIST OF ADDRESSES WHICH DEFINES THE LOCATIONS
:   OF MESSAGE STRINGS, HEADERS, DATA OR FORMAT.
```

```

1      ;THIS ROUTINE HANDLES UNEXPECTED TIMEOUTS
2
3 005332 011600
4 005334 005740
5 005336 022626
6 005340 104401 005346
   005344 000417
       005404
7 005404 010046
8 005406 104402
9 005410 000240
10
11
12
13
14 005412 000240
15 005414 005227 000000
16 005420 001375
17 005422 000005
18
19
   005424 012706 001114
   005430 005026
   005432 022706 001154
   005436 001374
   005440 012706 001100
       005444 012737 063210 000020
       005452 012737 000340 000022
       005460 012737 063710 000030
       005466 012737 000340 000032
       005474 012737 065450 000034
       005502 012737 000340 000036
       005510 012737 065556 000024
       005516 012737 000340 000026
       005524 013737 042172 042164
       005532 005037 001206
       005536 005037 001210
       005542 112737 000001 001131
       005550 012737 005550 001122
       005556 012737 005556 001124
       005564 013746 000004
       005570 012737 005624 000004
       005576 012737 177570 001154
       005604 012737 177570 001156
       005612 022777 177777 173334
       005620 001012
       005622 000403
       005624 012716 005632
       005630 000002
       005632 012737 000176 001154
       005640 012737 000174 001156

BADTMO: MOV      (SP),R0      ;SAVE PC WHERE THE TIME OUT OCCURED
        TST      -(R0)       ;ADJUST PC -2
        CMP      (SP)+,(SP)+ ;RESTORE STACK POINTER
        TYPE     ,65$        ;:TYPE ASCIZ STRING
        BR       64$         ;:GET OVER THE ASCIZ
        ;:65$: .ASCIZ <CRLF>/UNEXPECTED BUS TIMEOUT, PC=/
        64$:
        MOV      RO,-(SP)    ;SETUP FOR TYPING OUT PC
        TYPOC
        NOP
        ;PUT 'HALT(0)' INSTRUCTION HERE IF YOU WISH
        ;TO STOP ON UNEXPECTED TIMEOUT.

.SBTTL  START OF PROGRAM

START:  NOP
        INC      #0          ;TTY LOOP, WAIT FOR INCREMENT
        BNE     .-4          ;OF WORD
        RESET
        ;RESET THE WORLD

.SBTTL  INITIALIZE THE COMMON TAGS
        ;:CLEAR THE COMMON TAGS (%CMTAG) AREA
        MOV     #CMTAG,R6    ;:FIRST LOCATION TO BE CLEARED
        CLR     (R6)+       ;:CLEAR MEMORY LOCATION
        CMP     #SWR,R6     ;:DONE?
        BNE     .-6         ;:LOOP BACK IF NO
        MOV     #STACK,SP   ;:SETUP THE STACK POINTER
        ;:INITIALIZE A FEW VECTORS
        MOV     #SCOPE,@IOTVEC ;:IOT VECTOR FOR SCOPE ROUTINE
        MOV     #340,@IOTVEC+2 ;:LEVEL 7
        MOV     #ERROR,@EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE
        MOV     #340,@EMTVEC+2 ;:LEVEL 7
        MOV     #TRAP,@TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS
        MOV     #340,@TRAPVEC+2 ;:LEVEL 7
        MOV     #PWRDN,@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     $ESCAPE     ;:CLEAR THE ESCAPE ON ERROR ADDRESS
        MOVB   #1,$ERMAX    ;:ALLOW ONE ERROR PER TEST
        MOV     #,$LPADR    ;:INITIALIZE THE LOOP ADDRESS FOR SCOPE
        MOV     #,$LPERR    ;:SETUP THE ERROR LOOP ADDRESS
        ;:SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
        ;:EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
        MOV     @ERRVEC,-(SP) ;:SAVE ERROR VECTOR
        MOV     #64$,@ERRVEC ;:SET UP ERROR VECTOR
        MOV     #DSWR,SWR   ;:SETUP FOR A HARDWARE SWICH REGISTER
        MOV     #DDISP,DISPLAY ;:AND A HARDWARE DISPLAY REGISTER
        CMP     #-1,@SWR    ;:TRY TO REFERENCE HARDWARE SWR
        BNE     66$        ;:BRANCH IF NO TIMEOUT TRAP OCCURRED
        ;:AND THE HARDWARE SWR IS NOT = -1
        BR     65$         ;:BRANCH IF NO TIMEOUT
        64$: MOV     #65$, (SP) ;:SET UP FOR TRAP RETURN
        65$: MOV     #SWREG,SWR ;:POINT TO SOFTWARE SWR
        MOV     #DISPREG,DISPLAY
  
```

```

005646 012637 000004      66$:  MOV      (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
005652 005037 001230      CLR      $PASS              ;;CLEAR PASS COUNT
005656 132737 000200 001243  BITB     #APTSIZE,$ENVM    ;;TEST USER SIZE UNDER APT
005664 001403      BEQ      67$              ;;YES,USE NON-APT SWITCH
005666 012737 001244 001154  MOV      #$$SWREG,$SWR    ;;NO,USE APT SWITCH REGISTER
005674
20      ;SETUP "TIMEOUT" TRAP VECTOR FOR UNEXPECTED BUS TIMEOUTS
21 005674 012737 005332 000004  MOV      #BADTMO,ERRVEC  ;;SETUP FOR UNEXPECTED TIMEOUT
22 005702 012737 000300 000006  MOV      #PR6,ERRVEC+2  ;;LEVEL 6
23 005710 012746 000300      MOV      #PR6,-(SP)      ;;PUT NEW PS ON STACK
005714 012746 005722      MOV      #68$,-(SP)     ;;PUT NEW PC ON STACK
005720 000002      RTI                    ;;POP NEW PC AND PS
005722
24      68$:
25      .SBTTL  TYPE PROGRAM NAME
      ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
005722 005227 177777      INC      #-1              ;;FIRST TIME?
005726 001056      BNE      69$              ;;BRANCH IF NO
005730 022737 042326 000042  CMP      #SENDAD,@#42    ;;ACT-11?
005736 001452      BEQ      69$              ;;BRANCH IF YES
005740 104401 006006      TYPE     ,70$            ;;TYPE ASCIZ STRING
      .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
005744 005737 000042      TST     @#42              ;;ARE WE RUNNING UNDER XXDP/ACT?
005750 001012      BNE      71$              ;;BRANCH IF YES
005752 123727 001242 000001  CMPB    $ENV,#1          ;;ARE WE RUNNING UNDER APT?
005760 001406      BEQ      71$              ;;BRANCH IF YES
005762 023727 001154 000176  CMP      $SWR,#SWREG     ;;SOFTWARE SWITCH REG SELECTED?
005770 001005      BNE      72$              ;;BRANCH IF NO
005772 104407      GTSWR                    ;;GET SOFT-SWR SETTINGS
005774 000403      BR      72$
005776 112737 000001 001150 71$:  MOVB    #1,$AUTOB        ;;SET AUTO-MODE INDICATOR
006004      72$:
006004 000427      BR      69$              ;;GET OVER THE ASCIZ
006064      ;;70$: .ASCIZ <CRLF>@CZRMMAO - RM05/3/2 FUNCTIONAL TEST, PART 1@<CRLF>
      69$:
26      ;THE FOLLOWING FINDS OUT THE PROGRAM CONTROL MODE:
27      ;PAPER TAPE (MANUAL), ACT11, XXDP CHAIN OR DUMP
28
29
30 006064 005037 001330      CLR      XXDP              ;;CLEAR 'XXDP' LOAD DEVICE STORAGE
31 006070 122737 000016 000041  CMPB    #16,@#41        ;;LOADED FROM AN RM05/3/2 ?
32 006076 001160      BNE      3$              ;;BRANCH IF NOT
33 006100 013737 000040 001330  MOV      @#40,XXDP        ;;GET DEVICE INDICATOR AND NUMBER
34 006106 122737 000007 001330  CMPB    #7,XXDP         ;;IS IT A VALID NUMBER ?
35 006114 103002      BHIS    1$              ;;YES
36 006116 105037 001330      CLRB    XXDP            ;;NO, DEFAULT TO DRIVE 0
37 006122 005737 000042      1$:  TST     @#42              ;;CHAIN MODE OR ACT11 AUTO ACCEPT ?
38 006126 001425      BEQ      2$              ;;BR IF NEITHER
39 006130 104401 006136      TYPE     ,74$            ;;TYPE ASCIZ STRING
006134 000412      BR      73$              ;;GET OVER THE ASCIZ
      ;;74$: .ASCIZ <CRLF>/NOT TESTING DRIVE /
      73$:
40 006162 005046      CLR      -(SP)           ;;CLEAR WORD ON STACK
41 006164 113716 001330      MOVB    XXDP,(SP)       ;;GET DRIVE ADDRESS
42 006170 104403      TYPOS                    ;;TYPE THE ADDRESS
43 006172 001      .BYTE   1              ;;ONLY 1 CHARACTER

```

```

44 006173 000
45 006174 104401 001217 .BYTE 0 ;SUPRESS LEADING ZEROS
46 006200 000517 TYPE %$CRLF ;CR-LF
47 BR 3$ ;GET NUMBER OF DRIVES
48 006202 005227 177777 2$: INC #-1 ;FIRST TIME THRU HERE ?
49 006206 001114 BNE 3$ ;NO
50 006210 104401 006216 TYPE %$76$ ;TYPE ASCIZ STRING
006214 000410 BR %$75$ ;GET OVER THE ASCIZ
;:76$: .ASCIZ <CRLF>/TO TEST DRIVE /
;:75$:
51 006236 005046 CLR -(SP) ;CLEAR WORD ON STACK
52 006240 113716 001330 MOVB XXDP,(SP) ;GET DRIVE ADDRESS
53 006244 104403 TYPOS ;TYPE DRIVE ADDRESS
54 006246 001 .BYTE 1 ;ONLY 1 CHARACTER
55 006247 000 .BYTE 0 ;SUPRESS LEADING ZEROS
56 006250 104401 006256 TYPE %$78$ ;TYPE ASCIZ STRING
006254 000431 BR %$77$ ;GET OVER THE ASCIZ
;:78$: .ASCIZ /, HALT PROGRAM, REMOVE RRDPAK AND REPLACE IT/<CRLF>
;:77$:
57 006340 104401 006346 TYPE %$79$ ;TYPE ASCIZ STRING
006344 000435 BR 3$ ;GET OVER THE ASCIZ
;:79$: .ASCIZ /WITH A WORK PAK, CLEAR LOCATION 40 AND RESTART PROGRAM./<CRLF>
;:3$:
61 ;CHECK FOR AUTO MODE OR STANDLONE MODE
62 006440 105737 001150 TSTB $AUTOB ;RUNNING IN AUTO MODE ?
63 006444 001515 BEQ STANDALONE ;BR IF NO
64 006446 012737 000377 001300 MOV #377,$DEV ;SET DEVICE MAP FOR ALL DRIVES
65
66 ;PROGRAM IS RUNNING IN AUTO MODE - SEE IF SIZING IS ALLOWED
67 006454 XSIZ:
68 006454 132737 000200 001243 BITB #BIT7,$ENVM ;SIZING ALLOWED ?
69 006462 001075 BNE 7$ ;NO
70
71 006464 005001 CLR R1 ;START FROM DRIVE 0
72 006466 013700 001276 MOV $BASE,R0 ;LOAD THE BASE ADDRESS
73
74 006472 136137 067146 001300 1$: BITB ATNTBL(R1),$DEV ;IS DEVICE PRESENT IN MAP ?
75 006500 001462 BEQ 6$ ;BR IF NO
76 006502 012737 066745 006644 MOV #LODEV,5$ ;GET ADDRESS OF LOAD DEVICE MESSAGE
77 006510 005737 001330 TST XXDP ;LOADED FROM RM05/3/2 ?
78 006514 001403 BEQ 2$ ;NO
79 006516 123701 001330 CMPB XXDP,R1 ;IS THIS THE DRIVE ?
80 006522 001435 BEQ 3$ ;YES, TRY NEXT DRIVE
81
82 006524 012760 000040 000010 2$: MOV #CLR, RMCS2(R0) ;CLEAR MASS BUS
83 006532 010160 000010 MOV R1, RMCS2(R0) ;LOAD THE DRIVE ADDRESS
84 006536 005760 000012 TST RMD5(R0) ;TRY TO ACCESS AN RM DRIVE REGISTER
85 006542 012737 066765 006644 MOV #NOTPRS, 5$ ;GET ADDRESS OF NOT PRESENT MESSAGE
86 006550 032760 010000 000010 BIT #NED, RMCS2(R0) ;IS DRIVE PRESENT ?
87 006556 001017 BNE 3$ ;BR IF NO
88 006560 012737 067002 006644 MOV #NOTAVL, 5$ ;GET ADDRESS OF AVAILABLE MESSAGE
89 006566 032760 004000 000000 BIT #DVA, RMCS1(R0) ;IS DRIVE AVAILABLE ?
90 006574 001410 BEQ 3$ ;BR IF NO
91 006576 012737 067021 006644 MOV #OFFLIN, 5$ ;GET ADDRESS OF OFF LINE MESSAGE
92 006604 032760 010000 000012 BIT #MOL, RMD5(R0) ;IS MEDIUM ON LINE ?
93 006612 001401 BEQ 3$ ;BR IF NO
94 006614 000414 BR 6$

```

```

95
96 006616 146137 067146 001300 3$: BICB ATNTBL(R1), $DEVM ;CLEAR DEVICE FROM BIT MAP
97 006624 104401 001217 4$: TYPE , $CRLF ;CR-LF
98 006630 104401 066737 TYPE ,MSGDRV ;TYPE 'DRIVE'
99 006634 010146 MOV R1, -(SP) ;;SAVE R1 FOR TYPEOUT
    006636 104403 TYPOS ;;GO TYPE--OCTAL ASCII
    006640 002 .BYTE 2 ;;TYPE 2 DIGIT(S)
    006641 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
100 006642 104401 TYPE ;TYPE ERROR MESSAGE
101 006644 000000 5$: .WORD 0 ;ADDRESS OF MESSAGE GOES HERE
102
103 006646 005201 6$: INC R1 ;INCREMENT THE DRIVE ADDRESS
104 006650 020127 000007 CMP R1, #7 ;ALL DRIVES ARE CHECKED ?
105 006654 003706 BLE 1$ ;BRANCH IF NOT
106
107 006656 104401 001217 7$: TYPE , $CRLF ;CR-LF
108 006662 005004 CLR R4 ;THESE TWO LOOPS ARE ADDED TO
109 006664 005304 DEC R4 ;WAIT FOR TTY TO FINISH TYPING.
110 006666 001376 BNE .-2
111 006670 005304 DEC R4
112 006672 001376 BNE .-2
113
114 006674 000137 007604 JMP CMNSTART ;JUMP TO COMMON START
115
  
```



```

1          .SBTTL  STANDALONE INPUT ROUTINES
2
3          STANDALONE:
4 006700   JSR      PC,$TKINT      ;INITIALIZE CONSOLE
5
6 006704   004737  064120
7 006710   005227  177777
8
9          INC      #-1           ;FIRST TIME THRU HERE ?
          BEQ      3$           ;YES !!
          ;SEE IF THE USER WANTS TO KEEP SAME DEVICES FOR TESTING
10         1$:
11 006712   104401  066304        TYPE      ,CNSL00      ;MAINTAIN PREVIOUS PARAMETERS ?
12 006716   104411                RDCHR                ;GET RESPONSE
13 006720   012637  001176        MOV      (SP)+,$TMP1    ;ECHO RESPONSE
14 006724   104401  001176        TYPE      , $TMP1
15 006730   123727  001176  000131 CMPB     $TMP1,#'Y      ;YES RESPONSE ?
16 006736   001004                BNE      2$           ;NO!!
17 006740   104401  001217        TYPE      , $CR LF    ;CR-LF
18 006744   000137  007604        JMP      CMNSTART     ;KEEP PREVIOUS PARAMETERS
19
20 006750   123727  001176  000116 2$:  CMPB     $TMP1,#'N      ;NO RESPONSE ?
21 006756   001427                BEQ      5$           ;YES, GET NEW PARAMETERS
22 006760   104401  066701        TYPE      ,CNSL08    ;NO, TYPE ' ILLEGAL INPUT '
23 006764   000752                BR       1$           ;TRY AGAIN
24
25         ;SEE IF OPERATOR WANTS HELP TEXT
26 006766   104401  066220        3$:  TYPE      ,MSHELP     ;WANT HELP ?
27 006772   104411                RDCHR                ;GET RESPONSE
28 006774   012637  001176        MOV      (SP)+,$TMP1    ;SAVE AND ECHO RESPONSE
29 007000   104401  001176        TYPE      , $TMP1
30 007004   123727  001176  000131 CMPB     $TMP1,#'Y      ;WAS IT A YES RESPONSE ?
31 007012   001407                BEQ      4$           ;YES
32 007014   123727  001176  000116 CMPB     $TMP1,#'N      ;WAS IT A NO RESPONSE ?
33 007022   001405                BEQ      5$           ;YES
34 007024   104401  066701        TYPE      ,CNSL08    ;NO, TYPE ' ILLEGAL INPUT '
35 007030   000756                BR       3$           ;TRY AGAIN
36 007032   104401  103640        4$:  TYPE      ,HELP     ;YES - TYPE HELP TEXT
37
38         ;SEE IF USER WANTS TO CHANGE ADDRESSES
39 007036   104401  001217        5$:  TYPE      , $CR LF    ;CR-LF
40 007042   104401  066251        TYPE      ,UBUSQST    ;WANT TO CHANGE ADDRESS ?
41 007046   104411                RDCHR                ;GET RESPONSE
42 007050   012637  001176        MOV      (SP)+,$TMP1    ;SAVE AND ECHO RESPONSE
43 007054   104401  001176        TYPE      , $TMP1
44 007060   123727  001176  000131 CMPB     $TMP1,#'Y      ;WAS IT A YES RESPONSE ?
45 007066   001407                BEQ      6$           ;YES
46 007070   123727  001176  000116 CMPB     $TMP1,#'N      ;WAS IT A NO RESPONSE ?
47 007076   001525                BEQ      12$          ;YES
48 007100   104401  066701        TYPE      ,CNSL08    ;NO, TYPE ' ILLEGAL INPUT '
49 007104   000756                BR       5$+4        ;TRY AGAIN
50
51         ;DIALOGUE TO CHANGE THE UNIBUS ADDRESS, VECTOR ADDRESS AND INTERRUPT PRIORITY
52 007106   104401  066337        6$:  TYPE      ,CNSL01    ;TYPE CURRENT BUS ADDRESS
53 007112   013746  001276        MOV      $BASE,-(SP)   ;;SAVE $BASE FOR TYPEOUT
54 007116   104402                TYPOC                ;;GO TYPE--OCTAL ASCII(ALL DIGITS)

```

```

57 007120 104401 066211          TYPE      ,QUES      ;TYPE " ? "
58 007124 104413          RDOCT      ;GET NEW BUS ADDRESS
59 007126 012637 001176        MOV      (SP)+,$TMP1 ;CARRIAGE RETURN ?
60 007132 001412          BEQ      8$         ;YES-SKIP TO NEXT ENTRY
61 007134 022737 160000 001176  CMP      #160000,$TMP1 ;BASE ADDRESS IN I/O PAGE ?
62 007142 101403          BLOS     7$         ;YES
63 007144 104401 066355          TYPE      ,CNSLO2   ;TYPE WARNING MESSAGE
64 007150 000760          BR       6$+4      ;TRY AGAIN
65 007152 013737 001176 001276 7$:  MOV      $TMP1,$BASE ;STORE NEW BUS ADDRESS
66
67 007160 104401 066417          8$:      TYPE      ,CNSLO3
68 007164 005046          CLR      -(SP)
69 007166 113716 001272        MOVB     $VECT1,(SP) ;GET CURRENT VECTOR ADDRESS
70 007172 104403          TYPOS
71 007174      003        .BYTE   3          ;TYPE 3 DIGITS
72 007175      000        .BYTE   0          ;SUPPRESS LEADING ZEROS
73 007176 104401 066211          TYPE      ,QUES      ;TYPE " ? "
74 007202 104413          RDOCT      ;GET NEW VECTOR ADDRESS
75 007204 012637 001176        MOV      (SP)+,$TMP1 ;CARRIAGE RETURN?
76 007210 001412          BEQ      10$        ;YES-SKIP TO NEXT ENTRY
77 007212 022737 001000 001176  CMP      #1000,$TMP1 ;VECTOR ADDRESS < 1000 ?
78 007220 101003          BHI     9$         ;YES!!
79 007222 104401 066437          TYPE      ,CNSLO4   ;TYPE WARNING MESSAGE
80 007226 000754          BR       8$         ;RETRY
81 007230 113737 001176 001272 9$:  MOVB     $TMP1,$VECT1 ;STORE NEW VECTOR ADDRESS
82
83 007236 104401 066473          10$:   TYPE      ,CNSLO5
84 007242 005046          CLR      -(SP)
85 007244 113716 001273        MOVB     $VECT1+1,(SP) ;GET CURRENT BR LEVEL
86 007250 006216          ASR     (SP)
87 007252 006216          ASR     (SP)
88 007254 006216          ASR     (SP)
89 007256 006216          ASR     (SP)
90 007260 006216          ASR     (SP)
91 007262 104403          TYPOS
92 007264      001        .BYTE   1          ;ONLY 1 DIGIT
93 007265      000        .BYTE   0          ;SUPPRESS LEADING ZEROS
94 007266 104401 066211          TYPE      ,QUES
95 007272 104413          RDOCT      ;GET NEW PRIORITY
96 007274 012637 001176        MOV      (SP)+,$TMP1 ;CARRIAGE RETURN ?
97 007300 001424          BEQ      12$        ;YES-SKIP TO NEXT ENTRY
98 007302 023727 001176 000007  CMP      $TMP1,#7    ;LEGAL PRIORITY ?
99 007310 002403          BLT     11$        ;YES!!
100 007312 104401 066505          TYPE      ,CNSLO6   ;TYPE WARNING MESSAGE
101 007316 000747          BR       10$       ;TRY AGAIN
102 007320 006337 001176          11$:   ASL     $TMP1      ;STORE NEW PRIORITY
103 007324 006337 001176          ASL     $TMP1
104 007330 006337 001176          ASL     $TMP1
105 007334 006337 001176          ASL     $TMP1
106 007340 006337 001176          ASL     $TMP1
107 007344 113737 001176 001273  MOVB     $TMP1,$VECT1+1
108
109          ;DIALOGUE TO INPUT DEVICE NUMBERS
110 007352          12$:
111 007352 005227 177777          INC     #-1
112 007356 001002          BNE     13$
113 007360 104401 066536          TYPE      ,CNSLO7   ;TYPE INPUT INSTRUCTIONS

```



```

1          ;ASSEMBLE TEST QUE FROM DEVICE MAP
2 007604   CMNSTART:
3 007604   013700   001300   MOV     $DEVN,R0       ;R0 = DEVICE MAP
4 007610   012701   001466   MOV     #TSTQUE+2,R1  ;R1 = ADDRESS OF FIRST ENTRY IN QUE
5 007614   010137   001464   MOV     R1,TSTQUE    ;INITIALIZE ENTRY POINTER
6 007620   012702   000001   MOV     #1,R2        ;R2 = DEVICE POINTER
7 007624   005003           CLR     R3           ;R3 = DEVICE NUMBER
8 007626   030200   1$:    BIT     R2,R0        ;IS THIS DEVICE IN MAP ?
9 007630   001406           BEQ     2$          ;NO !!
10 007632   010311           MOV     R3,(R1)     ;YES - ENTER DEVICE NUMBER IN QUE
11 007634   116361   067146   000001  MOVB   ATNTBL(R3),1(R1);ENTER ATTENTION BIT IN QUE
12 007642   062701   000002           ADD     #2,R1      ;ADVANCE ENTRY POINTER
13 007646   006302   2$:    ASL     R2          ;ADVANCE DEVICE POINTER
14 007650   105702           TSTB   R2          ;DONE ALL DEVICES ?
15 007652   001402           BEQ     3$          ;YES
16 007654   005203           INC     R3          ;ADVANCE DEVICE NUMBER
17 007656   000763           BR     1$          ;ENTER NEXT DEVICE
18 007660   005011   3$:    CLR     (R1)      ;TERMINATE TEST QUE
19
20          ;SIZE FOR CLOCK
21 007662   004737   044644   JSR     PC,SIZCLK    ;SEE IF CLOCK PRESENT
22 007666   000413           BR     5$          ;YES - CLOCK IS PRESENT
23 007670   4$:
24 007672   104000           EMT
   007676   104401   007700   TYPE   ,65$         ;;TYPE ASCIZ STRING
   007676   000405           BR     64$         ;;GET OVER THE ASCIZ
   ;;65$: .ASCIZ <CRLF>/PROG HLT/
   64$:
25 007712   000000           HALT
26 007714   000765           BR     4$          ;PROGRAM HALT !!
27 007716   5$:
   .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
   007716   005737   000042   TST    @#42         ;;ARE WE RUNNING UNDER XXDP/ACT?
   007722   001012           BNE    66$         ;;BRANCH IF YES
   007724   123727   001242   000001  CMPB   $ENV,#1     ;;ARE WE RUNNING UNDER APT?
   007732   001406           BEQ    66$         ;;BRANCH IF YES
   007734   023727   001154   000176  CMP    SWR,#SWREG  ;;SOFTWARE SWITCH REG SELECTED?
   007742   001005           BNE    67$         ;;BRANCH IF NO
   007744   104407           GTSWR
   007746   000403           BR     67$         ;;GET SOFT-SWR SETTINGS
   007750   112737   000001   001150  66$:  MOVB   #1,$AUTOB   ;;SET AUTO-MODE INDICATOR
   67$:
28
29 007756   000240   READY:  NOP
30 007760   105737   001300   TSTB   $DEVN       ;READY TO START TEST
31 007764   001007           BNE    2$          ;ANY DRIVES IN MAP ?
32 007766   005737   000042   TST    @#42        ;BR IF YES
33 007772   001002           BNE    1$          ;ANY MONITOR PRESENT ?
34 007774   000137   005412   JMP     START      ;BR IF YES
35 010000   000137   042136   1$:    JMP     $EOP       ;JUMP TO START
36
37 010004   105037   001116   2$:    CLRB   $TSTNM     ;RETURN CONTROL TO MONITOR
38 010010   005037   001206   CLR     $TIMES     ;RESET TEST NUMBER
39 010014   005037   001326   CLR     CTLFG      ;INITIALIZE NUMBER OF ITERATIONS
40 010020   004737   064120   JSR     PC,$TKINT  ;CLEAR CONTROL-C FLAG
41 010024   012746   000300   MOV     #PR6,-(SP) ;INITIALIZE TTY
   010030   012746   010036   MOV     #64$,-(SP) ;PUT NEW PS ON STACK
   ;PUT NEW PC ON STACK

```

```

010034 000002          RTI          ;;POP NEW PC AND PS
010036
42 010036 117737 171422 001234 64$:  MOVB  @TSTQUE,$UNIT  ;LOAD UNIT NUMBER
43
44          ;CLEAR MASSBUS CONTROLLER, SELECT DRIVE AND DETERMINE THE LAST TRACK
45          ;OF THE DIFFERENT DRIVE TYPES
46 010044 012737 002000 001332      MOV  #TA4,LSTRK      ;ASSUME LAST TRACK FOR RM02/3 = 4.
47 010052 013700 001276              MOV  $BASE,R0        ;R0 = UNIBUS ADDRESS
48 010056 012760 000040 000010      MOV  #CLR,RMCS2(R0) ;CLEAR MASSBUS
49 010064 117760 171374 000010      MOVB @TSTQUE,RMCS2(R0);SELECT DEVICE UNDER TEST
50 010072 016002 000026              MOV  RMDT(R0),R2    ;GET RMDT AND
51 010076 042702 177770              BIC  #177770,R2     ;SAVE DRIVE TYPE BITS
52 010102 022702 000007              CMP  #7,R2          ;IS IT AN RM05 ?
53 010106 001003                    BNE  3$             ;NO, MUST BE AN RM02 OR RM03
54 010110 012737 011000 001332      MOV  #TA16!TA2,LSTRK;YES--SET LAST TRACK = 18.
55 010116          3$:

```

1
2

```

*****
*TEST 1          CONTROLLER ACCESS TEST
*****
TST1:
    
```

010116				SCOPE	:SCOPE CALL
010116	000004			NOP	:START OF TEST
010120	000240			MOV #STACK,SP	:INITIALIZE STACK POINTER
010122	012706	001100		MOV \$BASE,R0	:R0 = UNIBUS ADDRESS
010126	013700	001276		MOV TSTQUE,R1	:(R1) = DEVICE BEING TESTED
010132	013701	001464		MOV #1,\$TESTN	::SET TEST NUMBER IN APT MAIL BOX
010136	012737	000001	001226		
3					
4	010144	005001		CLR R1	
5	010146	013746	000004	MOV ERRVEC,-(SP)	::PUSH ERRVEC ON STACK
6	010152	013746	000006	MOV ERRVEC+2,-(SP)	::PUSH ERRVEC+2 ON STACK
7	010156	012737	010250	MOV #3\$,ERRVEC	
8	010164	012737	000300	MOV #PR6,ERRVEC+2	
9					
10	010172	110160	000001	MOVB R1,RMCS1+1(R0)	:MOVE HI BYTE TO RMCS1
11	010176	010160	000002	MOV R1,RMWC(R0)	:MOVE WORD COUNT REGISTER
12	010202	016002	000002	MOV RMWC(R0),R2	
13	010206	010160	000004	MOV R1,RMBA(R0)	:MOVE BUS ADDRESS REGISTER
14	010212	016002	000004	MOV RMBA(R0),R2	
15	010216	010160	000010	MOV R1,RMCS2(R0)	:MOVE CONTROL STATUS REGISTER
16	010222	016002	000010	MOV RMCS2(R0),R2	
17	010226	010160	000022	MOV R1,RMDB(R0)	:MOVE DATA BUFFER
18	010232	016002	000022	MOV RMDB(R0),R2	
19	010236	012637	000006	MOV (SP)+,ERRVEC+2	::POP STACK INTO ERRVEC+2
20	010242	012637	000004	MOV (SP)+,ERRVEC	::POP STACK INTO ERRVEC
21	010246	000415		BR 7\$:NO BUS TIMEOUT OCCURRED
22					
23	010250	022626		3\$: CMP (SP)+,(SP)+	:ADJUST STACK
24	010252	012637	000006	MOV (SP)+,ERRVEC+2	::POP STACK INTO ERRVEC+2
25	010256	012637	000004	MOV (SP)+,ERRVEC	::POP STACK INTO ERRVEC
26	010262	104110		EMT 110	
27	010264	005737	000042	TST @#42	:STAND ALONE MODE ?
28	010270	001002		BNE 5\$:NO!!
29	010272	000137	005412	JMP START	:YES-GO GET \$BASE
30					
31	010276	000137	042136	5\$: JMP \$EOP	:GO TO END OF PASS HANDLER
32	010302			7\$:	
33					
34					

```

*****
*TEST 2          DEVICE AVAILABLE TEST
*****
TST2:
    
```

010302				SCOPE	:SCOPE CALL
010302	000004			NOP	:START OF TEST
010304	000240			MOV #STACK,SP	:INITIALIZE STACK POINTER
010306	012706	001100		MOV \$BASE,R0	:R0 = UNIBUS ADDRESS
010312	013700	001276		MOV TSTQUE,R1	:(R1) = DEVICE BEING TESTED
010316	013701	001464		MOV #2,\$TESTN	::SET TEST NUMBER IN APT MAIL BOX
010322	012737	000002	001226		
35					
36	010330	004737	053530	JSR PC,CNTCLR	:GO ISSUE CONTROLLER CLEAR
	010334	000404		BR 2\$:GO TO 2\$ IF NO ERROR
	010336	000240		NOP	:RETURN HERE IF ERROR
	010340	104000		EMT	:ERROR NUMBER DEFINED BY SUBROUTINE

```

37 010342 000137 010462          2$: JMP 7$ ;GO TO 7$ IF ERROR
38 010346 013746 000004          MOV ERRVEC,-(SP) ;;PUSH ERRVEC ON STACK
39 010352 013746 000006          MOV ERRVEC+2,-(SP) ;;PUSH ERRVEC+2 ON STACK
40 010356 012737 010446 000004   MOV #5$,ERRVEC
41 010364 013737 000300 000006   MOV PR6,ERRVEC+2
42
43 010372 016037 000000 001176   MOV RMCS1(RO),$TMP1 ;GET DVA STATUS
44 010400 016037 000010 001174   MOV RMCS2(RO),$TMP0 ;GET NED STATUS
45 010406 012637 000006          MOV (SP)+,ERRVEC+2 ;;POP STACK INTO ERRVEC+2
46 010412 012637 000004          MOV (SP)+,ERRVEC ;;POP STACK INTO ERRVEC
47 010416 032737 010000 001174   BIT #NED,$TMP0 ;NONEXISTENT DEVICE ?
48 010424 001402          BEQ 3$ ;NO!!
49 010426 104111          EMT 111
50 010430 000414          BR 7$
51 010432 032737 004000 001176   3$: BIT #DVA,$TMP1 ;DEVICE AVAILABLE ?
52 010440 001012          BNE 9$ ;YES!!
53 010442 104112          EMT 112
54 010444 000406          BR 7$
55
56 010446 022626          5$: CMP (SP)+,(SP)+ ;ADJUST STACK
57 010450 012637 000006          MOV (SP)+,ERRVEC+2 ;;POP STACK INTO ERRVEC+2
58 010454 012637 000004          MOV (SP)+,ERRVEC ;;POP STACK INTO ERRVEC
59 010460 104113          EMT 113
60 010462 000137 042100          7$: JMP $EOSP
61 010466          9$:
62
63

```

```

*****
;*TEST 3 DRIVE TYPE TEST
*****
TST3:

```

```

010466
010466 000004          SCOPE ;SCOPE CALL
010470 000240          NOP ;START OF TEST
010472 012706 001100   MOV #STACK,SP ;INITIALIZE STACK POINTER
010476 013700 001276   MOV $BASE,RO ;RO = UNIBUS ADDRESS
010502 013701 001464   MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
010506 012737 000003 001226   MOV #3,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
64
65 010514 004737 053530          JSR PC,CNTCLR ;GO ISSUE CONTROLLER CLEAR
010520 000404          BR 2$ ;GO TO 2$ IF NO ERROR
010522 000240          NOP ;RETURN HERE IF ERROR
010524 104000          EMT ;ERROR NUMBER DEFINED BY SUBROUTINE
010526 000137 010664          JMP 4$ ;GO TO 4$ IF ERROR
66 010532          2$:
67 010532 112737 000026 001514   MOVB #RMDT,GETINX ;SETUP GET INDEX TABLE
010540 112737 000200 001515   MOVB #200,GETINX+1 ;SETUP TERMINATOR BYTE
010546 012737 010670 001362   MOV #5$,RMDTI ;SET RMDT INPUT BUFFER = 5$
010554 004737 044154          JSR PC,GET ;GO READ RMDT VIA GET SUBROUTINE
010560 000402          BR 3$ ;GO TO 3$ IF NO ERROR
010562 000240          NOP ;RETURN HERE IF ERROR
010564 104000          EMT ;ERROR DEFINED BY GET SUBROUTINE
68
69 010566 022737 020024 001362   3$: CMP #SNGPRT,RMDTI ;SINGLE PORT RM03 ?
70 010574 001435          BEQ 5$ ;YES !!
71 010576 022737 024024 001362   CMP #DULPRT,RMDTI ;DUAL PORT RM03 ?
72 010604 001431          BEQ 5$ ;YES !!
73

```

```

74 010606 022737 020025 001362      CMP      #SNGPRT!BIT0,RMDTI      ;SINGLE PORT RM02 ?
75 010614 001425                    BEQ      5$                      ;YES !!
76 010616 022737 024025 001362      CMP      #DULPRT!BIT0,RMDTI      ;DUAL PORT RM02 ?
77 010624 001421                    BEQ      5$                      ;YES !!
78
79 010626 022737 020027 001362      CMP      #SNGPRT!BIT1!BIT0,RMDTI ;SINGLE PORT RM05 ?
80 010634 001415                    BEQ      5$                      ;YES !!
81 010636 022737 024027 001362      CMP      #DULPRT!BIT1!BIT0,RMDTI ;DUAL PORT RM05 ?
82 010644 001411                    BEQ      5$
83
84 010646 012737 020024 001176      MOV      #SNGPRT,$TMP1          ;LOAD ACCEPTABLE VALUES
85 010654 012737 024024 001200      MOV      #DULPRT,$TMP2
86 010662 104114                    EMT      114
87
88 010664 000137 042100              4$:    JMP      $EOSP              ;GO TO SUBPASS HANDLER.
89 010670                    5$:
90
91
;*****
;*TEST 4          UNIBUS INITIALIZE TEST
;*****
TST4:
010670                    SCOPE          ;SCOPE CALL
010670 000004                    NOP          ;START OF TEST
010672 000240                    MOV      #STACK,SP      ;INITIALIZE STACK POINTER
010674 012706 001100            MOV      $BASE,R0       ;R0 = UNIBUS ADDRESS
010700 013700 001276            MOV      TSTQUE,R1      ;(R1) = DEVICE BEING TESTED
010704 013701 001464            MOV      #1,$TIMES      ;;DO 1 ITERATION
010710 012737 000001 001206      MOV      #4,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
010716 012737 000004 001226
92
93 010724 004737 053530          JSR      PC,CNTCLR      ;GO ISSUE CONTROLLER CLEAR
010730 000404                    BR        10$           ;GO TO 10$ IF NO ERROR
010732 000240                    NOP          ;RETURN HERE IF ERROR
010734 104000                    EMT
010736 000137 011540          JMP      220$          ;ERROR NUMBER DEFINED BY SUBROUTINE
94 010742                    10$:    ;GO TO 220$ IF ERROR
95 010742 012702 000101          MOV      #65.,R2       ;SET OR AND RESET IR
96 010746 012737 000000 001432      MOV      #0,RMDBO      ;WRITE ZEROS IN DATA SILO
97 010754 112737 000022 001543      MOVB     #RMDB,PUTINX   ;SETUP REGISTER INDEX
98 010762 112737 000200 001544      MOVB     #200,PUTINX+1
99 010770                    20$:
100 010770 004737 044424          JSR      PC,PUT        ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
010774 000404                    BR        30$          ;GO TO 30$ IF NO ERROR
010776 000240                    NOP          ;RETURN HERE IF ERROR
011000 104000                    EMT          ;ERROR # DEFINED BY PUT SUBROUTINE
011002 000137 011540          JMP      220$          ;GO TO 220$ IF ERROR
101 011006 005302                    30$:    DEC      R2            ;DECREMENT COUNT
102 011010 001367                    BNE     20$           ;WRITE SILO AGAIN
103 011012 012737 177777 001414      MOV      #-1,RMBAO     ;RMBA = ALL 1'S
104 011020 012737 177777 001424      MOV      #-1,RMER10    ;RMER1 = ALL 1'S
105 011026 012737 177777 001452      MOV      #-1,RMER20    ;RMER2 = ALL 1'S
106 011034 012737 040001 001434      MOV      #DMD!DBEN,RMMR10 ;SET DIAGNOSTIC MODE
107 011042 012737 003577 001410      MOV      #003577,RMCS10
108 011050 012737 021037 001420      MOV      #021037,RMCS20
109
110 011056 012702 001543          MOV      #PUTINX,R2     ;R2 = ADDRESS OF INDEX TABLE
111 011062 112722 000004          MOVB     #RMBA,(R2)+
112 011066 112722 000014          MOVB     #RMER1,(R2)+

```



```

113 011072 112722 000042      MOVB  #RMER2,(R2)+
114 011076 112722 000024      MOVB  #RMMR1,(R2)+
115 011102 112722 000000      MOVB  #RMCS1,(R2)+
116 011106 112722 000010      MOVB  #RMCS2,(R2)+
117 011112 112722 000200      MOVB  #200,(R2)+
118 011116 004737 044424      JSR   PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      011122 000404      BR    40$           ;GO TO 40$ IF NO ERROR
      011124 000240      NOP                    ;RETURN HERE IF ERROR
      011126 104000      EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      011130 000137 011540      JMP   220$          ;GO TO 220$ IF ERROR
119 011134 000005      40$: RESET          ;UNIBUS INITIALIZE
120 011136 004737 064120      JSR   PC,$TKINT      ;INITIALIZE CONSOLE
121 011142 111160 000010      MOVB  (R1),RMCS2(RO) ;SELECT DEVICE
122 011146 004737 044070      JSR   PC,GETSTS      ;GO SET UP FOR STATUS FETCH
123
124 011152 004737 044154      JSR   PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      011156 000403      BR    50$           ;GO TO 50$ IF NO ERROR
      011160 000240      NOP                    ;RETURN HERE IF ERROR
      011162 104000      EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
125 011164 000565      BR    220$          ;SKIP REMAINDER OF TEST
126
127 011166 013737 001334 001142 50$: MOV   RMCS1I,$BDDAT  ;VERIFY RMCS1
128 011174 042737 100000 001142      BIC   #SC,$BDDAT     ;IGNORE SPECIAL CONDITION
129 011202 012737 004200 001140      MOV   #DVA!RDY,$GDDAT ;EXPECT DVA & RDY
130 011210 023737 001140 001142      CMP   $GDDAT,$BDDAT  ;COMPARE EXPECTED, RECEIVED
131 011216 001401      BEQ   60$           ;BRANCH IF EQUAL
132 011220 104115      EMT   115
133
134 011222 005037 001140      60$: CLR   $GDDAT        ;VERIFY RMBA IS ZERO
135 011226 013737 001340 001142      MOV   RMBAI,$BDDAT
136 011234 001401      BEQ   70$           ;BRANCH IF ZERO
137 011236 104116      EMT   116
138
139 011240 013737 001344 001142 70$: MOV   RMCS2I,$BDDAT ;VERIFY RMCS2
140 011246 005046      CLR   -(SP)         ;EXPECT IR & UNIT NUMBER
141 011250 111116      MOVB  (R1),(SP)
142 011252 052716 000100      BIS   #IR,(SP)
143 011256 012637 001140      MOV   (SP)+,$GDDAT
144 011262 023737 001140 001142      CMP   $GDDAT,$BDDAT ;COMPARE EXPECTED & RECEIVED
145 011270 001401      BEQ   90$           ;BRANCH IF EQUAL
146 011272 104117      EMT   117
147
148 011274 005037 001140      90$: CLR   $GDDAT        ;VERIFY RMER1
149 011300 013737 001350 001142      MOV   RMER1I,$BDDAT
150 011306 042737 040000 001142      BIC   #UNS,$BDDAT    ;IGNORE UNSAFE
151 011314 001401      BEQ   110$          ;BRANCH IF ZERO
152 011316 104120      EMT   120
153
154 011320 013737 001352 001142 110$: MOV   RMAI,$BDDAT   ;VERIFY RMAI
155 011326 005002      CLR   R2            ;CLEAR ALL BUT THIS
156 011330 116102 000001      MOVB  1(R1),R2      ;DRIVES ATTENTION BIT
157 011334 000302      SWAB  R2
158 011336 005102      COM   R2
159 011340 000240      NOP
160 011342 040237 001142      BIC   R2,$BDDAT
161 011346 001401      BEQ   130$          ;BRANCH IF ITS 0
162 011350 104121      EMT   121

```

```

163
164 011352 013737 001360 001142 130$: MOV RMMR11,$BDDAT ;VERIFY RMMR
165 011360 042737 000046 001142 BIC #WC!LS!LST,$BDDAT ;IGNORE WORD CLOCK, SCT, TRK
166 011366 012737 000010 001140 MOV #MWD,$GDDAT ;EXPECT WRITE DATA BIT
167 011374 023737 001140 001142 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED AND RECEIVED
168 011402 001401 BEQ 170$ ;BRANCH IF 0
169 011404 104122 EMT 122
170
171 011406 005037 001140 170$: CLR $GDDAT ;EXPECT ZEROS
172 011412 013737 001402 001142 MOV RMEC2I,$BDDAT ;VERIFY RMEC2 = 0
173 011420 001401 BEQ 190$
174 011422 104124 EMT 124
175
176 011424 013737 001374 001142 190$: MOV RMMR2I,$BDDAT ;VERIFY RMMR2
177 011432 042737 140000 001142 BIC #RQA!RQB,$BDDAT
178 011440 012737 011777 001140 MOV #TST!1777,$GDDAT ;EXPECT TEST,TAG BIT ON
179 011446 023737 001140 001142 CMP $GDDAT,$BDDAT
180 011454 001401 BEQ 210$
181 011456 104125 EMT 125
182
183 011460 005037 001140 210$: CLR $GDDAT ;EXPECT ALL ZEROS
184 011464 013737 001376 001142 MOV RMER2I,$BDDAT ;VERIFY RMER2
185 011472 042737 040200 001142 BIC #SKI!DVC,$BDDAT ;IGNORE DEVICE ERRORS
186 011500 001401 BEQ 215$ ;BRANCH IF OTHER BITS 0
187 011502 104173 EMT 173
188 011504 013737 001346 001142 215$: MOV RMDSI,$BDDAT ;CHECK DRIVE STATUS REGISTER
189 011512 042737 177177 001142 BIC #^C<DPR!DRY>,$BDDAT
190 011520 012737 000600 001140 MOV #DPR!DRY,$GDDAT ;EXPECTED STATUS
191 011526 023737 001142 001140 CMP $BDDAT,$GDDAT ;COMPARE EXPECTED & RECEIVED STATUS
192 011534 001401 BEQ 220$ ;DRIVE STATUS IS OK
193 011536 104123 EMT 123
194 011540 220$:
195
196
*****
;*TEST 5 CONTROLLER CLEAR TEST
*****
TST5:
011540 SCOPE ;SCOPE CALL
011540 000004 NOP ;START OF TEST
011542 000240 MOV #STACK,SP ;INITIALIZE STACK POINTER
011544 012706 001100 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
011550 013700 001276 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
011554 013701 001464 MOV #5,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
011560 012737 000005 001226
197
198 011566 004737 053530 JSR PC,CNTCLR ;GO ISSUE CONTROLLER CLEAR
011572 000404 BR 10$ ;GO TO 10$ IF NO ERROR
011574 000240 NOP ;RETURN HERE IF ERROR
011576 104000 EMT ;ERROR NUMBER DEFINED BY SUBROUTINE
011600 000137 012076 JMP 80$ ;GO TO 80$ IF ERROR
199 011604 10$:
200 011604 012702 000101 MOV #65.,R2
201 011610 012737 000000 001432 MOV #0,RMDBO ;WRITE ZEROS IN DATA SILO
202 011616 112737 000022 001543 MOVB #RMDB,PUTINX ;SETUP REGISTER INDEX TABLE
203 011624 112737 000200 001544 MOVB #200,PUTINX+1
204 011632 20$:
205 011632 004737 044424 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
011636 000404 BR 30$ ;GO TO 30$ IF NO ERROR

```

```
011640 000240 NOP ;RETURN HERE IF ERROR
011642 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
011644 000137 012076 JMP 80$ ;GO TO 80$ IF ERROR
206 011650 005302 30$: DEC R2 ;DECREMENT COUNT
207 011652 001367 BNE 20$ ;AND WRITE SILO AGAIN IF NOT DONE
208 011654 012737 177777 001414 MOV #-1,RMBAO
209 011662 012737 177777 001424 MOV #-1,RMER10
210 011670 012737 177777 001452 MOV #-1,RMER20
211 011676 012737 040001 001434 MOV #DMD!DBEN,RMMR10
212 011704 012737 003577 001410 MOV #003577,RMCS10
213 011712 012737 021037 001420 MOV #021037,RMCS20
214 011720 012702 001543 MOV #PUTINX,R2 ;R2 = ADDRESS OF INDEX TABLE
215 011724 112722 000004 MOVB #RMBA,(R2)+
216 011730 112722 000014 MOVB #RMER1,(R2)+
217 011734 112722 000042 MOVB #RMER2,(R2)+
218 011740 112722 000024 MOVB #RMMR1,(R2)+
219 011744 112722 000000 MOVB #RMCS1,(R2)+
220 011750 112722 000010 MOVB #RMCS2,(R2)+
221 011754 112722 000200 MOVB #200,(R2)+
222 011760 004737 044424 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
011764 000403 BR 40$ ;GO TO 40$ IF NO ERROR
011766 000240 NOP ;RETURN HERE IF ERROR
011770 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
223 011772 000430 BR 70$
224
225 011774 40$: JSR PC,CNTCLR ;GO ISSUE CONTROLLER CLEAR
226 011774 004737 053530 BR 50$ ;GO TO 50$ IF NO ERROR
012000 000404 NOP ;RETURN HERE IF ERROR
012002 000240 EMT ;ERROR NUMBER DEFINED BY SUBROUTINE
012004 104000 JMP 80$ ;GO TO 80$ IF ERROR
227 012006 000137 012076 50$: JSR PC,GETSTS ;SETUP TO READ ALL REGISTERS
228 012012 004737 044070
229 012016 004737 044154 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
012022 000404 BR 60$ ;GO TO 60$ IF NO ERROR
012024 000240 NOP ;RETURN HERE IF ERROR
012026 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
012030 000137 012076 JMP 80$ ;GO TO 80$ IF ERROR
230 012034 60$: JSR PC,CLRSTS ;GO VERIFY CONTROLLER CLEAR OPERATION
231 012034 004737 053646 BR 70$ ;GO TO 70$ IF NO ERROR
012040 000405 NOP ;RETURN HERE IF ERROR
012042 000240 EMT ;ERROR # DEFINED BY CLRSTS SUBROUTINE
012044 104000 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
012046 004736 JMP 80$ ;GO TO 80$ IF ERROR
232 012054 70$: MOV #NOP,RMCS10 ;CHANGE FUNCTION CODE FOR ERROR CHECK
233 012054 012737 000000 001410
234
235 012062 004737 046004 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
012066 000403 BR 80$ ;GO TO 80$ IF NO ERROR
012070 000240 NOP ;RETURN HERE IF ERROR
012072 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
012074 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
236 012076 80$:
237
238
```

.....
: *TEST 6 ERROR CLEAR TEST

```

012076
012076 000004
012100 000240
012102 012706 001100
012106 013700 001276
012112 013701 001464
012116 012737 000006 001226
239
240 012124 004737 053530
012130 000404
012132 000240
012134 104000
012136 000137 012302
241 012142
242 012142 112737 000000 001543
012150 112737 000200 001544
012156 012737 040000 001410
012164 004737 044424
012170 000403
012172 000240
012174 104000
243 012176 000441
244 012200 112737 000000 001514
245 012206 112737 000010 001515
246 012214 112737 000200 001516
247
248 012222 004737 044154
012226 000403
012230 000240
012232 104000
249 012234 000422
250 012236 013737 001334 001142
251 012244 005037 001140
252 012250 042737 117777 001142
253 012256 001401
254 012260 104137
255
256 012262 013737 001344 001142
257 012270 042737 104377 001142
258 012276 001401
259 012300 104140
260
261 012302
262
263

```

```

.....
TST6:
SCOPE ;SCOPE CALL
NOP ;START OF TEST
MOV #STACK,SP ;INITIALIZE STACK POINTER
MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
MOV #6,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

JSR PC,CNTCLR ;GO ISSUE CONTROLLER CLEAR
BR 10$ ;GO TO 10$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
EMT ;ERROR NUMBER DEFINED BY SUBROUTINE
JMP 50$ ;GO TO 50$ IF ERROR

10$:
MOVB #RMCS1,PUTINX ;SETUP PUT INDEX TABLE
MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
MOV #TRE,RMCS10 ;SET RMCS1 OUTPUT BUFFER = TRE
JSR PC,PUT ;GO WRITE RMCS1 VIA PUT SUBROUTINE
BR 20$ ;GO TO 20$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
EMT ;ERROR DEFINED BY PUT SUBROUTINE
BR 50$ ;SKIP REST OF TEST

20$:
MOVB #RMCS1,GETINX
MOVB #RMCS2,GETINX+1
MOVB #200,GETINX+2

JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
BR 30$ ;GO TO 30$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
EMT ;ERROR # DEFINED BY GET SUBROUTINE
BR 50$ ;SKIP REST OF TEST

30$:
MOV RMCS11,$BDDAT ;CHECK TRE & MCPE
CLR $GDDAT ;EXPECT 0'S
BIC #^C<TRE!MCPE>,$BDDAT
BEQ 40$ ;BRANCH IF TRE & MCPE = 0
EMT 137

40$:
MOV RMCS21,$BDDAT ;CHECK RMCS2
BIC #^C<WCE!UPE!NED!PGE!MXF!MDPE>,$BDDAT
BEQ 50$
EMT 140

50$:

```

```

.....
*TEST 7 DRIVE STATUS TEST
.....
TST7:

```

```

012302
012302 000004
012304 000240
012306 012706 001100
012312 013700 001276
012316 013701 001464
012322 012737 000007 001226
264
265 012330 004737 053530

```

```

SCOPE ;SCOPE CALL
NOP ;START OF TEST
MOV #STACK,SP ;INITIALIZE STACK POINTER
MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
MOV #7,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

JSR PC,CNTCLR ;GO ISSUE CONTROLLER CLEAR

```

```
012334 000404 BR 10$ ;GO TO 10$ IF NO ERROR
012336 000240 NOP ;RETURN HERE IF ERROR
012340 104000 EMT ;ERROR NUMBER DEFINED BY SUBROUTINE
012342 000137 012616 JMP 100$ ;GO TO 100$ IF ERROR
266 012346 10$: JSR PC,GETSTS
267 012346 004737 044070
268
269 012352 004737 044154 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
012356 000403 BR 20$ ;GO TO 20$ IF NO ERROR
012360 000240 NOP ;RETURN HERE IF ERROR
012362 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
270 012364 000514 BR 100$ ;SKIP REST OF TEST
271 012366 032737 010000 001346 20$: BIT #MOL,RMDSI ;MEDIUM ON LINE??
272 012374 001016 BNE 30$ ;YES!!
273 012376 013737 001346 001140 MOV RMDSI,$GDDAT ;EXPECTED STATUS
274 012404 042737 162000 001140 BIC #ATA!ERR!PIP!LBT,$GDDAT
275 012412 052737 010000 001140 BIS #MOL,$GDDAT
276 012420 013737 001346 001142 MOV RMDSI,$BDDAT ;RECEIVED STATUS
277 012426 104151 EMT 151 ;MOL STATUS INCORRECT
278 012430 000440 BR 70$
279
280 012432 032737 000200 001376 30$: BIT #DVC,RMER2I ;IS THERE A DEVICE CHECK??
281 012440 001422 BEQ 60$ ;BR IF NO
282 012442 013737 001376 001142 MOV RMER2I,$BDDAT ;RECEIVED STATUS
283 012450 005037 001140 CLR $GDDAT ;EXPECTED STATUS
284 012454 104152 EMT 152 ;DRIVE FAULT
285
286 012456 032737 040000 001350 BIT #UNS,RMER1I ;IS UNS SET??
287 012464 001022 BNE 70$ ;YES!!
288 012466 013737 001350 001142 MOV RMER1I,$BDDAT ;RECEIVED STATUS
289 012474 012737 040000 001140 MOV #UNS,$GDDAT ;EXPECTED STATUS
290 012502 104153 EMT 153 ;'DVC' IS SET BUT 'UNS' IS NOT
291 012504 000412 BR 70$
292
293 012506 032737 040000 001350 60$: BIT #UNS,RMER1I ;IS UNS SET??
294 012514 001410 BEQ 80$ ;NO!!
295 012516 013737 001350 001142 MOV RMER1I,$BDDAT ;RECEIVED STATUS
296 012524 005037 001140 CLR $GDDAT ;EXPECTED STATUS
297 012530 104154 EMT 154 ;'DVC' IS NOT SET, BUT 'UNS' IS
298 012532 000137 042100 70$: JMP $EOSP ;SKIP REST OF TEST
299
300 012536 032737 004000 001346 80$: BIT #WRL,RMDSI ;IS WRITE PROTECT ON??
301 012544 001412 BEQ 90$ ;NO!!
302 012546 013737 001346 001140 MOV RMDSI,$GDDAT ;EXPECTED STATUS
303 012554 042737 166076 001140 BIC #^C<MOL!PGM!DPR!DRY!VV!OM>,$GDDAT
304 012562 013737 001346 001142 MOV RMDSI,$BDDAT ;RECEIVED STATUS
305 012570 104175 EMT 175 ;SELECTED DEVICE IN WRITE PROTECT
306
307 012572 032737 040000 001376 90$: BIT #SKI,RMER2I ;IS SKI SET??
308 012600 001406 BEQ 100$ ;NO!!
309 012602 013737 001376 001142 MOV RMER2I,$BDDAT ;RECEIVED STATUS
310 012610 005037 001140 CLR $GDDAT ;EXPECTED STATUS
311 012614 104205 EMT 205 ;PERSISTENT SEEK INCOMPLETE ERROR
312 012616 100$:
313
314
```

:*TEST 10 PRIMARY/SECONDARY ERROR TEST

```

*****
TST10:
012616 012616 000004          SCOPE          :SCOPE CALL
012620 012620 000240          NOP            :START OF TEST
012622 012706 001100          MOV #STACK,SP :INITIALIZE STACK POINTER
012626 013700 001276          MOV $BASE,R0  :R0 = UNIBUS ADDRESS
012632 013701 001464          MOV TSTQUE,R1 : (R1) = DEVICE BEING TESTED
012636 012737 000010 001226  MOV #10,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
315
316 012644 004737 053530          JSR PC,CNTCLR :GO ISSUE CONTROLLER CLEAR
012650 000404          BR 1$         :GO TO 1$ IF NO ERROR
012652 000240          NOP           :RETURN HERE IF ERROR
012654 104000          EMT          :ERROR NUMBER DEFINED BY SUBROUTINE
012656 000137 012776          JMP 5$        :GO TO 5$ IF ERROR
317
318 012662 112737 000000 001543 1$:  MOVB #RMCS1,PUTINX :SETUP PUT INDEX TABLE
012670 112737 000200 001544  MOVB #200,PUTINX+1 :SET TERMINATOR BYTE
012676 012737 000023 001410  MOV #PAKACK!GO,RMCS10 :SET RMCS1 OUTPUT BUFFER = PAKACK!GO
012704 004737 044424          JSR PC,PUT    :GO WRITE RMCS1 VIA PUT SUBROUTINE
012710 000403          BR 2$        :GO TO 2$ IF NO ERROR
012712 000240          NOP           :RETURN HERE IF ERROR
012714 104000          EMT          :ERROR DEFINED BY PUT SUBROUTINE
319 012716 000427          BR 5$        :SKIP REST OF TEST IF ERROR
320 012720 004737 044070 2$:  JSR PC,GETSTS :SETUP FOR STATUS FETCH
321 012724 004737 044766          JSR PC,TIMOUT :WAIT FOR COMPLETION OF NOP
322
323 012730 004737 044154          JSR PC,GET    :GO READ REGISTER(S) WITH GET SUBROUTINE
012734 000403          BR 3$        :GO TO 3$ IF NO ERROR
012736 000240          NOP           :RETURN HERE IF ERROR
012740 104000          EMT          :ERROR # DEFINED BY GET SUBROUTINE
324 012742 000415          BR 5$        :SKIP REST OF TEST IF ERROR
325
326 012744 004737 045152 3$:  JSR PC,PRIERR :GO CHECK FOR PRIMARY ERRORS
012750 000404          BR 4$        :GO TO 4$ IF NO ERROR
012752 000240          NOP           :RETURN HERE IF ERROR
012754 104000          EMT          :ERROR # DEFINED BY PRIERR SUBROUTINE
012756 004736          JSR PC,@(SP)+ :GO BACK FOR MORE ERROR CHECKS
327 012760 000406          BR 5$        :SKIP REST OF TEST IF ERROR
328
329 012762 004737 046004 4$:  JSR PC,SECERR :GO CHECK FOR SECONDARY ERRORS
012766 000403          BR 5$        :GO TO 5$ IF NO ERROR
012770 000240          NOP           :RETURN HERE IF ERROR
012772 104000          EMT          :ERROR # DEFINED BY SECERR SUBROUTINE
012774 004736          JSR PC,@(SP)+ :GO BACK FOR MORE ERROR CHECKS
330
331
332
*****
:*TEST 11          DIAGNOSTIC MODE TEST
*****
TST11:
012776 012776 000004          SCOPE          :SCOPE CALL
013000 013000 000240          NOP            :START OF TEST
013002 012706 001100          MOV #STACK,SP :INITIALIZE STACK POINTER
013006 013700 001276          MOV $BASE,R0  :R0 = UNIBUS ADDRESS
013012 013701 001464          MOV TSTQUE,R1 : (R1) = DEVICE BEING TESTED
013016 012737 000011 001226  MOV #11,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
333

```

334	013024	004737	053530			JSR	PC,CNTCLR	:GO ISSUE CONTROLLER CLEAR
	013030	000404				BR	1\$:GO TO 1\$ IF NO ERROR
	013032	000240				NOP		:RETURN HERE IF ERROR
	013034	104000				EMT		:ERROR NUMBER DEFINED BY SUBROUTINE
	013036	000137	014050			JMP	22\$:GO TO 22\$ IF ERROR
335	013042				1\$:			
336	013042	112737	000024	001543		MOV	#RMMR1,PUTINX	:SETUP PUT INDEX TABLE
	013050	112737	000200	001544		MOV	#200,PUTINX+1	:SET TERMINATOR BYTE
	013056	012737	000001	001434		MOV	#DMD,RMMR10	:SET RMMR1 OUTPUT BUFFER = DMD
	013064	004737	044424			JSR	PC,PUT	:GO WRITE RMMR1 VIA PUT SUBROUTINE
	013070	000404				BR	2\$:GO TO 2\$ IF NO ERROR
	013072	000240				NOP		:RETURN HERE IF ERROR
	013074	104000				EMT		:ERROR DEFINED BY PUT SUBROUTINE
	013076	000137	014050			JMP	22\$:GO TO 22\$ IF ERROR
337	013102	004737	044070		2\$:	JSR	PC,GETSTS	:SETUP FOR STATUS FETCH
338								
339	013106	004737	044154			JSR	PC,GET	:GO READ REGISTER(S) WITH GET SUBROUTINE
	013112	000404				BR	3\$:GO TO 3\$ IF NO ERROR
	013114	000240				NOP		:RETURN HERE IF ERROR
	013116	104000				EMT		:ERROR # DEFINED BY GET SUBROUTINE
	013120	000137	014050			JMP	22\$:GO TO 22\$ IF ERROR
340	013124	032737	000001	001360	3\$:	BIT	#DMD,RMMR11	:IS DIAGNOSTIC MODE SET??
341	013132	001011				BNE	5\$:YES!!
342	013134	012737	000001	001140		MOV	#DMD,\$GDDAT	:EXPECTED STATUS
343	013142	013737	001360	001142		MOV	RMMR11,\$BDDAT	:RECEIVED STATUS
344	013150	104176				EMT	176	
345	013152	000137	014050			JMP	22\$:SKIP REST OF TEST IF DMD = 0
346	013156	032737	010000	001346	5\$:	BIT	#MOL,RMDSI	:IS 'MOL' = 0 ??
347	013164	001411				BEQ	6\$:YES!!
348	013166	013737	001346	001142		MOV	RMDSI,\$BDDAT	:SETUP BAD DATA FOR TYPEOUT
349	013174	042737	167777	001142		BIC	#^CMOL,\$BDDAT	
350	013202	005037	001140			CLR	\$GDDAT	:SETUP GOOD DATA FOR TYPEOUT
351	013206	104177				EMT	177	
352	013210	032737	020000	001346	6\$:	BIT	#PIP,RMDSI	:IS PIP SET??
353	013216	001012				BNE	7\$:YES!!
354	013220	013737	001346	001142		MOV	RMDSI,\$BDDAT	:SETUP BAD DATA FOR TYPEOUT
355	013226	042737	157777	001142		BIC	#^CPIP,\$BDDAT	
356	013234	012737	020000	001140		MOV	#PIP,\$GDDAT	:EXPECTED PIP SET
357	013242	104200				EMT	200	
358	013244	032737	004000	001346	7\$:	BIT	#WRL,RMDSI	:IS WRITE LOCK OFF??
359	013252	001411				BEQ	8\$:YES!!
360	013254	013737	001346	001142		MOV	RMDSI,\$BDDAT	:SETUP BAD DATA FOR TYPEOUT
361	013262	042737	173777	001142		BIC	#^CWRL,\$BDDAT	
362	013270	005037	001140			CLR	\$GDDAT	:EXPECTED WRL = 0
363	013274	104201				EMT	201	
364	013276	032737	040000	001376	8\$:	BIT	#SKI,RMER21	:IS SKI = 0
365	013304	001411				BEQ	9\$:YES!!
366	013306	013737	001376	001142		MOV	RMER21,\$BDDAT	:SETUP BAD DATA FOR TYPEOUT
367	013314	042737	137777	001142		BIC	#^CSKI,\$BDDAT	
368	013322	005037	001140			CLR	\$GDDAT	:SKI SHOULD BE 0
369	013326	104202				EMT	202	
370	013330	032737	000200	001376	9\$:	BIT	#DVC,RMER21	:IS DEVICE CHECK = 0??
371	013336	001411				BEQ	10\$:YES!!
372	013340	013737	001376	001142		MOV	RMER21,\$BDDAT	:SETUP BAD DATA FOR TYPEOUT
373	013346	042737	177577	001142		BIC	#^CDVC,\$BDDAT	
374	013354	005037	001140			CLR	\$GDDAT	:DVC SHOULD BE 0
375	013360	104203				EMT	203	

```

376 013362
377 013362 112737 000024 001543 10$: MOVB #RMMR1,PUTINX ;SETUP PUT INDEX TABLE
    013370 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
    013376 012737 001711 001434 MOV #DMD!MUR!MOC!MSER!MDF!MWP,RMMR10 ;SET RMMR1 OUTPUT BUFFER = DMD!MUR!M
    013404 004737 044424 JSR PC,PUT ;GO WRITE RMMR1 VIA PUT SUBROUTINE
    013410 000404 BR 11$ ;GO TO 11$ IF NO ERROR
    013412 000240 NOP ;RETURN HERE IF ERROR
    013414 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
    013416 000137 014050 JMP 22$ ;GO TO 22$ IF ERROR
378 013422
379 013422 004737 044154 11$: JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
    013426 000404 BR 12$ ;GO TO 12$ IF NO ERROR
    013430 000240 NOP ;RETURN HERE IF ERROR
    013432 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
    013434 000137 014050 JMP 22$ ;GO TO 22$ IF ERROR
380 013440 032737 000001 001360 12$: BIT #DMD,RMMR1I ;IS DIAGNOSTIC MODE SET??
381 013446 001011 BNE 125$ ;YES!!
382 013450 012737 000001 001140 MOV #DMD,$GDDAT ;EXPECTED STATUS
383 013456 013737 001360 001142 MOV RMMR1I,$BDDAT ;RECEIVED STATUS
384 013464 104176 EMT 176
385 013466 000137 014050 JMP 22$ ;SKIP REST OF TEST IF DMD = 0
386 013472 032737 010000 001346 125$: BIT #MOL,RMDSI ;IS MOL = 1??
387 013500 001012 BNE 13$ ;YES!!
388 013502 013737 001346 001142 MOV RMDSI,$BDDAT ;SETUP BAD DATA FOR TYPEOUT
389 013510 042737 167777 001142 BIC #^CMOL,$BDDAT
390 013516 012737 010000 001140 MOV #MOL,$GDDAT ;EXPECTED MOL = 1
391 013524 104177 EMT 177
392 013526 032737 020000 001346 13$: BIT #PIP,RMDSI ;IS PIP = 0 ?
393 013534 001411 BEQ 14$ ;YES!!
394 013536 013737 001346 001142 MOV RMDSI,$BDDAT ;SETUP BAD DATA FOR TYPEOUT
395 013544 042737 157777 001142 BIC #^CPIP,$BDDAT
396 013552 005037 001140 CLR $GDDAT ;EXPECTED PIP STATUS
397 013556 104200 EMT 200
398 013560 032737 004000 001346 14$: BIT #WRL,RMDSI ;IS WRL SET??
399 013566 001012 BNE 15$ ;YES!!
400 013570 013737 001346 001142 MOV RMDSI,$BDDAT ;SETUP BAD DATA FOR TYPEOUT
401 013576 042737 173777 001142 BIC #^CWRL,$BDDAT
402 013604 012737 004000 001140 MOV #WRL,$GDDAT ;EXPECTED GOOD STATUS
403 013612 104201 EMT 201
404 013614 032737 040000 001376 15$: BIT #SKI,RMER2I ;IS SKI SET??
405 013622 001012 BNE 16$ ;YES!!
406 013624 013737 001376 001142 MOV RMER2I,$BDDAT ;BAD DATA FOR TYPEOUT
407 013632 042737 137777 001142 BIC #^CSKI,$BDDAT
408 013640 012737 040000 001140 MOV #SKI,$GDDAT ;EXPECTED SKI ON
409 013646 104202 EMT 202
410 013650 032737 000200 001376 16$: BIT #DVC,RMER2I ;IS DVC SET ??
411 013656 001012 BNE 17$ ;YES!!
412 013660 013737 001376 001142 MOV RMER2I,$BDDAT ;BAD DATA FOR TYPEOUT
413 013666 042737 177577 001142 BIC #^CDVC,$BDDAT
414 013674 012737 000200 001140 MOV #DVC,$GDDAT ;EXPECTED DVC ON
415 013702 104203 EMT 203
416 013704 032737 000100 001346 17$: BIT #VV,RMDSI ;MUR SHOULD HAVE RESET VOLUME VALID
417 013712 001411 BEQ 19$ ;BRANCH IF IT DID
418 013714 013737 001346 001142 MOV RMDSI,$BDDAT ;BAD DATA FOR TYPEOUT
419 013722 042737 177677 001142 BIC #^CVV,$BDDAT
420 013730 005037 001140 CLR $GDDAT ;GOOD DATA FOR TYPEOUT
421 013734 104204 EMT 204
  
```



```

422 013736      19$:
423 013736 112737 000000 001543  MOVB #RMCS1,PUTINX ;SETUP PUT INDEX TABLE
    013744 112737 000200 001544  MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
    013752 012737 000011 001410  MOV #DRVCLR!GO,RMCS10 ;SET RMCS1 OUTPUT BUFFER = DRVCLR!GO
    013760 004737 044424  JSR PC,PUT ;GO WRITE RMCS1 VIA PUT SUBROUTINE
    013764 000404  BR 20$ ;GO TO 20$ IF NO ERROR
    013766 000240  NOP ;RETURN HERE IF ERROR
    013770 104000  EMT ;ERROR DEFINED BY PUT SUBROUTINE
    013772 000137 014050  JMP 22$ ;GO TO 22$ IF ERROR
424 013776      20$:
425 013776 004737 044154  JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
    014002 000404  BR 21$ ;GO TO 21$ IF NO ERROR
    014004 000240  NOP ;RETURN HERE IF ERROR
    014006 104000  EMT ;ERROR # DEFINED BY GET SUBROUTINE
    014010 000137 014050  JMP 22$ ;GO TO 22$ IF ERROR
426 014014 032737 002000 001376 21$: BIT #LBC,RMER2I ;IS LBC SET ??
427 014022 001012  BNE 22$ ;YES!!
428 014024 013737 001376 001142  MOV RMER2I,$BDDAT ;BAD DATA FOR TYPEOUT
429 014032 012737 002000 001140  MOV #LBC,$GDDAT ;GOOD DATA FOR TYPEOUT
430 014040 042737 002000 001142  BIC #LBC,$BDDAT
431 014046 104262  EMT 262
432 014050      22$:
433
434

```

```

*****
;*TEST 12 PACK ACKNOWLEDGE TEST
*****
TST12:

```

```

014050      SCOPE ;SCOPE CALL
014050 000004  NOP ;START OF TEST
014052 000240  MOV #STACK,SP ;INITIALIZE STACK POINTER
014054 012706 001100  MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
014060 013700 001276  MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
014064 013701 001464  MOV #12,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
014070 012737 000012 001226
435
436 014076 004737 053530  JSR PC,CNTCLR ;GO ISSUE CONTROLLER CLEAR
    014102 000404  BR 1$ ;GO TO 1$ IF NO ERROR
    014104 000240  NOP ;RETURN HERE IF ERROR
    014106 104000  EMT ;ERROR NUMBER DEFINED BY SUBROUTINE
    014110 000137 014350  JMP 9$ ;GO TO 9$ IF ERROR
437 014114      1$:
438 014114 112737 000024 001543  MOVB #RMMR1,PUTINX ;SETUP PUT INDEX TABLE
    014122 112737 000200 001544  MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
    014130 012737 000001 001434  MOV #DMD,RMMR10 ;SET RMMR1 OUTPUT BUFFER = DMD
    014136 004737 044424  JSR PC,PUT ;GO WRITE RMMR1 VIA PUT SUBROUTINE
    014142 000404  BR 2$ ;GO TO 2$ IF NO ERROR
    014144 000240  NOP ;RETURN HERE IF ERROR
    014146 104000  EMT ;ERROR DEFINED BY PUT SUBROUTINE
    014150 000137 014350  JMP 9$ ;GO TO 9$ IF ERROR
439 014154      2$:
440 014154 112737 000024 001543  MOVB #RMMR1,PUTINX ;SETUP PUT INDEX TABLE
    014162 112737 000200 001544  MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
    014170 012737 000000 001434  MOV #0,RMMR10 ;SET RMMR1 OUTPUT BUFFER = 0
    014176 004737 044424  JSR PC,PUT ;GO WRITE RMMR1 VIA PUT SUBROUTINE
    014202 000404  BR 3$ ;GO TO 3$ IF NO ERROR
    014204 000240  NOP ;RETURN HERE IF ERROR
    014206 104000  EMT ;ERROR DEFINED BY PUT SUBROUTINE
    014210 000137 014350  JMP 9$ ;GO TO 9$ IF ERROR

```

```

441 014214
442 014214 112737 000000 001543 3$: MOVB #RMCS1,PUTINX ;SETUP PUT INDEX TABLE
    014222 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
    014230 012737 000023 001410 MOV #PAKACK!GO,RMCS10 ;SET RMCS1 OUTPUT BUFFER = PAKACK!GO
    014236 004737 044424 JSR PC,PUT ;GO WRITE RMCS1 VIA PUT SUBROUTINE
    014242 000404 BR 4$ ;GO TO 4$ IF NO ERROR
    014244 000240 NOP ;RETURN HERE IF ERROR
    014246 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
    014250 000137 014350 JMP 9$ ;GO TO 9$ IF ERROR
443 014254 004737 044070 4$: JSR PC,GETSTS
444 014260 004737 044766 JSR PC,TIMOUT ;WAIT FOR COMPLETION
445
446 014264 004737 044154 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
    014270 000403 BR 5$ ;GO TO 5$ IF NO ERROR
    014272 000240 NOP ;RETURN HERE IF ERROR
    014274 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
447 014276 000424 BR 9$ ;SKIP REMAINDER OF TEST
448 014300
449 014300 004737 045152 5$: JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
    014304 000404 BR 6$ ;GO TO 6$ IF NO ERROR
    014306 000240 NOP ;RETURN HERE IF ERROR
    014310 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
    014312 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
450 014314 000415 BR 9$ ;SKIP TO NEXT TEST
451 014316
452 014316 004737 054526 6$: JSR PC,ACKSTS ;GO VERIFY PACK ACKNOWLEDGE
    014322 000404 BR 7$ ;GO TO 7$ IF NO ERROR
    014324 000240 NOP ;RETURN HERE IF ERROR
    014326 104000 EMT ;ERROR # DEFINED BY ACKSTS SUBROUTINE
    014330 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
453 014332 000406 BR 9$ ;SKIP TO NEXT TEST
454
455 014334
456 014334 004737 046004 7$: JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
    014340 000403 BR 9$ ;GO TO 9$ IF NO ERROR
    014342 000240 NOP ;RETURN HERE IF ERROR
    014344 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
    014346 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
457 014350
458
459

```

```

:*****
:*TEST 13 RECALIBRATE TEST
:*****
TST13:

```

```

014350
014350 000004 SCOPE ;SCOPE CALL
014352 000240 NOP ;START OF TEST
014354 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
014360 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
014364 013701 001464 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
014370 012737 000001 001206 MOV #1,$TIMES ;DO 1 ITERATION
014376 012737 000013 001226 MOV #13,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
460
461 014404 004737 043150 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
    014410 050020 .WORD 050020 ;TASK DESCRIPTOR AS FOLLOWS:
    ;CLEAR CONTROLLER & SELECT DEVICE
    ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
    ;VERIFY PACK ACKNOWLEDGE

```

```

014412 000404 BR 5$ ;GO TO 5$ IF NO ERROR
014414 000240 NOP ;RETURN HERE IF ERROR
014416 104000 EMT ;ERROR # DEFINED BY TSTPRP SUBROUTINE
014420 000137 014566 JMP 10$ ;GO TO 10$ IF ERROR
462 014424 5$:
463 014424 112737 000000 001543 MOV# #RMCS1,PUTINX ;SETUP PUT INDEX TABLE
014432 112737 000200 001544 MOV# #200,PUTINX+1 ;SET TERMINATOR BYTE
014440 012737 000007 001410 MOV #RECAL!GO,RMCS10 ;SET RMCS1 OUTPUT BUFFER = RECAL!GO
014446 004737 044424 JSR PC,PUT ;GO WRITE RMCS1 VIA PUT SUBROUTINE
014452 000404 BR 6$ ;GO TO 6$ IF NO ERROR
014454 000240 NOP ;RETURN HERE IF ERROR
014456 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
014460 000137 014566 JMP 10$ ;GO TO 10$ IF ERROR
464 014464 004737 044070 6$: JSR PC,GETSTS ;GO SETUP FOR STATUS FETCH
465 014470 004737 044766 JSR PC,TIMOUT ;WAIT FOR COMPLETION
466
467 014474 004737 044154 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
014500 000404 BR 7$ ;GO TO 7$ IF NO ERROR
014502 000240 NOP ;RETURN HERE IF ERROR
014504 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
014506 000137 014566 JMP 10$ ;GO TO 10$ IF ERROR
468 014512 7$:
469 014512 004737 045152 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
014516 000405 BR 8$ ;GO TO 8$ IF NO ERROR
014520 000240 NOP ;RETURN HERE IF ERROR
014522 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
014524 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
014526 000137 014566 JMP 10$ ;GO TO 10$ IF ERROR
470 014532 8$:
471 014532 004737 055322 JSR PC,RCLSTS ;GO VERIFY RECALIBRATE OPERATION
014536 000405 BR 9$ ;GO TO 9$ IF NO ERROR
014540 000240 NOP ;RETURN HERE IF ERROR
014542 104000 EMT ;ERROR # DEFINED BY RCLSTS SUBROUTINE
014544 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
014546 000137 014566 JMP 10$ ;GO TO 10$ IF ERROR
472 014552 9$:
473 014552 004737 046004 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
014556 000403 BR 10$ ;GO TO 10$ IF NO ERROR
014560 000240 NOP ;RETURN HERE IF ERROR
014562 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
014564 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
474 014566 10$:
475
476

```

```

*****
:*TEST 14 ABORT RECALIBRATE TEST
*****
TST14:

```

```

014566 000004 SCOPE ;SCOPE CALL
014570 000240 NOP ;START OF TEST
014572 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
014576 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
014602 013701 001464 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
014606 012737 000001 001206 MOV #1,$TIMES ;DO 1 ITERATION
014614 012737 000014 001226 MOV #14,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
477
478 014622 004737 043150 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
014626 054130 .WORD 054130 ;TASK DESCRIPTOR AS FOLLOWS:

```

```

                                ;CLEAR CONTROLLER & SELECT DEVICE
                                ;VERIFY CONTROLLER CLEAR OPERATION
                                ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                ;VERIFY PACK ACKNOWLEDGE
                                ;RECALIBRATE IF "SKI" OR "PIP" IS SET
                                ;VERIFY RECALIBRATION
                                ;GO TO 8$ IF NO ERROR
                                ;RETURN HERE IF ERROR
                                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                ;GO TO 15$ IF ERROR
014630 000404 BR 8$
014632 000240 NOP
014634 104000 EMT
014636 000137 015074 JMP 15$
479 014642 8$:
480 014642 112737 000014 001543 MOVB #RMER1,PUTINX ;SETUP PUT INDEX TABLE
014650 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
014656 012737 040000 001424 MOV #UNS,RMER10 ;SET RMER1 OUTPUT BUFFER = UNS
014664 004737 044424 JSR PC,PUT ;GO WRITE RMER1 VIA PUT SUBROUTINE
014670 000404 BR 9$ ;GO TO 9$ IF NO ERROR
014672 000240 NOP ;RETURN HERE IF ERROR
014674 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
014676 000137 015074 JMP 15$ ;GO TO 15$ IF ERROR
481 014702 9$:
482 014702 112737 000000 001543 MOVB #RMCS1,PUTINX ;SETUP PUT INDEX TABLE
014710 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
014716 012737 000007 001410 MOV #RECAL!GO,RMCS10 ;SET RMCS1 OUTPUT BUFFER = RECAL!GO
014724 004737 044424 JSR PC,PUT ;GO WRITE RMCS1 VIA PUT SUBROUTINE
014730 000404 BR 10$ ;GO TO 10$ IF NO ERROR
014732 000240 NOP ;RETURN HERE IF ERROR
014734 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
014736 000137 015074 JMP 15$ ;GO TO 15$ IF ERROR
483 014742 004737 044070 10$: JSR PC,GETSTS ;SETUP FOR STATUS FETCH
484
485 014746 004737 044154 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
014752 000404 BR 11$ ;GO TO 11$ IF NO ERROR
014754 000240 NOP ;RETURN HERE IF ERROR
014756 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
014760 000137 015074 JMP 15$ ;GO TO 15$ IF ERROR
486 014764 032737 020000 001346 11$: BIT #PIP,RMDSI ;DID THE DRIVE RECALIBRATE??
487 014772 001401 BEQ 12$ ;NO!!
488 014774 104214 EMT 214
489 014776 004737 044766 12$: JSR PC,TIMOUT ;WAIT FOR COMPLETION
490
491 015002 004737 044154 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
015006 000404 BR 13$ ;GO TO 13$ IF NO ERROR
015010 000240 NOP ;RETURN HERE IF ERROR
015012 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
015014 000137 015074 JMP 15$ ;GO TO 15$ IF ERROR
492 015020 13$:
493 015020 004737 045152 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
015024 000405 BR 14$ ;GO TO 14$ IF NO ERROR
015026 000240 NOP ;RETURN HERE IF ERROR
015030 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
015032 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
015034 000137 015074 JMP 15$ ;GO TO 15$ IF ERROR
494 015040 14$:
495 015040 004737 061232 JSR PC,STCDRVSTS ;GO CHECK FOR CHANGES IN DRIVE STATUS
015044 000405 BR 141$ ;GO TO 141$ IF NO ERROR
015046 000240 NOP ;RETURN HERE IF ERROR
015050 104000 EMT ;ERROR # DEFINED BY STCDRVSTS SUBROUTINE

```

```
015052 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
015054 000137 015074 JMP 15$ ;GO TO 15$ IF ERROR
496 015060 141$: JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
497 015060 004737 046004 BR 15$ ;GO TO 15$ IF NO ERROR
015064 000403 NOP ;RETURN HERE IF ERROR
015066 000240 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
015070 104000 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
015072 004736
498 015074 15$:
499
500
*****
;*TEST 15 IVC RECALIBRATE TEST
*****
TST15:
015074 SCOPE ;SCOPE CALL
015074 000004 NOP ;START OF TEST
015076 000240 MOV #STACK,SP ;INITIALIZE STACK POINTER
015100 012706 001100 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
015104 013700 001276 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
015110 013701 001464 MOV #1,$TIMES ;DO 1 ITERATION
015114 012737 000001 001206 MOV #15,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
015122 012737 000015 001226
501
502 015130 004737 043150 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
015134 054130 .WORD 054130 ;TASK DESCRIPTOR AS FOLLOWS:
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
;VERIFY RECALIBRATION
015136 000404 BR 8$ ;GO TO 8$ IF NO ERROR
015140 000240 NOP ;RETURN HERE IF ERROR
015142 104000 EMT ;ERROR # DEFINED BY TSTPRP SUBROUTINE
015144 000137 015404 JMP 15$ ;GO TO 15$ IF ERROR
503 015150 8$:
504 015150 112737 000024 001543 MOVB #RMMR1,PUTINX ;SETUP PUT INDEX TABLE
015156 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
015164 012737 000001 001434 MOV #DMD,RMMR10 ;SET RMMR1 OUTPUT BUFFER = DMD
015172 004737 044424 JSR PC,PUT ;GO WRITE RMMR1 VIA PUT SUBROUTINE
015176 000404 BR 9$ ;GO TO 9$ IF NO ERROR
015200 000240 NOP ;RETURN HERE IF ERROR
015202 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
015204 000137 015404 JMP 15$ ;GO TO 15$ IF ERROR
505 015210 9$:
506 015210 112737 000024 001543 MOVB #RMMR1,PUTINX ;SETUP PUT INDEX TABLE
015216 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
015224 012737 000000 001434 MOV #0,RMMR10 ;SET RMMR1 OUTPUT BUFFER = 0
015232 004737 044424 JSR PC,PUT ;GO WRITE RMMR1 VIA PUT SUBROUTINE
015236 000404 BR 10$ ;GO TO 10$ IF NO ERROR
015240 000240 NOP ;RETURN HERE IF ERROR
015242 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
015244 000137 015404 JMP 15$ ;GO TO 15$ IF ERROR
507 015250 10$:
508 015250 112737 000000 001543 MOVB #RMCS1,PUTINX ;SETUP PUT INDEX TABLE
015256 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
015264 012737 000007 001410 MOV #RECAL!GO,RMCS10 ;SET RMCS1 OUTPUT BUFFER = RECAL!GO
015272 004737 044424 JSR PC,PUT ;GO WRITE RMCS1 VIA PUT SUBROUTINE
```

```

015276 000404 BR 11$ :GO TO 11$ IF NO ERROR
015300 000240 NOP :RETURN HERE IF ERROR
015302 104000 EMT :ERROR DEFINED BY PUT SUBROUTINE
509 015304 000137 015404 JMP 15$ :GO TO 15$ IF ERROR
510 015310 004737 044070 11$: JSR PC,GETSTS :SETUP FOR STATUS FETCH
511 015314 004737 044766 JSR PC,TIMOUT :WAIT FOR RECAL TO COMPLETE
512 015320 004737 044154 JSR PC,GET :GO READ REGISTER(S) WITH GET SUBROUTINE
015324 000404 BR 12$ :GO TO 12$ IF NO ERROR
015326 000240 NOP :RETURN HERE IF ERROR
015330 104000 EMT :ERROR # DEFINED BY GET SUBROUTINE
015332 000137 015404 JMP 15$ :GO TO 15$ IF ERROR
513 015336 12$: JSR PC,PRIERR :GO CHECK FOR PRIMARY ERRORS
514 015336 004737 045152 BR 13$ :GO TO 13$ IF NO ERROR
015342 000405 NOP :RETURN HERE IF ERROR
015344 000240 EMT :ERROR # DEFINED BY PRIERR SUBROUTINE
015346 104000 JSR PC,@(SP)+ :GO BACK FOR MORE ERROR CHECKS
015350 004736 JMP 15$ :GO TO 15$ IF ERROR
515 015356 032737 010000 001376 13$: BIT #IVC,RMER21 :IVC SHOULD BE SET
516 015364 001001 BNE 14$ :IT IS - GO CHECK SECONDARY ERRORS
517 015366 104215 EMT 215
518 015370 14$: JSR PC,SECERR :GO CHECK FOR SECONDARY ERRORS
519 015370 004737 046004 BR 15$ :GO TO 15$ IF NO ERROR
015374 000403 NOP :RETURN HERE IF ERROR
015376 000240 EMT :ERROR # DEFINED BY SECERR SUBROUTINE
015400 104000 JSR PC,@(SP)+ :GO BACK FOR MORE ERROR CHECKS
015402 004736
520 015404 15$:
521
522

```

```

*****
:*TEST 16 IAE RECALIBRATE TEST
*****
TST16:

```

```

015404 000004 SCOPE :SCOPE CALL
015406 000240 NOP :START OF TEST
015410 012706 001100 MOV #STACK,SP :INITIALIZE STACK POINTER
015414 013700 001276 MOV $BASE,R0 :R0 = UNIBUS ADDRESS
015420 013701 001464 MOV TSTQUE,R1 : (R1) = DEVICE BEING TESTED
015424 012737 000001 001206 MOV #1,$TIMES :DO 1 ITERATION
015432 012737 000016 001226 MOV #16,$TESTN :SET TEST NUMBER IN APT MAIL BOX
523
524 015440 004737 043150 JSR PC,TSTPRP :PREPARE DEVICE FOR TEST
015444 054130 .WORD 054130 :TASK DESCRIPTOR AS FOLLOWS:
: CLEAR CONTROLLER & SELECT DEVICE
: VERIFY CONTROLLER CLEAR OPERATION
: PACK ACKNOWLEDGE IF VOLUME NOT VALID
: VERIFY PACK ACKNOWLEDGE
: RECALIBRATE IF "SKI" OR "PIP" IS SET
: VERIFY RECALIBRATION
015446 000404 BR 70$ :GO TO 70$ IF NO ERROR
015450 000240 NOP :RETURN HERE IF ERROR
015452 104000 EMT :ERROR # DEFINED BY TSTPRP SUBROUTINE
015454 000137 015646 JMP 120$ :GO TO 120$ IF ERROR
525 015460 70$:
526 015460 012737 177777 001444 MOV #-1,RMDCO :RMDC WILL BE ALL ONES
527 015466 012737 177777 001416 MOV #-1,RMDAO :RMDA WILL BE ALL ONES

```

```

528 015474 012737 000007 001410      MOV      #RECAL!GO, RMCS10
529 015502 012702 001543      MOV      #PUTINX, R2          ;R2 POINTS TO PUT INDEX TABLE
530 015506 112722 000034      MOV      #RMDC, (R2)+        ;SETUP PUT INDEX TABLE
531 015512 112722 000006      MOV      #RMDA, (R2)+
532 015516 112722 000000      MOV      #RMCS1, (R2)+
533 015522 112722 000200      MOV      #200, (R2)+
534 015526 004737 044424      JSR      PC, PUT              ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
    015532 000404      BR      80$                  ;GO TO 80$ IF NO ERROR
    015534 000240      NOP                               ;RETURN HERE IF ERROR
    015536 104000      EMT                               ;ERROR # DEFINED BY PUT SUBROUTINE
    015540 000137 015646      JMP      120$                 ;GO TO 120$ IF ERROR
535 015544 004737 044070      JSR      PC, GETSTS           ;SETUP FOR STATUS FETCH
536 015550 004737 044766      JSR      PC, TIMEOUT         ;WAIT FOR GO TO RESET
537
538 015554 004737 044154      JSR      PC, GET              ;GO READ REGISTER(S) WITH GET SUBROUTINE
    015560 000404      BR      90$                  ;GO TO 90$ IF NO ERROR
    015562 000240      NOP                               ;RETURN HERE IF ERROR
    015564 104000      EMT                               ;ERROR # DEFINED BY GET SUBROUTINE
    015566 000137 015646      JMP      120$                 ;GO TO 120$ IF ERROR
539 015572
540 015572 004737 045152      JSR      PC, PRIERR          ;GO CHECK FOR PRIMARY ERRORS
    015576 000405      BR      100$                 ;GO TO 100$ IF NO ERROR
    015600 000240      NOP                               ;RETURN HERE IF ERROR
    015602 104000      EMT                               ;ERROR # DEFINED BY PRIERR SUBROUTINE
    015604 004736      JSR      PC, @ (SP)+         ;GO BACK FOR MORE ERROR CHECKS
    015606 000137 015646      JMP      120$                 ;GO TO 120$ IF ERROR
541 015612
542 015612 004737 055322      JSR      PC, RCLSTS          ;GO VERIFY RECALIBRATE OPERATION
    015616 000405      BR      110$                 ;GO TO 110$ IF NO ERROR
    015620 000240      NOP                               ;RETURN HERE IF ERROR
    015622 104000      EMT                               ;ERROR # DEFINED BY RCLSTS SUBROUTINE
    015624 004736      JSR      PC, @ (SP)+         ;GO BACK FOR MORE ERROR CHECKS
    015626 000137 015646      JMP      120$                 ;GO TO 120$ IF ERROR
543 015632
544 015632 004737 046004      JSR      PC, SECERR          ;GO CHECK FOR SECONDARY ERRORS
    015636 000403      BR      120$                 ;GO TO 120$ IF NO ERROR
    015640 000240      NOP                               ;RETURN HERE IF ERROR
    015642 104000      EMT                               ;ERROR # DEFINED BY SECERR SUBROUTINE
    015644 004736      JSR      PC, @ (SP)+         ;GO BACK FOR MORE ERROR CHECKS
545 015646
546
547

```

```

*****
;*TEST 17      RECALIBRATE AT OFFSET
*****
TST17:

```

```

015646
015646 000004
015650 000240
015652 012706 001100
015656 013700 001276
015662 013701 001464
015666 012737 000001 001206
015674 012737 000017 001226
548
549 015702 004737 043150      JSR      PC, TSTPRP          ;PREPARE DEVICE FOR TEST
    015706 050020      .WORD  050020              ;TASK DESCRIPTOR AS FOLLOWS:
                                                ;CLEAR CONTROLLER & SELECT DEVICE
                                                ;PACK ACKNOWLEDGE IF VOLUME NOT VALID

```

```

015710 000404          BR      5$      ;VERIFY PACK ACKNOWLEDGE
015712 000240          NOP          ;GO TO 5$ IF NO ERROR
015714 104000          EMT          ;RETURN HERE IF ERROR
015716 000137 016124  JMP      80$      ;ERROR # DEFINED BY TSTPRP SUBROUTINE
550 015722          JMP      80$      ;GO TO 80$ IF ERROR
551 015722 112737 000000 001543 5$:      MOVB   #RMCS1,PUTINX ;SETUP PUT INDEX TABLE
015730 112737 000200 001544      MOVB   #200,PUTINX+1 ;SET TERMINATOR BYTE
015736 012737 000015 001410      MOV    #OFFSET!GO,RMCS10 ;SET RMCS1 OUTPUT BUFFER = OFFSET!GO
015744 004737 044424      JSR    PC,PUT        ;GO WRITE RMCS1 VIA PUT SUBROUTINE
015750 000404          BR      10$      ;GO TO 10$ IF NO ERROR
015752 000240          NOP          ;RETURN HERE IF ERROR
015754 104000          EMT          ;ERROR DEFINED BY PUT SUBROUTINE
015756 000137 016124  JMP      80$      ;GO TO 80$ IF ERROR
552 015762          JMP      80$      ;GO TO 80$ IF ERROR
553 015762 112737 000000 001543 10$:     MOVB   #RMCS1,PUTINX ;SETUP PUT INDEX TABLE
015770 112737 000200 001544      MOVB   #200,PUTINX+1 ;SET TERMINATOR BYTE
015776 012737 000007 001410      MOV    #RECAL!GO,RMCS10 ;SET RMCS1 OUTPUT BUFFER = RECAL!GO
016004 004737 044424      JSR    PC,PUT        ;GO WRITE RMCS1 VIA PUT SUBROUTINE
016010 000404          BR      20$      ;GO TO 20$ IF NO ERROR
016012 000240          NOP          ;RETURN HERE IF ERROR
016014 104000          EMT          ;ERROR DEFINED BY PUT SUBROUTINE
016016 000137 016124  JMP      80$      ;GO TO 80$ IF ERROR
554 016022          JMP      80$      ;GO TO 80$ IF ERROR
555
556          ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
016022 004737 044070      JSR    PC,GETSTS    ;GO TO GETSTS SUBROUTINE
557
558          ;WAIT FOR COMMAND TO COMPLETE
016026 004737 044766      JSR    PC,TIMOUT    ;GO TO TIMEOUT SUBROUTINE
559
560 016032 004737 044154      JSR    PC,GET        ;GO READ REGISTER(S) WITH GET SUBROUTINE
016036 000404          BR      50$      ;GO TO 50$ IF NO ERROR
016040 000240          NOP          ;RETURN HERE IF ERROR
016042 104000          EMT          ;ERROR # DEFINED BY GET SUBROUTINE
016044 000137 016124  JMP      80$      ;GO TO 80$ IF ERROR
561 016050          JMP      80$      ;GO TO 80$ IF ERROR
562 016050 004737 045152 50$:     JSR    PC,PRIERR    ;GO CHECK FOR PRIMARY ERRORS
016054 000405          BR      60$      ;GO TO 60$ IF NO ERROR
016056 000240          NOP          ;RETURN HERE IF ERROR
016060 104000          EMT          ;ERROR # DEFINED BY PRIERR SUBROUTINE
016062 004736          JSR    PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
016064 000137 016124  JMP      80$      ;GO TO 80$ IF ERROR
563 016070          JMP      80$      ;GO TO 80$ IF ERROR
564 016070 004737 055322 60$:     JSR    PC,RCLSTS    ;GO VERIFY RECALIBRATE OPERATION
016074 000405          BR      70$      ;GO TO 70$ IF NO ERROR
016076 000240          NOP          ;RETURN HERE IF ERROR
016100 104000          EMT          ;ERROR # DEFINED BY RCLSTS SUBROUTINE
016102 004736          JSR    PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
016104 000137 016124  JMP      80$      ;GO TO 80$ IF ERROR
565 016110          JMP      80$      ;GO TO 80$ IF ERROR
566 016110 004737 046004 70$:     JSR    PC,SECERR    ;GO CHECK FOR SECONDARY ERRORS
016114 000403          BR      80$      ;GO TO 80$ IF NO ERROR
016116 000240          NOP          ;RETURN HERE IF ERROR
016120 104000          EMT          ;ERROR # DEFINED BY SECERR SUBROUTINE
016122 004736          JSR    PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
567 016124          JMP      80$      ;GO TO 80$ IF ERROR

```


568
 569

```

:*****
:TEST 20      DRIVE CLEAR TEST
:*****
TST20:

```

016124	000004			SCOPE		:SCOPE CALL
016124	000240			NOP		:START OF TEST
016126	000240			MOV	#STACK,SP	:INITIALIZE STACK POINTER
016130	012706	001100		MOV	\$BASE,R0	:R0 = UNIBUS ADDRESS
016134	013700	001276		MOV	TSTQUE,R1	:(R1) = DEVICE BEING TESTED
016140	013701	001464		MOV	#20,\$TESTN	::SET TEST NUMBER IN APT MAIL BOX
016144	012737	000020	001226			
570						
571	016152	004737	043150	JSR	PC,TSTPRP	:PREPARE DEVICE FOR TEST
	016156	054130		.WORD	054130	:TASK DESCRIPTOR AS FOLLOWS:
						:CLEAR CONTROLLER & SELECT DEVICE
						:VERIFY CONTROLLER CLEAR OPERATION
						:PACK ACKNOWLEDGE IF VOLUME NOT VALID
						:VERIFY PACK ACKNOWLEDGE
						:RECALIBRATE IF "SKI" OR "PIP" IS SET
						:VERIFY RECALIBRATION
						:GO TO 2\$ IF NO ERROR
						:RETURN HERE IF ERROR
						:ERROR # DEFINED BY TSTPRP SUBROUTINE
						:GO TO 21\$ IF ERROR
016160	000404			BR	2\$	
016162	000240			NOP		
016164	104000			EMT		
016166	000137	016374		JMP	21\$	
572	016172					
573	016172	012737	000000	MOV	#0,RMDAO	
574	016200	012737	000011	MOV	#DRVCLR!GO,RMCS10	
575	016206	012737	177777	MOV	#-1,RMER10	
576	016214	012737	177777	MOV	#-1,RMER20	
577	016222	042737	004000	BIC	#LSC,RMER20	:DELETE FOR PASS 2 ETCH
578	016230	012702	001543	MOV	#PUTINX,R2	:R2 POINTS TO INDEX TABLE
579	016234	112722	000006	MOVB	#RMDA,(R2)+	:RMDA CLEARS LBT
580	016240	112722	000042	MOVB	#RMER2,(R2)+	
581	016244	112722	000014	MOVB	#RMER1,(R2)+	
582	016250	112722	000000	MOVB	#RMCS1,(R2)+	
583	016254	112722	000200	MOVB	#200,(R2)+	
584	016260	004737	044424	JSR	PC,PUT	:GO WRITE REGISTER(S) WITH PUT SUBROUTINE
	016264	000403		BR	3\$:GO TO 3\$ IF NO ERROR
	016266	000240		NOP		:RETURN HERE IF ERROR
	016270	104000		EMT		:ERROR # DEFINED BY PUT SUBROUTINE
585	016272	000440		BR	21\$:ESCAPE IF ERROR
586	016274	004737	044070	JSR	PC,GETSTS	:SETUP FOR STATUS FETCH
587	016300	004737	044766	JSR	PC,TIMOUT	:WAIT FOR DRIVE CLEAR TO COMPLETE
588						
589	016304	004737	044154	JSR	PC,GET	:GO READ REGISTER(S) WITH GET SUBROUTINE
	016310	000403		BR	4\$:GO TO 4\$ IF NO ERROR
	016312	000240		NOP		:RETURN HERE IF ERROR
	016314	104000		EMT		:ERROR # DEFINED BY GET SUBROUTINE
590	016316	000426		BR	21\$:SKIP REMAINDER OF TEST
591	016320					
592	016320	004737	045152	JSR	PC,PRIERR	:GO CHECK FOR PRIMARY ERRORS
	016324	000405		BR	5\$:GO TO 5\$ IF NO ERROR
	016326	000240		NOP		:RETURN HERE IF ERROR
	016330	104000		EMT		:ERROR # DEFINED BY PRIERR SUBROUTINE
	016332	004736		JSR	PC,@(SP)+	:GO BACK FOR MORE ERROR CHECKS
	016334	000137	016374	JMP	21\$:GO TO 21\$ IF ERROR
593	016340					

```
594 016340 004737 057064 JSR PC,DRVSTS ;GO VERIFY DRIVE CLEAR
    016344 000405 BR 6$ ;GO TO 6$ IF NO ERROR
    016346 000240 NOP ;RETURN HERE IF ERROR
    016350 104000 EMT ;ERROR # DEFINED BY DRVSTS SUBROUTINE
    016352 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
    016354 000137 016374 JMP 21$ ;GO TO 21$ IF ERROR
595 016360 6$:
596 016360 004737 046004 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
    016364 000403 BR 21$ ;GO TO 21$ IF NO ERROR
    016366 000240 NOP ;RETURN HERE IF ERROR
    016370 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
    016372 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
597 016374 21$:
598
599
```

```
::*****
:*TEST 21 NOP TEST
:*****
TST21:
```

```
016374 000004 SCOPE ;SCOPE CALL
016374 000004 NOP ;START OF TEST
016376 000240 MOV #STACK,SP ;INITIALIZE STACK POINTER
016400 012706 001100 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
016404 013700 001276 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
016410 013701 001464 MOV #21,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
016414 012737 000021 001226
600
601 016422 004737 043150 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
    016426 054130 .WORD 054130 ;TASK DESCRIPTOR AS FOLLOWS:
    ;CLEAR CONTROLLER & SELECT DEVICE
    ;VERIFY CONTROLLER CLEAR OPERATION
    ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
    ;VERIFY PACK ACKNOWLEDGE
    ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
    ;VERIFY RECALIBRATION
    016430 000404 BR 70$ ;GO TO 70$ IF NO ERROR
    016432 000240 NOP ;RETURN HERE IF ERROR
    016434 104000 EMT ;ERROR # DEFINED BY TSTPRP SUBROUTINE
    016436 000137 016624 JMP 130$ ;GO TO 130$ IF ERROR
602 016442 70$:
603 016442 112737 000000 001543 MOVB #RMCS1,PUTINX ;SETUP PUT INDEX TABLE
    016450 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
    016456 012737 000001 001410 MOV #NOP!GO,RMCS10 ;SET RMCS1 OUTPUT BUFFER = NOP!GO
    016464 004737 044424 JSR PC,PUT ;GO WRITE RMCS1 VIA PUT SUBROUTINE
    016470 000404 BR 80$ ;GO TO 80$ IF NO ERROR
    016472 000240 NOP ;RETURN HERE IF ERROR
    016474 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
    016476 000137 016624 JMP 130$ ;GO TO 130$ IF ERROR
604 016502 004737 044070 80$: JSR PC,GETSTS ;SETUP FOR STATUS FETCH
605
606 016506 004737 044154 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
    016512 000404 BR 90$ ;GO TO 90$ IF NO ERROR
    016514 000240 NOP ;RETURN HERE IF ERROR
    016516 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
    016520 000137 016624 JMP 130$ ;GO TO 130$ IF ERROR
607 016524 90$:
608 016524 004737 045152 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
    016530 000405 BR 100$ ;GO TO 100$ IF NO ERROR
    016532 000240 NOP ;RETURN HERE IF ERROR
```

```

016534 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
016536 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
016540 000137 016624 JMP 130$ ;GO TO 130$ IF ERROR
609 016544 100$:
610 016544 004737 051644 JSR PC,CMPERRSTS ;CHECK ANY ERRORS NOT MASKED
016550 115760 .WORD ND1MSK ;MASK FOR RMER1
016552 000010 .WORD DPE ;MASK FOR RMER2
016554 000405 BR 110$ ;GO TO 110$ IF NO ERROR
016556 000240 NOP ;RETURN HERE IF ERROR
016560 104000 EMT ;ERROR # DEFINED BY CMPERRSTS SUBROUTINE
016562 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
016564 000137 016624 JMP 130$ ;GO TO 130$ IF ERROR
611 016570 110$:
612 016570 004737 061232 JSR PC,STCDRVSTS ;GO CHECK FOR CHANGES IN DRIVE STATUS
016574 000405 BR 120$ ;GO TO 120$ IF NO ERROR
016576 000240 NOP ;RETURN HERE IF ERROR
016600 104000 EMT ;ERROR # DEFINED BY STCDRVSTS SUBROUTINE
016602 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
016604 000137 016624 JMP 130$ ;GO TO 130$ IF ERROR
613 016610 120$:
614 016610 004737 046004 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
016614 000403 BR 130$ ;GO TO 130$ IF NO ERROR
016616 000240 NOP ;RETURN HERE IF ERROR
016620 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
016622 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
615 016624 130$:
616
617
;*****
;*TEST 22 OFFSET TEST
;*****
TST22:
016624 000004 SCOPE ;SCOPE CALL
016626 000240 NOP ;START OF TEST
016630 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
016634 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
016640 013701 001464 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
016644 012737 000022 001226 MOV #22,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
618
619 016652 004737 043150 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
016656 054130 .WORD 054130 ;TASK DESCRIPTOR AS FOLLOWS:
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
;VERIFY RECALIBRATION
016660 000404 BR 40$ ;GO TO 40$ IF NO ERROR
016662 000240 NOP ;RETURN HERE IF ERROR
016664 104000 EMT ;ERROR # DEFINED BY TSTPRP SUBROUTINE
016666 000137 017150 JMP 100$ ;GO TO 100$ IF ERROR
620 016672 40$:
621 016672 112737 000000 001543 MOVB #RMCS1,PUTINX ;SETUP PUT INDEX TABLE
016700 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
016706 012737 000015 001410 MOV #OFFSET!GO,RMCS10 ;SET RMCS1 OUTPUT BUFFER = OFFSET!GO
016714 004737 044424 JSR PC,PUT ;GO WRITE RMCS1 VIA PUT SUBROUTINE
016720 000404 BR 50$ ;GO TO 50$ IF NO ERROR
016722 000240 NOP ;RETURN HERE IF ERROR

```

```

T22      016724 104000
        016726 000137 017150
622 016732 004737 044070      50$:
623 016736 004737 044766      JSR      PC,GETSTS      ;ERROR DEFINED BY PIT SUBROUTINE
        JSR      PC,TIMOUT ;GO TO 100$ IF ERROR
624
625 016742 004737 044154      JSR      PC,GET        ;SETUP FOR STATUS FETCH
        016746 000404      BR        60$          ;WAIT FOR GO TO RESET
        016750 000240      NOP
        016752 104000      EMT
        016754 000137 017150  JMP      100$         ;GO READ REGISTER(S) WITH GET SUBROUTINE
626 016760      60$:
627 016760 004737 045152      JSR      PC,PRIERR     ;GO CHECK FOR PRIMARY ERRORS
        016764 000405      BR        70$         ;GO TO 70$ IF NO ERROR
        016766 000240      NOP                 ;RETURN HERE IF ERROR
        016770 104000      EMT                 ;ERROR # DEFINED BY PRIERR SUBROUTINE
        016772 004736      JSR      PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
        016774 000137 017150  JMP      100$         ;GO TO 100$ IF ERROR
628 017000 032737 000001 001346 70$:
629 017006 001012      BIT      #OM,RMDSI    ;OFFSET MODE ON??
        017010 013737 001346 001142  BNE
        017016 042737 177776 001142  MOV      RMDSI,$BDDAT ;YES!!
        017024 012737 000001 001140  BIC      #^COM,$BDDAT ;BAD DATA FOR TYPEOUT
        017032 104156      MOV      #OM,$GDDAT  ;GOOD DATA FOR TYPEOUT
        017034 032737 100000 001346 80$:
        017042 001012      EMT      156
        017044 013737 001346 001142  BIT      #ATA,RMDSI   ;WAS ATTENTION SET ??
        017052 042737 077777 001142  BNE      90$          ;YES!!
        017060 012737 100000 001140  MOV      RMDSI,$BDDAT ;BAD DATA FOR TYPEOUT
        017066 104172      BIC      #^CATA,$BDDAT
        017070      MOV      #ATA,$GDDAT ;GOOD DATA FOR TYPEOUT
        017070      EMT      172
641 017070 004737 051644      90$:
        017074 115760      JSR      PC,CMPERRSTS ;CHECK ANY ERRORS NOT MASKED
        017076 000010      .WORD   NDIMSK       ;MASK FOR RMER1
        017100 000405      .WORD   DPE          ;MASK FOR RMER2
        017102 000240      BR        91$        ;GO TO 91$ IF NO ERROR
        017104 104000      NOP                 ;RETURN HERE IF ERROR
        017106 004736      EMT                 ;ERROR # DEFINED BY CMPERRSTS SUBROUTINE
        017110 000137 017150  JSR      PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
        017114      JMP      100$         ;GO TO 100$ IF ERROR
642 017114      91$:
643 017114 004737 061232      JSR      PC,STCDRVSTS ;GO CHECK FOR CHANGES IN DRIVE STATUS
        017120 000405      BR        92$        ;GO TO 92$ IF NO ERROR
        017122 000240      NOP                 ;RETURN HERE IF ERROR
        017124 104000      EMT                 ;ERROR # DEFINED BY STCDRVSTS SUBROUTINE
        017126 004736      JSR      PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
        017130 000137 017150  JMP      100$         ;GO TO 100$ IF ERROR
644 017134      92$:
645 017134 004737 046004      JSR      PC,SECERR    ;GO CHECK FOR SECONDARY ERRORS
        017140 000403      BR        100$       ;GO TO 100$ IF NO ERROR
        017142 000240      NOP                 ;RETURN HERE IF ERROR
        017144 104000      EMT                 ;ERROR # DEFINED BY SECERR SUBROUTINE
        017146 004736      JSR      PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
646 017150      100$:
647
648
        ;*****
        ;*TEST 23      GO/ATA TEST
        ;*****
TST23:
017150

```

	017150	000004			SCOPE		:SCOPE CALL
	017152	000240			NOP		:START OF TEST
	017154	012706	001100		MOV	#STACK,SP	:INITIALIZE STACK POINTER
	017160	013700	001276		MOV	\$BASE,R0	:R0 = UNIBUS ADDRESS
	017164	013701	001464		MOV	TSTQUE,R1	:(R1) = DEVICE BEING TESTED
	017170	012737	000023	001226	MOV	#23,\$TESTN	::SET TEST NUMBER IN APT MAIL BOX
649							
650	017176	004737	043150		JSR	PC,TSTPRP	:PREPARE DEVICE FOR TEST
	017202	054130			.WORD	054130	:TASK DESCRIPTOR AS FOLLOWS:
							:CLEAR CONTROLLER & SELECT DEVICE
							:VERIFY CONTROLLER CLEAR OPERATION
							:PACK ACKNOWLEDGE IF VOLUME NOT VALID
							:VERIFY PACK ACKNOWLEDGE
							:RECALIBRATE IF "SKI" OR "PIP" IS SET
							:VERIFY RECALIBRATION
							:GO TO 70\$ IF NO ERROR
							:RETURN HERE IF ERROR
							:ERROR # DEFINED BY TSTPRP SUBROUTINE
							:GO TO 180\$ IF ERROR
	017204	000404			BR	70\$	
	017206	000240			NOP		
	017210	104000			EMT		
	017212	000137	017570		JMP	180\$	
651	017216			70\$:			
652	017216	112737	000000	001543	MOVB	#RMCS1,PUTINX	:SETUP PUT INDEX TABLE
	017224	112737	000200	001544	MOVB	#200,PUTINX+1	:SET TERMINATOR BYTE
	017232	012737	000015	001410	MOV	#OFFSET!GO,RMCS10	:SET RMCS1 OUTPUT BUFFER = OFFSET!GO
	017240	004737	044424		JSR	PC,PUT	:GO WRITE RMCS1 VIA PUT SUBROUTINE
	017244	000404			BR	80\$:GO TO 80\$ IF NO ERROR
	017246	000240			NOP		:RETURN HERE IF ERROR
	017250	104000			EMT		:ERROR DEFINED BY PUT SUBROUTINE
	017252	000137	017570		JMP	180\$:GO TO 180\$ IF ERROR
653	017256	004737	044070	80\$:	JSR	PC,GETSTS	:SETUP TO READ ALL REGISTERS
654							
655	017262	004737	044154		JSR	PC,GET	:GO READ REGISTER(S) WITH GET SUBROUTINE
	017266	000404			BR	90\$:GO TO 90\$ IF NO ERROR
	017270	000240			NOP		:RETURN HERE IF ERROR
	017272	104000			EMT		:ERROR # DEFINED BY GET SUBROUTINE
	017274	000137	017570		JMP	180\$:GO TO 180\$ IF ERROR
656	017300	032737	100000	001346	90\$:	BIT	#ATA,RMDSI
657	017306	001012			BNE	100\$:IS ATTENTION SET??
658	017310	012737	100000	001140	MOV	#ATA,\$GDDAT	:GOOD DATA FOR TYPEOUT
659	017316	013737	001346	001142	MOV	RMDSI,\$BDDAT	:BAD DATA FOR TYPEOUT
660	017324	042737	077777	001142	BIC	#^CATA,\$BDDAT	
661	017332	104172			EMT	172	
662	017334			100\$:			
663	017334	004737	051644		JSR	PC,CMPERRSTS	:CHECK ANY ERRORS NOT MASKED
	017340	000000			.WORD	0	:MASK FOR RMER1
	017342	000000			.WORD	0	:MASK FOR RMER2
	017344	000405			BR	110\$:GO TO 110\$ IF NO ERROR
	017346	000240			NOP		:RETURN HERE IF ERROR
	017350	104000			EMT		:ERROR # DEFINED BY CMPERRSTS SUBROUTINE
	017352	004736			JSR	PC,@(SP)+	:GO BACK FOR MORE ERROR CHECKS
	017354	000137	017570		JMP	180\$:GO TO 180\$ IF ERROR
664	017360			110\$:			
665	017360	112737	000000	001543	MOVB	#RMCS1,PUTINX	:SETUP PUT INDEX TABLE
	017366	112737	000200	001544	MOVB	#200,PUTINX+1	:SET TERMINATOR BYTE
	017374	012737	000001	001410	MOV	#NOP!GO,RMCS10	:SET RMCS1 OUTPUT BUFFER = NOP!GO
	017402	004737	044424		JSR	PC,PUT	:GO WRITE RMCS1 VIA PUT SUBROUTINE
	017406	000404			BR	120\$:GO TO 120\$ IF NO ERROR
	017410	000240			NOP		:RETURN HERE IF ERROR

```

017412 104000          EMT          ;ERROR DEFINED BY PUT SUBROUTINE
017414 000137 017570   JMP          180$          ;GO TO 180$ IF ERROR
666 017420          120$:
667 017420 004737 044154   JSR          PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
017424 000404          BR          130$          ;GO TO 130$ IF NO ERROR
017426 000240          NOP          ;RETURN HERE IF ERROR
017430 104000          EMT          ;ERROR # DEFINED BY GET SUBROUTINE
017432 000137 017570   JMP          180$          ;GO TO 180$ IF ERROR
668 017436          130$:
669 017436 004737 045152   JSR          PC,PRIERR       ;GO CHECK FOR PRIMARY ERRORS
017442 000405          BR          140$          ;GO TO 140$ IF NO ERROR
017444 000240          NOP          ;RETURN HERE IF ERROR
017446 104000          EMT          ;ERROR # DEFINED BY PRIERR SUBROUTINE
017450 004736          JSR          PC,@(SP)+       ;GO BACK FOR MORE ERROR CHECKS
017452 000137 017570   JMP          180$          ;GO TO 180$ IF ERROR
670 017456 032737 100000 001346 140$:
671 017464 001411          BIT          #ATA,RMDSI     ;IS ATTENTION RESET??
672 017466 013737 001346 001142   BEQ          150$          ;YES
673 017474 042737 077777 001142   MOV          RMDSI,$BDDAT    ;BAD DATA FOR TYPEOUT
674 017502 005037 001140          BIC          #^CATA,$BDDAT
675 017506 104246          CLR          $GDDAT         ;GOOD DATA FOR TYPEOUT
676 017510          150$:
677 017510 004737 051644   JSR          PC,CMPEERRSTS   ;CHECK ANY ERRORS NOT MASKED
017514 115760          .WORD       NDIMSK        ;MASK FOR RMER1
017516 000010          .WORD       DPE           ;MASK FOR RMER2
017520 000405          BR          160$          ;GO TO 160$ IF NO ERROR
017522 000240          NOP          ;RETURN HERE IF ERROR
017524 104000          EMT          ;ERROR # DEFINED BY CMPEERRSTS SUBROUTINE
017526 004736          JSR          PC,@(SP)+       ;GO BACK FOR MORE ERROR CHECKS
017530 000137 017570   JMP          180$          ;GO TO 180$ IF ERROR
678 017534          160$:
679 017534 004737 061232   JSR          PC,STCDRVSTS    ;GO CHECK FOR CHANGES IN DRIVE STATUS
017540 000405          BR          170$          ;GO TO 170$ IF NO ERROR
017542 000240          NOP          ;RETURN HERE IF ERROR
017544 104000          EMT          ;ERROR # DEFINED BY STCDRVSTS SUBROUTINE
017546 004736          JSR          PC,@(SP)+       ;GO BACK FOR MORE ERROR CHECKS
017550 000137 017570   JMP          180$          ;GO TO 180$ IF ERROR
680 017554          170$:
681 017554 004737 046004   JSR          PC,SECERR       ;GO CHECK FOR SECONDARY ERRORS
017560 000403          BR          180$          ;GO TO 180$ IF NO ERROR
017562 000240          NOP          ;RETURN HERE IF ERROR
017564 104000          EMT          ;ERROR # DEFINED BY SECERR SUBROUTINE
017566 004736          JSR          PC,@(SP)+       ;GO BACK FOR MORE ERROR CHECKS
682 017570          180$:
683
684

```

```

*****
:*TEST 24      WRITE ATA TEST
*****

```

```

TST24:
017570          SCOPE          ;SCOPE CALL
017570 000004          NOP          ;START OF TEST
017572 000240          MOV          #STACK,SP    ;INITIALIZE STACK POINTER
017574 012706 001100          MOV          $BASE,R0      ;R0 = UNIBUS ADDRESS
017600 013700 001276          MOV          TSTQUE,R1     ;(R1) = DEVICE BEING TESTED
017604 013701 001464          MOV          #24,$TESTN    ;SET TEST NUMBER IN APT MAIL BOX
017610 012737 000024 001226
685
686 017616 004737 043150   JSR          PC,TSTPRP      ;PREPARE DEVICE FOR TEST

```

```

017622 054130 .WORD 054130 ;TASK DESCRIPTOR AS FOLLOWS:
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF "SKI" OR "PIP" IS SET
;VERIFY RECALIBRATION
;GO TO 70$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 150$ IF ERROR

687 017624 000404 BR 70$
688 017626 000240 NOP
017630 104000 EMT
017632 000137 020134 JMP 150$
687 017636 112737 000000 001543 70$: MOVB #RMCS1,PUTINX ;SETUP PUT INDEX TABLE
688 017644 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
017652 012737 000015 001410 MOV #OFFSET!GO,RMCS10 ;SET RMCS1 OUTPUT BUFFER = OFFSET!GO
017660 004737 044424 JSR PC,PUT ;GO WRITE RMCS1 VIA PUT SUBROUTINE
017664 000404 BR 80$ ;GO TO 80$ IF NO ERROR
017666 000240 NOP ;RETURN HERE IF ERROR
017670 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
017672 000137 020134 JMP 150$ ;GO TO 150$ IF ERROR
689 017676 004737 044070 80$: JSR PC,GETSTS ;SETUP TO READ ALL REGISTERS
690
691 017702 004737 044154 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
017706 000404 BR 90$ ;GO TO 90$ IF NO ERROR
017710 000240 NOP ;RETURN HERE IF ERROR
017712 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
017714 000137 020134 JMP 150$ ;GO TO 150$ IF ERROR
692 017720 032737 100000 001346 90$: BIT #ATA,RMDSI ;IS ATTENTION SET??
693 017726 001012 BNE 100$ ;YES!!
694 017730 013737 001346 001142 MOV RMDSI,$BDDAT ;BAD DATA FOR TYPEOUT
695 017736 042737 077777 001142 BIC #^CATA,$BDDAT
696 017744 012737 100000 001140 MOV #ATA,$GDDAT ;GOOD DATA FOR TYPEOUT
697 017752 104172 EMT 172
698 017754 116102 000001 100$: MOVB 1(R1),R2 ;R2 = ATTENTION BIT
699 017760 042702 177400 BIC #^CATNMSK,R2 ;CLEAR SIGN EXTENSION
700 017764 000240 NOP
701 017766 010237 001426 MOV R2,RMASO ;PUT RMAS BIT IN OUTPUT BUF
702 017772 112737 000016 001543 MOVB #RMAS,PUTINX ;WRITE REGISTER INDEX IN TABLE
703 020000 112737 000200 001544 MOVB #200,PUTINX+1 ;WRITE TERMINATOR
704 020006 004737 044424 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
020012 000404 BR 110$ ;GO TO 110$ IF NO ERROR
020014 000240 NOP ;RETURN HERE IF ERROR
020016 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
020020 000137 020134 JMP 150$ ;GO TO 150$ IF ERROR
705 020024 110$:
706 020024 004737 044154 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
020030 000404 BR 120$ ;GO TO 120$ IF NO ERROR
020032 000240 NOP ;RETURN HERE IF ERROR
020034 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
020036 000137 020134 JMP 150$ ;GO TO 150$ IF ERROR
707 020042 120$:
708 020042 004737 045152 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
020046 000405 BR 130$ ;GO TO 130$ IF NO ERROR
020050 000240 NOP ;RETURN HERE IF ERROR
020052 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
020054 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
020056 000137 020134 JMP 150$ ;GO TO 150$ IF ERROR

```

```

709 020062 032737 100000 001346 130$: BIT #ATA,RMDSI ;IS ATTENTION RESET??
710 020070 001411 BEQ 140$ ;YES!!
711 020072 013737 001346 001142 MOV RMDSI,$BDDAT ;BAD DATA FOR TYPEOUT
712 020100 042737 077777 001142 BIC #^CATA,$BDDAT
713 020106 005037 001140 CLR $GDDAT ;GOOD DATA FOR TYPEOUT
714 020112 104247 EMT 247
715 020114 140$:
716 020114 004737 051644 JSR PC,CMPERRSTS ;CHECK ANY ERRORS NOT MASKED
020120 000000 .WORD 0 ;MASK FOR RMER1
020122 000000 .WORD 0 ;MASK FOR RMER2
020124 000403 BR 150$ ;GO TO 150$ IF NO ERROR
020126 000240 NOP ;RETURN HERE IF ERROR
020130 104000 EMT ;ERROR # DEFINED BY CMPERRSTS SUBROUTINE
020132 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
717 020134 150$:
718
719
;*****
;*TEST 25 ERROR/ATA TEST
;*****
TST25:
020134 000004 SCOPE ;SCOPE CALL
020136 000240 NOP ;START OF TEST
020140 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
020144 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
020150 013701 001464 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
020154 012737 000025 001226 MOV #25,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
720
721 020162 004737 043150 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
020166 054130 .WORD 054130 ;TASK DESCRIPTOR AS FOLLOWS:
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF "SKI" OR "PIP" IS SET
;VERIFY RECALIBRATION
020170 000404 BR 70$ ;GO TO 70$ IF NO ERROR
020172 000240 NOP ;RETURN HERE IF ERROR
020174 104000 EMT ;ERROR # DEFINED BY TSTPRP SUBROUTINE
020176 000137 020526 JMP 160$ ;GO TO 160$ IF ERROR
722 020202 70$:
723 020202 112737 000000 001543 MOVB #RMCS1,PUTINX ;SETUP PUT INDEX TABLE
020210 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
020216 012737 000015 001410 MOV #OFFSET!GO,RMCS10 ;SET RMCS1 OUTPUT BUFFER = OFFSET!GO
020224 004737 044424 JSR PC,PUT ;GO WRITE RMCS1 VIA PUT SUBROUTINE
020230 000404 BR 80$ ;GO TO 80$ IF NO ERROR
020232 000240 NOP ;RETURN HERE IF ERROR
020234 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
020236 000137 020526 JMP 160$ ;GO TO 160$ IF ERROR
724 020242 004737 044070 80$:
725
726 020246 004737 044154 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
020252 000404 BR 90$ ;GO TO 90$ IF NO ERROR
020254 000240 NOP ;RETURN HERE IF ERROR
020256 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
020260 000137 020526 JMP 160$ ;GO TO 160$ IF ERROR
727 020264 032737 100000 001346 90$:
728 020272 001012 BIT #ATA,RMDSI ;IS ATA SET??
BNE 100$ ;YES!!

```



```

729 020274 013737 001346 001142      MOV      RMDSI,$BDDAT      ;BAD DATA FOR TYPEOUT
730 020302 042737 077777 001142      BIC      #^CATA,$BDDAT
731 020310 012737 100000 001140      MOV      #ATA,$GDDAT      ;GOOD DATA FOR TYPEOUT
732 020316 104172      EMT      172
733 020320      100$:
734 020320 112737 000014 001543      MOVB     #RMER1,PUTINX     ;SETUP PUT INDEX TABLE
      020326 112737 000200 001544      MOVB     #200,PUTINX+1     ;SET TERMINATOR BYTE
      020334 012737 040000 001424      MOV      #UNS,RMER10      ;SET RMER1 OUTPUT BUFFER = UNS
      020342 004737 044424      JSR      PC,PUT           ;GO WRITE RMER1 VIA PUT SUBROUTINE
      020346 000404      BR      110$             ;GO TO 110$ IF NO ERROR
      020350 000240      NOP
      020352 104000      EMT
      020354 000137 020526      JMP      160$             ;ERROR DEFINED BY PUT SUBROUTINE
      ;GO TO 160$ IF ERROR
735 020360      110$:
736 020360 112737 000000 001543      MOVB     #RMCS1,PUTINX     ;SETUP PUT INDEX TABLE
      020366 112737 000200 001544      MOVB     #200,PUTINX+1     ;SET TERMINATOR BYTE
      020374 012737 000001 001410      MOV      #NOP!GO,RMCS10   ;SET RMCS1 OUTPUT BUFFER = NOP!GO
      020402 004737 044424      JSR      PC,PUT           ;GO WRITE RMCS1 VIA PUT SUBROUTINE
      020406 000404      BR      120$             ;GO TO 120$ IF NO ERROR
      020410 000240      NOP
      020412 104000      EMT
      020414 000137 020526      JMP      160$             ;ERROR DEFINED BY PUT SUBROUTINE
      ;GO TO 160$ IF ERROR
737 020420      120$:
738 020420 004737 044154      JSR      PC,GET           ;GO READ REGISTER(S) WITH GET SUBROUTINE
      020424 000404      BR      130$             ;GO TO 130$ IF NO ERROR
      020426 000240      NOP
      020430 104000      EMT
      020432 000137 020526      JMP      160$             ;ERROR # DEFINED BY GET SUBROUTINE
      ;GO TO 160$ IF ERROR
739 020436 032737 100000 001346 130$:
740 020444 001012      BIT      #ATA,RMDSI       ;IS ATA STILL SET??
741 020446 012737 100000 001140      BNE     140$             ;YES!!
742 020454 013737 001346 001142      MOV      #ATA,$GDDAT      ;GOOD DATA FOR TYPEOUT
743 020462 042737 077777 001142      MOV      RMDSI,$BDDAT     ;BAD DATA FOR TYPEOUT
744 020470 104250      BIC      #^CATA,$BDDAT
745 020472      EMT      250
746 020472 004737 061232      140$:
      JSR      PC,STCDRVSTS   ;GO CHECK FOR CHANGES IN DRIVE STATUS
      020476 000405      BR      150$             ;GO TO 150$ IF NO ERROR
      020500 000240      NOP
      020502 104000      EMT
      020504 004736      JSR      PC,@(SP)+       ;GO CHECK FOR SECONDARY ERRORS
      020506 000137 020526      JMP      160$             ;GO TO 160$ IF NO ERROR
      ;GO BACK FOR MORE ERROR CHECKS
747 020512      150$:
748 020512 004737 046004      JSR      PC,SECERR        ;GO CHECK FOR SECONDARY ERRORS
      020516 000403      BR      160$             ;GO TO 160$ IF NO ERROR
      020520 000240      NOP
      020522 104000      EMT
      020524 004736      JSR      PC,@(SP)+       ;ERROR # DEFINED BY SECERR SUBROUTINE
      ;GO BACK FOR MORE ERROR CHECKS
749 020526      160$:
750
751

```

```

:*****
:*TEST 26      PROGRAM INTERRUPT TEST
:*****
TST26:
      SCOPE      ;SCOPE CALL
      NOP        ;START OF TEST
      MOV      #STACK,SP ;INITIALIZE STACK POINTER
      MOV      $BASE,R0  ;R0 = UNIBUS ADDRESS

```

```

T26      020542 013701 001464      MOV    TSTQUE,R1      ;(R1) = DEVICE BEING TESTED
          020546 012737 000026 001226  MOV    #26,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX
752      020554 004737 043150      JSR    PC,TSTPRP     ;PREPARE DEVICE FOR TEST
753      020560 054130                .WORD  054130        ;TASK DESCRIPTOR AS FOLLOWS:
                                     ;CLEAR CONTROLLER & SELECT DEVICE
                                     ;VERIFY CONTROLLER CLEAR OPERATION
                                     ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                     ;VERIFY PACK ACKNOWLEDGE
                                     ;RECALIBRATE IF "SKI" OR "PIP" IS SET
                                     ;VERIFY RECALIBRATION
          020562 000404                BR     70$          ;GO TO 70$ IF NO ERROR
          020564 000240                NOP
          020566 104000                EMT
          020570 000137 021216                JMP    210$        ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                     ;GO TO 210$ IF ERROR
754      020574                70$:
          020574 113702 001272      MOVB   $VECT1,R2    ;R2 = INTERRUPT ADDRESS
          020600 042702 177400      BIC   #^C<377>,R2  ;CLEAR SIGN EXTENSION
          020604 011204                MOV    (R2),R4      ;SAVE HANDLER ADDRESS
          020606 016205 000002      MOV    2(R2),R5    ;SAVE HANDLER PRIORITY
          020612 012712 021056      MOV    #150$(R2)   ;WRITE HANDLER ADDRESS AND
          020616 012762 000300 000002  MOV    #PR6,2(R2)  ;PRIORITY FOR THIS TEST
          020624 113703 001273      MOVB   $VECT1+1,R3 ;R3 = INTERRUPT LEVEL
          020630 042703 177400      BIC   #^C<377>,R3  ;CLEAR SIGN EXTENSION
          020634 000303                SWAB   R3
          020636 005303                DEC    R3           ;DECREMENT INTERRUPT LEVEL
          020640 100001                BPL   80$
          020642 005003                CLR   R3
          020644 042703 177437      80$:  BIC   #^CPR7,R3
          020650 000240                NOP
          020652 012746 000300      MOV    #PR6,-(SP)  ;;PUSH #PR6 ON STACK
          020656 012746 021014      MOV    #120$,-(SP) ;;PUSH #120$ ON STACK
          020662 010346                MOV    R3,-(SP)    ;;PUSH R3 ON STACK
          020664 012746 021006      MOV    #110$,-(SP) ;;PUSH #110$ ON STACK
          020670 012737 177777 001426  MOV    #-1,RMASO   ;WRITE ONES IN RMAS TO
          020676 112737 000016 001543  MOVB   #RMAS,PUTINX ;CLEAR ALL ATTENTIONS
          020704 112737 000200 001543  MOVB   #200,PUTINX
776      020712 004737 044424      JSR    PC,PUT       ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
          020716 000404                BR     90$          ;GO TO 90$ IF NO ERROR
          020720 000240                NOP
          020722 104000                EMT
          020724 000137 021216                JMP    210$        ;ERROR # DEFINED BY PUT SUBROUTINE
                                     ;GO TO 210$ IF ERROR
778      020730                90$:
          020730 112737 000000 001543  MOVB   #RMCS1,PUTINX ;SETUP PUT INDEX TABLE
          020736 112737 000200 001544  MOVB   #200,PUTINX+1 ;SET TERMINATOR BYTE
          020744 012737 000115 001410  MOV    #OFFSET!GO!IE,RMCS10 ;SET RMCS1 OUTPUT BUFFER = OFFSET!GO!IE
          020752 004737 044424      JSR    PC,PUT       ;GO WRITE RMCS1 VIA PUT SUBROUTINE
          020756 000404                BR     100$        ;GO TO 100$ IF NO ERROR
          020760 000240                NOP
          020762 104000                EMT
          020764 000137 021216                JMP    210$        ;ERROR DEFINED BY PUT SUBROUTINE
                                     ;GO TO 210$ IF ERROR
780      020770                100$:
          020770 004737 044766      JSR    PC,TIMOUT    ;WAIT FOR COMPLETION
          020774 004737 044070      JSR    PC,GETSTS    ;SETUP TO READ ALL REGISTERS
          021000 012703 000205      MOV    #205,R3     ;R3 = GROSS TIMEOUT
          021004 000002                RTI                ;DROP CP PRIORITY

```

```

785 021006 005303      110$: DEC R3          ;TIMEOUT THE INTERRUPT
786 021010 100376      BPL 110$
787
788 ;TIMEOUT BEFORE INTERRUPT
789
790 021012 000002      RTI          ;RAISE CP PRIORITY
791 021014
792 021014 004737 044154 120$: JSR PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
      021020 000404      BR 130$          ;GO TO 130$ IF NO ERROR
      021022 000240      NOP          ;RETURN HERE IF ERROR
      021024 104000      EMT          ;ERROR # DEFINED BY GET SUBROUTINE
      021026 000137 021216 JMP 210$        ;GO TO 210$ IF ERROR
793 021032
794 021032 004737 045152 130$: JSR PC,PRIERR    ;GO CHECK FOR PRIMARY ERRORS
      021036 000405      BR 140$          ;GO TO 140$ IF NO ERROR
      021040 000240      NOP          ;RETURN HERE IF ERROR
      021042 104000      EMT          ;ERROR # DEFINED BY PRIERR SUBROUTINE
      021044 004736      JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      021046 000137 021216 JMP 210$        ;GO TO 210$ IF ERROR
795 021052
796 021052 104251      140$: EMT 251
797 021054 000423      BR 180$
798 021056 022626      150$: CMP (SP)+,(SP)+ ;ADJUST STACK
799 021060 012716 021066 MOV #160$,(SP) ;CHANGE RETURN ADDRESS
800 021064 000002      RTI          ;RAISE CP PRIORITY
801 021066
802 021066 004737 044154 160$: JSR PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
      021072 000404      BR 170$          ;GO TO 170$ IF NO ERROR
      021074 000240      NOP          ;RETURN HERE IF ERROR
      021076 104000      EMT          ;ERROR # DEFINED BY GET SUBROUTINE
      021100 000137 021216 JMP 210$        ;GO TO 210$ IF ERROR
803 021104
804 021104 004737 045152 170$: JSR PC,PRIERR    ;GO CHECK FOR PRIMARY ERRORS
      021110 000405      BR 180$          ;GO TO 180$ IF NO ERROR
      021112 000240      NOP          ;RETURN HERE IF ERROR
      021114 104000      EMT          ;ERROR # DEFINED BY PRIERR SUBROUTINE
      021116 004736      JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      021120 000137 021216 JMP 210$        ;GO TO 210$ IF ERROR
805 021124
806 021124 004737 051644 180$: JSR PC,CMPESTS    ;CHECK ANY ERRORS NOT MASKED
      021130 115760      .WORD NDIMSK ;MASK FOR RMER1
      021132 000010      .WORD DPE    ;MASK FOR RMER2
      021134 000405      BR 190$          ;GO TO 190$ IF NO ERROR
      021136 000240      NOP          ;RETURN HERE IF ERROR
      021140 104000      EMT          ;ERROR # DEFINED BY CMPESTS SUBROUTINE
      021142 004736      JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      021144 000137 021216 JMP 210$        ;GO TO 210$ IF ERROR
807 021150
808 021150 004737 061232 190$: JSR PC,STCDRVSTS ;GO CHECK FOR CHANGES IN DRIVE STATUS
      021154 000405      BR 200$          ;GO TO 200$ IF NO ERROR
      021156 000240      NOP          ;RETURN HERE IF ERROR
      021160 104000      EMT          ;ERROR # DEFINED BY STCDRVSTS SUBROUTINE
      021162 004736      JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      021164 000137 021216 JMP 210$        ;GO TO 210$ IF ERROR
809 021170
810 021170 004737 046004 200$: JSR PC,SECERR    ;GO CHECK FOR SECONDARY ERRORS
  
```

```
021174 000405 BR 205$ ;GO TO 205$ IF NO ERROR
021176 000240 NOP ;RETURN HERE IF ERROR
021200 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
021202 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
021204 000137 021216 JMP 210$ ;GO TO 210$ IF ERROR
811 021210 205$:
812 021210 010412 MOV R4,(R2)
813 021212 010562 000002 MOV R5,2(R2)
814 021216 210$:
815
816 ;*****
;*TEST 27 INHIBIT INTERRUPT TEST
;*****
TST27:
021216 000004 SCOPE ;SCOPE CALL
021220 000240 NOP ;START OF TEST
021222 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
021226 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
021232 013701 001464 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
021236 012737 000027 001226 MOV #27,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
817
818 021244 004737 043150 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
021250 054130 .WORD 054130 ;TASK DESCRIPTOR AS FOLLOWS:
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF "SKI" OR "PIP" IS SET
;VERIFY RECALIBRATION
;GO TO 70$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 210$ IF ERROR
021252 000404 BR 70$
021254 000240 NOP
021256 104000 EMT
021260 000137 021564 JMP 210$
819 021264 70$:
820 021264 113702 001272 MOVB $VECT1,R2 ;R2 = INTERRUPT ADDRESS
821 021270 042702 177400 BIC #^C<377>,R2 ;CLEAR SIGN EXTENSION
822 021274 011204 MOV (R2),R4 ;SAVE HANDLER ADDRESS
823 021276 016205 000002 MOV 2(R2),R5 ;SAVE HANDLER PRIORITY
824 021302 012712 021524 MOV #130,(R2) ;WRITE HANDLER ADDRESS AND
825 021306 012762 000300 000002 MOV #PR6,2(R2) ;PRIORITY FOR THIS TEST
826 021314 113703 001273 MOVB $VECT1+1,R3 ;R3 = INTERRUPT LEVEL
827 021320 042703 177400 BIC #^C<377>,R3 ;CLEAR SIGN EXTENSION
828 021324 012746 000300 MOV #PR6,-(SP) ;PUSH #PR6 ON STACK
829 021330 012746 021556 MOV #180,-(SP) ;PUSH #180$ ON STACK
830 021334 010346 MOV R3,-(SP) ;PUSH R3 ON STACK
831 021336 005316 DEC (SP)
832 021340 100001 BPL 80$
833 021342 005016 CLR (SP)
834 021344 042716 177437 80$: BIC #^CPR7,(SP)
835 021350 012746 021512 MOV #120,-(SP) ;PUSH #120$ ON STACK
836 021354 010346 MOV R3,-(SP) ;PUSH R3 ON STACK
837 021356 012746 021440 MOV #100,-(SP) ;PUSH #100$ ON STACK
838 021362 004737 044070 JSR PC,GETSTS ;SETUP TO READ ALL REGISTERS
839
840 021366 112737 000000 001543 MOVB #RMCS1,PUTINX ;SETUP PUT INDEX TABLE
021374 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
021402 012737 000115 001410 MOV #OFFSET!GO!IE,RMCS10 ;SET RMCS1 OUTPUT BUFFER = OFFSET!GO!IE
```

```

021410 004737 044424      JSR    PC,PUT          ;GO WRITE RMCS1 VIA PUT SUBROUTINE
021414 000404              BR     90$            ;GO TO 90$ IF NO ERROR
021416 000240              NOP                    ;RETURN HERE IF ERROR
021420 104000              EMT                    ;ERROR DEFINED BY PUT SUBROUTINE
021422 000137 021564      JMP    210$           ;GO TO 210$ IF ERROR
841 021426                90$:
842 021426 004737 044766      JSR    PC,TIMOUT       ;WAIT FOR GO = 0
843 021432 012703 000205      MOV    #205,R3        ;R3 = GROSS TIMER
844 021436 000002              RTI
845
846 021440 005303          100$: DEC    R3          ;DELAY
847 021442 100376          BPL    100$
848
849 021444 112737 000000 001543  MOVB   #RMCS1,PUTINX   ;SETUP PUT INDEX TABLE
021452 112737 000200 001544  MOVB   #200,PUTINX+1   ;SET TERMINATOR BYTE
021460 012737 000000 001410  MOV    #NOP,RMCS10    ;SET RMCS1 OUTPUT BUFFER = NOP
021466 004737 044424      JSR    PC,PUT          ;GO WRITE RMCS1 VIA PUT SUBROUTINE
021472 000404              BR     110$           ;GO TO 110$ IF NO ERROR
021474 000240              NOP                    ;RETURN HERE IF ERROR
021476 104000              EMT                    ;ERROR DEFINED BY PUT SUBROUTINE
021500 000137 021564      JMP    210$           ;GO TO 210$ IF ERROR
850 021504 012703 000205      110$: MOV    #205,R3
851 021510 000002              RTI
852
853 021512 012712 021526      120$: MOV    #140$,(R2)
854 021516 005303          125$: DEC    R3
855 021520 100376          BPL    125$
856 021522 000002              RTI                    ;NO ERROR
857
858 021524 022626          130$: CMP    (SP)+,(SP)+ ;ADJUST STACK
859 021526 022626          140$: CMP    (SP)+,(SP)+ ;ADJUST STACK
860 021530 012716 021536      MOV    #160$,(SP)
861 021534 000002              RTI
862 021536
863 021536 004737 044154      160$: JSR    PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
021542 000404              BR     170$           ;GO TO 170$ IF NO ERROR
021544 000240              NOP                    ;RETURN HERE IF ERROR
021546 104000              EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
021550 000137 021564      JMP    210$           ;GO TO 210$ IF ERROR
864 021554          170$: EMT    252
021554 104252
865
866 021556 010412          180$: MOV    R4,(R2)
867 021560 010562 000002      MOV    R5,2(R2)
868 021564          210$:
869
870

```

```

*****
:*TEST 30      RETURN TO CENTERLINE TEST
*****

```

```

TST30:
021564
021564 000004              SCOPE ;SCOPE CALL
021566 000240              NOP   ;START OF TEST
021570 012706 001100      MOV    #STACK,SP     ;INITIALIZE STACK POINTER
021574 013700 001276      MOV    $BASE,R0      ;R0 = UNIBUS ADDRESS
021600 013701 001464      MOV    TSTQUE,R1     ;(R1) = DEVICE BEING TESTED
021604 012737 000030 001226  MOV    #30,$TESTN    ;SET TEST NUMBER IN APT MAIL BOX
871

```



```

895 022052
896 022052 004737 061232 130$: JSR PC,STCDRVSTS ;GO CHECK FOR CHANGES IN DRIVE STATUS
    022056 000405 BR 140$ ;GO TO 140$ IF NO ERROR
    022060 000240 NOP ;RETURN HERE IF ERROR
    022062 104000 EMT ;ERROR # DEFINED BY STCDRVSTS SUBROUTINE
    022064 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
    022066 000137 022106 JMP 150$ ;GO TO 150$ IF ERROR
897 022072
898 022072 004737 046004 140$: JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
    022076 000403 BR 150$ ;GO TO 150$ IF NO ERROR
    022100 000240 NOP ;RETURN HERE IF ERROR
    022102 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
    022104 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
899 022106
900
901
    
```

```

:*****
:*TEST 31 READ IN PRESET TEST
:*****
TST31:
    
```

```

022106
022106 000004 SCOPE ;SCOPE CALL
022110 000240 NOP ;START OF TEST
022112 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
022116 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
022122 013701 001464 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
022126 012737 000031 001226 MOV #31,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
902
903 022134 004737 043150 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
    022140 054130 .WORD 054130 ;TASK DESCRIPTOR AS FOLLOWS:
    ;CLEAR CONTROLLER & SELECT DEVICE
    ;VERIFY CONTROLLER CLEAR OPERATION
    ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
    ;VERIFY PACK ACKNOWLEDGE
    ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
    ;VERIFY RECALIBRATION
    022142 000404 BR 70$ ;GO TO 70$ IF NO ERROR
    022144 000240 NOP ;RETURN HERE IF ERROR
    022146 104000 EMT ;ERROR # DEFINED BY TSTPRP SUBROUTINE
    022150 000137 022470 JMP 160$ ;GO TO 160$ IF ERROR
904 022154
905 022154 012702 001543 70$: MOV #PUTINX,R2 ;R2 POINTS TO INDEX TABLE
906 022160 112722 000006 MOVB #RMDA,(R2)+
907 022164 112722 000034 MOVB #RMDC,(R2)+
908 022170 112722 000032 MOVB #RMOF,(R2)+
909 022174 112722 000000 MOVB #RMCS1,(R2)+
910 022200 112722 000200 MOVB #200,(R2)+
911 022204 012737 177777 001416 MOV #-1,RMDAO
912 022212 012737 177777 001444 MOV #-1,RMDCO
913 022220 012737 016200 001442 MOV #FMT16!ECI!HCI!OFD,RMOFO
914 022226 012737 000021 001410 MOV #RIP!GO,RMCS1O
915 022234 004737 044424 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
    022240 000404 BR 80$ ;GO TO 80$ IF NO ERROR
    022242 000240 NOP ;RETURN HERE IF ERROR
    022244 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
    022246 000137 022470 JMP 160$ ;GO TO 160$ IF ERROR
916 022252 004737 044070 80$: JSR PC,GETSTS ;SETUP FOR STATUS FETCH
917 022256 004737 044766 JSR PC,TIMOUT ;WAIT FOR RIP TO COMPLETE
918
    
```

```
919 022262 004737 044154 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
    022266 000404 BR 90$ ;GO TO 90$ IF NO ERROR
    022270 000240 NOP ;RETURN HERE IF ERROR
    022272 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
    022274 000137 022470 JMP 160$ ;GO TO 160$ IF ERROR
920 022300 90$:
921 022300 004737 045152 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
    022304 000405 BR 100$ ;GO TO 100$ IF NO ERROR
    022306 000240 NOP ;RETURN HERE IF ERROR
    022310 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
    022312 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
    022314 000137 022470 JMP 160$ ;GO TO 160$ IF ERROR
922 022320 013737 001366 001142 100$: MOV RMOFI,$BDDAT ;IS RMOF RESET??
923 022326 042737 161577 001142 BIC #^C<FMT16!ECI!HCI!OFD>,$BDDAT
924 022334 001403 BEQ 110$ ;YES!!
925 022336 005037 001140 CLR $GDDAT ;GOOD DATA FOR TYPEOUT
926 022342 104160 EMT 160
927 022344 005737 001342 110$: TST RMDAI ;IS RMDA RESET??
928 022350 001406 BEQ 120$ ;YES!!
929 022352 013737 001342 001142 MOV RMDAI,$BDDAT ;BAD DATA FOR TYPEOUT
930 022360 005037 001140 CLR $GDDAT ;GOOD DATA FOR TYPEOUT
931 022364 104161 EMT 161
932 022366 005737 001370 120$: TST RMDCI ;IS RMDC RESET??
933 022372 001406 BEQ 130$ ;YES!!
934 022374 013737 001370 001142 MOV RMDCI,$BDDAT ;BAD DATA FOR TYPEOUT
935 022402 005037 001140 CLR $GDDAT ;GOOD DATA FOR TYPEOUT
936 022406 104162 EMT 162
937 022410 130$:
938 022410 004737 051644 JSR PC,MPERRSTS ;CHECK ANY ERRORS NOT MASKED
    022414 115760 .WORD NDIMSK ;MASK FOR RMER1
    022416 000010 .WORD DPE ;MASK FOR RMER2
    022420 000405 BR 140$ ;GO TO 140$ IF NO ERROR
    022422 000240 NOP ;RETURN HERE IF ERROR
    022424 104000 EMT ;ERROR # DEFINED BY MPERRSTS SUBROUTINE
    022426 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
    022430 000137 022470 JMP 160$ ;GO TO 160$ IF ERROR
939 022434 140$:
940 022434 004737 061232 JSR PC,STCDRVSTS ;GO CHECK FOR CHANGES IN DRIVE STATUS
    022440 000405 BR 150$ ;GO TO 150$ IF NO ERROR
    022442 000240 NOP ;RETURN HERE IF ERROR
    022444 104000 EMT ;ERROR # DEFINED BY STCDRVSTS SUBROUTINE
    022446 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
    022450 000137 022470 JMP 160$ ;GO TO 160$ IF ERROR
941 022454 150$:
942 022454 004737 046004 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
    022460 000403 BR 160$ ;GO TO 160$ IF NO ERROR
    022462 000240 NOP ;RETURN HERE IF ERROR
    022464 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
    022466 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
943 022470 160$:
944
945
*****
;*TEST 32 RMDC CLEAR OFFSET TEST
*****
TST32:
    022470 SCOPE ;SCOPE CALL
    022470 000004 ;START OF TEST
    022472 000240 NOP
```



```

022474 012706 001100      MOV    #STACK,SP      ;INITIALIZE STACK POINTER
022500 013700 001276      MOV    $BASE,R0       ;R0 = UNIBUS ADDRESS
022504 013701 001464      MOV    TSTQUE,R1      ;(R1) = DEVICE BEING TESTED
022510 012737 000032 001226  MOV    #32,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
946
947 022516 004737 043150      JSR    PC,TSTPRP      ;PREPARE DEVICE FOR TEST
022522 054130      .WORD 054130         ;TASK DESCRIPTOR AS FOLLOWS:
                                ;CLEAR CONTROLLER & SELECT DEVICE
                                ;VERIFY CONTROLLER CLEAR OPERATION
                                ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                ;VERIFY PACK ACKNOWLEDGE
                                ;RECALIBRATE IF "SKI" OR "PIP" IS SET
                                ;VERIFY RECALIBRATION
022524 000404      BR     70$           ;GO TO 70$ IF NO ERROR
022526 000240      NOP
022530 104000      EMT
022532 000137 023006      JMP    140$         ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                ;GO TO 140$ IF ERROR
948 022536      70$:
949 022536 112737 000000 001543  MOVB   #RMCS1,PUTINX  ;SETUP PUT INDEX TABLE
022544 112737 000200 001544  MOVB   #200,PUTINX+1 ;SET TERMINATOR BYTE
022552 012737 000015 001410  MOV    #OFFSET!GO,RMCS10 ;SET RMCS1 OUTPUT BUFFER = OFFSET!GO
022560 004737 044424      JSR    PC,PUT        ;GO WRITE RMCS1 VIA PUT SUBROUTINE
022564 000404      BR     80$           ;GO TO 80$ IF NO ERROR
022566 000240      NOP
022570 104000      EMT
022572 000137 023006      JMP    140$         ;ERROR DEFINED BY PUT SUBROUTINE
                                ;GO TO 140$ IF ERROR
950 022576 004737 044070 80$: JSR    PC,GETSTS     ;SETUP FOR STATUS
951
952 022602 004737 044154      JSR    PC,GET        ;GO READ REGISTER(S) WITH GET SUBROUTINE
022606 000404      BR     90$           ;GO TO 90$ IF NO ERROR
022610 000240      NOP
022612 104000      EMT
022614 000137 023006      JMP    140$         ;ERROR # DEFINED BY GET SUBROUTINE
                                ;GO TO 140$ IF ERROR
953 022620      90$:
954 022620 004737 045152      JSR    PC,PRIERR     ;GO CHECK FOR PRIMARY ERRORS
022624 000405      BR     100$          ;GO TO 100$ IF NO ERROR
022626 000240      NOP
022630 104000      EMT
022632 004736      JSR    PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
022634 000137 023006      JMP    140$         ;GO TO 140$ IF ERROR
955 022640 032737 000001 001346 100$: BIT    #OM,RMDSI     ;OFFSET ON??
956 022646 001013      BNE    110$          ;YES
957 022650 013737 001346 001142  MOV    RMDSI,$BDDAT  ;BAD DATA FOR TYPEOUT
958 022656 042737 177776 001142  BIC    #^COM,$BDDAT
959 022664 012737 000001 001140  MOV    #OM,$GDDAT   ;GOOD DATA FOR TYPEOUT
960 022672 104156      EMT    156
961 022674 000444      BR     140$          ;SKIP REST OF TEST
962 022676      110$:
963 022676 112737 000034 001543  MOVB   #RMDC,PUTINX  ;SETUP PUT INDEX TABLE
022704 112737 000200 001544  MOVB   #200,PUTINX+1 ;SET TERMINATOR BYTE
022712 012737 000000 001444  MOV    #0,RMDCO     ;SET RMDC OUTPUT BUFFER = 0
022720 004737 044424      JSR    PC,PUT        ;GO WRITE RMDC VIA PUT SUBROUTINE
022724 000404      BR     120$          ;GO TO 120$ IF NO ERROR
022726 000240      NOP
022730 104000      EMT
022732 000137 023006      JMP    140$         ;ERROR DEFINED BY PUT SUBROUTINE
                                ;GO TO 140$ IF ERROR
964 022736      120$:

```

```

965 022736 004737 044154      JSR    PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      022742 000404          BR     130$           ;GO TO 130$ IF NO ERROR
      022744 000240          NOP                    ;RETURN HERE IF ERROR
      022746 104000          EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      022750 000137 023006    JMP     140$           ;GO TO 140$ IF ERROR
966 022754 032737 000001 001346 130$: BIT    #OM,RMDSI       ;DID OFFSET MODE RESET??
967 022762 001411          BEQ    140$
968 022764 013737 001346 001142 MOV    RMDSI,$BDDAT   ;BAD DATA FOR TYPEOUT
969 022772 042737 177776 001142 BIC    #^COM,$BDDAT
970 023000 005037 001140    CLR    $GDDAT        ;GOOD DATA FOR TYPEOUT
971 023004 104253          EMT    253
972 023006          140$:
973
974

```

```

;*****
;*TEST 33      ILLEGAL FUNCTION TEST
;*****
TST33:

```

```

023006
023006 000004          SCOPE                ;SCOPE CALL
023010 000240          NOP                    ;START OF TEST
023012 012706 001100    MOV    #STACK,SP     ;INITIALIZE STACK POINTER
023016 013700 001276    MOV    $BASE,R0      ;R0 = UNIBUS ADDRESS
023022 013701 001464    MOV    TSTQUE,R1     ;(R1) = DEVICE BEING TESTED
023026 012737 000033 001226 MOV    #33,$TESTN    ;SET TEST NUMBER IN APT MAIL BOX
975
976 023034 004737 043150    JSR    PC,TSTPRP     ;PREPARE DEVICE FOR TEST
      023040 040000    .WORD 040000        ;TASK DESCRIPTOR AS FOLLOWS:
      023042 000404          BR     5$             ;CLEAR CONTROLLER & SELECT DEVICE
      023044 000240          NOP                    ;GO TO 5$ IF NO ERROR
      023046 104000          EMT                    ;RETURN HERE IF ERROR
      023050 000137 023572    JMP     110$          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
      023054          5$: JSR    PC,GETSTS       ;GO TO 110$ IF ERROR
977 023054          JSR    PC,GETSTS       ;SETUP FOR STATUS
978 023054 004737 044070    MOV    #NOP,R2
979 023060 012702 000000    10$:
980 023064          JSR    PC,CNTCLR      ;GO ISSUE CONTROLLER CLEAR
981 023064 004737 053530    BR     20$           ;GO TO 20$ IF NO ERROR
      023070 000404          NOP                    ;RETURN HERE IF ERROR
      023072 000240          EMT                    ;ERROR NUMBER DEFINED BY SUBROUTINE
      023074 104000          JMP     110$          ;GO TO 110$ IF ERROR
      023076 000137 023572    20$:
982 023102
983
984          ;SET DIAGNOSTIC MODE AND DIAGNOSTIC UNIT READY, DIAGNOSTIC ON
985          ;CYLINDER
986 023102 112737 000024 001543 MOVB   #RMMR1,PUTINX  ;SETUP PUT INDEX TABLE
      023110 112737 000200 001544 MOVB   #200,PUTINX+1 ;SET TERMINATOR BYTE
      023116 012737 000001 001434 MOV    #DMD,RMMR1O   ;SET RMMR1 OUTPUT BUFFER = DMD
      023124 004737 044424    JSR    PC,PUT        ;GO WRITE RMMR1 VIA PUT SUBROUTINE
      023130 000404          BR     30$           ;GO TO 30$ IF NO ERROR
      023132 000240          NOP                    ;RETURN HERE IF ERROR
      023134 104000          EMT                    ;ERROR DEFINED BY PUT SUBROUTINE
      023136 000137 023572    JMP     110$          ;GO TO 110$ IF ERROR
987 023142          30$:
988 023142 112737 000024 001543 MOVB   #RMMR1,PUTINX  ;SETUP PUT INDEX TABLE
      023150 112737 000200 001544 MOVB   #200,PUTINX+1 ;SET TERMINATOR BYTE
      023156 012737 001401 001434 MOV    #DMD!MUR!MOC,RMMR1O ;SET RMMR1 OUTPUT BUFFER = DMD!MUR!MOC
      023164 004737 044424    JSR    PC,PUT        ;GO WRITE RMMR1 VIA PUT SUBROUTINE

```

```

023170 000404 BR 35$ ;GO TO 35$ IF NO ERROR
023172 000240 NOP ;RETURN HERE IF ERROR
023174 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
023176 000137 023572 JMP 110$ ;GO TO 110$ IF ERROR
989 023202 35$:
990
991 ;SET VOLUME VALID BEFORE ISSUING FUNCTION CODE BEING TESTED
992 023202 112737 000000 001543 MOVB #RMCS1,PUTINX ;SETUP PUT INDEX TABLE
023210 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
023216 012737 000023 001410 MOV #PAKACK!GO,RMCS10 ;SET RMCS1 OUTPUT BUFFER = PAKACK!GO
023224 004737 044424 JSR PC,PUT ;GO WRITE RMCS1 VIA PUT SUBROUTINE
023230 000404 BR 40$ ;GO TO 40$ IF NO ERROR
023232 000240 NOP ;RETURN HERE IF ERROR
023234 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
023236 000137 023572 JMP 110$ ;GO TO 110$ IF ERROR
993 023242 40$:
994 023242 012737 000001 001410 MOV #GO,RMCS10
995 023250 050237 001410 BIS R2,RMCS10 ;WRITE FUNCTION CODE IN BUFFER
996 023254 012737 103640 001414 MOV #BUFONE,RMBAO ;DUMMY BUS ADDRESS
997 023262 012737 177777 001412 MOV #-1,RMWCO ;DUMMY WORD COUNT
998 023270 005037 001416 CLR RMDAO ;CLEAR DISK ADDRESS
999 023274 005037 001444 CLR RMDCO ;CLEAR CYLINDER ADDRESS
1000 023300 012737 010000 001442 MOV #FMT16,RMOFO ;16 BIT FORMAT
1001 023306 012703 001543 MOV #PUTINX,R3 ;WRITE REGISTER INDEX TABLE
1002 023312 112723 000004 MOVB #RMBA,(R3)+
1003 023316 112723 000002 MOVB #RMWC,(R3)+
1004 023322 112723 000032 MOVB #RMOF,(R3)+
1005 023326 112723 000006 MOVB #RMDA,(R3)+
1006 023332 112723 000034 MOVB #RMDC,(R3)+
1007 023336 112723 000000 MOVB #RMCS1,(R3)+
1008 023342 112713 000200 MOVB #200,(R3)
1009 023346 004737 044424 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
023352 000404 BR 50$ ;GO TO 50$ IF NO ERROR
023354 000240 NOP ;RETURN HERE IF ERROR
023356 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
023360 000137 023572 JMP 110$ ;GO TO 110$ IF ERROR
1010 023364 004737 044766 50$: JSR PC,TIMOUT ;WAIT FOR GO TO RESET
1011
1012 023370 004737 044154 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
023374 000404 BR 60$ ;GO TO 60$ IF NO ERROR
023376 000240 NOP ;RETURN HERE IF ERROR
023400 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
023402 000137 023572 JMP 110$ ;GO TO 110$ IF ERROR
1013 023406 60$:
1014 023406 004737 045152 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
023412 000405 BR 70$ ;GO TO 70$ IF NO ERROR
023414 000240 NOP ;RETURN HERE IF ERROR
023416 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
023420 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
023422 000137 023572 JMP 110$ ;GO TO 110$ IF ERROR
1015 023426 016237 067046 001140 70$: MOV FNCDTB(R2),$GDDAT ;GET ENTRY FROM FUNCTION CODE TABLE
1016 023434 042737 177776 001140 BIC #*CILF,$GDDAT
1017 023442 013737 001350 001142 MOV RMER11,$BDDAT ;BAD DATA FOR TYPEOUT
1018 023450 042737 177776 001142 BIC #*CILF,$BDDAT
1019 023456 023737 001140 001142 CMP $GDDAT,$BDDAT ;IS ILF STATUS CORRECT?
1020 023464 001402 BEQ 80$ ;YES!!
1021 023466 104254 EMT 254
  
```

```

1022 023470 000440 BR 110$
1023 023472 016237 067046 001140 80$: MOV FNCDTB(R2), $GDDAT
1024 023500 032737 040000 001346 BIT #ERR, RMDSI ;WAS AN ERROR DETECTED??
1025 023506 001403 BEQ 90$ ;NO!!
1026 023510 052737 100000 001140 BIS #ATA, $GDDAT ;YES - ATA SHOULD BE ON
1027 023516 042737 077777 001140 90$: BIC #^CATA, $GDDAT
1028 023524 013737 001346 001142 MOV RMDSI, $BDDAT ;GET DRIVE'S ATTENTION
1029 023532 042737 077777 001142 BIC #^CATA, $BDDAT
1030 023540 023737 001140 001142 CMP $GDDAT, $BDDAT ;IS ATA STATUS OK??
1031 023546 001402 BEQ 100$ ;YES!!
1032 023550 104255 EMT 255
1033 023552 000407 BR 110$
1034 023554 062702 000002 100$: ADD #2, R2 ;GO TO NEXT FUNCTION CODE
1035 023560 022702 000076 CMP #ILF76, R2 ;DONE??
1036 023564 103402 BLO 110$ ;YES !!
1037 023566 000137 023064 JMP 10$ ;TEST NEXT FUNCTION
1038 023572
1039
1040
  
```

```

*****
;*TEST 34 INVALID COMMAND TEST
*****
TST34:
  
```

```

023572 000004 SCOPE ;SCOPE CALL
023572 000240 NOP ;START OF TEST
023574 000240 MOV #STACK, SP ;INITIALIZE STACK POINTER
023576 012706 001100 MOV $BASE, R0 ;R0 = UNIBUS ADDRESS
023602 013700 001276 MOV TSTQUE, R1 ;(R1) = DEVICE BEING TESTED
023606 013701 001464 MOV #34, $TESTN ;;SET TEST NUMBER IN APT MAIL BOX
023612 012737 000034 001226
1041
1042 023620 004737 043150 JSR PC, TSTPRP ;PREPARE DEVICE FOR TEST
023624 040000 .WORD 040000 ;TASK DESCRIPTOR AS FOLLOWS:
;CLEAR CONTROLLER & SELECT DEVICE
023626 000404 BR 5$ ;GO TO 5$ IF NO ERROR
023630 000240 NOP ;RETURN HERE IF ERROR
023632 104000 EMT ;ERROR # DEFINED BY TSTPRP SUBROUTINE
023634 000137 024316 JMP 110$ ;GO TO 110$ IF ERROR
1043 023640 5$:
1044 023640 004737 044070 JSR PC, GETSTS ;SETUP FOR STATUS
1045 023644 012702 000000 MOV #NOP, R2
1046 023650 10$:
1047 023650 004737 053530 JSR PC, CNTCLR ;GO ISSUE CONTROLLER CLEAR
023654 000404 BR 20$ ;GO TO 20$ IF NO ERROR
023656 000240 NOP ;RETURN HERE IF ERROR
023660 104000 EMT ;ERROR NUMBER DEFINED BY SUBROUTINE
023662 000137 024316 JMP 110$ ;GO TO 110$ IF ERROR
1048 023666 20$:
1049
1050 ;SET DIAGNOSTIC MODE AND DIAGNOSTIC UNIT READY, DIAGNOSTIC ON
1051 ;CYLINDER
1052 023666 112737 000024 001543 MOVB #RMMR1, PUTINX ;SETUP PUT INDEX TABLE
023674 112737 000200 001544 MOVB #200, PUTINX+1 ;SET TERMINATOR BYTE
023702 012737 000001 001434 MOV #DMD, RMMR10 ;SET RMMR1 OUTPUT BUFFER = DMD
023710 004737 044424 JSR PC, PUT ;GO WRITE RMMR1 VIA PUT SUBROUTINE
023714 000404 BR 30$ ;GO TO 30$ IF NO ERROR
023716 000240 NOP ;RETURN HERE IF ERROR
023720 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
023722 000137 024316 JMP 110$ ;GO TO 110$ IF ERROR
  
```

```

1053 023726
1054 023726 112737 000024 001543 30$: MOVB #RMMR1,PUTINX ;SETUP PUT INDEX TABLE
      023734 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
      023742 012737 001401 001434 MOV #DMD!MUR!MOC,RMMR10 ;SET RMMR1 OUTPUT BUFFER = DMD!MUR!MOC
      023750 004737 044424 JSR PC,PUT ;GO WRITE RMMR1 VIA PUT SUBROUTINE
      023754 000404 BR 35$ ;GO TO 35$ IF NO ERROR
      023756 000240 NOP ;RETURN HERE IF ERROR
      023760 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
      023762 000137 024316 JMP 110$ ;GO TO 110$ IF ERROR
1055 023766 35$:
1056
1057
1058 ;VOLUME VALID IS LEFT RESET FOR THE TEST
1059
1060 023766 40$:
1061 023766 012737 000001 001410 MOV #GO,RMCS10
1062 023774 050237 001410 BIS R2,RMCS10 ;WRITE FUNCTION CODE IN BUFFER
1063 024000 012737 103640 001414 MOV #BUFONE,RMBA0 ;DUMMY BUS ADDRESS
1064 024006 012737 177777 001412 MOV #-1,RMWC0 ;DUMMY WORD COUNT
1065 024014 005037 001416 CLR RMDA0 ;CLEAR DISK ADDRESS
1066 024020 005037 001444 CLR RMDC0 ;CLEAR CYLINDER ADDRESS
1067 024024 012737 010000 001442 MOV #FMT16,RMOFO ;16 BIT FORMAT
1068 024032 012703 001543 MOV #PUTINX,R3 ;WRITE REGISTER INDEX TABLE
1069 024036 112723 000004 MOVBA,(R3)+
1070 024042 112723 000002 MOVBMWC,(R3)+
1071 024046 112723 000032 MOVBMOF,(R3)+
1072 024052 112723 000006 MOVBMODA,(R3)+
1073 024056 112723 000034 MOVBMDC,(R3)+
1074 024062 112723 000000 MOVBMCS1,(R3)+
1075 024066 112713 000200 MOVBM200,(R3)
1076 024072 004737 044424 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      024076 000404 BR 50$ ;GO TO 50$ IF NO ERROR
      024100 000240 NOP ;RETURN HERE IF ERROR
      024102 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
      024104 000137 024316 JMP 110$ ;GO TO 110$ IF ERROR
1077 024110 004737 044766 50$: JSR PC,TIMOUT ;WAIT FOR GO TO RESET
1078
1079 024114 004737 044154 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
      024120 000404 BR 60$ ;GO TO 60$ IF NO ERROR
      024122 000240 NOP ;RETURN HERE IF ERROR
      024124 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
      024126 000137 024316 JMP 110$ ;GO TO 110$ IF ERROR
1080 024132 60$:
1081 024132 004737 045152 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
      024136 000405 BR 70$ ;GO TO 70$ IF NO ERROR
      024140 000240 NOP ;RETURN HERE IF ERROR
      024142 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
      024144 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      024146 000137 024316 JMP 110$ ;GO TO 110$ IF ERROR
1082 024152 016237 067046 001140 70$: MOV FNCDTB(R2),$GDDAT ;GET ENTRY FROM FUNCTION CODE TABLE
1083 024160 042737 167777 001140 BIC #*CIVC,$GDDAT
1084 024166 013737 001376 001142 MOV RMER21,$BDDAT ;BAD DATA FOR TYPEOUT
1085 024174 042737 167777 001142 BIC #*CIVC,$BDDAT
1086 024202 023737 001140 001142 CMP $GDDAT,$BDDAT ;IS IVC STATUS CORRECT?
1087 024210 001402 BEQ 80$ ;YES!!
1088 024212 104216 EMT 216
1089 024214 000440 BR 110$
  
```

```

1090 024216 016237 067046 001140 80$: MOV FNCDTB(R2), $GDDAT
1091 024224 032737 040000 001346 BIT #ERR, RMDSI ; WAS AN ERROR DETECTED??
1092 024232 001403 BEQ 90$ ; NO!!
1093 024234 052737 100000 001140 BIS #ATA, $GDDAT ; YES - ATA SHOULD BE ON
1094 024242 042737 077777 001140 90$: BIC #^CATA, $GDDAT
1095 024250 013737 001346 001142 MOV RMDSI, $BDDAT ; GET DRIVE'S ATTENTION
1096 024256 042737 077777 001142 BIC #^CATA, $BDDAT
1097 024264 023737 001140 001142 CMP $GDDAT, $BDDAT ; IS ATA STATUS OK??
1098 024272 001402 BEQ 100$ ; YES!!
1099 024274 104255 EMT 255
1100 024276 000407 BR 110$
1101 024300 062702 000002 100$: ADD #2, R2 ; GO TO NEXT FUNCTION CODE
1102 024304 022702 000076 CMP #ILF76, R2 ; DONE??
1103 024310 103402 BLO 110$ ; YES !!
1104 024312 000137 023650 JMP 10$ ; TEST NEXT FUNCTION
1105 024316
1106
1107
  
```

```

*****
;*TEST 35 INVALID ADDRESS ERROR TEST
*****
TST35:
  
```

```

024316 000004 SCOPE ; SCOPE CALL
024320 000240 NOP ; START OF TEST
024322 012706 001100 MOV #STACK, SP ; INITIALIZE STACK POINTER
024326 013700 001276 MOV $BASE, R0 ; R0 = UNIBUS ADDRESS
024332 013701 001464 MOV TSTQUE, R1 ; (R1) = DEVICE BEING TESTED
024336 012737 000035 001226 MOV #35, $TESTN ; SET TEST NUMBER IN APT MAIL BOX
1108
1109 024344 004737 043150 JSR PC, TSTPRP ; PREPARE DEVICE FOR TEST
024350 040000 .WORD 040000 ; TASK DESCRIPTOR AS FOLLOWS:
; CLEAR CONTROLLER & SELECT DEVICE
024352 000404 BR 5$ ; GO TO 5$ IF NO ERROR
024354 000240 NOP ; RETURN HERE IF ERROR
024356 104000 EMT ; ERROR # DEFINED BY TSTPRP SUBROUTINE
024360 000137 025106 JMP 110$ ; GO TO 110$ IF ERROR
1110 024364 5$: JSR PC, GETSTS ; SETUP FOR STATUS
1111 024364 004737 044070 MOV #NOP, R2
1112 024370 012702 000000
1113 024374 10$: JSR PC, CNTCLR ; GO ISSUE CONTROLLER CLEAR
1114 024374 004737 053530 BR 20$ ; GO TO 20$ IF NO ERROR
024400 000404 NOP ; RETURN HERE IF ERROR
024402 000240 EMT ; ERROR NUMBER DEFINED BY SUBROUTINE
024404 104000 JMP 110$ ; GO TO 110$ IF ERROR
024406 000137 025106
1115 024412 20$:
1116
1117 ; SET DIAGNOSTIC MODE AND DIAGNOSTIC UNIT READY, DIAGNOSTIC ON
1118 ; CYLINDER
1119 024412 112737 000024 001543 MOVB #RMMR1, PUTINX ; SETUP PUT INDEX TABLE
024420 112737 000200 001544 MOVB #200, PUTINX+1 ; SET TERMINATOR BYTE
024426 012737 000001 001434 MOV #DMD, RMMR10 ; SET RMMR1 OUTPUT BUFFER = DMD
024434 004737 044424 JSR PC, PUT ; GO WRITE RMMR1 VIA PUT SUBROUTINE
024440 000404 BR 30$ ; GO TO 30$ IF NO ERROR
024442 000240 NOP ; RETURN HERE IF ERROR
024444 104000 EMT ; ERROR DEFINED BY PUT SUBROUTINE
024446 000137 025106 JMP 110$ ; GO TO 110$ IF ERROR
1120 024452 30$:
  
```

```

1121 024452 112737 000024 001543      MOVB   #RMMR1,PUTINX      ;SETUP PUT INDEX TABLE
      024460 112737 000200 001544      MOVB   #200,PUTINX+1     ;SET TERMINATOR BYTE
      024466 012737 001401 001434      MOV    #DMD!MUR!MOC,RMMR10 ;SET RMMR1 OUTPUT BUFFER = DMD!MUR!MOC
      024474 004737 044424                JSR    PC,PUT             ;GO WRITE RMMR1 VIA PUT SUBROUTINE
      024500 000404                BR     35$               ;GO TO 35$ IF NO ERROR
      024502 000240                NOP                                ;RETURN HERE IF ERROR
      024504 104000                EMT                                ;ERROR DEFINED BY PUT SUBROUTINE
      024506 000137 025106                JMP    110$              ;GO TO 110$ IF ERROR
1122 024512 35$:
1123
1124 ;SET VOLUME VALID BEFORE ISSUING FUNCTION CODE BEING TESTED
1125 024512 112737 000000 001543      MOVB   #RMCS1,PUTINX     ;SETUP PUT INDEX TABLE
      024520 112737 000200 001544      MOVB   #200,PUTINX+1     ;SET TERMINATOR BYTE
      024526 012737 000023 001410      MOV    #PAKACK!GO,RMCS10 ;SET RMCS1 OUTPUT BUFFER = PAKACK!GO
      024534 004737 044424                JSR    PC,PUT             ;GO WRITE RMCS1 VIA PUT SUBROUTINE
      024540 000404                BR     40$               ;GO TO 40$ IF NO ERROR
      024542 000240                NOP                                ;RETURN HERE IF ERROR
      024544 104000                EMT                                ;ERROR DEFINED BY PUT SUBROUTINE
      024546 000137 025106                JMP    110$              ;GO TO 110$ IF ERROR
1126 024552 40$:
1127 024552 012737 000001 001410      MOV    #GO,RMCS10
1128 024560 050237 001410      BIS    R2,RMCS10         ;WRITE FUNCTION CODE IN BUFFER
1129 024564 012737 103640 001414      MOV    #BUFONE,RMBAO     ;DUMMY BUS ADDRESS
1130 024572 012737 177777 001412      MOV    #-1,RMWCO         ;DUMMY WORD COUNT
1131 024600 012737 177777 001416      MOV    #-1,RMDAO         ;USE INVALID DISK ADDRESS
1132 024606 012737 177777 001444      MOV    #-1,RMDCO         ;USE INVALID CYLINDER ADDRESS
1133 024614 012737 010000 001442      MOV    #FMT16,RMOFO      ;16 BIT FORMAT
1134 024622 012703 001543      MOV    #PUTINX,R3        ;WRITE REGISTER INDEX TABLE
1135 024626 112723 000004      MOVB   #RMBA,(R3)+
1136 024632 112723 000002      MOVB   #RMWC,(R3)+
1137 024636 112723 000032      MOVB   #RMOF,(R3)+
1138 024642 112723 000006      MOVB   #RMDA,(R3)+
1139 024646 112723 000034      MOVB   #RMDC,(R3)+
1140 024652 112723 000000      MOVB   #RMCS1,(R3)+
1141 024656 112713 000200      MOVB   #200,(R3)
1142 024662 004737 044424      JSR    PC,PUT             ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      024666 000404                BR     50$               ;GO TO 50$ IF NO ERROR
      024670 000240                NOP                                ;RETURN HERE IF ERROR
      024672 104000                EMT                                ;ERROR # DEFINED BY PUT SUBROUTINE
      024674 000137 025106                JMP    110$              ;GO TO 110$ IF ERROR
1143 024700 004737 044766      JSR    PC,TIMOUT         ;WAIT FOR GO TO RESET
1144
1145 024704 004737 044154      JSR    PC,GET            ;GO READ REGISTER(S) WITH GET SUBROUTINE
      024710 000404                BR     60$               ;GO TO 60$ IF NO ERROR
      024712 000240                NOP                                ;RETURN HERE IF ERROR
      024714 104000                EMT                                ;ERROR # DEFINED BY GET SUBROUTINE
      024716 000137 025106                JMP    110$              ;GO TO 110$ IF ERROR
1146 024722 60$:
1147 024722 004737 045152      JSR    PC,PRIERR         ;GO CHECK FOR PRIMARY ERRORS
      024726 000405                BR     70$               ;GO TO 70$ IF NO ERROR
      024730 000240                NOP                                ;RETURN HERE IF ERROR
      024732 104000                EMT                                ;ERROR # DEFINED BY PRIERR SUBROUTINE
      024734 004736                JSR    PC,@(SP)+         ;GO BACK FOR MORE ERROR CHECKS
      024736 000137 025106                JMP    110$              ;GO TO 110$ IF ERROR
1148 024742 016237 067046 001140 70$:      MOV    FNCDTB(R2),%GDDAT ;GET ENTRY FROM FUNCTION CODE TABLE
1149 024750 042737 175777 001140      BIC    #^CIAE,%GDDAT
1150 024756 013737 001350 001142      MOV    RMER1I,%BDDAT     ;BAD DATA FOR TYPEOUT
    
```

```

1151 024764 042737 175777 001142 BIC #^CIAE,$BDDAT
1152 024772 023737 001140 001142 CMP $GDDAT,$BDDAT ;IS IAE STATUS CORRECT?
1153 025000 001402 BEQ 80$ ;YES!!
1154 025002 104217 EMT 217
1155 025004 000440 BR 110$
1156 025006 016237 067046 001140 80$: MOV FNCDTB(R2),$GDDAT
1157 025014 032737 040000 001346 BIT #ERR,RMSI ;WAS AN ERROR DETECTED??
1158 025022 001403 BEQ 90$ ;NO!!
1159 025024 052737 100000 001140 BIS #ATA,$GDDAT ;YES - ATA SHOULD BE ON
1160 025032 042737 077777 001140 90$: BIC #^CATA,$GDDAT
1161 025040 013737 001346 001142 MOV RMSI,$BDDAT ;GET DRIVE'S ATTENTION
1162 025046 042737 077777 001142 BIC #^CATA,$BDDAT
1163 025054 023737 001140 001142 CMP $GDDAT,$BDDAT ;IS ATA STATUS OK??
1164 025062 001402 BEQ 100$ ;YES!!
1165 025064 104255 EMT 255
1166 025066 000407 BR 110$
1167 025070 062702 000002 100$: ADD #2,R2 ;GO TO NEXT FUNCTION CODE
1168 025074 022702 000076 CMP #ILF76,R2 ;DONE??
1169 025100 103402 BLO 110$ ;YES !!
1170 025102 000137 024374 JMP 10$ ;TEST NEXT FUNCTION
1171 025106 110$:
1172
1173

```

 :*TEST 36 WRITE LOCK ERROR TEST

```

025106 000004 TST36: SCOPE ;SCOPE CALL
025110 000240 NOP ;START OF TEST
025112 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
025116 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
025122 013701 001464 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
025126 012737 000036 001226 MOV #36,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
1174
1175 025134 004737 043150 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
025140 040000 .WORD 040000 ;TASK DESCRIPTOR AS FOLLOWS:
;CLEAR CONTROLLER & SELECT DEVICE
025142 000404 BR 5$ ;GO TO 5$ IF NO ERROR
025144 000240 NOP ;RETURN HERE IF ERROR
025146 104000 EMT ;ERROR # DEFINED BY TSTPRP SUBROUTINE
025150 000137 025672 JMP 110$ ;GO TO 110$ IF ERROR
1176 025154 5$: JSR PC,GETSTS ;SETUP FOR STATUS
1177 025154 004737 044070 MOV #NOP,R2
1178 025160 012702 000000 10$: JSR PC,CNTCLR ;GO ISSUE CONTROLLER CLEAR
025164 004737 053530 BR 20$ ;GO TO 20$ IF NO ERROR
025170 000404 NOP ;RETURN HERE IF ERROR
025172 000240 EMT ;ERROR NUMBER DEFINED BY SUBROUTINE
025174 104000 JMP 110$ ;GO TO 110$ IF ERROR
025176 000137 025672 20$:
1181 025202 ;SET DIAGNOSTIC MODE AND DIAGNOSTIC UNIT READY, DIAGNOSTIC ON
1182 ;CYLINDER, AND DIAGNOSTIC WRITE LOCK
1183
1184
1185 025202 112737 000024 001543 MOVB #RMMR1,PUTINX ;SETUP PUT INDEX TABLE
025210 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
025216 012737 000001 001434 MOV #DMD,RMMR10 ;SET RMMR1 OUTPUT BUFFER = DMD
025224 004737 044424 JSR PC,PUT ;GO WRITE RMMR1 VIA PUT SUBROUTINE

```



```

025230 000404 BR 30$ ;GO TO 30$ IF NO ERROR
025232 000240 NOP ;RETURN HERE IF ERROR
025234 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
025236 000137 025672 JMP 110$ ;GO TO 110$ IF ERROR
1186 025242 30$:
1187 025242 112737 000024 001543 MOVB #RMMR1,PUTINX ;SETUP PUT INDEX TABLE
025250 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
025256 012737 001411 001434 MOV #DMD!MUR!MOC!MWP,RMMR10 ;SET RMMR1 OUTPUT BUFFER = DMD!MUR!MOC!MWP
025264 004737 044424 JSR PC,PUT ;GO WRITE RMMR1 VIA PUT SUBROUTINE
025270 000404 BR 35$ ;GO TO 35$ IF NO ERROR
025272 000240 NOP ;RETURN HERE IF ERROR
025274 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
025276 000137 025672 JMP 110$ ;GO TO 110$ IF ERROR
1188 025302 35$:
1189
1190 ;SET VOLUME VALID BEFORE ISSUING FUNCTION CODE BEING TESTED
1191 025302 112737 000000 001543 MOVB #RMCS1,PUTINX ;SETUP PUT INDEX TABLE
025310 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
025316 012737 000023 001410 MOV #PAKACK!GO,RMCS10 ;SET RMCS1 OUTPUT BUFFER = PAKACK!GO
025324 004737 044424 JSR PC,PUT ;GO WRITE RMCS1 VIA PUT SUBROUTINE
025330 000404 BR 40$ ;GO TO 40$ IF NO ERROR
025332 000240 NOP ;RETURN HERE IF ERROR
025334 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
025336 000137 025672 JMP 110$ ;GO TO 110$ IF ERROR
1192 025342 40$:
1193 025342 012737 000001 001410 MOV #GO,RMCS10
1194 025350 050237 001410 BIS R2,RMCS10 ;WRITE FUNCTION CODE IN BUFFER
1195 025354 012737 103640 001414 MOV #BUFONE,RMBAO ;DUMMY BUS ADDRESS
1196 025362 012737 177777 001412 MOV #-1,RMWC0 ;DUMMY WORD COUNT
1197 025370 005037 001416 CLR RMDAO ;CLEAR DISK ADDRESS
1198 025374 005037 001444 CLR RMDCO ;CLEAR CYLINDER ADDRESS
1199 025400 012737 010000 001442 MOV #FMT16,RMOFO ;16 BIT FORMAT
1200 025406 012703 001543 MOV #PUTINX,R3 ;WRITE REGISTER INDEX TABLE
1201 025412 112723 000004 MOVB #RMBA,(R3)+
1202 025416 112723 000002 MOVB #RMWC,(R3)+
1203 025422 112723 000032 MOVB #RMOF,(R3)+
1204 025426 112723 000006 MOVB #RMDA,(R3)+
1205 025432 112723 000034 MOVB #RMDC,(R3)+
1206 025436 112723 000000 MOVB #RMCS1,(R3)+
1207 025442 112713 000200 MOVB #200,(R3)
1208 025446 004737 044424 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
025452 000404 BR 50$ ;GO TO 50$ IF NO ERROR
025454 000240 NOP ;RETURN HERE IF ERROR
025456 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
025460 000137 025672 JMP 110$ ;GO TO 110$ IF ERROR
1209 025464 004737 044766 50$: JSR PC,TIMOUT ;WAIT FOR GO TO RESET
1210
1211 025470 004737 044154 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
025474 000404 BR 60$ ;GO TO 60$ IF NO ERROR
025476 000240 NOP ;RETURN HERE IF ERROR
025500 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
025502 000137 025672 JMP 110$ ;GO TO 110$ IF ERROR
1212 025506 60$:
1213 025506 004737 045152 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
025512 000405 BR 70$ ;GO TO 70$ IF NO ERROR
025514 000240 NOP ;RETURN HERE IF ERROR
025516 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE

```

```
025520 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
025522 000137 025672 JMP 110$ ;GO TO 110$ IF ERROR
1214 025526 016237 067046 001140 70$: MOV FNCDTB(R2),$GDDAT ;GET ENTRY FROM FUNCTION CODE TABLE
1215 025534 042737 173777 001140 BIC #^CWLE,$GDDAT
1216 025542 013737 001350 001142 MOV RMERT1,$BDDAT ;BAD DATA FOR TYPEOUT
1217 025550 042737 173777 001142 BIC #^CWLE,$BDDAT
1218 025556 023737 001140 001142 CMP $GDDAT,$BDDAT ;IS WLE STATUS CORRECT?
1219 025564 001402 BEQ 80$ ;YES!!
1220 025566 104220 EMT 220
1221 025570 000440 BR 110$
1222 025572 016237 067046 001140 80$: MOV FNCDTB(R2),$GDDAT
1223 025600 032737 040000 001346 BIT #ERR,RMDSI ;WAS AN ERROR DETECTED??
1224 025606 001403 BEQ 90$ ;NO!!
1225 025610 052737 100000 001140 BIS #ATA,$GDDAT ;YES - ATA SHOULD BE ON
1226 025616 042737 077777 001140 90$: BIC #^CATA,$GDDAT
1227 025624 013737 001346 001142 MOV RMDSI,$BDDAT ;GET DRIVE'S ATTENTION
1228 025632 042737 077777 001142 BIC #^CATA,$BDDAT
1229 025640 023737 001140 001142 CMP $GDDAT,$BDDAT ;IS ATA STATUS OK??
1230 025646 001402 BEQ 100$ ;YES!!
1231 025650 104255 EMT 255
1232 025652 000407 BR 110$
1233 025654 062702 000002 100$: ADD #2,R2 ;GO TO NEXT FUNCTION CODE
1234 025660 022702 000076 CMP #ILF76,R2 ;DONE??
1235 025664 103402 BLO 110$ ;YES !!
1236 025666 000137 025164 JMP 10$ ;TEST NEXT FUNCTION
1237 025672 110$:
1238
1239
```

```
::*****
:*TEST 37 ERROR ABORT TESTS
:*****
TST37:
```

```
025672 000004 SCOPE ;SCOPE CALL
025674 000240 NOP ;START OF TEST
025676 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
025702 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
025706 013701 001464 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
025712 012737 000037 001226 MOV #37,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
1240
1241 025720 004737 043150 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
025724 054130 .WORD 054130 ;TASK DESCRIPTOR AS FOLLOWS:
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
;VERIFY RECALIBRATION
;GO TO 10$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 80$ IF ERROR
025726 000404 BR 10$
025730 000240 NOP
025732 104000 EMT
025734 000137 026224 JMP 80$
1242 025740 10$:
1243
1244 ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
1245 025740 004737 044070 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
1246 025744 012702 000000 MOV #NOP,R2 ;R2 = FUNCTION CODE
1247 025750 004737 053530 JSR PC,CNTCLR ;GO ISSUE CONTROLLER CLEAR
```

```

025754 000404 BR 30$ ;GO TO 30$ IF NO ERROR
025756 000240 NOP ;RETURN HERE IF ERROR
025760 104000 EMT ;ERROR # DEFINED BY CNTCLR SUBROUTINE
025762 000137 026224 JMP 80$ ;GO TO 80$ IF ERROR
1248 025766 30$:
1249 025766 112737 000014 001543 MOVB #RMER1,PUTINX ;SETUP PUT INDEX TABLE
025774 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
026002 012737 040000 001424 MOV #UNS,RMER10 ;SET RMER1 OUTPUT BUFFER = UNS
026010 004737 044424 JSR PC,PUT ;GO WRITE RMER1 VIA PUT SUBROUTINE
026014 000404 BR 40$ ;GO TO 40$ IF NO ERROR
026016 000240 NOP ;RETURN HERE IF ERROR
026020 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
026022 000137 026224 JMP 80$ ;GO TO 80$ IF ERROR
1250 026026 40$:
1251 026026 012737 000001 001410 MOV #GO,RMCS10
1252 026034 050237 001410 BIS R2,RMCS10 ;WRITE FUNCTION CODE IN BUFFER
1253 026040 012737 103640 001414 MOV #BUFONE,RMBA0 ;DUMMY BUS ADDRESS
1254 026046 012737 177777 001412 MOV #-1,RMWCO ;DUMMY WORD COUNT
1255 026054 012737 000000 001416 MOV #0,RMDAO ;CLEAR DISK ADDRESS
1256 026062 012737 000000 001444 MOV #0,RMDCO ;CLEAR CYLINDER ADDRESS
1257 026070 012737 010000 001442 MOV #FMT16,RMOFO ;16 BIT FORMAT
1258 026076 012703 001543 MOV #PUTINX,R3 ;WRITE REGISTER INDEX TABLE
1259 026102 112723 000004 MOVB #RMBA,(R3)+
1260 026106 112723 000002 MOVB #RMWC,(R3)+
1261 026112 112723 000032 MOVB #RMOF,(R3)+
1262 026116 112723 000006 MOVB #RMDA,(R3)+
1263 026122 112723 000034 MOVB #RMDC,(R3)+
1264 026126 112723 000000 MOVB #RMCS1,(R3)+
1265 026132 112713 000200 MOVB #200,(R3)
1266 026136 004737 044424 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
026142 000402 BR 45$ ;GO TO 45$ IF NO ERROR
026144 000240 NOP ;RETURN HERE IF ERROR
026146 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
1267 026150 45$:
1268
1269 ;WAIT FOR COMMAND TO COMPLETE
026150 004737 044766 JSR PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
1270
1271 026154 004737 044154 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
026160 000404 BR 50$ ;GO TO 50$ IF NO ERROR
026162 000240 NOP ;RETURN HERE IF ERROR
026164 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
026166 000137 026224 JMP 80$ ;GO TO 80$ IF ERROR
1272 026172 50$:
1273 026172 004737 045152 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
026176 000405 BR 60$ ;GO TO 60$ IF NO ERROR
026200 000240 NOP ;RETURN HERE IF ERROR
026202 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
026204 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
026206 000137 026224 JMP 80$ ;GO TO 80$ IF ERROR
1274 026212 60$:
1275 026212 70$:
1276 026212 062702 000002 ADD #2,R2 ;ADVANCE FUNCTION CODE
1277 026216 022702 000076 CMP #1LF76,R2 ;DONE ALL COMMANDS
1278 026222 103252 BHIS 20$ ;NO !!
1279 026224 80$:
1280

```

1281

```

*****
: *TEST 40      RMR TEST
*****
TST40:

```

026224	000004			SCOPE	:SCOPE CALL
026224	000240			NCP	:START OF TEST
026226	012706	001100		MOV #STACK,SP	:INITIALIZE STACK POINTER
026234	013700	001276		MOV \$BASE,R0	:R0 = UNIBUS ADDRESS
026240	013701	001464		MOV TSTQUE,R1	:(R1) = DEVICE BEING TESTED
026244	012737	000040	001226	MOV #40,\$TESTN	::SET TEST NUMBER IN APT MAIL BOX
1282					
1283	026252	004737	043150	JSR PC,TSTPRP	:PREPARE DEVICE FOR TEST
	026256	040000		.WORD 040000	:TASK DESCRIPTOR AS FOLLOWS:
					:CLEAR CONTROLLER & SELECT DEVICE
	026260	000404		BR 10\$:GO TO 10\$ IF NO ERROR
	026262	000240		NOP	:RETURN HERE IF ERROR
	026264	104000		EMT	:ERROR # DEFINED BY TSTPRP SUBROUTINE
	026266	000137	026664	JMP 100\$:GO TO 100\$ IF ERROR
1284	026272			10\$:	
1285					
1286				:SETUP GET INDEX TABLE TO READ ALL REGISTERS	
	026272	004737	044070	JSR PC,GETSTS	:GO TO GETSTS SUBROUTINE
1287	026276			20\$:	
1288	026276	112737	000024 001543	MOVB #RMMR1,PUTINX	:SETUP PUT INDEX TABLE
	026304	112737	000200 001544	MOVB #200,PUTINX+1	:SET TERMINATOR BYTE
	026312	012737	000001 001434	MOV #DMD,RMMR10	:SET RMMR1 OUTPUT BUFFER = DMD
	026320	004737	044424	JSR PC,PUT	:GO WRITE RMMR1 VIA PUT SUBROUTINE
	026324	000404		BR 30\$:GO TO 30\$ IF NO ERROR
	026326	000240		NOP	:RETURN HERE IF ERROR
	026330	104000		EMT	:ERROR DEFINED BY PUT SUBROUTINE
	026332	000137	026664	JMP 100\$:GO TO 100\$ IF ERROR
1289	026336			30\$:	
1290	026336	112737	000024 001543	MOVB #RMMR1,PUTINX	:SETUP PUT INDEX TABLE
	026344	112737	000200 001544	MOVB #200,PUTINX+1	:SET TERMINATOR BYTE
	026352	012737	001401 001434	MOV #DMD!MUR!MOC,RMMR10	:SET RMMR1 OUTPUT BUFFER = DMD!MUR!MOC
	026360	004737	044424	JSR PC,PUT	:GO WRITE RMMR1 VIA PUT SUBROUTINE
	026364	000404		BR 40\$:GO TO 40\$ IF NO ERROR
	026366	000240		NOP	:RETURN HERE IF ERROR
	026370	104000		EMT	:ERROR DEFINED BY PUT SUBROUTINE
	026372	000137	026664	JMP 100\$:GO TO 100\$ IF ERROR
1291	026376			40\$:	
1292	026376	112737	000000 001543	MOVB #RMCS1,PUTINX	:SETUP PUT INDEX TABLE
	026404	112737	000200 001544	MOVB #200,PUTINX+1	:SET TERMINATOR BYTE
	026412	012737	000023 001410	MOV #PAKACK!GO,RMCS10	:SET RMCS1 OUTPUT BUFFER = PAKACK!GO
	026420	004737	044424	JSR PC,PUT	:GO WRITE RMCS1 VIA PUT SUBROUTINE
	026424	000404		BR 50\$:GO TO 50\$ IF NO ERROR
	026426	000240		NOP	:RETURN HERE IF ERROR
	026430	104000		EMT	:ERROR DEFINED BY PUT SUBROUTINE
	026432	000137	026664	JMP 100\$:GO TO 100\$ IF ERROR
1293	026436			50\$:	
1294	026436	004737	044154	JSR PC,GET	:GO READ REGISTER(S) WITH GET SUBROUTINE
	026442	000404		BR 60\$:GO TO 60\$ IF NO ERROR
	026444	000240		NOP	:RETURN HERE IF ERROR
	026446	104000		EMT	:ERROR # DEFINED BY GET SUBROUTINE
	026450	000137	026664	JMP 100\$:GO TO 100\$ IF ERROR
1295	026454			60\$:	
1296	026454	000240		NOP	

```
1297 026456 112737 000024 001543      MOVB   #RMMR1,PUTINX      ;SETUP PUT INDEX TABLE
      026464 112737 000200 001544      MOVB   #200,PUTINX+1     ;SET TERMINATOR BYTE
      026472 012737 041401 001434      MOV    #DMD!MUR!MOC!DBEN,RMMR10 ;SET RMMR1 OUTPUT BUFFER = DMD!MUR!MOC!DBEN
      026500 004737 044424      JSR    PC,PUT            ;GO WRITE RMMR1 VIA PUT SUBROUTINE
      026504 000404      BR     70$              ;GO TO 70$ IF NO ERROR
      026506 000240      NOP                    ;RETURN HERE IF ERROR
      026510 104000      EMT                    ;ERROR DEFINED BY PUT SUBROUTINE
      026512 000137 026664      JMP    100$            ;GO TO 100$ IF ERROR
1298 026516 112737 000000 001543      MOVB   #RMCS1,PUTINX     ;SETUP PUT INDEX TABLE
1299 026524 112737 000200 001544      MOVB   #200,PUTINX+1     ;SET TERMINATOR BYTE
      026532 012737 000001 001410      MOV    #NOP!GO,RMCS10    ;SET RMCS1 OUTPUT BUFFER = NOP!GO
      026540 004737 044424      JSR    PC,PUT            ;GO WRITE RMCS1 VIA PUT SUBROUTINE
      026544 000404      BR     80$              ;GO TO 80$ IF NO ERROR
      026546 000240      NOP                    ;RETURN HERE IF ERROR
      026550 104000      EMT                    ;ERROR DEFINED BY PUT SUBROUTINE
      026552 000137 026664      JMP    100$            ;GO TO 100$ IF ERROR
1300 026556 112737 000000 001543      MOVB   #RMCS1,PUTINX     ;SETUP PUT INDEX TABLE
1301 026564 112737 000200 001544      MOVB   #200,PUTINX+1     ;SET TERMINATOR BYTE
      026572 012737 000015 001410      MOV    #OFFSET!GO,RMCS10 ;SET RMCS1 OUTPUT BUFFER = OFFSET!GO
      026600 004737 044424      JSR    PC,PUT            ;GO WRITE RMCS1 VIA PUT SUBROUTINE
      026604 000404      BR     85$              ;GO TO 85$ IF NO ERROR
      026606 000240      NOP                    ;RETURN HERE IF ERROR
      026610 104000      EMT                    ;ERROR DEFINED BY PUT SUBROUTINE
      026612 000137 026664      JMP    100$            ;GO TO 100$ IF ERROR
1302 026616 004737 044154      JSR    PC,GET            ;GO READ REGISTER(S) WITH GET SUBROUTINE
1303 026622 000404      BR     90$              ;GO TO 90$ IF NO ERROR
      026624 000240      NOP                    ;RETURN HERE IF ERROR
      026626 104000      EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      026630 000137 026664      JMP    100$            ;GO TO 100$ IF ERROR
1304 026634 000240      NOP                    ;
1305 026636 032737 000004 001350      BIT    #RMR,RMER11       ;IS RMR SET ??
1307 026644 001007      BNE    100$             ;YES !!
1308 026646 012737 000004 001140      MOV    #RMR,$GDDAT       ;EXPECTED STATUS
1309 026654 013737 001350 001142      MOV    RMER11,$BDDAT     ;RECEIVED STATUS
1310 026662 104222      EMT                    222
1311 026664
1312
1313
```

```
:::*****
:*TEST 41      PARITY ERROR TEST
:::*****
```

```
TST41:
026664      SCOPE                ;SCOPE CALL
026664 000004      NOP                    ;START OF TEST
026666 000240      MOV    #STACK,SP       ;INITIALIZE STACK POINTER
026670 012706 001100      MOV    $BASE,R0         ;R0 = UNIBUS ADDRESS
026674 013700 001276      MOV    TSTQUE,R1        ;(R1) = DEVICE BEING TESTED
026700 013701 001464      MOV    #41,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
026704 012737 000041 001226
1314
1315 026712 004737 043150      JSR    PC,TSTPRP        ;PREPARE DEVICE FOR TEST
      026716 040000      .WORD 040000          ;TASK DESCRIPTOR AS FOLLOWS:
                                ;CLEAR CONTROLLER & SELECT DEVICE
      026720 000404      BR     10$             ;GO TO 10$ IF NO ERROR
      026722 000240      NOP                    ;RETURN HERE IF ERROR
```

```

026724 104000          EMT          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
026726 000137 027104   JMP          70$          ;GO TO 70$ IF ERROR
1316 026732          10$:
1317 026732 012702 000001   MOV          #1,R2        ;R2 = DATA PATTERN
1318 026736          20$:
1319 026736 004737 053530   JSR          PC,CNTCLR    ;GO ISSUE CONTROLLER CLEAR
026742 000404          BR          30$          ;GO TO 30$ IF NO ERROR
026744 000240          NOP          ;RETURN HERE IF ERROR
026746 104000          EMT          ;ERROR # DEFINED BY CNTCLR SUBROUTINE
026750 000137 027104   JMP          70$          ;GO TO 70$ IF ERROR
1320 026754          30$:
1321 026754 111103          MOVB         (R1),R3      ;SETUP RMCS2
1322 026756 042703 177770   BIC          #^CUNTMSK,R3
1323 026762 052703 000020   BIS          #PAT,R3
1324 026766 010337 001420   MOV          R3,RMCS20   ;OUTPUT VALUE TO RMCS2
1325 026772 010237 001416   MOV          R2,RMDAO    ;VALUE TO RMDA
1326 026776 012703 001543   MOV          #PUTINX,R3  ;WRITE REGISTER OUTPUT INDEX
1327 027002 112723 000010   MOVB         #RMCS2,(R3)+
1328 027006 112723 000006   MOVB         #RMDA,(R3)+
1329 027012 112723 000200   MOVB         #200,(R3)+
1330
1331 027016 004737 044424   JSR          PC,PUT      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
027022 000402          BR          40$          ;GO TO 40$ IF NO ERROR
027024 000240          NOP          ;RETURN HERE IF ERROR
027026 104000          EMT          ;ERROR # DEFINED BY PUT SUBROUTINE
1332 027030          40$:
1333 027030 004737 044154   JSR          PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
027034 000404          BR          50$          ;GO TO 50$ IF NO ERROR
027036 000240          NOP          ;RETURN HERE IF ERROR
027040 104000          EMT          ;ERROR # DEFINED BY GET SUBROUTINE
027042 000137 027104   JMP          70$          ;GO TO 70$ IF ERROR
1334 027046          50$:
1335 027046 032737 000010 001350   BIT          #PAR,RMER11 ;IS PARITY ERROR SET ??
1336 027054 001011          BNE         60$          ;YES !!
1337 027056 012737 000010 001140   MOV          #PAR,$GDDAT ;EXPECTED STATUS
1338 027064 013737 001350 001142   MOV          RMER11,$BDDAT ;RECEIVED STATUS
1339 027072 010237 001174          MOV          R2,$TMPO    ;DATA PATTERN
1340 027076 104223          EMT          223
1341 027100          60$:
1342 027100 006302          ASL          R2          ;ADVANCE DATA PATTERN
1343 027102 001315          BNE         70$          ;BRANCH IF NOT DONE
1344 027104          70$:
1345
1346

```

```

:*****
:*TEST 42      ILLEGAL REGISTER TEST
:*****
TST42:

```

```

027104
027104 000004          SCOPE          ;SCOPE CALL
027106 000240          NOP          ;START OF TEST
027110 012706 001100          MOV          #STACK,SP  ;INITIALIZE STACK POINTER
027114 013700 001276          MOV          $BASE,R0   ;R0 = UNIBUS ADDRESS
027120 013701 001464          MOV          TSTQUE,R1  ;(R1) = DEVICE BEING TESTED
027124 012737 000042 001226   MOV          #42,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
1347
1348 027132 004737 043150   JSR          PC,TSTPRP  ;PREPARE DEVICE FOR TEST
027136 040000          .WORD       040000   ;TASK DESCRIPTOR AS FOLLOWS:
;CLEAR CONTROLLER & SELECT DEVICE

```

```

027140 000404 BR 10$ ;GO TO 10$ IF NO ERROR
027142 000240 NOP ;RETURN HERE IF ERROR
027144 104000 EMT ;ERROR # DEFINED BY TSTPRP SUBROUTINE
027146 000137 027432 JMP 120$ ;GO TO 120$ IF ERROR
1349 027152 10$:
1350 027152 005005 CLR R5 ;R5 = EXPECTED STATUS
1351 027154 004737 044070 JSR PC,GETSTS ;SETUP FOR STATUS
1352 027160 013746 000004 MOV ERRVEC,-(SP) ;;PUSH ERRVEC ON STACK
1353 027164 013746 000006 MOV ERRVEC+2,-(SP) ;;PUSH ERRVEC+2 ON STACK
1354 027170 012737 027430 000004 MOV #110$,ERRVEC ;SETUP FOR BUS TIMEOUT
1355 027176 012737 000300 000006 MOV #PR6,ERRVEC+2
1356 027204 005002 CLR R2 ;R2 = REGISTER INDEX
1357 027206 20$:
1358 027206 004737 053530 JSR PC,CNTCLR ;GO ISSUE CONTROLLER CLEAR
027212 000404 BR 30$ ;GO TO 30$ IF NO ERROR
027214 000240 NOP ;RETURN HERE IF ERROR
027216 104000 EMT ;ERROR NUMBER DEFINED BY SUBROUTINE
027220 000137 027344 JMP 70$ ;GO TO 70$ IF ERROR
1359 027224 010003 30$: MOV R0,R3 ;R3 = REGISTER ADDRESS
1360 027226 060203 ADD R2,R3
1361 027230 005013 CLR (R3) ;CLEAR THE REGISTER
1362 027232 004737 052056 JSR PC,DEVSEL ;GO SELECT DEVICE
027236 000404 BR 35$ ;GO TO 35$ IF NO ERROR
027240 000240 NOP ;RETURN HERE IF ERROR
027242 104000 EMT ;ERROR # DEFINED BY DEVSEL SUBROUTINE
027244 000137 027432 JMP 120$ ;GO TO 120$ IF ERROR
1363 027250 35$:
1364 027250 004737 044154 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
027254 000404 BR 40$ ;GO TO 40$ IF NO ERROR
027256 000240 NOP ;RETURN HERE IF ERROR
027260 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
027262 000137 027344 JMP 70$ ;GO TO 70$ IF ERROR
1365 027266 40$:
1366 027266 004737 045152 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
027272 000405 BR 50$ ;GO TO 50$ IF NO ERROR
027274 000240 NOP ;RETURN HERE IF ERROR
027276 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
027300 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
027302 000137 027344 JMP 70$ ;GO TO 70$ IF ERROR
1367 027306 010537 001140 50$: MOV R5,$GDDAT
1368 027312 013737 001350 001142 MOV RMER11,$BDDAT ;GET DRIVE'S ILR STATUS
1369 027320 042737 177775 001142 BIC #^CILR,$BDDAT
1370 027326 023737 001140 001142 CMP $GDDAT,$BDDAT ;IS ILR STATUS OK??
1371 027334 001403 BEQ 70$ ;YES!!
1372 027336 010237 001174 MOV R2,$TMPO ;SAVE R2 FOR ERROR DATA
1373 027342 104256 EMT 256
1374 :70$: ADD #2,R2
1375 : CMP #RMBAE,R2 ;IS NEXT REGISTER RMBAE??
1376 : BNE 80$ ;NO!!
1377 : MOV #A16!A17,RMCS1(R0) ;SET A16 & A17
1378 : MOV RMBAE(R0),R3 ;IS THIS AN RH70??
1379 : BIC #^C<BIT0!BIT1>,R3 ;CLEAR ALL BUT ADDRESS BITS
1380 : CMP #BIT0!BIT1,R3 ;ARE ADDRESS BITS ON ??
1381 : BEQ 100$ ;YES
1382 : MOV #ILR,R5 ;NO - EXPECT ILR = 1
1383 : BR 100$
1384 :80$: CMP #RMCS3+2,R2 ;SHOULD ILR BE SET??

```

```

1385      :      BNE      90$      :NO!!
1386      :      MOV      #ILR,R5   :YES!!
1387      :      BR       100$
1388      :90$:    CMP      #76,R2   :DONE??
1389      :      BLO      120$      :YES!!
1390      :100$:   BR       20$
1391      :THE FOLLOWING CODING FOR HANDLE RH70
1392      :REGISTER NUMBERS IS EITHER 22 OR 32
1393 027344 062702 000002 70$:    ADD      #2,R2      :INCREMENT THE REGISTER ADDRESS
1394 027350 022702 000050      CMP      #50,R2      :TIME TO CHECK THE EXTEND ADDRESS REG ?
1395 027354 101314      BHI      20$          :BRANCH IF NOT
1396 027356 103420      BLO      80$          :BRANCH IF ALREADY SET UP
1397 027360 012705 000002      MOV      #ILR,R5     :EXCEPT ILR HAPPEND
1398 027364 012760 001400 000000      MOV      #A16!A17,RMCS1(R0) :SET EXTENDED ADDRESS BITS
1399 027372 012702 000054      MOV      #54,R2      :SET ADDRESS TO 54,IF 22 REGISTERS
1400 027376 016003 000050      MOV      50(R0),R3   :IS THIS 22 REG RH70 ?
1401 027402 042703 177774      BIC      #177774,R3  :LEFT ONLY BIT 0 AND BIT 1
1402 027406 022703 000003      CMP      #BIT0!BIT1,R3 :BIT 0 AND BIT 1 SET AT THE SAME TIME ?
1403 027412 001402      BEQ      80$        :BRANCH IF SO
1404 027414 012702 000050      MOV      #50,R2     :OTHERWISE,SET ADDRESS TO 50
1405 027420 022702 000074 80$:    CMP      #74,R2   :ALL REGISTERS CHECKED ?
1406 027424 101402      BLOS    120$        :BRANCH IF SO
1407 027426 000667      BR       20$        :NEXT REGISTER
1408
1409 027430 022626 110$:    CMP      (SP)+,(SP)+
1410 027432 120$:    MOV      (SP)+,ERRVEC+2  ;;POP STACK INTO ERRVEC+2
1411 027432 012637 000006      MOV      (SP)+,ERRVEC  ;;POP STACK INTO ERRVEC
1411 027436 012637 000004
1412
1413

```

```

:*****
:*TEST 43      SEEK LAST CYLINDER
:*****
TST43:

```

```

027442      :SCOPE      :SCOPE CALL
027442 000004      NOP          :START OF TEST
027444 000240      MOV      #STACK,SP  :INITIALIZE STACK POINTER
027446 012706 001100      MOV      $BASE,R0   :R0 = UNIBUS ADDRESS
027452 013700 001276      MOV      TSTQUE,R1  :(R1) = DEVICE BEING TESTED
027456 013701 001464      MOV      #1,$TIMES  ;;DO 1 ITERATION
027462 012737 000001 001206      MOV      #43,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
027470 012737 000043 001226
1414
1415 027476 10$:    JSR      PC,TSTPRP  :PREPARE DEVICE FOR TEST
1416 027476 004737 043150      .WORD   054130    :TASK DESCRIPTOR AS FOLLOWS:
                                :CLEAR CONTROLLER & SELECT DEVICE
                                :VERIFY CONTROLLER CLEAR OPERATION
                                :PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                :VERIFY PACK ACKNOWLEDGE
                                :RECALIBRATE IF "SKI" OR "PIP" IS SET
                                :VERIFY RECALIBRATION
                                :GO TO 80$ IF NO ERROR
                                :RETURN HERE IF ERROR
                                :ERROR # DEFINED BY TSTPRP SUBROUTINE
                                :GO TO 140$ IF ERROR
                                :LAST CYLINDER
                                :TRACK = 0, SECTOR = 0
027504 000404      BR       80$
027506 000240      NOP
027510 104000      EMT
027512 000137 027730      JMP      140$
1417 027516 80$:    MOV      #822,RMDCO
1418 027516 012737 001466 001444      MOV      #0,RMDAO
1419 027524 012737 000000 001416

```



```

1420 027532 012737 000005 001410      MOV      #SEEK!GO, RMCS10 ;LOAD SEEK COMMAND IN BUFFER,
1421 027540 012702 001543      MOV      #PUTINX, R2    ;R2 POINTS TO REGISTER TABLE
1422 027544 112722 000034      MOVVB   #RMDC, (R2)+    ;WRITE REGISTER INDEX TABLE
1423 027550 112722 000006      MOVVB   #RMDA, (R2)+
1424 027554 112722 000000      MOVVB   #RMCS1, (R2)+
1425 027560 112722 000200      MOVVB   #200, (R2)+    ;WRITE TERMINATOR
1426 027564 004737 044424      JSR     PC, PUT        ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      027570 000404      BR      90$          ;GO TO 90$ IF NO ERROR
      027572 000240      NOP
      027574 104000      EMT     ;RETURN HERE IF ERROR
      027576 000137 027730      JMP     140$        ;ERROR # DEFINED BY PUT SUBROUTINE
1427 027602 004737 044070      JSR     PC, GETSTS    ;GO TO 140$ IF ERROR
1428 027606 004737 044766      JSR     PC, TIMEOUT   ;SETUP FOR STATUS FETCH
      1429
1430 027612 004737 044154      JSR     PC, GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
      027616 000404      BR      100$       ;GO TO 100$ IF NO ERROR
      027620 000240      NOP
      027622 104000      EMT     ;RETURN HERE IF ERROR
      027624 000137 027730      JMP     140$        ;ERROR # DEFINED BY GET SUBROUTINE
1431 027630
1432 027630 004737 045152      JSR     PC, PRIERR   ;GO CHECK FOR PRIMARY ERRORS
      027634 000405      BR      110$       ;GO TO 110$ IF NO ERROR
      027636 000240      NOP
      027640 104000      EMT     ;RETURN HERE IF ERROR
      027642 004736      JSR     PC, @ (SP)+  ;ERROR # DEFINED BY PRIERR SUBROUTINE
      027644 000137 027730      JMP     140$        ;GO BACK FOR MORE ERROR CHECKS
1433 027650
1434 027650 004737 052270      JSR     PC, SEKSTS   ;GO CHECK FOR SECONDARY ERRORS
      027654 000405      BR      120$       ;GO TO 120$ IF NO ERROR
      027656 000240      NOP
      027660 104000      EMT     ;RETURN HERE IF ERROR
      027662 004736      JSR     PC, @ (SP)+  ;ERROR # DEFINED BY SEKSTS SUBROUTINE
      027664 000137 027730      JMP     140$        ;GO BACK FOR MORE ERROR CHECKS
1435 027670
1436 027670 004737 051644      JSR     PC, CMPERRSTS ;CHECK ANY ERRORS NOT MASKED
      027674 115760      .WORD  NDTMSK      ;MASK FOR RMER1
      027676 000010      .WORD  DPE         ;MASK FOR RMER2
      027700 000405      BR      130$       ;GO TO 130$ IF NO ERROR
      027702 000240      NOP
      027704 104000      EMT     ;RETURN HERE IF ERROR
      027706 004736      JSR     PC, @ (SP)+  ;ERROR # DEFINED BY CMPERRSTS SUBROUTINE
      027710 000137 027730      JMP     140$        ;GO BACK FOR MORE ERROR CHECKS
1437 027714
1438 027714 004737 046004      JSR     PC, SECERR   ;GO CHECK FOR SECONDARY ERRORS
      027720 000403      BR      140$       ;GO TO 140$ IF NO ERROR
      027722 000240      NOP
      027724 104000      EMT     ;RETURN HERE IF ERROR
      027726 004736      JSR     PC, @ (SP)+  ;ERROR # DEFINED BY SECERR SUBROUTINE
1439 027730
1440
1441

```

```

*****
;*TEST 44      SEEK FIRST CYLINDER
*****
TST44:
      SCOPE      ;SCOPE CALL
      NOP
      MOV      #STACK, SP ;INITIALIZE STACK POINTER

```

	027740	013700	001276		MOV	\$BASE,R0	:R0 = UNIBUS ADDRESS
	027744	013701	001464		MOV	TSTQUE,R1	:(R1) = DEVICE BEING TESTED
	027750	012737	000001	001206	MOV	#1,\$TIMES	::DO 1 ITERATION
	027756	012737	000044	001226	MOV	#44,\$TESTN	::SET TEST NUMBER IN APT MAIL BOX
1442							
1443	027764						
1444	027764	004737	043150	10\$:	JSR	PC,TSTPRP	:PREPARE DEVICE FOR TEST
	027770	054130			.WORD	054130	:TASK DESCRIPTOR AS FOLLOWS:
							:CLEAR CONTROLLER & SELECT DEVICE
							:VERIFY CONTROLLER CLEAR OPERATION
							:PACK ACKNOWLEDGE IF VOLUME NOT VALID
							:VERIFY PACK ACKNOWLEDGE
							:RECALIBRATE IF "SKI" OR "PIP" IS SET
							:VERIFY RECALIBRATION
							:GO TO 80\$ IF NO ERROR
							:RETURN HERE IF ERROR
							:ERROR # DEFINED BY TSTPRP SUBROUTINE
							:GO TO 140\$ IF ERROR
	027772	000404			BR	80\$	
	027774	000240			NOP		
	027776	104000			EMT		
	030000	000137	030216		JMP	140\$	
1445	030004			80\$:			
1446	030004	012737	000000	001444	MOV	#0.,RMDCO	:CYLINDER = 0
1447	030012	012737	000000	001416	MOV	#0,RMDAO	:TRACK = 0, SECTOR = 0
1448	030020	012737	000005	001410	MOV	#SEEK!GO,RMCS10	:LOAD SEEK COMMAND IN BUFFER
1449	030026	012702	001543		MOV	#PUTINX,R2	:R2 POINTS TO REGISTER TABLE
1450	030032	112722	000034		MOVB	#RMDC,(R2)+	:WRITE REGISTER INDEX TABLE
1451	030036	112722	000006		MOVB	#RMDA,(R2)+	
1452	030042	112722	000000		MOVB	#RMCS1,(R2)+	
1453	030046	112722	000200		MOVB	#200,(R2)+	:WRITE TERMINATOR
1454	030052	004737	044424		JSR	PC,PUT	:GO WRITE REGISTER(S) WITH PUT SUBROUTINE
	030056	000404			BR	90\$:GO TO 90\$ IF NO ERROR
	030060	000240			NOP		:RETURN HERE IF ERROR
	030062	104000			EMT		:ERROR # DEFINED BY PUT SUBROUTINE
	030064	000137	030216		JMP	140\$:GO TO 140\$ IF ERROR
1455	030070	004737	044070	90\$:	JSR	PC,GETSTS	:SETUP FOR STATUS FETCH
1456	030074	004737	044766		JSR	PC,TIMOUT	:WAIT FOR SEEK TO COMPLETE
1457							
1458	030100	004737	044154		JSR	PC,GET	:GO READ REGISTER(S) WITH GET SUBROUTINE
	030104	000404			BR	100\$:GO TO 100\$ IF NO ERROR
	030106	000240			NOP		:RETURN HERE IF ERROR
	030110	104000			EMT		:ERROR # DEFINED BY GET SUBROUTINE
	030112	000137	030216		JMP	140\$:GO TO 140\$ IF ERROR
1459	030116			100\$:			
1460	030116	004737	045152		JSR	PC,PRIERR	:GO CHECK FOR PRIMARY ERRORS
	030122	000405			BR	110\$:GO TO 110\$ IF NO ERROR
	030124	000240			NOP		:RETURN HERE IF ERROR
	030126	104000			EMT		:ERROR # DEFINED BY PRIERR SUBROUTINE
	030130	004736			JSR	PC,@(SP)+	:GO BACK FOR MORE ERROR CHECKS
	030132	000137	030216		JMP	140\$:GO TO 140\$ IF ERROR
1461	030136			110\$:			
1462	030136	004737	052270		JSR	PC,SEKSTS	:GO VERIFY RESULTS OF SEEK OPERATION
	030142	000405			BR	120\$:GO TO 120\$ IF NO ERROR
	030144	000240			NOP		:RETURN HERE IF ERROR
	030146	104000			EMT		:ERROR # DEFINED BY SEKSTS SUBROUTINE
	030150	004736			JSR	PC,@(SP)+	:GO BACK FOR MORE ERROR CHECKS
	030152	000137	030216		JMP	140\$:GO TO 140\$ IF ERROR
1463	030156			120\$:			
1464	030156	004737	051644		JSR	PC,CMPERRSTS	:CHECK ANY ERRORS NOT MASKED
	030162	115760			.WORD	NDIMSK	:MASK FOR RMER!

```
030164 000010 .WORD DPE ;MASK FOR RMER2
030166 000405 BR 130$ ;GO TO 130$ IF NO ERROR
030170 000240 NOP ;RETURN HERE IF ERROR
030172 104000 EMT ;ERROR # DEFINED BY CMPERRSTS SUBROUTINE
030174 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
030176 000137 030216 JMP 140$ ;GO TO 140$ IF ERROR
1465 030202 130$:
1466 030202 004737 046004 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
030206 000403 BR 140$ ;GO TO 140$ IF NO ERROR
030210 000240 NOP ;RETURN HERE IF ERROR
030212 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
030214 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
1467 030216 140$:
1468
1469
```

```
::*****
:*TEST 45 SEEK PRIME CYLINDERS
:*****
TST45:
```

```
030216 000004 SCOPE ;SCOPE CALL
030220 000240 NOP ;START OF TEST
030222 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
030226 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
030232 013701 001464 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
030236 012737 000002 001206 MOV #2,$TIMES ;DO 2 ITERATIONS
030244 012737 000045 001226 MOV #45,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
1470
1471 030252 012737 000001 001444 MOV #1,RMDCO ;FIRST CYLINDER = 1
1472 030260 10$:
1473 030260 004737 043150 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
030264 054130 .WORD 054130 ;TASK DESCRIPTOR AS FOLLOWS:
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF "SKI" OR "PIP" IS SET
;VERIFY RECALIBRATION
030266 000404 BR 80$ ;GO TO 80$ IF NO ERROR
030270 000240 NOP ;RETURN HERE IF ERROR
030272 104000 EMT ;ERROR # DEFINED BY TSTPRP SUBROUTINE
030274 000137 030530 JMP 150$ ;GO TO 150$ IF ERROR
1474 030300 80$:
1475 030300 012737 000000 001416 MOV #0,RMDAO ;TRACK = 0, SECTOR = 0
1476 030306 012737 000005 001410 MOV #SEEK!GO,RMCS10 ;LOAD SEEK COMMAND IN BUFFER
1477 030314 012702 001543 MOV #PUTINX,R2 ;R2 POINTS TO REGISTER TABLE
1478 030320 112722 000034 MOVB #RMDC,(R2)+ ;WRITE REGISTER INDEX TABLE
1479 030324 112722 000006 MOVB #RMDA,(R2)+
1480 030330 112722 000000 MOVB #RMCS1,(R2)+
1481 030334 112722 000200 MOVB #200,(R2)+ ;WRITE TERMINATOR
1482 030340 004737 044424 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
030344 000404 BR 90$ ;GO TO 90$ IF NO ERROR
030346 000240 NOP ;RETURN HERE IF ERROR
030350 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
030352 000137 030530 JMP 150$ ;GO TO 150$ IF ERROR
1483 030356 90$:
1484 030356 004737 044070 JSR PC,GETSTS ;SETUP FOR STATUS FETCH
1485 030362 004737 044766 JSR PC,TIMOUT ;WAIT FOR SEEK TO COMPLETE
1486
```

```

1487 030366 004737 044154      JSR    PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      030372 000404          BR     100$           ;GO TO 100$ IF NO ERROR
      030374 000240          NOP                    ;RETURN HERE IF ERROR
      030376 104000          EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      030400 000137 030530      JMP    150$           ;GO TO 150$ IF ERROR
1488 030404          100$:
1489 030404 004737 045152      JSR    PC,PRIERR       ;GO CHECK FOR PRIMARY ERRORS
      030410 000405          BR     110$           ;GO TO 110$ IF NO ERROR
      030412 000240          NOP                    ;RETURN HERE IF ERROR
      030414 104000          EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      030416 004736          JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      030420 000137 030530      JMP    150$           ;GO TO 150$ IF ERROR
1490 030424          110$:
1491 030424 004737 052270      JSR    PC,SEKSTS       ;GO VERIFY RESULTS OF SEEK OPERATION
      030430 000405          BR     120$           ;GO TO 120$ IF NO ERROR
      030432 000240          NOP                    ;RETURN HERE IF ERROR
      030434 104000          EMT                    ;ERROR # DEFINED BY SEKSTS SUBROUTINE
      030436 004736          JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      030440 000137 030530      JMP    150$           ;GO TO 150$ IF ERROR
1492 030444          120$:
1493 030444 004737 051644      JSR    PC,CMPEERRSTS   ;CHECK ANY ERRORS NOT MASKED
      030450 115760          .WORD  ND1MSK        ;MASK FOR RMER1
      030452 000010          .WORD  DPE           ;MASK FOR RMER2
      030454 000405          BR     130$           ;GO TO 130$ IF NO ERROR
      030456 000240          NOP                    ;RETURN HERE IF ERROR
      030460 104000          EMT                    ;ERROR # DEFINED BY CMPEERRSTS SUBROUTINE
      030462 004736          JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      030464 000137 030530      JMP    150$           ;GO TO 150$ IF ERROR
1494 030470          130$:
1495 030470 004737 046004      JSR    PC,SECERR       ;GO CHECK FOR SECONDARY ERRORS
      030474 000405          BR     140$           ;GO TO 140$ IF NO ERROR
      030476 000240          NOP                    ;RETURN HERE IF ERROR
      030500 104000          EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      030502 004736          JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      030504 000137 030530      JMP    150$           ;GO TO 150$ IF ERROR
1496 030510          140$:
1497 030510 006337 001444      ASL    RMDCO           ;SHIFT TO NEXT PRIME CYLINDER
1498 030514 022737 001000 001444  CMP    #512.,RMDCO     ;IS THE TEST DONE??
1499 030522 103402          BLO    150$           ;YES!!
1500 030524 000137 030260      JMP    10$            ;GO DO NEXT CYLINDER
1501 030530          150$:
1502
1503

```

```

*****
;*TEST 46      SEEK ZERO DIFFERENCE
*****
TST46:

```

```

030530          SCOPE          ;SCOPE CALL
030530 000004          NOP                    ;START OF TEST
030532 000240          MOV     #STACK,SP      ;INITIALIZE STACK POINTER
030534 012706 001100          MOV     $BASE,R0       ;R0 = UNIBUS ADDRESS
030540 013700 001276          MOV     TSTQUE,R1      ;(R1) = DEVICE BEING TESTED
030544 013701 001464          MOV     #1,$TIMES     ;;DO 1 ITERATION
030550 012737 000001 001206  MOV     #46,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX
030556 012737 000046 001226
1504
1505 030564          10$:
1506 030564 004737 043150      JSR    PC,TSTPRP       ;PREPARE DEVICE FOR TEST
      030570 054130          .WORD  054130        ;TASK DESCRIPTOR AS FOLLOWS:

```

```

030572 000404 BR 80$ ;CLEAR CONTROLLER & SELECT DEVICE
030574 000240 NOP ;VERIFY CONTROLLER CLEAR OPERATION
030576 104000 EMT ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
030600 000137 031036 JMP 170$ ;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
;VERIFY RECALIBRATION
;GO TO 80$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 170$ IF ERROR
1507 030604 80$:
1508 030604 012703 000003 MOV #3,R3 ;R3 = NUMBER OF SEEKS
1509 030610 012737 000000 001444 MOV #0,RMDCO ;CYLINDER = 0
1510 030616 012737 000000 001416 MOV #0,RMDAO ;TRACK = 0, SECTOR = 0
1511 030624 012737 000005 001410 MOV #SEEK!GO,RMCS10 ;FUNCTION CODE FOR SEEK
1512 030632 012702 001543 MOV #PUTINX,R2 ;R2 POINTS TO REGISTER INDEX
1513 030636 112722 000006 MOVB #RMDA,(R2)+ ;WRITE REGISTER INDEX TABLE
1514 030642 112722 000034 MOVB #RMDC,(R2)+
1515 030646 112722 000000 MOVB #RMCS1,(R2)+
1516 030652 112722 000200 MOVB #200,(R2)+ ;TERMINATE TABLE
1517 030656 90$:
1518 030656 004737 044424 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
030662 000404 BR 100$ ;GO TO 100$ IF NO ERROR
030664 000240 NOP ;RETURN HERE IF ERROR
030666 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
030670 000137 031036 JMP 170$ ;GO TO 170$ IF ERROR
1519 030674 004737 044070 100$: JSR PC,GETSTS ;SETUP FOR READING STATUS
1520 030700 004737 044766 JSR PC,TIMOUT ;WAIT FOR COMPLETION
1521
1522 030704 004737 044154 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
030710 000404 BR 120$ ;GO TO 120$ IF NO ERROR
030712 000240 NOP ;RETURN HERE IF ERROR
030714 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
030716 000137 031036 JMP 170$ ;GO TO 170$ IF ERROR
1523 030722 120$:
1524 030722 004737 045152 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
030726 000405 BR 130$ ;GO TO 130$ IF NO ERROR
030730 000240 NOP ;RETURN HERE IF ERROR
030732 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
030734 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
030736 000137 031036 JMP 170$ ;GO TO 170$ IF ERROR
1525 030742 130$:
1526 030742 004737 052270 JSR PC,SEKSTS ;GO VERIFY RESULTS OF SEEK OPERATION
030746 000405 BR 140$ ;GO TO 140$ IF NO ERROR
030750 000240 NOP ;RETURN HERE IF ERROR
030752 104000 EMT ;ERROR # DEFINED BY SEKSTS SUBROUTINE
030754 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
030756 000137 031036 JMP 170$ ;GO TO 170$ IF ERROR
1527 030762 140$:
1528 030762 004737 051644 JSR PC,CMPERRSTS ;CHECK ANY ERRORS NOT MASKED
030766 115760 .WORD ND1MSK ;MASK FOR RMER1
030770 000010 .WORD DPE ;MASK FOR RMER2
030772 000405 BR 150$ ;GO TO 150$ IF NO ERROR
030774 000240 NOP ;RETURN HERE IF ERROR
030776 104000 EMT ;ERROR # DEFINED BY CMPERRSTS SUBROUTINE
031000 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
031002 000137 031036 JMP 170$ ;GO TO 170$ IF ERROR

```

```

1529 031006
1530 031006 004737 046004
      031012 000405
      031014 000240
      031016 104000
      031020 004736
      031022 000137 031036
1531 031026
1532 031026 005303
1533 031030 001402
1534 031032 000137 030656
1535 031036
1536
1537

```

```

150$: JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
      BR 160$ ;GO TO 160$ IF NO ERROR
      NOP ;RETURN HERE IF ERROR
      EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
      JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      JMP 170$ ;GO TO 170$ IF ERROR

160$: DEC R3 ;DONE ALL SEEKS??
      BEQ 170$ ;YES!!
      JMP 90$ ;NO - GO DO NEXT SEEK

170$:

```

```

;*****
;*TEST 47 SEEK MAXIMUM DIFFERENCE FORWARD
;*****
TST47:

```

```

031036
031036 000004
031040 000240
031042 012706 001100
031046 013700 001276
031052 013701 001464
031056 012737 000002 001206
031064 012737 000047 001226

```

```

SCOPE ;SCOPE CALL
NOP ;START OF TEST
MOV #STACK,SP ;INITIALIZE STACK POINTER
MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
MOV #2,$TIMES ;DO 2 ITERATIONS
MOV #47,$TESTN ;SET TEST NUMBER IN APT MAIL BOX

```

```

1538
1539 031072
1540 031072 004737 043150
      031076 054130

```

```

10$: JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
      .WORD 054130 ;TASK DESCRIPTOR AS FOLLOWS:
      ;CLEAR CONTROLLER & SELECT DEVICE
      ;VERIFY CONTROLLER CLEAR OPERATION
      ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
      ;VERIFY PACK ACKNOWLEDGE
      ;RECALIBRATE IF "SKI" OR "PIP" IS SET
      ;VERIFY RECALIBRATION
      ;GO TO 80$ IF NO ERROR
      ;RETURN HERE IF ERROR
      ;ERROR # DEFINED BY TSTPRP SUBROUTINE
      ;GO TO 160$ IF ERROR

80$: BR 80$
      NOP
      EMT
      JMP 160$

```

```

031100 000404
031102 000240
031104 104000
031106 000137 031350
1541 031112
1542 031112 012737 000000 001444
1543 031120 012737 000000 001416
1544 031126 012737 000005 001410
1545 031134 012702 001543
1546 031140 112722 000006
1547 031144 112722 000034
1548 031150 112722 000000
1549 031154 112722 000200
1550 031160

```

```

MOV #0,RMDCO ;SEEK TO CYLINDER 0
MOV #0,RMDAO ;TRACK = 0, SECTOR = 0
MOV #SEEK!GO,RMCS10 ;FUNCTION CODE FOR SEEK
MOV #PUTINX,R2 ;R2 POINTS TO REGISTER TABLE
MOVB #RMDA,(R2)+ ;WRITE REGISTER INDEX TABLE
MOVB #RMDC,(R2)+
MOVB #RMCS1,(R2)+
MOVB #200,(R2)+ ;STEUP TERMINATOR BYTE IN TABLE

```

```

1551 031160 004737 044424
      031164 000404
      031166 000240
      031170 104000
      031172 000137 031350

```

```

90$: JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      BR 100$ ;GO TO 100$ IF NO ERROR
      NOP ;RETURN HERE IF ERROR
      EMT ;ERROR # DEFINED BY PUT SUBROUTINE
      JMP 160$ ;GO TO 160$ IF ERROR

```

```

1552 031176
1553 031176 004737 044070
1554 031202 004737 044766
1555

```

```

100$: JSR PC,GETSTS ;SETUP TO READ ALL REGISTERS
      JSR PC,TIMOUT ;WAIT FOR SEEK TO COMPLETE

```

```

1556 031206 004737 044154      JSR    PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      031212 000404          BR      110$          ;GO TO 110$ IF NO ERROR
      031214 000240          NOP                    ;RETURN HERE IF ERROR
      031216 104000          EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      031220 000137 031350      JMP     160$          ;GO TO 160$ IF ERROR
1557 031224          110$:
1558 031224 004737 045152      JSR    PC,PRIERR       ;GO CHECK FOR PRIMARY ERRORS
      031230 000405          BR      120$          ;GO TO 120$ IF NO ERROR
      031232 000240          NOP                    ;RETURN HERE IF ERROR
      031234 104000          EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      031236 004736          JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      031240 000137 031350      JMP     160$          ;GO TO 160$ IF ERROR
1559 031244          120$:
1560 031244 004737 052270      JSR    PC,SEKSTS       ;GO VERIFY RESULTS OF SEEK OPERATION
      031250 000405          BR      130$          ;GO TO 130$ IF NO ERROR
      031252 000240          NOP                    ;RETURN HERE IF ERROR
      031254 104000          EMT                    ;ERROR # DEFINED BY SEKSTS SUBROUTINE
      031256 004736          JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      031260 000137 031350      JMP     160$          ;GO TO 160$ IF ERROR
1561 031264          130$:
1562 031264 004737 051644      JSR    PC,CMPERRSTS    ;CHECK ANY ERRORS NOT MASKED
      031270 115760          .WORD  NDTMSK         ;MASK FOR RMER1
      031272 000010          .WORD  DPE           ;MASK FOR RMER2
      031274 000405          BR      140$          ;GO TO 140$ IF NO ERROR
      031276 000240          NOP                    ;RETURN HERE IF ERROR
      031300 104000          EMT                    ;ERROR # DEFINED BY CMPERRSTS SUBROUTINE
      031302 004736          JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      031304 000137 031350      JMP     160$          ;GO TO 160$ IF ERROR
1563 031310          140$:
1564 031310 004737 046004      JSR    PC,SECERR       ;GO CHECK FOR SECONDARY ERRORS
      031314 000405          BR      150$          ;GO TO 150$ IF NO ERROR
      031316 000240          NOP                    ;RETURN HERE IF ERROR
      031320 104000          EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      031322 004736          JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      031324 000137 031350      JMP     160$          ;GO TO 160$ IF ERROR
1565 031330          150$:
1566 031330 005737 001444      TST    RMDCO           ;IS TEST DONE ?
1567 031334 001005          BNE    160$          ;YES!!
1568 031336 012737 001466 001444  MOV    #822.,RMDCO     ;NO - SEEK TO LAST CYLINDER
1569 031344 000137 031160      JMP     90$
1570 031350          160$:
1571
1572

```

```

*****
;*TEST 50      SEEK ADJACENT FORWARD
*****
TST50:

```

```

031350          SCOPE          ;SCOPE CALL
031350 000004          NOP                    ;START OF TEST
031352 000240          MOV    #STACK,SP      ;INITIALIZE STACK POINTER
031354 012706 001100      MOV    $BASE,R0        ;R0 = UNIBUS ADDRESS
031360 013700 001276      MOV    TSTQUE,R1       ;(R1) = DEVICE BEING TESTED
031364 013701 001464      MOV    #1,$TIMES       ;DO 1 ITERATION
031370 012737 000001 001206  MOV    #50,$TESTN      ;SET TEST NUMBER IN APT MAIL BOX
031376 012737 000050 001226
1573
1574 031404          10$:
1575 031404 004737 043150      JSR    PC,TSTPRP       ;PREPARE DEVICE FOR TEST
      031410 054130          .WORD  054130         ;TASK DESCRIPTOR AS FOLLOWS:

```

```

                                ;CLEAR CONTROLLER & SELECT DEVICE
                                ;VERIFY CONTROLLER CLEAR OPERATION
                                ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                ;VERIFY PACK ACKNOWLEDGE
                                ;RECALIBRATE IF "SKI" OR "PIP" IS SET
                                ;VERIFY RECALIBRATION
                                ;GO TO 80$ IF NO ERROR
                                ;RETURN HERE IF ERROR
                                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                ;GO TO 160$ IF ERROR
031412 000404 BR 80$
031414 000240 NOP
031416 104000 EMT
031420 000137 031660 JMP 160$
1576 031424 80$:
1577 031424 012737 000000 001444 MOV #0,RMDCO ;CYLINDER = 0
1578 031432 012737 000000 001416 MOV #0,RMDAO ;TRACK = 0, SECTOR = 0
1579 031440 012737 000005 001410 MOV #SEEK!GO,RMCS10 ;FUNCTION CODE FOR SEEK
1580 031446 012702 001543 MOV #PUTINX,R2 ;R2 POINTS TO REGISTER INDEX
1581 031452 112722 000034 MOVB #RMDC,(R2)+ ;WRITE REGISTER INDEX TABLE
1582 031456 112722 000006 MOVB #RMDA,(R2)+
1583 031462 112722 000000 MOVB #RMCS1,(R2)+
1584 031466 112722 000200 MOVB #200,(R2)+ ;TERMINATE REGISTER TABLE
1585 031472 90$:
1586 031472 004737 044424 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
031476 000404 BR 100$ ;GO TO 100$ IF NO ERROR
031500 000240 NOP ;RETURN HERE IF ERROR
031502 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
031504 000137 031660 JMP 160$ ;GO TO 160$ IF ERROR
1587 031510 004737 044070 100$: JSR PC,GETSTS ;SETUP FOR STATUS FETCH
1588 031514 004737 044766 JSR PC,TIMOUT ;WAIT FOR COMPLETION
1589
1590 031520 004737 044154 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
031524 000404 BR 110$ ;GO TO 110$ IF NO ERROR
031526 000240 NOP ;RETURN HERE IF ERROR
031530 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
031532 000137 031660 JMP 160$ ;GO TO 160$ IF ERROR
1591 031536 110$:
1592 031536 004737 045152 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
031542 000405 BR 120$ ;GO TO 120$ IF NO ERROR
031544 000240 NOP ;RETURN HERE IF ERROR
031546 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
031550 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
031552 000137 031660 JMP 160$ ;GO TO 160$ IF ERROR
1593 031556 120$:
1594 031556 004737 052270 JSR PC,SEKSTS ;GO VERIFY RESULTS OF SEEK OPERATION
031562 000405 BR 130$ ;GO TO 130$ IF NO ERROR
031564 000240 NOP ;RETURN HERE IF ERROR
031566 104000 EMT ;ERROR # DEFINED BY SEKSTS SUBROUTINE
031570 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
031572 000137 031660 JMP 160$ ;GO TO 160$ IF ERROR
1595 031576 130$:
1596 031576 004737 051644 JSR PC,CMPEPERRSTS ;CHECK ANY ERRORS NOT MASKED
031602 115760 .WORD ND1MSK ;MASK FOR RMER1
031604 000010 .WORD DPE ;MASK FOR RMER2
031606 000405 BR 140$ ;GO TO 140$ IF NO ERROR
031610 000240 NOP ;RETURN HERE IF ERROR
031612 104000 EMT ;ERROR # DEFINED BY CMPEPERRSTS SUBROUTINE
031614 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
031616 000137 031660 JMP 160$ ;GO TO 160$ IF ERROR
1597 031622 140$:

```



```

1598 031622 004737 046004      JSR    PC,SECERR      ;GO CHECK FOR SECONDARY ERRORS
      031626 000405      BR     150$          ;GO TO 150$ IF NO ERROR
      031630 000240      NOP                    ;RETURN HERE IF ERROR
      031632 104000      EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      031634 004736      JSR    PC,@(SP)+     ;GO BACK FOR MORE ERROR CHECKS
      031636 000137 031660      JMP    160$          ;GO TO 160$ IF ERROR
1599 031642 005737 001444      150$: TST    RMDCO        ;FIRST TIME THROUGH??
1600 031642 001004      BNE   160$          ;NO!!
1601 031646 005237 001444      INC   RMDCO        ;YES - SEEK TO ADJACENT CYLINDER FORWARD
1602 031650 000137 031472      JMP    90$
1603 031654 000137 031472
1604 031660
1605
1606

```

```

*****
;*TEST 51      SEEK ADJACENT REVERSE
*****
TST51:

```

```

031660 000004      SCOPE                ;SCOPE CALL
031660 000240      NOP                  ;START OF TEST
031664 012706 001100      MOV    #STACK,SP    ;INITIALIZE STACK POINTER
031670 013700 001276      MOV    $BASE,R0     ;R0 = UNIBUS ADDRESS
031674 013701 001464      MOV    TSTQUE,R1    ;(R1) = DEVICE BEING TESTED
031700 012737 000001 001206      MOV    #1,$TIMES    ;DO 1 ITERATION
031706 012737 000051 001226      MOV    #51,$TESTN   ;SET TEST NUMBER IN APT MAIL BOX
1607
1608 031714
1609 031714 004737 043150      10$: JSR    PC,TSTPRP    ;PREPARE DEVICE FOR TEST
      031720 054130      .WORD 054130      ;TASK DESCRIPTOR AS FOLLOWS:
                                      ;CLEAR CONTROLLER & SELECT DEVICE
                                      ;VERIFY CONTROLLER CLEAR OPERATION
                                      ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                      ;VERIFY PACK ACKNOWLEDGE
                                      ;RECALIBRATE IF "SKI" OR "PIP" IS SET
                                      ;VERIFY RECALIBRATION
                                      ;GO TO 80$ IF NO ERROR
                                      ;RETURN HERE IF ERROR
                                      ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                      ;GO TO 160$ IF ERROR
031722 000404      BR     80$
031724 000240      NOP
031726 104000      EMT
031730 000137 032172      JMP    160$
1610 031734 012737 000001 001444      80$: MOV    #1,RMDCO    ;CYLINDER = 1
1611 031734 012737 000000 001416      MOV    #0,RMDAO    ;TRACK = 0, SECTOR = 0
1612 031742 012737 000005 001410      MOV    #SEEK!GO,RMCS10 ;FUNCTION CODE FOR SEEK
1613 031750 012737 001543      MOV    #PUTINX,R2  ;R2 POINTS TO REGISTER INDEX
1614 031756 012702 000006      MOV    #RMDA,(R2)+ ;WRITE REGISTER INDEX TABLE
1615 031762 112722 000034      MOV    #RMDC,(R2)+
1616 031766 112722 000000      MOV    #RMCS1,(R2)+
1617 031772 112722 000200      MOV    #200,(R2)+
1618 031776 112722 000200
1619 032002
1620 032002 004737 044424      90$: JSR    PC,PUT      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      032006 000404      BR     100$       ;GO TO 100$ IF NO ERROR
      032010 000240      NOP                    ;RETURN HERE IF ERROR
      032012 104000      EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      032014 000137 032172      JMP    160$        ;GO TO 160$ IF ERROR
1621 032020 004737 044070      100$: JSR   PC,GETSTS ;SETUP TO READ ALL REGISTERS
1622 032024 004737 044766      JSR   PC,TIMOUT    ;WAIT FOR SEEK TO COMPLETE
1623
1624 032030 004737 044154      JSR   PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE

```

```

032034 000404 BR 110$ ;GO TO 110$ IF NO ERROR
032036 000240 NOP ;RETURN HERE IF ERROR
032040 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
1625 032042 000137 032172 JMP 160$ ;GO TO 160$ IF ERROR
1626 032046 004737 045152 110$: JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
032052 000405 BR 120$ ;GO TO 120$ IF NO ERROR
032054 000240 NOP ;RETURN HERE IF ERROR
032056 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
032060 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
1627 032062 000137 032172 JMP 160$ ;GO TO 160$ IF ERROR
1628 032066 004737 052270 120$: JSR PC,SEKSTS ;GO VERIFY RESULTS OF SEEK OPERATION
032072 000405 BR 130$ ;GO TO 130$ IF NO ERROR
032074 000240 NOP ;RETURN HERE IF ERROR
032076 104000 EMT ;ERROR # DEFINED BY SEKSTS SUBROUTINE
032100 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
1629 032102 000137 032172 JMP 160$ ;GO TO 160$ IF ERROR
1630 032106 004737 051644 130$: JSR PC,CMPERRSTS ;CHECK ANY ERRORS NOT MASKED
032112 115760 .WORD NDTMSK ;MASK FOR RMER1
032114 000010 .WORD DPE ;MASK FOR RMER2
032116 000405 BR 140$ ;GO TO 140$ IF NO ERROR
032120 000240 NOP ;RETURN HERE IF ERROR
032122 104000 EMT ;ERROR # DEFINED BY CMPERRSTS SUBROUTINE
032124 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
1631 032126 000137 032172 JMP 160$ ;GO TO 160$ IF ERROR
1632 032132 004737 046004 140$: JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
032136 000405 BR 150$ ;GO TO 150$ IF NO ERROR
032140 000240 NOP ;RETURN HERE IF ERROR
032142 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
032144 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
1633 032146 000137 032172 JMP 160$ ;GO TO 160$ IF ERROR
1634 032152 022737 000001 001444 150$: CMP #1,RMDCO ;IS THIS THE FIRST SEEK??
1635 032160 001004 BNE 160$ ;NO!!
1636 032162 005337 001444 DEC RMDCO ;YES - SEEK TO ADJACENT CYLINDER
1637 032166 000137 032002 JMP 90$ ;GO SEEK
1638 032172
1639
1640

```

```

*****
;*TEST 52 SEEK INVALID SECTOR
*****

```

```

TST52:
032172 SCOPE ;SCOPE CALL
032172 000004 NOP ;START OF TEST
032174 000240 MOV #STACK,SP ;INITIALIZE STACK POINTER
032202 012706 001100 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
032206 013700 001276 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
032212 012737 000052 001226 MOV #52,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
1641
1642 032220 10$: JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
1643 032220 004737 043150 .WORD 054130 ;TASK DESCRIPTOR AS FOLLOWS:
032224 054130 ;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION

```

```
032226 000404 BR 80$ ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
032230 000240 NOP ;VERIFY PACK ACKNOWLEDGE
032232 104000 EMT ;RECALIBRATE IF "SKI" OR "PIP" IS SET
032234 000137 032610 JMP 180$ ;VERIFY RECALIBRATION
;GO TO 80$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 180$ IF ERROR
1644 032240 80$:
1645 032240 012737 000005 001410 MOV #SEEK!GO,RMCS10 ;SEEK COMMAND
1646 032246 012737 000000 001442 MOV #0,RMOFO ;FORMAT 18 BIT
1647 032254 012737 000000 001444 MOV #0,RMDCO ;CYLINDER = 0
1648 032262 012737 000036 001416 MOV #30.,RMDAO ;INVALID SECTOR = 30., TRACK = 0
1649 032270 012702 001543 MOV #PUTINX,R2 ;R2 POINTS TO REGISTER TABLE
1650 032274 112722 000032 MOVB #RMOF,(R2)+ ;WRITE REGISTER INDEX TABLE
1651 032300 112722 000006 MOVB #RMDA,(R2)+
1652 032304 112722 000034 MOVB #RMDC,(R2)+
1653 032310 112722 000000 MOVB #RMCS1,(R2)+
1654 032314 112722 000200 MOVB #200,(R2)+
1655 032320 004737 044070 JSR PC,GETSTS ;TERMINATE TABLE
1656 032324 90$: ;SETUP INPUT REGISTER INDEX
1657 032324 004737 053530 JSR PC,CNTCLR ;GO ISSUE CONTROLLER CLEAR
032330 000404 BR 95$ ;GO TO 95$ IF NO ERROR
032332 000240 NOP ;RETURN HERE IF ERROR
032334 104000 EMT ;ERROR NUMBER DEFINED BY SUBROUTINE
032336 000137 032610 JMP 180$ ;GO TO 180$ IF ERROR
1658 032342 95$:
1659 032342 004737 044154 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
032346 000404 BR 100$ ;GO TO 100$ IF NO ERROR
032350 000240 NOP ;RETURN HERE IF ERROR
032352 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
032354 000137 032610 JMP 180$ ;GO TO 180$ IF ERROR
1660 032360 100$:
1661 032360 004737 053646 JSR PC,CLRSTS ;GO VERIFY CONTROLLER CLEAR OPERATION
032364 000405 BR 110$ ;GO TO 110$ IF NO ERROR
032366 000240 NOP ;RETURN HERE IF ERROR
032370 104000 EMT ;ERROR # DEFINED BY CLRSTS SUBROUTINE
032372 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
032374 000137 032610 JMP 180$ ;GO TO 180$ IF ERROR
1662 032400 110$:
1663 032400 004737 044424 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
032404 000404 BR 120$ ;GO TO 120$ IF NO ERROR
032406 000240 NOP ;RETURN HERE IF ERROR
032410 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
032412 000137 032610 JMP 180$ ;GO TO 180$ IF ERROR
1664 032416 120$:
1665 032416 004737 044766 JSR PC,TIMOUT ;WAIT FOR GO TO RESET
1666
1667 032422 004737 044154 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
032426 000404 BR 125$ ;GO TO 125$ IF NO ERROR
032430 000240 NOP ;RETURN HERE IF ERROR
032432 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
032434 000137 032610 JMP 180$ ;GO TO 180$ IF ERROR
1668 032440 125$:
1669 032440 004737 045152 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
032444 000405 BR 130$ ;GO TO 130$ IF NO ERROR
032446 000240 NOP ;RETURN HERE IF ERROR
```

```

032450 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
032452 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
032454 000137 032610 JMP 180$ ;GO TO 180$ IF ERROR
1670 032460 130$:
1671 032460 004737 052270 JSR PC,SEKSTS ;GO VERIFY RESULTS OF SEEK OPERATION
032464 000405 BR 140$ ;GO TO 140$ IF NO ERROR
032466 000240 NOP ;RETURN HERE IF ERROR
032470 104000 EMT ;ERROR # DEFINED BY SEKSTS SUBROUTINE
032472 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
032474 000137 032610 JMP 180$ ;GO TO 180$ IF ERROR
1672 032500 140$:
1673 032500 150$:
1674 032500 004737 051644 JSR PC,CMPERRSTS ;CHECK ANY ERRORS NOT MASKED
032504 117760 .WORD ;MASK FOR RMER1
032506 000010 .WORD DPE ;MASK FOR RMER2
032510 000405 BR 160$ ;GO TO 160$ IF NO ERROR
032512 000240 NOP ;RETURN HERE IF ERROR
032514 104000 EMT ;ERROR # DEFINED BY CMPERRSTS SUBROUTINE
032516 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
032520 000137 032610 JMP 180$ ;GO TO 180$ IF ERROR
1675 032524 160$:
1676 032524 004737 046004 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
032530 000405 BR 170$ ;GO TO 170$ IF NO ERROR
032532 000240 NOP ;RETURN HERE IF ERROR
032534 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
032536 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
032540 000137 032610 JMP 180$ ;GO TO 180$ IF ERROR
1677 032544 170$:
1678 032544 005237 001416 INC RMDAO ;INCREMENT SECTOR ADDRESS
1679 032550 022737 000037 001416 CMP #31.,RMDAO ;DONE??
1680 032556 103262 BHIS 90$ ;NO - TEST NEXT SECTOR
1681
1682 032560 032737 010000 001442 BIT #FMT16,RMOFO ;WAS TEST DONE FOR 16 BIT MODE YET ?
1683 032566 001010 BNE 180$ ;BR IF YES
1684 032570 012737 010000 001442 MOV #FMT16,RMOFO ;SET 16 BIT FORMAT MODE
1685 032576 012737 000037 001416 MOV #31.,RMDAO ;SET INVALID SECTOR 31.
1686 032604 000137 032324 JMP 90$ ;NO - TEST NEXT SECTOR
1687 032610 180$:
1688
1689

```

```

*****
*TEST 53 SEEK INVALID TRACK
*****
TST53:

```

```

032610 SCOPE ;SCOPE CALL
032610 000004 NOP ;START OF TEST
032612 000240 MOV #STACK,SP ;INITIALIZE STACK POINTER
032614 012706 001100 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
032620 013700 001276 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
032624 013701 001464 MOV #53,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
032630 012737 000053 001226
1690
1691 032636 004737 043150 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
032642 054130 .WORD 054130 ;TASK DESCRIPTOR AS FOLLOWS:
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF "SKI" OR "PIP" IS SET

```

```

032644 000404 BR 80$ ;VERIFY RECALIBRATION
032646 000240 NOP ;GO TO 80$ IF NO ERROR
032650 104000 EMT ;RETURN HERE IF ERROR
032652 000137 033236 JMP 200$ ;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 200$ IF ERROR
1692 032656 80$:
1693 032656 012737 000005 001410 MOV #SEEK!GO,RMCS10 ;SEEK COMMAND
1694 032664 012737 000000 001444 MOV #0,RMDCO ;CYLINDER = 0
1695 032672 013737 001332 001416 MOV LSTRK,RMDAO ;LAST TRACK, SECTOR = 0
1696 032700 105237 001417 INCB RMDAO+1 ;SETUP FIRST INVALID TRACK
1697 032704 012737 000000 001442 MOV #0,RMOFO ;18 BIT FORMAT
1698 032712 012702 001543 MOV #PUTINX,R2 ;R2 POINTS TO REGISTER TABLE
1699 032716 112722 000032 MOVB #RMOF,(R2)+ ;WRITE REGISTER INDEX
1700 032722 112722 000034 MOVB #RMDC,(R2)+ ;TABLE FOR OUTPUT
1701 032726 112722 000006 MOVB #RMDA,(R2)+
1702 032732 112722 000000 MOVB #RMCS1,(R2)+
1703 032736 112722 000200 MOVB #200,(R2)+ ;TERMINATE TABLE
1704 032742 004737 044070 JSR PC,GETSTS ;SETUP INPUT TABLE FOR STATUS
1705 032746 90$:
1706 032746 004737 053530 JSR PC,CNTCLR ;GO ISSUE CONTROLLER CLEAR
032752 000404 BR 95$ ;GO TO 95$ IF NO ERROR
032754 000240 NOP ;RETURN HERE IF ERROR
032756 104000 EMT ;ERROR NUMBER DEFINED BY SUBROUTINE
032760 000137 033236 JMP 200$ ;GO TO 200$ IF ERROR
1707 032764 95$:
1708 032764 004737 044154 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
032770 000404 BR 100$ ;GO TO 100$ IF NO ERROR
032772 000240 NOP ;RETURN HERE IF ERROR
032774 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
032776 000137 033236 JMP 200$ ;GO TO 200$ IF ERROR
1709 033002 100$:
1710 033002 004737 053646 JSR PC,CLRSTS ;GO VERIFY CONTROLLER CLEAR OPERATION
033006 000405 BR 110$ ;GO TO 110$ IF NO ERROR
033010 000240 NOP ;RETURN HERE IF ERROR
033012 104000 EMT ;ERROR # DEFINED BY CLRSTS SUBROUTINE
033014 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
033016 000137 033236 JMP 200$ ;GO TO 200$ IF ERROR
1711 033022 110$:
1712 033022 004737 044424 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
033026 000404 BR 120$ ;GO TO 120$ IF NO ERROR
033030 000240 NOP ;RETURN HERE IF ERROR
033032 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
033034 000137 033236 JMP 200$ ;GO TO 200$ IF ERROR
1713 033040 120$:
1714 033040 004737 044766 JSR PC,TIMOUT ;WAIT FOR GO TO RESET
1715
1716 033044 004737 044154 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
033050 000404 BR 130$ ;GO TO 130$ IF NO ERROR
033052 000240 NOP ;RETURN HERE IF ERROR
033054 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
033056 000137 033236 JMP 200$ ;GO TO 200$ IF ERROR
1717 033062 130$:
1718 033062 004737 045152 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
033066 000405 BR 140$ ;GO TO 140$ IF NO ERROR
033070 000240 NOP ;RETURN HERE IF ERROR
033072 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
033074 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS

```

```

1719 033076 000137 033236          JMP      200$          ;GO TO 200$ IF ERROR
1720 033102 004737 052270          140$: JSR      PC,SEKSTS   ;GO VERIFY RESULTS OF SEEK OPERATION
      033106 000405                BR       150$          ;GO TO 150$ IF NO ERROR
      033110 000240                NOP                      ;RETURN HERE IF ERROR
      033112 104000                EMT                      ;ERROR # DEFINED BY SEKSTS SUBROUTINE
      033114 004736                JSR      PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      033116 000137 033236          JMP      200$          ;GO TO 200$ IF ERROR
1721 033122 150$:
1722 033122 160$:
1723 033122 004737 051644          JSR      PC,CMPERRSTS ;CHECK ANY ERRORS NOT MASKED
      033126 117760                .WORD   IAE!NDTMSK     ;MASK FOR RMER1
      033130 000010                .WORD   DPE            ;MASK FOR RMER2
      033132 000405                BR       170$          ;GO TO 170$ IF NO ERROR
      033134 000240                NOP                      ;RETURN HERE IF ERROR
      033136 104000                EMT                      ;ERROR # DEFINED BY CMPERRSTS SUBROUTINE
      033140 004736                JSR      PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      033142 000137 033236          JMP      200$          ;GO TO 200$ IF ERROR
1724 033146 170$:
1725 033146 004737 046004          JSR      PC,SECERR    ;GO CHECK FOR SECONDARY ERRORS
      033152 000405                BR       180$          ;GO TO 180$ IF NO ERROR
      033154 000240                NOP                      ;RETURN HERE IF ERROR
      033156 104000                EMT                      ;ERROR # DEFINED BY SECERR SUBROUTINE
      033160 004736                JSR      PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      033162 000137 033236          JMP      200$          ;GO TO 200$ IF ERROR
1726 033166 180$:
1727 033166 105237 001417          INCB    RMDAO+1        ;INCREMENT TRACK ADDRESS
1728 033172 123727 001417 000200  CMPB    RMDAO+1,#128. ;DONE??
1729 033200 101414                BLOS    195$          ;NO-DO NEXT TRACK
1730
1731 033202 032737 010000 001442  BIT     #FMT16,RMOFO   ;WAS TEST DONE FOR 16 BIT??
1732 033210 001012                BNE     200$          ;YES!!
1733 033212 012737 010000 001442  MOV     #FMT16,RMOFO   ;SET FORMAT 16
1734 033220 013737 001332 001416  MOV     LSTRK,RMDAO    ;LAST TRACK, SECTOR = 0
1735 033226 105237 001417          INCB    RMDAO+1        ;SETUP FIRST INVALID TRACK ADDRESS
1736 033232 000137 032746          195$: JMP     90$        ;DO TEST FOR 16 BIT FORMAT
1737
1738 033236 200$:
1739
1740

```

```

:*****
:*TEST 54      SEEK INVALID CYLINDER
:*****
TST54:

```

```

033236 000004          SCOPE          ;SCOPE CALL
033240 000240          NOP           ;START OF TEST
033242 012706 001100  MOV     #STACK,SP ;INITIALIZE STACK POINTER
033246 013700 001276  MOV     $BASE,R0  ;R0 = UNIBUS ADDRESS
033252 013701 001464  MOV     TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
033256 012737 000054 001226  MOV     #54,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
1741
1742 033264 004737 043150          JSR      PC,TSTPRP   ;PREPARE DEVICE FOR TEST
      033270 054130                .WORD   054130      ;TASK DESCRIPTOR AS FOLLOWS:
                                          ;CLEAR CONTROLLER & SELECT DEVICE
                                          ;VERIFY CONTROLLER CLEAR OPERATION
                                          ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                          ;VERIFY PACK ACKNOWLEDGE
                                          ;RECALIBRATE IF "SKI" OR "PIP" IS SET

```

```

033272 000404 BR 80$ ;VERIFY RECALIBRATION
033274 000240 NOP ;GO TO 80$ IF NO ERROR
033276 104000 EMT ;RETURN HERE IF ERROR
033300 000137 033654 JMP 200$ ;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 200$ IF ERROR
1743 033304 80$:
1744 033304 012737 000005 001410 MOV #SEEK!GO,RMCS10 ;SEEK FUNCTION CODE
1745 033312 012737 001467 001444 MOV #823.,RMDCO ;START AT FIRST INVALID CYLINDER
1746 033320 012737 000000 001416 MOV #0,RMDAO ;TRACK = 0, SECTOR = 0
1747 033326 012737 000000 001442 MOV #0,RMOFO ;18 BIT FORMAT
1748 033334 012702 001543 MOV #PUTINX,R2 ;R2 POINTS TO INDEX TABLE
1749 033340 112722 000006 MOVB #RMDA,(R2)+ ;WRITE REGISTER TABLE
1750 033344 112722 000034 MOVB #RMDC,(R2)+
1751 033350 112722 000032 MOVB #RMOF,(R2)+
1752 033354 112722 000000 MOVB #RMCS1,(R2)+
1753 033360 112722 000200 MOVB #200,(R2)+ ;TERMINATE TABLE
1754 033364 004737 044070 JSR PC,GETSTS ;SETUP FOR STATUS
1755 033370 90$:
1756 033370 004737 053530 JSR PC,CNTCLR ;GO ISSUE CONTROLLER CLEAR
033374 000404 BR 95$ ;GO TO 95$ IF NO ERROR
033376 000240 NOP ;RETURN HERE IF ERROR
033400 104000 EMT ;ERROR NUMBER DEFINED BY SUBROUTINE
033402 000137 033654 JMP 200$ ;GO TO 200$ IF ERROR
1757 033406 95$:
1758 033406 004737 044154 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
033412 000404 BR 100$ ;GO TO 100$ IF NO ERROR
033414 000240 NOP ;RETURN HERE IF ERROR
033416 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
033420 000137 033654 JMP 200$ ;GO TO 200$ IF ERROR
1759 033424 100$:
1760 033424 004737 053646 JSR PC,CLRSTS ;GO VERIFY CONTROLLER CLEAR OPERATION
033430 000405 BR 110$ ;GO TO 110$ IF NO ERROR
033432 000240 NOP ;RETURN HERE IF ERROR
033434 104000 EMT ;ERROR # DEFINED BY CLRSTS SUBROUTINE
033436 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
033440 000137 033654 JMP 200$ ;GO TO 200$ IF ERROR
1761 033444 110$:
1762 033444 004737 044424 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
033450 000404 BR 120$ ;GO TO 120$ IF NO ERROR
033452 000240 NOP ;RETURN HERE IF ERROR
033454 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
033456 000137 033654 JMP 200$ ;GO TO 200$ IF ERROR
1763 033462 120$:
1764 033462 004737 044766 JSR PC,TIMOUT ;WAIT FOR GO TO RESET
1765
1766 033466 004737 044154 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
033472 000404 BR 130$ ;GO TO 130$ IF NO ERROR
033474 000240 NOP ;RETURN HERE IF ERROR
033476 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
033500 000137 033654 JMP 200$ ;GO TO 200$ IF ERROR
1767 033504 130$:
1768 033504 004737 045152 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
033510 000405 BR 140$ ;GO TO 140$ IF NO ERROR
033512 000240 NOP ;RETURN HERE IF ERROR
033514 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
033516 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
033520 000137 033654 JMP 200$ ;GO TO 200$ IF ERROR

```

```

1769 033524
1770 033524 004737 052270
      033530 000405
      033532 000240
      033534 104000
      033536 004736
      033540 000137 033654
1771 033544
1772 033544 004737 051644
      033550 117760
      033552 000010
      033554 000405
      033556 000240
      033560 104000
      033562 004736
      033564 000137 033654
1773 033570
1774 033570 004737 046004
      033574 000405
      033576 000240
      033600 104000
      033602 004736
      033604 000137 033654
1775 033610
1776 033610 005237 001444
1777 033614 023727 001444 002000
1778 033622 002412
1779
1780 033624 032737 010000 001442
1781 033632 001010
1782 033634 012737 010000 001442
1783 033642 012737 001467 001444
1784 033650 000137 033370
1785
1786 033654
1787
1788

```

```

140$:
JSR PC,SEKSTS ;GO VERIFY RESULTS OF SEEK OPERATION
BR 150$ ;GO TO 150$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
EMT ;ERROR # DEFINED BY SEKSTS SUBROUTINE
JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 200$ ;GO TO 200$ IF ERROR

150$:
JSR PC,CMPEERRSTS ;CHECK ANY ERRORS NOT MASKED
.WORD IAE!NDTMSK ;MASK FOR RMER1
.WORD DPE ;MASK FOR RMER2
BR 170$ ;GO TO 170$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
EMT ;ERROR # DEFINED BY CMPEERRSTS SUBROUTINE
JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 200$ ;GO TO 200$ IF ERROR

170$:
JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
BR 180$ ;GO TO 180$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 200$ ;GO TO 200$ IF ERROR

180$:
INC RMDCO ;INCREMENT CYLINDER ADDRESS
CMP RMDCO,#1024. ;DONE??
BLT 195$ ;NO-DO NEXT CYLINDER

195$:
BIT #FMT16,RMOFO ;16 BIT FORMAT TESTED??
BNE 200$ ;YES !!
MOV #FMT16,RMOFO ;SETUP FOR 16 BIT TEST
MOV #823.,RMDCO ;START AT FIRST INVALID CYLINDER
JMP 90$

200$:

```

```

*****
;*TEST 55 IVC SEEK TEST
*****

```

```

TST55:
033654
033654 000004
033656 000240
033660 012706 001100
033664 013700 001276
033670 013701 001464
033674 012737 000055 001226
1789
1790 033702 004737 053530
      033706 000404
      033710 000240
      033712 104000
      033714 000137 034214
1791 033720
1792 033720 112737 000024 001543
      033726 112737 000200 001544
      033734 012737 000001 001434
      033742 004737 044424

```

```

SCOPE ;SCOPE CALL
NOP ;START OF TEST
MOV #STACK,SP ;INITIALIZE STACK POINTER
MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
MOV #55,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

10$:
JSR PC,CNTCLR ;GO ISSUE CONTROLLER CLEAR
BR 10$ ;GO TO 10$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
EMT ;ERROR NUMBER DEFINED BY SUBROUTINE
JMP 90$ ;GO TO 90$ IF ERROR

MOV #RMMR1,PUTINX ;SETUP PUT INDEX TABLE
MOV #200,PUTINX+1 ;SET TERMINATOR BYTE
MOV #DMD,RMMR10 ;SET RMMR1 OUTPUT BUFFER = DMD
JSR PC,PUT ;GO WRITE RMMR1 VIA PUT SUBROUTINE

```



```

033746 000404 BR 20$ ;GO TO 20$ IF NO ERROR
033750 000240 NOP ;RETURN HERE IF ERROR
033752 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
033754 000137 034214 JMP 90$ ;GO TO 90$ IF ERROR
1793 033760 20$:
1794 033760 012737 000005 001410 MOV #SEEK!GO,RMCS10
1795 033766 012737 000000 001434 MOV #0,RMMR10
1796 033774 012737 000000 001444 MOV #0,RMDCO ;CYLINDER = 0
1797 034002 012737 000000 001416 MOV #0,RMDAO ;TRACK = 0, SECTOR = 0
1798 034010 012702 001543 MOV #PUTINX,R2 ;WRITE REGISTER INDEX
1799 034014 112722 000024 MOVB #RMMR1,(R2)+ ;TABLE
1800 034020 112722 000006 MOVB #RMDA,(R2)+
1801 034024 112722 000034 MOVB #RMDC,(R2)+
1802 034030 112722 000000 MOVB #RMCS1,(R2)+
1803 034034 112722 000200 MOVB #200,(R2)+
1804 034040 004737 044070 JSR PC,GETSTS ;SETUP FOR STATUS
1805 034044 004737 044424 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
034050 000404 BR 30$ ;GO TO 30$ IF NO ERROR
034052 000240 NOP ;RETURN HERE IF ERROR
034054 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
034056 000137 034214 JMP 90$ ;GO TO 90$ IF ERROR
1806 034062 30$:
1807 034062 004737 044154 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
034066 000404 BR 40$ ;GO TO 40$ IF NO ERROR
034070 000240 NOP ;RETURN HERE IF ERROR
034072 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
034074 000137 034214 JMP 90$ ;GO TO 90$ IF ERROR
1808 034100 40$:
1809 034100 004737 045152 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
034104 000405 BR 50$ ;GO TO 50$ IF NO ERROR
034106 000240 NOP ;RETURN HERE IF ERROR
034110 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
034112 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
034114 000137 034214 JMP 90$ ;GO TO 90$ IF ERROR
1810 034120 032737 010000 001376 50$:
1811 034126 001012 BIT #IVC,RMER2I ;DID INVALID COMMAND SET??
1812 034130 012737 010000 001140 BNE 60$ ;YES!!
1813 034136 013737 001376 001142 MOV #IVC,$GDDAT ;GOOD DATA FOR TYPEOUT
1814 034144 042737 167777 001142 MOV RMER2I,$BDDAT ;BAD DATA FOR TYPEOUT
1815 034152 104064 BIC #^CIVC,$BDDAT
1816 034154 60$:
1817 034154 004737 051644 JSR PC,CMPEERRSTS ;CHECK ANY ERRORS NOT MASKED
034160 115760 .WORD ND1MSK ;MASK FOR RMER1
034162 010010 .WORD IVC!DPE ;MASK FOR RMER2
034164 000405 BR 80$ ;GO TO 80$ IF NO ERROR
034166 000240 NOP ;RETURN HERE IF ERROR
034170 104000 EMT ;ERROR # DEFINED BY CMPEERRSTS SUBROUTINE
034172 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
034174 000137 034214 JMP 90$ ;GO TO 90$ IF ERROR
1818 034200 80$:
1819 034200 004737 046004 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
034204 000403 BR 90$ ;GO TO 90$ IF NO ERROR
034206 000240 NOP ;RETURN HERE IF ERROR
034210 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
034212 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
1820 034214 90$:
1821

```

1822

```

*****
:TEST 56      ABORT SEEK TEST
*****
TST56:

```

034214	000004				SCOPE		:SCOPE CALL	
034214	000240				NOP		:START OF TEST	
034220	012706	001100			MOV	#STACK,SP	:INITIALIZE STACK POINTER	
034224	013700	001276			MOV	\$BASE,R0	:R0 = UNIBUS ADDRESS	
034230	013701	001464			MOV	TSTQUE,R1	:(R1) = DEVICE BEING TESTED	
034234	012737	000056	001226		MOV	#56,\$TESTN	:;SET TEST NUMBER IN APT MAIL BOX	
1823								
1824	034242	004737	043150		JSR	PC,TSTPRP	:PREPARE DEVICE FOR TEST	
	034246	054130			.WORD	054130	:TASK DESCRIPTOR AS FOLLOWS:	
							:CLEAR CONTROLLER & SELECT DEVICE	
							:VERIFY CONTROLLER CLEAR OPERATION	
							:PACK ACKNOWLEDGE IF VOLUME NOT VALID	
							:VERIFY PACK ACKNOWLEDGE	
							:RECALIBRATE IF "SKI" OR "PIP" IS SET	
							:VERIFY RECALIBRATION	
							:GO TO 80\$ IF NO ERROR	
							:RETURN HERE IF ERROR	
							:ERROR # DEFINED BY 1STPRP SUBROUTINE	
							:GO TO 160\$ IF ERROR	
034250	000404				BR	80\$		
034252	000240				NOP			
034254	104000				EMT			
034256	000137	034556			JMP	160\$		
1825	034262							
1826	034262	012737	040000	001424	80\$:	MOV	#UNS,RMER10	:SET UNSAFE ERROR
1827	034270	012737	000000	001444		MOV	#0,RMDCO	:CYLINDER = 0
1828	034276	012737	000000	001416		MOV	#0,RMDAO	:TRACK = 0, SECTOR = 0
1829	034304	012737	000005	001410		MOV	#SEEK!GO,RMCS10	:SEEK COMMAND
1830	034312	012702	001543			MOV	#PUTINX,R2	:SETUP REGISTER INDEX TABLE
1831	034316	112722	000006			MOVB	#RMDA,(R2)+	:AND OUTPUT BUFFER
1832	034322	112722	000034			MOVB	#RMDC,(R2)+	
1833	034326	112722	000014			MOVB	#RMER1,(R2)+	
1834	034332	112722	000000			MOVB	#RMCS1,(R2)+	
1835	034336	112722	000200			MOVB	#200,(R2)+	
1836	034342	004737	044070			JSR	PC,GETSTS	:SETUP FOR STATUS FETCH
1837	034346	004737	044424			JSR	PC,PUT	:GO WRITE REGISTER(S) WITH PUT SUBROUTINE
	034352	000404				BR	90\$:GO TO 90\$ IF NO ERROR
	034354	000240				NOP		:RETURN HERE IF ERROR
	034356	104000				EMT		:ERROR # DEFINED BY PUT SUBROUTINE
	034360	000137	034556			JMP	160\$:GO TO 160\$ IF ERROR
1838	034364				90\$:			
1839	034364	004737	044154			JSR	PC,GET	:GO READ REGISTER(S) WITH GET SUBROUTINE
	034370	000404				BR	100\$:GO TO 100\$ IF NO ERROR
	034372	000240				NOP		:RETURN HERE IF ERROR
	034374	104000				EMT		:ERROR # DEFINED BY GET SUBROUTINE
	034376	000137	034556			JMP	160\$:GO TO 160\$ IF ERROR
1840	034402	032737	020000	001346	100\$:	BIT	#PIP,RMDSI	:DID DRIVE START SEEK??
1841	034410	001411				BEQ	110\$:NO!!
1842	034412	013737	001346	001142		MOV	RMDSI,\$BDDAT	:BAD DATA FOR TYPEOUT
1843	034420	042737	157777	001142		BIC	#^CPIP,\$BDDAT	
1844	034426	005037	001140			CLR	\$GDDAT	:GOOD DATA FOR TYPEOUT
1845	034432	104065				EMT	65	
1846	034434	004737	044766		110\$:	JSR	PC,TIMOUT	:WAIT FOR GO TO RESET
1847								
1848	034440	004737	044154			JSR	PC,GET	:GO READ REGISTER(S) WITH GET SUBROUTINE
	034444	000404				BR	120\$:GO TO 120\$ IF NO ERROR
	034446	000240				NOP		:RETURN HERE IF ERROR

```

1849 034450 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
034452 000137 034556 JMP 160$ ;GO TO 160$ IF ERROR
1850 034456 004737 045152 120$: JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
034462 000405 BR 130$ ;GO TO 130$ IF NO ERROR
034464 000240 NOP ;RETURN HERE IF ERROR
034466 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
034470 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
034472 000137 034556 JMP 160$ ;GO TO 160$ IF ERROR
1851 034476 004737 061232 130$: JSR PC,STCDRVSTS ;GO CHECK FOR CHANGES IN DRIVE STATUS
034502 000405 BR 140$ ;GO TO 140$ IF NO ERROR
034504 000240 NOP ;RETURN HERE IF ERROR
034506 104000 EMT ;ERROR # DEFINED BY STCDRVSTS SUBROUTINE
034510 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
034512 000137 034556 JMP 160$ ;GO TO 160$ IF ERROR
1853 034516 004737 051644 140$: JSR PC,CMPEERRSTS ;CHECK ANY ERRORS NOT MASKED
034522 155760 .WORD NDIMSK!UNS ;MASK FOR RMER1
034524 000010 .WORD DPE ;MASK FOR RMER2
034526 000405 BR 150$ ;GO TO 150$ IF NO ERROR
034530 000240 NOP ;RETURN HERE IF ERROR
034532 104000 EMT ;ERROR # DEFINED BY CMPEERRSTS SUBROUTINE
034534 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
034536 000137 034556 JMP 160$ ;GO TO 160$ IF ERROR
1855 034542 004737 046004 150$: JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
034546 000403 BR 160$ ;GO TO 160$ IF NO ERROR
034550 000240 NOP ;RETURN HERE IF ERROR
034552 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
034554 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
1857 034556 160$:
1858
1859

```

```

*****
;*TEST 57 SEEK AT OFFSET
*****
TST57:

```

```

034556 000004 SCOPE ;SCOPE CALL
034560 000240 NOP ;START OF TEST
034562 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
034566 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
034572 013701 001464 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
034576 012737 000001 001206 MOV #1,$TIMES ;DO 1 ITERATION
034604 012737 000057 001226 MOV #57,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
1860
1861 034612 004737 043150 10$: JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
034616 054130 .WORD 054130 ;TASK DESCRIPTOR AS FOLLOWS:
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF "SKI" OR "PIP" IS SET
;VERIFY RECALIBRATION
034620 000404 BR 20$ ;GO TO 20$ IF NO ERROR
034622 000240 NOP ;RETURN HERE IF ERROR
034624 104000 EMT ;ERROR # DEFINED BY TSTPRP SUBROUTINE

```

```

1863 034626 000137 035110          JMP      140$          ;GO TO 140$ IF ERROR
1863 034632          20$:
1864 034632 012737 000000 001444  MOV      #0,RMDCO      ;CYLINDER = 0
1865 034640 012737 000000 001416  MOV      #0,RMDAO      ;TRACK = 0, SECTOR = 0
1866 034646 012737 000015 001410  MOV      #OFFSET!GO,RMCS10 ;LOAD OFFSET COMMAND IN BUFFER
1867 034654 012702 001543          MOV      #PUTINX,R2    ;R2 POINTS TO REGISTER TABLE
1868 034660 112722 000034          MOVVB   #RMDC,(R2)+    ;WRITE REGISTER INDEX TABLE
1869 034664 112722 000006          MOVVB   #RMDA,(R2)+
1870 034670 112722 000000          MOVVB   #RMCS1,(R2)+
1871 034674 112722 000200          MOVVB   #200,(R2)+    ;WRITE TERMINATOR
1872 034700 004737 044424          JSR     PC,PUT        ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
          034704 000404          BR      30$          ;GO TO 30$ IF NO ERROR
          034706 000240          NOP                    ;RETURN HERE IF ERROR
          034710 104000          EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
          034712 000137 035110          JMP     140$          ;GO TO 140$ IF ERROR
1873 034716 004737 044070          30$:  JSR     PC,GETSTS     ;SETUP FOR STATUS FETCH
1874 034722 004737 044766          JSR     PC,TIMOUT     ;WAIT FOR OFFSET TO COMPLETE
1875 034726          40$:
1876 034726 012737 000005 001410  MOV      #SEEK!GO,RMCS10 ;LOAD SEEK COMMAND
1877 034734 112737 000000 001543  MOVVB   #RMCS1,PUTINX  ;LOAD REGISTER INDEX TABLE
1878 034742 112737 000200 001544  MOVVB   #200,PUTINX+1
1879 034750 004737 044424          JSR     PC,PUT        ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
          034754 000404          BR      50$          ;GO TO 50$ IF NO ERROR
          034756 000240          NOP                    ;RETURN HERE IF ERROR
          034760 104000          EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
          034762 000137 035110          JMP     140$          ;GO TO 140$ IF ERROR
1880 034766          50$:
1881
1882          ;WAIT FOR COMMAND TO COMPLETE
          JSR     PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
1883
1884 034772 004737 044154          JSR     PC,GET        ;GO READ REGISTER(S) WITH GET SUBROUTINE
          034776 000404          BR      100$         ;GO TO 100$ IF NO ERROR
          035000 000240          NOP                    ;RETURN HERE IF ERROR
          035002 104000          EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
          035004 000137 035110          JMP     140$         ;GO TO 140$ IF ERROR
1885 035010          100$:
1886 035010 004737 045152          JSR     PC,PRIERR     ;GO CHECK FOR PRIMARY ERRORS
          035014 000405          BR      110$         ;GO TO 110$ IF NO ERROR
          035016 000240          NOP                    ;RETURN HERE IF ERROR
          035020 104000          EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
          035022 004736          JSR     PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
          035024 000137 035110          JMP     140$         ;GO TO 140$ IF ERROR
1887 035030          110$:
1888 035030 004737 052270          JSR     PC,SEKSTS     ;GO VERIFY RESULTS OF SEEK OPERATION
          035034 000405          BR      120$         ;GO TO 120$ IF NO ERROR
          035036 000240          NOP                    ;RETURN HERE IF ERROR
          035040 104000          EMT                    ;ERROR # DEFINED BY SEKSTS SUBROUTINE
          035042 004736          JSR     PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
          035044 000137 035110          JMP     140$         ;GO TO 140$ IF ERROR
1889 035050          120$:
1890 035050 004737 051644          JSR     PC,CMPESTS    ;CHECK ANY ERRORS NOT MASKED
          035054 115760          .WORD  ND1MSK        ;MASK FOR RMER1
          035056 000010          .WORD  DPE           ;MASK FOR RMER2
          035060 000405          BR      130$         ;GO TO 130$ IF NO ERROR
          035062 000240          NOP                    ;RETURN HERE IF ERROR
          035064 104000          EMT                    ;ERROR # DEFINED BY CMPESTS SUBROUTINE
    
```

```

1891 035066 004736 035110 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
035070 000137 035110 JMP 140$ ;GO TO 140$ IF ERROR
1892 035074 004737 046004 130$: JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
035100 000403 BR 140$ ;GO TO 140$ IF NO ERROR
035102 000240 NOP ;RETURN HERE IF ERROR
035104 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
035106 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
1893 035110 140$:
1894
1895
:*****
:*TEST 60 LOOK AHEAD TEST
:*****
TST60:
035110 SCOPE ;SCOPE CALL
035110 000004 NOP ;START OF TEST
035112 000240 MOV #STACK,SP ;INITIALIZE STACK POINTER
035114 012706 001100 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
035120 013700 001276 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
035124 013701 001464 MOV #60,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
035130 012737 000060 001226
1896
1897 035136 004737 053530 JSR PC,CNTCLR ;GO ISSUE CONTROLLER CLEAR
035142 000404 BR 10$ ;GO TO 10$ IF NO ERROR
035144 000240 NOP ;RETURN HERE IF ERROR
035146 104000 EMT ;ERROR NUMBER DEFINED BY SUBROUTINE
035150 000137 035536 JMP 160$ ;GO TO 160$ IF ERROR
1898
1899 035154 012703 000037 10$: MOV #31.,R3 ;R3 = SAMPLE COUNT FOR 18 BIT MODE
1900 035160 012737 000000 001442 MOV #0,RMOFO ;START WITH 18 BIT MODE
1901 035166 012737 000000 001444 MOV #0,RMDCO ;CYLINDER = 0
1902 035174 012737 000000 001416 MOV #0,RMDAO ;SEARCH TRACK = 0, SECTOR = 0
1903 035202 012737 000031 001410 MOV #SEARCH!GO,RMCS10 ;RECALIBRATE COMMAND
1904 035210 012702 001543 MOV #PUTINX,R2
1905 035214 112722 000006 MOVB #RMDA,(R2)+
1906 035220 112722 000034 MOVB #RMDC,(R2)+
1907 035224 112722 000032 MOVB #RMOF,(R2)+
1908 035230 112722 000000 MOVB #RMCS1,(R2)+
1909 035234 112722 000200 MOVB #200,(R2)+
1910 035240 004737 044070 JSR PC,GETSTS
1911 035244
1912 035244 004737 044424 20$: JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
035250 000404 BR 30$ ;GO TO 30$ IF NO ERROR
035252 000240 NOP ;RETURN HERE IF ERROR
035254 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
035256 000137 035526 JMP 150$ ;GO TO 150$ IF ERROR
1913 035262 004737 044766 30$: JSR PC,TIMOUT ;WAIT FOR COMPLETION
1914
1915 035266 004737 044154 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
035272 000404 BR 40$ ;GO TO 40$ IF NO ERROR
035274 000240 NOP ;RETURN HERE IF ERROR
035276 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
035300 000137 035526 JMP 150$ ;GO TO 150$ IF ERROR
1916 035304
1917 035304 004737 045152 40$: JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
035310 000405 BR 50$ ;GO TO 50$ IF NO ERROR
035312 000240 NOP ;RETURN HERE IF ERROR
035314 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE

```

```

035316 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
035320 000137 035526 JMP 150$ ;GO TO 150$ IF ERROR
1918 035324 50$:
1919 035324 012704 103640 60$: MOV #BUFFER,R4 ;R4 = STORAGE ADDRESS
1920 035330 012705 177777 MOV #-1,R5 ;R5 = GROSS TIMER
1921 035334 016014 000020 70$: MOV RMLA(R0),(R4) ;STORE RMLA SAMPLE
1922 035340 016002 000020 MOV RMLA(R0),R2 ;GET ANOTHER LOOK
1923 035344 020214 CMP R2,(R4) ;ARE SAMPLES SAME??
1924 035346 001403 BEQ 80$ ;YES!!
1925 035350 005305 DEC R5 ;TIMEOUT??
1926 035352 001370 BNE 70$ ;NO - TRY AGAIN
1927 035354 000411 BR 100$ ;PROGRAM TIMEOUT
1928 035356 062704 000002 80$: ADD #2,R4 ;ADVANCE STORAGE ADDRESS
1929 035362 005303 DEC R3 ;ALL SAMPLES TAKEN??
1930 035364 001410 BEQ 110$ ;YES!!
1931 035366 026002 000020 90$: CMP RMLA(R0),R2 ;WAIT FOR CHANGE
1932 035372 001360 BNE 70$ ;YES!!
1933 035374 005305 DEC R5 ;TIMEOUT??
1934 035376 001373 BNE 90$ ;NO
1935 035400 100$:
035400 104352 EMT 352
1936 035402 000137 035536 JMP 160$
1937
1938 ;SET 18 BIT SAMPLE PARAMETERS
1939 035406 012702 003500 110$: MOV #3500,R2 ;R2 = MAXIMUM SAMPLE
1940 035412 012703 000036 MOV #30,R3 ;R3 = NUMBER OF COMPARES
1941 035416 012704 103640 MOV #BUFFER,R4 ;R4 = BUFFER ADDRESS
1942 035422 032737 010000 001442 BIT #FMT16,RMOFO ;IS SAMPLE FOR 18 BIT MODE ?
1943 035430 001404 BEQ 120$ ;BR IF YES
1944
1945 ;SET 16 BIT SAMPLE PARAMETERS
1946 035432 012702 003700 MOV #3700,R2 ;R2 = MAXIMUM SAMPLE
1947 035436 012703 000040 MOV #32,R3 ;R3 = NUMBER OF COMPARES
1948 035442 012405 120$: MOV (R4)+,R5 ;GET A SAMPLE AND INCREMENT
1949 035444 062705 000100 ADD #100,R5 ;FOR EXPECTED VALUE OF NEXT
1950 035450 020205 CMP R2,R5 ;SHOULD NEXT BE 0 ?
1951 035452 103001 BHIS 130$ ;NO!!
1952 035454 005005 CLR R5 ;YES - CHANGE EXPECTED
1953 035456 020514 130$: CMP R5,(R4) ;IS NEXT SAMPLE CORRECT??
1954 035460 001406 BEQ 140$ ;YES!!
1955 035462 010537 001140 MOV R5,$GDDAT ;EXPECTED VALUE
1956 035466 011437 001142 MOV (R4),$BDDAT ;RECEIVED VALUE
1957 035472 104353 EMT 353
1958 035474 000414 BR 150$
1959 035476 005303 140$: DEC R3 ;ALL SAMPLES CHECKED??
1960 035500 001360 BNE 120$ ;NO - TEST NEXT SAMPLE
1961
1962 035502 032737 010000 001442 BIT #FMT16,RMOFO ;TEST DONE FOR 16 BIT MODE ?
1963 035510 001006 BNE 150$ ;BR IF YES
1964 035512 012737 010000 001442 MOV #FMT16,RMOFO ;SET 16 BIT MODE AND
1965 035520 012703 000041 MOV #33,R3 ;SET SAMPLE COUNT.
1966 035524 000647 BR 20$ ;TEST AGAIN
1967
1968 035526 150$:
035526 012637 000006 MOV (SP)+,ERRVEC+2 ;:POP STACK INTO ERRVEC+2
1969 035532 012637 000004 MOV (SP)+,ERRVEC ;:POP STACK INTO ERRVEC
1970 035536 160$:

```

1971
1972

: *TEST 61 SEARCH ON CYLINDER

```

035536          000004          SCOPE          ;SCOPE CALL
035536          000240          NOP           ;START OF TEST
035540          012706 001100    MOV          #STACK,SP ;INITIALIZE STACK POINTER
035542          013700 001276    MOV          $BASE,R0   ;R0 = UNIBUS ADDRESS
035546          013701 001464    MOV          TSTQUE,R1  ;(R1) = DEVICE BEING TESTED
035552          012737 000061 001226  MOV          #61,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
035556          004737 043150    JSR          PC,TSTPRP  ;PREPARE DEVICE FOR TEST
1973 035564          054130    .WORD       054130     ;TASK DESCRIPTOR AS FOLLOWS:
1974 035570          ;CLEAR CONTROLLER & SELECT DEVICE
          ;VERIFY CONTROLLER CLEAR OPERATION
          ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
          ;VERIFY PACK ACKNOWLEDGE
          ;RECALIBRATE IF "SKI" OR "PIP" IS SET
          ;VERIFY RECALIBRATION
          ;GO TO 80$ IF NO ERROR
          ;RETURN HERE IF ERROR
          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
          ;GO TO 200$ IF ERROR
035572          000404          BR           80$
035574          000240          NOP
035576          104000          EMT
035600          000137 036214    JMP          200$
1975 035604          80$:
035604          012703 000035    MOV          #29.,R3   ;R3 = MAXIMUM SECTOR ADDRESS
1976 035610          012737 000000 001444    MOV          #0,RMDCO  ;CYLINDER = 0
1977 035616          012737 000000 001416    MOV          #0,RMDAO  ;TRACK = 0, SECTOR = 0
1978 035624          012737 000000 001442    MOV          #0,RMOFO  ;18 BIT FORMAT
1979 035632          012737 000005 001410 85$: MOV          #SEEK!GO,RMCS10 ;SEEK COMMAND
1980 035640          012702 001543    MOV          #PUTINX,R2 ;SETUP REGISTER INDEX TABLE
1981 035644          112722 000006    MOVB        #RMDA,(R2)+ ;FOR SEEK TO CYLINDER 0
1982 035650          112722 000034    MOVB        #RMDC,(R2)+
1983 035654          112722 000032    MOVB        #RMOF,(R2)+
1984 035660          112722 000000    MOVB        #RMCS1,(R2)+
1985 035664          112722 000200    MOVB        #200,(R2)+
1986 035670          004737 044070    JSR          PC,GETSTS ;SETUP FOR STATUS FETCH
1987 035674          004737 044424    JSR          PC,PUT    ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
1988 035700          000404          BR           90$
          035702          000240          NOP
          035704          104000          EMT
          035706          000137 036214    JMP          200$
1989 035712          004737 044766    JSR          PC,TIMOUT ;GO TO 200$ IF ERROR
1990
1991 035716          004737 044154    JSR          PC,GET    ;GO READ REGISTER(S) WITH GET SUBROUTINE
          035722          000404          BR           100$
          035724          000240          NOP
          035726          104000          EMT
          035730          000137 036214    JMP          200$
1992 035734          100$:
1993 035734          004737 045152    JSR          PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
          035740          000405          BR           110$
          035742          000240          NOP
          035744          104000          EMT
          035746          004736    JSR          PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
1994 035750          000137 036214    JMP          200$
          035754          110$:

```

1995	035754	004737	052270			JSR	PC,SEKSTS	:GO VERIFY RESULTS OF SEEK OPERATION
	035760	000405				BR	120\$:GO TO 120\$ IF NO ERROR
	035762	000240				NOP		:RETURN HERE IF ERROR
	035764	104000				EMT		:ERROR # DEFINED BY SEKSTS SUBROUTINE
	035766	004736				JSR	PC,@(SP)+	:GO BACK FOR MORE ERROR CHECKS
	035770	000137	036214			JMP	200\$:GO TO 200\$ IF ERROR
1996	035774	012737	000031	001410	120\$:	MOV	#SEARCH!GO,RMCS10	:STORE SEARCH COMMAND
1997	036002	004737	044424			JSR	PC,PUT	:GO WRITE REGISTER(S) WITH PUT SUBROUTINE
	036006	000404				BR	130\$:GO TO 130\$ IF NO ERROR
	036010	000240				NOP		:RETURN HERE IF ERROR
	036012	104000				EMT		:ERROR # DEFINED BY PUT SUBROUTINE
	036014	000137	036214			JMP	200\$:GO TO 200\$ IF ERROR
1998	036020				130\$:			
1999	036020	004737	044766			JSR	PC,TIMOUT	:WAIT FOR SEARCH TO COMPLETE
2000								
2001	036024	004737	044154			JSR	PC,GET	:GO READ REGISTER(S) WITH GET SUBROUTINE
	036030	000404				BR	140\$:GO TO 140\$ IF NO ERROR
	036032	000240				NOP		:RETURN HERE IF ERROR
	036034	104000				EMT		:ERROR # DEFINED BY GET SUBROUTINE
	036036	000137	036214			JMP	200\$:GO TO 200\$ IF ERROR
2002	036042				140\$:			
2003	036042	004737	045152			JSR	PC,PRIERR	:GO CHECK FOR PRIMARY ERRORS
	036046	000405				BR	150\$:GO TO 150\$ IF NO ERROR
	036050	000240				NOP		:RETURN HERE IF ERROR
	036052	104000				EMT		:ERROR # DEFINED BY PRIERR SUBROUTINE
	036054	004736				JSR	PC,@(SP)+	:GO BACK FOR MORE ERROR CHECKS
	036056	000137	036214			JMP	200\$:GO TO 200\$ IF ERROR
2004	036062				150\$:			
2005	036062	004737	057666			JSR	PC,SCHSTS	:GO VERIFY SEARCH OPERATION
	036066	000405				BR	160\$:GO TO 160\$ IF NO ERROR
	036070	000240				NOP		:RETURN HERE IF ERROR
	036072	104000				EMT		:ERROR # DEFINED BY SCHSTS SUBROUTINE
	036074	004736				JSR	PC,@(SP)+	:GO BACK FOR MORE ERROR CHECKS
	036076	000137	036214			JMP	200\$:GO TO 200\$ IF ERROR
2006	036102				160\$:			
2007	036102	004737	051644			JSR	PC,CMPERRSTS	:CHECK ANY ERRORS NOT MASKED
	036106	115760				.WORD	NDTMSK	:MASK FOR RMER1
	036110	000010				.WORD	DPE	:MASK FOR RMER2
	036112	000405				BR	170\$:GO TO 170\$ IF NO ERROR
	036114	000240				NOP		:RETURN HERE IF ERROR
	036116	104000				EMT		:ERROR # DEFINED BY CMPERRSTS SUBROUTINE
	036120	004736				JSR	PC,@(SP)+	:GO BACK FOR MORE ERROR CHECKS
	036122	000137	036214			JMP	200\$:GO TO 200\$ IF ERROR
2008	036126				170\$:			
2009	036126	004737	046004			JSR	PC,SECERR	:GO CHECK FOR SECONDARY ERRORS
	036132	000405				BR	180\$:GO TO 180\$ IF NO ERROR
	036134	000240				NOP		:RETURN HERE IF ERROR
	036136	104000				EMT		:ERROR # DEFINED BY SECERR SUBROUTINE
	036140	004736				JSR	PC,@(SP)+	:GO BACK FOR MORE ERROR CHECKS
	036142	000137	036214			JMP	200\$:GO TO 200\$ IF ERROR
2010	036146				180\$:			
2011	036146	005237	001416			INC	RMDAO	:INCREMENT SECTOR ADDRESS
2012	036152	020337	001416			CMP	R3,RMDAO	:DONE ALL SECTORS??
2013	036156	103306				BHIS	120\$:NO!!
2014								
2015	036160	032737	010000	001442		BIT	#FMT16,RMOFO	:IS 16 BIT FORMAT DONE??
2016	036166	012737	010000	001442		MOV	#FMT16,RMOFO	:SET 16 BIT FORMAT


```

2017 036174 001007          BNE      200$          ;BR IF YES
2018 036176 012703 000037    MOV      #31.,R3      ;R3 = MAXIMUM SECTOR ADDRESS
2019 036202 012737 000000    MOV      #0,RMDAO    ;START AT TRACK = 0, SECTOR = 0
2020 036210 000137 035632    JMP      85$          ;GO TEST AGAIN
2021 036214
2022
2023
200$:
*****
:*TEST 62          SEARCH OFF CYLINDER
*****
TST62:
036214          SCOPE          ;SCOPE CALL
036214 000004    NOP          ;START OF TEST
036216 000240    MOV      #STACK,SP  ;INITIALIZE STACK POINTER
036220 012706 001100    MOV      $BASE,R0   ;R0 = UNIBUS ADDRESS
036224 013700 001276    MOV      TSTQUE,R1  ;(R1) = DEVICE BEING TESTED
036230 013701 001464    MOV      #62,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
036234 012737 000062    201226
2024
2025 036242 004737 043150    JSR      PC,TSTPRP  ;PREPARE DEVICE FOR TEST
036246 054130    .WORD    054130    ;TASK DESCRIPTOR AS FOLLOWS:
                                ;CLEAR CONTROLLER & SELECT DEVICE
                                ;VERIFY CONTROLLER CLEAR OPERATION
                                ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                ;VERIFY PACK ACKNOWLEDGE
                                ;RECALIBRATE IF "SKI" OR "PIP" IS SET
                                ;VERIFY RECALIBRATION
                                ;GO TO 80$ IF NO ERROR
                                ;RETURN HERE IF ERROR
                                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                ;GO TO 230$ IF ERROR
036250 000404    BR      80$
036252 000240    NOP
036254 104000    EMT
036256 000137 036720    JMP      230$
2026 036262          80$:
2027 036262 012737 000000    001416 MOV      #0,RMDAO
2028 036270 012737 000000    001442 MOV      #0,RMOFO
2029 036276 012704 000035    MOV      #29.,R4    ;START WITH 18 BIT MODE
2030 036302 012703 000632    MOV      #410.,R3   ;R4 = MAXIMUM SECTOR ADDRESS
2031 036306 012705 000633    MOV      #411.,R5   ;R3 = SEARCH CYLINDER
2032 036312 012737 000005    001410 90$: MOV      #SEEK!GO,RMCS10 ;R5 = SEEK CYLINDER
2033 036320 010537 001444    MOV      R5,RMDCO   ;LOAD SEEK COMMAND
2034 036324 012702 001543    MOV      #PUTINX,R2 ;LOAD CYLINDER ADDRESS
2035 036330 112722 000032    MOVVB   #RMOF,(R2)+ ;R2 POINTS TO REGISTER TABLE
2036 036334 112722 000034    MOVVB   #RMDC,(R2)+ ;SETUP REGISTER INDEX TABLE
2037 036340 112722 000006    MOVVB   #RMDA,(R2)+ ;FOR SEEK COMMAND
2038 036344 112722 000000    MOVVB   #RMCS1,(R2)+
2039 036350 112722 000200    MOVVB   #200,(R2)+
2040 036354 004737 044070    JSR      PC,GETSTS  ;TERMINATE TABLE
2041 036360 004737 044424    JSR      PC,PUT     ;SETUP FOR STATUS
036364 000404    BR      100$       ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
036366 000240    NOP              ;GO TO 100$ IF NO ERROR
036370 104000    EMT              ;RETURN HERE IF ERROR
036372 000137 036720    JMP      230$     ;ERROR # DEFINED BY PUT SUBROUTINE
                                ;GO TO 230$ IF ERROR
2042 036376          100$:
2043 036376 004737 044766    JSR      PC,TIMOUT  ;WAIT FOR SEEK TO COMPLETE
2044
2045 036402 004737 044154    JSR      PC,GET     ;GO READ REGISTER(S) WITH GET SUBROUTINE
036406 000404    BR      110$       ;GO TO 110$ IF NO ERROR
036410 000240    NOP              ;RETURN HERE IF ERROR
036412 104000    EMT              ;ERROR # DEFINED BY GET SUBROUTINE
036414 000137 036720    JMP      230$     ;GO TO 230$ IF ERROR
    
```

2046	036420			110\$:		
2047	036420	004737	045152		JSR	PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
	036424	000405			BR	120\$;GO TO 120\$ IF NO ERROR
	036426	000240			NOP	;RETURN HERE IF ERROR
	036430	104000			EMT	;ERROR # DEFINED BY PRIERR SUBROUTINE
	036432	004736			JSR	PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
	036434	000137	036720		JMP	230\$;GO TO 230\$ IF ERROR
2048	036440			120\$:		
2049	036440	004737	052270		JSR	PC,SEKSTS ;GO VERIFY RESULTS OF SEEK OPERATION
	036444	000405			BR	130\$;GO TO 130\$ IF NO ERROR
	036446	000240			NOP	;RETURN HERE IF ERROR
	036450	104000			EMT	;ERROR # DEFINED BY SEKSTS SUBROUTINE
	036452	004736			JSR	PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
	036454	000137	036720		JMP	230\$;GO TO 230\$ IF ERROR
2050	036460			130\$:		
2051	036460	010337	001444		MOV	R3,RMDCO ;LOAD CYLINDER ADDRESS
2052	036464	012737	000031	001410	MOV	#SEARCH!GO,RMCS10 ;LOAD SEARCH COMMAND
2053	036472	004737	044424		JSR	PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
	036476	000404			BR	140\$;GO TO 140\$ IF NO ERROR
	036500	000240			NOP	;RETURN HERE IF ERROR
	036502	104000			EMT	;ERROR # DEFINED BY PUT SUBROUTINE
	036504	000137	036720		JMP	230\$;GO TO 230\$ IF ERROR
2054	036510			140\$:		
2055	036510	004737	044766		JSR	PC,TIMOUT ;WAIT FOR SEARCH TO COMPLETE
2056						
2057	036514	004737	044154		JSR	PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
	036520	000404			BR	150\$;GO TO 150\$ IF NO ERROR
	036522	000240			NOP	;RETURN HERE IF ERROR
	036524	104000			EMT	;ERROR # DEFINED BY GET SUBROUTINE
	036526	000137	036720		JMP	230\$;GO TO 230\$ IF ERROR
2058	036532			150\$:		
2059	036532	004737	045152		JSR	PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
	036536	000405			BR	160\$;GO TO 160\$ IF NO ERROR
	036540	000240			NOP	;RETURN HERE IF ERROR
	036542	104000			EMT	;ERROR # DEFINED BY PRIERR SUBROUTINE
	036544	004736			JSR	PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
	036546	000137	036720		JMP	230\$;GO TO 230\$ IF ERROR
2060	036552			160\$:		
2061	036552	004737	057666		JSR	PC,SCHSTS ;GO VERIFY SEARCH OPERATION
	036556	000405			BR	170\$;GO TO 170\$ IF NO ERROR
	036560	000240			NOP	;RETURN HERE IF ERROR
	036562	104000			EMT	;ERROR # DEFINED BY SCHSTS SUBROUTINE
	036564	004736			JSR	PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
	036566	000137	036720		JMP	230\$;GO TO 230\$ IF ERROR
2062	036572			170\$:		
2063	036572	004737	051644		JSR	PC,CMPEERRSTS ;CHECK ANY ERRORS NOT MASKED
	036576	115760			.WORD	NDTMSK ;MASK FOR RMER1
	036600	000010			.WORD	DPE ;MASK FOR RMER2
	036602	000405			BR	180\$;GO TO 180\$ IF NO ERROR
	036604	000240			NOP	;RETURN HERE IF ERROR
	036606	104000			EMT	;ERROR # DEFINED BY CMPEERRSTS SUBROUTINE
	036610	004736			JSR	PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
	036612	000137	036720		JMP	230\$;GO TO 230\$ IF ERROR
2064	036616			180\$:		
2065	036616	004737	046004		JSR	PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
	036622	000405			BR	190\$;GO TO 190\$ IF NO ERROR
	036624	000240			NOP	;RETURN HERE IF ERROR

```

036626 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
036630 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
036632 000137 036720 JMP 230$ ;GO TO 230$ IF ERROR
2066 036636 190$: DEC R3 ;DECREMENT SEARCH CYLINDER
2067 036636 005303 INC R5 ;INCREMENT SEEK CYLINDER
2068 036640 005205 INC RMDAO ;INCREMENT SECTOR ADDRESS
2069 036642 005237 001416 CMP R4,RMDAO ;DONE ALL SECTORS??
2070 036646 020437 001416 BHIS 225$ ;NO - DO NEXT SECTOR
2071 036652 103020
2072
2073 036654 032737 010000 001442 BIT #FMT16,RMOFO ;DONE 16 BIT MODE YET ?
2074 036662 001016 BNE 230$ ;BR IF YES
2075 036664 012737 010000 001442 MOV #FMT16,RMOFO ;SET 16 BIT FORMAT
2076 036672 012704 000037 MOV #31.,R4 ;R4 = MAXIMUM SECTOR ADDRESS
2077 036676 012703 000632 MOV #410.,R3 ;R3 = SEARCH CYLINDER
2078 036702 012705 000633 MOV #411.,R5 ;R5 = SEEK CYLINDER
2079 036706 012737 000000 001416 MOV #0,RMDAO ;START AT TRACK = 0, SECTOR = 0
2080 036714 000137 036312 JMP 90$ ;GO TEST AGAIN
2081 036720
2082
2083

```

 ;*TEST 63 SEARCH INVALID SECTOR
 ;*****

```

036720
036720 000004 TST63: SCOPE ;SCOPE CALL
036722 000240 NOP ;START OF TEST
036724 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
036730 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
036734 013701 001464 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
036740 012737 000063 001226 MOV #63,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
2084
2085 036746 004737 043150 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
036752 054130 .WORD 054130 ;TASK DESCRIPTOR AS FOLLOWS:
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF "SKI" OR "PIP" IS SET
;VERIFY RECALIBRATION
036754 000404 BR 80$ ;GO TO 80$ IF NO ERROR
036756 000240 NOP ;RETURN HERE IF ERROR
036760 104000 EMT ;ERROR # DEFINED BY TSTPRP SUBROUTINE
036762 000137 037320 JMP 190$ ;GO TO 190$ IF ERROR
2086 036766 80$: MOV #0,RMOFO ;SET 18 BIT FORMAT
2087 036766 012737 000000 001442 MOV #0,RMDCO ;CYLINDER = 0
2088 036774 012737 000000 001444 MOV #30.,RMDAO ;FIRST INVALID SECTOR = 30.
2089 037002 012737 000036 001416 MOV #SEARCH!GO,RMCS10 ;SEARCH COMMAND
2090 037010 012737 000031 001410 MOV #PUTINX,R2 ;WRITE REGISTER TABLE FOR
2091 037016 012702 001543 MOV #RMDC,(R2)+ ;COMMAND
2092 037022 112722 000034 MOV #RMOF,(R2)+
2093 037026 112722 000032 MOV #RMDA,(R2)+
2094 037032 112722 000006 MOV #RMCS1,(R2)+
2095 037036 112722 000000 MOV #200,(R2)+ ;TERMINATE TABLE
2096 037042 112722 000200 JSR PC,GETSTS ;SETUP STATUS FETCH
2097 037046 004737 044070
2098 037052
2099 037052 004737 053530 JSR PC,CNTCLR ;GO ISSUE CONTROLLER CLEAR

```

	037056	000404		BR	95\$:GO TO 95\$ IF NO ERROR
	037060	000240		NOP		:RETURN HERE IF ERROR
	037062	104000		EMT		:ERROR NUMBER DEFINED BY SUBROUTINE
2100	037064	000137	037320	JMP	190\$:GO TO 190\$ IF ERROR
	037070				95\$:	
2101	037070	004737	044424	JSR	PC,PUT	:GO WRITE REGISTER(S) WITH PUT SUBROUTINE
	037074	000404		BR	100\$:GO TO 100\$ IF NO ERROR
	037076	000240		NOP		:RETURN HERE IF ERROR
	037100	104000		EMT		:ERROR # DEFINED BY PUT SUBROUTINE
2102	037102	000137	037320	JMP	190\$:GO TO 190\$ IF ERROR
	037106				100\$:	
2103	037106	004737	044766	JSR	PC,TIMOUT	:WAIT FOR GO TO RESET
2104						
2105	037112	004737	044154	JSR	PC,GET	:GO READ REGISTER(S) WITH GET SUBROUTINE
	037116	000404		BR	120\$:GO TO 120\$ IF NO ERROR
	037120	000240		NOP		:RETURN HERE IF ERROR
	037122	104000		EMT		:ERROR # DEFINED BY GET SUBROUTINE
2106	037124	000137	037320	JMP	190\$:GO TO 190\$ IF ERROR
	037130				120\$:	
2107	037130	004737	045152	JSR	PC,PRIERR	:GO CHECK FOR PRIMARY ERRORS
	037134	000405		BR	130\$:GO TO 130\$ IF NO ERROR
	037136	000240		NOP		:RETURN HERE IF ERROR
	037140	104000		EMT		:ERROR # DEFINED BY PRIERR SUBROUTINE
	037142	004736		JSR	PC,@(SP)+	:GO BACK FOR MORE ERROR CHECKS
2108	037144	000137	037320	JMP	190\$:GO TO 190\$ IF ERROR
	037150				130\$:	
2109	037150	004737	057666	JSR	PC,SCHSTS	:GO VERIFY SEARCH OPERATION
	037154	000405		BR	140\$:GO TO 140\$ IF NO ERROR
	037156	000240		NOP		:RETURN HERE IF ERROR
	037160	104000		EMT		:ERROR # DEFINED BY SCHSTS SUBROUTINE
	037162	004736		JSR	PC,@(SP)+	:GO BACK FOR MORE ERROR CHECKS
2110	037164	000137	037320	JMP	190\$:GO TO 190\$ IF ERROR
	037170				140\$:	
2111	037170	004737	061232	JSR	PC,STCDRVSTS	:GO CHECK FOR CHANGES IN DRIVE STATUS
	037174	000405		BR	150\$:GO TO 150\$ IF NO ERROR
	037176	000240		NOP		:RETURN HERE IF ERROR
	037200	104000		EMT		:ERROR # DEFINED BY STCDRVSTS SUBROUTINE
	037202	004736		JSR	PC,@(SP)+	:GO BACK FOR MORE ERROR CHECKS
2112	037204	000137	037320	JMP	190\$:GO TO 190\$ IF ERROR
	037210				150\$:	
2113	037210	004737	051644	JSR	PC,COMPERRSTS	:CHECK ANY ERRORS NOT MASKED
	037214	117760		.WORD	IAE!NDTMSK	:MASK FOR RMER1
	037216	000010		.WORD	DPE	:MASK FOR RMER2
	037220	000405		BR	160\$:GO TO 160\$ IF NO ERROR
	037222	000240		NOP		:RETURN HERE IF ERROR
	037224	104000		EMT		:ERROR # DEFINED BY COMPERRSTS SUBROUTINE
	037226	004736		JSR	PC,@(SP)+	:GO BACK FOR MORE ERROR CHECKS
2114	037230	000137	037320	JMP	190\$:GO TO 190\$ IF ERROR
	037234				160\$:	
2115	037234	004737	046004	JSR	PC,SECERR	:GO CHECK FOR SECONDARY ERRORS
	037240	000405		BR	170\$:GO TO 170\$ IF NO ERROR
	037242	000240		NOP		:RETURN HERE IF ERROR
	037244	104000		EMT		:ERROR # DEFINED BY SECERR SUBROUTINE
	037246	004736		JSR	PC,@(SP)+	:GO BACK FOR MORE ERROR CHECKS
2116	037250	000137	037320	JMP	190\$:GO TO 190\$ IF ERROR
	037254				170\$:	
2117	037254	005237	001416	INC	RMDAO	:INCREMENT SECTOR ADDRESS

```

2118 037260 022737 000037 001416      CMP      #31.,RMDAO      ;DONE ?
2119 037266 103012                BHIS     175$          ;BR IF NO
2120
2121 037270 032737 010000 001442      BIT      #FMT16,RMOFO  ;TEST FOR 16 BIT MODE YET ?
2122 037276 001010                BNE     190$          ;BR IF YES
2123 037300 012737 010000 001442      MOV      #FMT16,RMOFO  ;SET 16 BIT MODE
2124 037306 012737 000037 001416      MOV      #31.,RMDAO    ;SET INVALID SECTOR = 31.
2125 037314 000137 037052                JMP      90$          ;TEST NEXT SECTOR
2126 037320                175$:
2127                190$:
2128
    
```

```

:*****
:*TEST 64      SEARCH INVALID TRACK
:*****
TST64:
    
```

```

037320
037320 000004                SCOPE      ;SCOPE CALL
037322 000240                NOP        ;START OF TEST
037324 012706 001100      MOV      #STACK,SP    ;INITIALIZE STACK POINTER
037330 013700 001276      MOV      $BASE,R0     ;R0 = UNIBUS ADDRESS
037334 013701 001464      MOV      TSTQUE,R1    ;(R1) = DEVICE BEING TESTED
037340 012737 000064 001226      MOV      #64,$TESTN   ;:SET TEST NUMBER IN APT MAIL BOX
2129
2130 037346 004737 043150      JSR      PC,TSTPRP    ;PREPARE DEVICE FOR TEST
037352 054130      .WORD   054130      ;TASK DESCRIPTOR AS FOLLOWS:
                                ;CLEAR CONTROLLER & SELECT DEVICE
                                ;VERIFY CONTROLLER CLEAR OPERATION
                                ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                ;VERIFY PACK ACKNOWLEDGE
                                ;RECALIBRATE IF "SKI" OR "PIP" IS SET
                                ;VERIFY RECALIBRATION
                                ;GO TO 80$ IF NO ERROR
                                ;RETURN HERE IF ERROR
                                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                ;GO TO 180$ IF ERROR
037354 000404                BR        80$
037356 000240                NOP
037360 104000                EMT
037362 000137 037730      JMP      180$
2131 037366                80$:
2132 037366 012737 000031 001410      MOV      #SEARCH!GO,RMCS10 ;SEARCH COMMAND
2133 037374 012737 000000 001444      MOV      #0,RMDCO     ;CYLINDER = 0
2134 037402 013737 001332 001416      MOV      LSTRK,RMDAO  ;LAST TRACK, SECTOR = 0
2135 037410 105237 001417                INCB     RMDAO+1      ;SETUP FIRST INVALID TRACK
2136 037414 012737 000000 001442      MOV      #0,RMOFO     ;SET 18 BIT FORMAT
2137 037422 012702 001543      MOV      #PUTINX,R2   ;WRITE REGISTER INDEX TABLE
2138 037426 112722 000006      MOV      #RMDA,(R2)+
2139 037432 112722 000034      MOV      #RMDC,(R2)+
2140 037436 112722 000032      MOV      #RMOF,(R2)+
2141 037442 112722 000000      MOV      #RMCS1,(R2)+
2142 037446 112722 000200      MOV      #200,(R2)+   ;TERMINATE TABLE
2143 037452 004737 044070      MOV      #200,(R2)+
2144 037456                JSR      PC,GETSTS
2145 037456 004737 053530      JSR      PC,CNTCLR    ;GO ISSUE CONTROLLER CLEAR
037462 000404                BR        95$        ;GO TO 95$ IF NO ERROR
037464 000240                NOP        ;RETURN HERE IF ERROR
037466 104000                EMT        ;ERROR NUMBER DEFINED BY SUBROUTINE
037470 000137 037730      JMP      180$        ;GO TO 180$ IF ERROR
2146 037474                95$:
2147 037474 004737 044424      JSR      PC,PUT       ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
037500 000404                BR        100$      ;GO TO 100$ IF NO ERROR
037502 000240                NOP        ;RETURN HERE IF ERROR
037504 104000                EMT        ;ERROR # DEFINED BY PUT SUBROUTINE
    
```

2148	037506	000137	037730		JMP	180\$;GO TO 180\$ IF ERROR
2149	037512	004737	044766	100\$:	JSR	PC,TIMOUT	;WAIT FOR GO TO RESET
2150	037512	004737	044154		JSR	PC,GET	;GO READ REGISTER(S) WITH GET SUBROUTINE
2151	037516	000404			BR	110\$;GO TO 110\$ IF NO ERROR
	037522	000240			NOP		;RETURN HERE IF ERROR
	037524	104000			EMT		;ERROR # DEFINED BY GET SUBROUTINE
	037526	000137	037730		JMP	180\$;GO TO 180\$ IF ERROR
2152	037534	004737	045152	110\$:	JSR	PC,PRIERR	;GO CHECK FOR PRIMARY ERRORS
2153	037534	000405			BR	120\$;GO TO 120\$ IF NO ERROR
	037540	000240			NOP		;RETURN HERE IF ERROR
	037542	104000			EMT		;ERROR # DEFINED BY PRIERR SUBROUTINE
	037544	004736			JSR	PC,@(SP)+	;GO BACK FOR MORE ERROR CHECKS
	037546	000137	037730		JMP	180\$;GO TO 180\$ IF ERROR
2154	037554	004737	057666	120\$:	JSR	PC,SCHSTS	;GO VERIFY SEARCH OPERATION
2155	037554	000405			BR	130\$;GO TO 130\$ IF NO ERROR
	037560	000240			NOP		;RETURN HERE IF ERROR
	037562	104000			EMT		;ERROR # DEFINED BY SCHSTS SUBROUTINE
	037564	004736			JSR	PC,@(SP)+	;GO BACK FOR MORE ERROR CHECKS
	037566	000137	037730		JMP	180\$;GO TO 180\$ IF ERROR
2156	037574	004737	061232	130\$:	JSR	PC,STCDRVSTS	;GO CHECK FOR CHANGES IN DRIVE STATUS
2157	037574	000405			BR	140\$;GO TO 140\$ IF NO ERROR
	037600	000240			NOP		;RETURN HERE IF ERROR
	037602	104000			EMT		;ERROR # DEFINED BY STCDRVSTS SUBROUTINE
	037604	004736			JSR	PC,@(SP)+	;GO BACK FOR MORE ERROR CHECKS
	037606	000137	037730		JMP	180\$;GO TO 180\$ IF ERROR
2158	037614	004737	051644	140\$:	JSR	PC,COMPERRSTS	;CHECK ANY ERRORS NOT MASKED
2159	037614	117760			.WORD	NDTMSK!IAE	;MASK FOR RMER1
	037620	000010			.WORD	DPE	;MASK FOR RMER2
	037622	000405			BR	150\$;GO TO 150\$ IF NO ERROR
	037624	000240			NOP		;RETURN HERE IF ERROR
	037626	104000			EMT		;ERROR # DEFINED BY COMPERRSTS SUBROUTINE
	037630	004736			JSR	PC,@(SP)+	;GO BACK FOR MORE ERROR CHECKS
	037632	000137	037730		JMP	180\$;GO TO 180\$ IF ERROR
2160	037640	004737	046004	150\$:	JSR	PC,SECERR	;GO CHECK FOR SECONDARY ERRORS
2161	037640	000405			BR	160\$;GO TO 160\$ IF NO ERROR
	037644	000240			NOP		;RETURN HERE IF ERROR
	037646	104000			EMT		;ERROR # DEFINED BY SECERR SUBROUTINE
	037650	004736			JSR	PC,@(SP)+	;GO BACK FOR MORE ERROR CHECKS
	037652	000137	037730		JMP	180\$;GO TO 180\$ IF ERROR
2162	037660	105237	001417	160\$:	INCB	RMDAO+1	;INCREMENT TRACK ADDRESS
2163	037660	123727	001417	000200	CMPB	RMDAO+1,#128.	;DONE ALL TRACKS ?
2164	037664	101414			BLOS	170\$;NO!!
2165	037672	032737	010000	001442	BIT	#FMT16,RMOFO	;DONE 16 BIT MODE ?
2166	037674	001012			BNE	180\$;YES!!
2167	037702	012737	010000	001442	MOV	#FMT16,RMOFO	;SET 16 BIT FORMAT
2168	037704	013737	001332	001416	MOV	LSTRK,RMDAO	;LAST TRACK, SECTOR = 0
2169	037712	105237	001417		INCB	RMDAO+1	;SETUP FIRST INVALID TRACK ADDRESS
2170	037720	000137	037456		JMP	90\$;TEST NEXT TRACK
2171	037724						
2172							

2173
2174 037730
2175
2176

180\$:

*TEST 65 SEARCH INVALID CYLINDER

TST65:

037730
037730 000004
037732 000240
037734 012706 001100
037740 013700 001276
037744 013701 001464
037750 012737 000065 001226
2177
2178 037756 004737 043150
037762 054130

SCOPE ;SCOPE CALL
NOP ;START OF TEST
MOV #STACK,SP ;INITIALIZE STACK POINTER
MOV \$BASE,R0 ;R0 = UNIBUS ADDRESS
MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
MOV #65,\$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
.WORD 054130 ;TASK DESCRIPTOR AS FOLLOWS:
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF "SKI" OR "PIP" IS SET
;VERIFY RECALIBRATION
BR 80\$;GO TO 80\$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
EMT ;ERROR # DEFINED BY TSTPRP SUBROUTINE
JMP 190\$;GO TO 190\$ IF ERROR

037764 000404
037766 000240
037770 104000
037772 000137 040330
2179 037776
2180 037776 012737 000031 001410
2181 040004 012737 001467 001444
2182 040012 012737 000000 001416
2183 040020 012737 000000 001442
2184 040026 012702 001543
2185 040032 112722 000032
2186 040036 112722 000034
2187 040042 112722 000006
2188 040046 112722 000000
2189 040052 112722 000200
2190 040056 004737 044070
2191 040062

80\$:

MOV #SEARCH!GO,RMCS10 ;SEARCH COMMAND
MOV #823.,RMDCO ;START AT FIRST INVALID CYLINDER
MOV #0,RMDAO ;TRACK = 0, SECTOR = 0
MOV #0,RMOFO ;SET 18 BIT FORMAT
MOV #PUTINX,R2 ;WRITE REGISTER INDEX TABLE
MOVB #RMOF,(R2)+
MOVB #RMDC,(R2)+
MOVB #RMDA,(R2)+
MOVB #RMCS1,(R2)+
MOVB #200,(R2)+
JSR PC,GETSTS ;SETUP STATUS FETCH

2192 040062 004737 053530
040066 000404
040070 000240
040072 104000
040074 000137 040330

90\$:

JSR PC,CNTCLR ;GO ISSUE CONTROLLER CLEAR
BR 95\$;GO TO 95\$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
EMT ;ERROR NUMBER DEFINED BY SUBROUTINE
JMP 190\$;GO TO 190\$ IF ERROR

2193 040100
2194 040100 004737 044424
040104 000404
040106 000240
040110 104000
040112 000137 040330

95\$:

JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
BR 100\$;GO TO 100\$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
EMT ;ERROR # DEFINED BY PUT SUBROUTINE
JMP 190\$;GO TO 190\$ IF ERROR

2195 040116
2196 040116 004737 044766
2197
2198 040122 004737 044154
040126 000404
040130 000240
040132 104000

100\$:

JSR PC,TIMOUT

JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
BR 110\$;GO TO 110\$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
EMT ;ERROR # DEFINED BY GET SUBROUTINE

```

165 SEARCH INVALID CYLINDER
2199 040134 000137 040330      110$: JMP      190$      ;GO TO 190$ IF ERROR
2200 040140 004737 045152      JSR      PC,PRIERR   ;GO CHECK FOR PRIMARY ERRORS
      040144 000405      BR        120$      ;GO TO 120$ IF NO ERROR
      040146 000240      NOP       ;RETURN HERE IF ERROR
      040150 104000      EMT      ;ERROR # DEFINED BY PRIERR SUBROUTINE
      040152 004736      JSR      PC,@(SP)+  ;GO BACK FOR MORE ERROR CHECKS
      040154 000137 040330      JMP      190$      ;GO TO 190$ IF ERROR
2201 040160 004737 057666      120$: JSR      PC,SCHSTS   ;GO VERIFY SEARCH OPERATION
2202 040164 000405      BR        130$      ;GO TO 130$ IF NO ERROR
      040166 000240      NOP       ;RETURN HERE IF ERROR
      040170 104000      EMT      ;ERROR # DEFINED BY SCHSTS SUBROUTINE
      040172 004736      JSR      PC,@(SP)+  ;GO BACK FOR MORE ERROR CHECKS
      040174 000137 040330      JMP      190$      ;GO TO 190$ IF ERROR
2203 040200 004737 061232      130$: JSR      PC,STCDRVSTS ;GO CHECK FOR CHANGES IN DRIVE STATUS
2204 040204 000405      BR        140$      ;GO TO 140$ IF NO ERROR
      040206 000240      NOP       ;RETURN HERE IF ERROR
      040210 104000      EMT      ;ERROR # DEFINED BY STCDRVSTS SUBROUTINE
      040212 004736      JSR      PC,@(SP)+  ;GO BACK FOR MORE ERROR CHECKS
      040214 000137 040330      JMP      190$      ;GO TO 190$ IF ERROR
2205 040220 004737 051644      140$: JSR      PC,CMPEERRSTS ;CHECK ANY ERRORS NOT MASKED
2206 040224 117760      .WORD    IAE!NDTMSK ;MASK FOR RMER1
      040226 000010      .WORD    DPE        ;MASK FOR RMER2
      040230 000405      BR        150$      ;GO TO 150$ IF NO ERROR
      040232 000240      NOP       ;RETURN HERE IF ERROR
      040234 104000      EMT      ;ERROR # DEFINED BY CMPEERRSTS SUBROUTINE
      040236 004736      JSR      PC,@(SP)+  ;GO BACK FOR MORE ERROR CHECKS
      040240 000137 040330      JMP      190$      ;GO TO 190$ IF ERROR
2207 040244 004737 046004      150$: JSR      PC,SECERR    ;GO CHECK FOR SECONDARY ERRORS
2208 040250 000405      BR        160$      ;GO TO 160$ IF NO ERROR
      040252 000240      NOP       ;RETURN HERE IF ERROR
      040254 104000      EMT      ;ERROR # DEFINED BY SECERR SUBROUTINE
      040256 004736      JSR      PC,@(SP)+  ;GO BACK FOR MORE ERROR CHECKS
      040260 000137 040330      JMP      190$      ;GO TO 190$ IF ERROR
2209 040264 005237 001444      160$: INC      RMDCO      ;INCREMENT CYLINDER ADDRESS
2210 040264 023727 001444 002000  CMP      RMDCO,#1024. ;DONE ALL CYLINDERS ?
2211 040270 002412      BLT      170$      ;NO!!
2212 040276 032737 010000 001442  BIT      #FMT16,RMOFO ;SET 16 BIT FORMAT
2213 040300 001010      BNE      190$      ;YES!!
2214 040306 012737 010000 001442  MOV      #FMT16,RMOFO ;SET 16 BIT FORMAT
2215 040310 012737 001467 001444  MOV      #823.,RMDCO  ;START AT FIRST INVALID CYLINDER
2216 040316 000137 040062      170$: JMP      90$
2217 040324 000137 040062
2218 040330
2219 040330
2220 040330      190$:
2221
2222

```

```

:*****
:*TEST 66      IVC SEARCH TEST
:*****

```

```

TST66:
040330      SCOPE      ;SCOPE CALL
040330 000004      NOP        ;START OF TEST
040332 000240

```



```

040334 012706 001100      MOV    #STACK,SP      ;INITIALIZE STACK POINTER
040340 013700 001276      MOV    $BASE,R0       ;R0 = UNIBUS ADDRESS
040344 013701 001464      MOV    TSTQUE,R1      ;(R1) = DEVICE BEING TESTED
040350 012737 000066 001226  MOV    #66,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX

2223
2224 040356 004737 043150  JSR    PC,TSTPRP      ;PREPARE DEVICE FOR TEST
040362 054130          .WORD 054130        ;TASK DESCRIPTOR AS FOLLOWS:
                                           ;CLEAR CONTROLLER & SELECT DEVICE
                                           ;VERIFY CONTROLLER CLEAR OPERATION
                                           ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                           ;VERIFY PACK ACKNOWLEDGE
                                           ;RECALIBRATE IF "SKI" OR "PIP" IS SET
                                           ;VERIFY RECALIBRATION
040364 000404          BR     80$           ;GO TO 80$ IF NO ERROR
040366 000240          NOP
040370 104000          EMT
040372 000137 040710  JMP    160$         ;RETURN HERE IF ERROR
                                           ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                           ;GO TO 160$ IF ERROR

2225 040376          80$:
2226 040376 112737 000024 001543  MOVB  #RMMR1,PUTINX   ;SETUP PUT INDEX TABLE
040404 112737 000200 001544  MOVB  #200,PUTINX+1   ;SET TERMINATOR BYTE
040412 012737 000001 001434  MOV   #DMD,RMMR10     ;SET RMMR1 OUTPUT BUFFER = DMD
040420 004737 044424  JSR   PC,PUT          ;GO WRITE RMMR1 VIA PUT SUBROUTINE
040424 000404          BR     90$           ;GO TO 90$ IF NO ERROR
040426 000240          NOP
040430 104000          EMT
040432 000137 040710  JMP    160$         ;RETURN HERE IF ERROR
                                           ;ERROR DEFINED BY PUT SUBROUTINE
                                           ;GO TO 160$ IF ERROR

2227 040436 012737 000000 001434 90$:  MOV   #0,RMMR10       ;RESET DIAGNOSTIC MODE
2228 040444 012737 000000 001444  MOV   #0,RMDCO        ;CYLINDER = 0
2229 040452 012737 000000 001416  MOV   #0,RMDAO        ;TRACK = 0, SECTOR = 0
2230 040460 012737 000000 001442  MOV   #0,RMOFO        ;18 BIT FORMAT
2231 040466 012737 000031 001410  MOV   #SEARCH!GO,RMCS10 ;SEARCH COMMAND
2232 040474 012702 001543  MOV   #PUTINX,R2      ;LOAD INDEX TABLE
2233 040500 112722 000024  MOVB  #RMMR1,(R2)+
2234 040504 112722 000034  MOVB  #RMDC,(R2)+
2235 040510 112722 000006  MOVB  #RMDA,(R2)+
2236 040514 112722 000032  MOVB  #RMOF,(R2)+
2237 040520 112722 000000  MOVB  #RMCS1,(R2)+
2238 040524 112722 000200  MOVB  #200,(R2)+
2239 040530 004737 044070  JSR   PC,GETSTS
2240 040534 004737 044424  JSR   PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
040540 000404          BR     100$         ;GO TO 100$ IF NO ERROR
040542 000240          NOP
040544 104000          EMT
040546 000137 040710  JMP    160$         ;RETURN HERE IF ERROR
                                           ;ERROR # DEFINED BY PUT SUBROUTINE
                                           ;GO TO 160$ IF ERROR

2241 040552          100$:
2242 040552 004737 044766  JSR   PC,TIMOUT      ;WAIT FOR GO TO RESET
2243
2244 040556 004737 044154  JSR   PC,GET         ;GO READ REGISTER(S) WITH GET SUBROUTINE
040562 000404          BR     110$         ;GO TO 110$ IF NO ERROR
040564 000240          NOP
040566 104000          EMT
040570 000137 040710  JMP    160$         ;RETURN HERE IF ERROR
                                           ;ERROR # DEFINED BY GET SUBROUTINE
                                           ;GO TO 160$ IF ERROR

2245 040574          110$:
2246 040574 004737 045152  JSR   PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
040600 000405          BR     120$         ;GO TO 120$ IF NO ERROR
040602 000240          NOP
040604 104000          EMT
                                           ;RETURN HERE IF ERROR
                                           ;ERROR # DEFINED BY PRIERR SUBROUTINE

```

```

040606 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
040610 000137 040710 JMP 160$ ;GO TO 160$ IF ERROR
2247 040614 120$: BIT #IVC,RMER2I ;DID IVC SET??
2248 040614 032737 010000 001376 BNE 130$ ;YES!!
2249 040622 001012 MOV #IVC,$GDDAT ;EXPECTED STATUS
2250 040624 012737 010000 001140 MOV RMER2I,$BDDAT ;RECEIVED STATUS
2251 040632 013737 001376 001142 BIC #^CIVC,$BDDAT
2252 040640 042737 167777 001142 EMT 260
2253 040646 104260
2254 040650 130$: JSR PC,CMPERRSTS ;CHECK ANY ERRORS NOT MASKED
2255 040650 004737 051644 .WORD NDTMSK ;MASK FOR RMER1
040654 115760 .WORD IVC!DPE ;MASK FOR RMER2
040656 010010 BR 150$ ;GO TO 150$ IF NO ERROR
040660 000405 NOP ;RETURN HERE IF ERROR
040662 000240 EMT ;ERROR # DEFINED BY CMPERRSTS SUBROUTINE
040664 104000 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
040666 004736 JMP 160$ ;GO TO 160$ IF ERROR
2256 040670 000137 040710
2257 040674 150$: JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
040674 004737 046004 BR 160$ ;GO TO 160$ IF NO ERROR
040700 000403 NOP ;RETURN HERE IF ERROR
040702 000240 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
040704 104000 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
040706 004736
2258 040710 160$:
2259
2260

```

```

:*****
:*TEST 67 ABORT SEARCH TEST
:*****
TST67:

```

```

040710 000004 SCOPE ;SCOPE CALL
040710 000004 NOP ;START OF TEST
040712 000240 MOV #STACK,SP ;INITIALIZE STACK POINTER
040714 012706 001100 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
040720 013700 001276 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
040724 013701 001464 MOV #67,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
2261 040730 012737 000067 001226
2262 040736 004737 043150 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
040742 054130 .WORD 054130 ;TASK DESCRIPTOR AS FOLLOWS:
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF "SKI" OR "PIP" IS SET
;VERIFY RECALIBRATION
;GO TO 80$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 160$ IF ERROR
040744 000404 BR 80$
040746 000240 NOP
040750 104000 EMT
040752 000137 041254 JMP 160$
2263 040756 80$: MOV #UNS,RMER10 ;SET UNSAFE ERROR
2264 040756 012737 040000 001424 MOV #SEARCH!GO,RMCS10 ;SEARCH COMMAND
2265 040764 012737 000031 001410 MOV #0,RMDCO ;CYLINDER 0
2266 040772 012737 000000 001444 MOV #0,RMDAO ;SECTOR = 0, TRACK = 0
2267 041000 012737 000000 001416 MOV #PUTINX,R2 ;WRITE REGISTER INDEX TABLE
2268 041006 012702 001543 MOVB #RMDC,(R2)+
2269 041012 112722 000034 MOVB #RMDA,(R2)+
2270 041016 112722 000006

```

2271	041022	112722	000014		MOVB	#RMER1,(R2)+	
2272	041026	112722	000000		MOVB	#RMCS1,(R2)+	
2273	041032	112722	000200		MOVB	#200,(R2)+	: TERMINATE TABLE
2274	041036	004737	044070		JSR	PC,GETSTS	: SETUP FOR STATUS
2275	041042	004737	044424		JSR	PC,PUT	: GO WRITE REGISTER(S) WITH PUT SUBROUTINE
	041046	000404			BR	90\$: GO TO 90\$ IF NO ERROR
	041050	000240			NOP		: RETURN HERE IF ERROR
	041052	104000			EMT		: ERROR # DEFINED BY PUT SUBROUTINE
	041054	000137	041254		JMP	160\$: GO TO 160\$ IF ERROR
2276	041060			90\$:			
2277	041060	004737	044154		JSR	PC,GET	: GO READ REGISTER(S) WITH GET SUBROUTINE
	041064	000404			BR	100\$: GO TO 100\$ IF NO ERROR
	041066	000240			NOP		: RETURN HERE IF ERROR
	041070	104000			EMT		: ERROR # DEFINED BY GET SUBROUTINE
	041072	000137	041254		JMP	160\$: GO TO 160\$ IF ERROR
2278	041076			100\$:			
2279	041076	032737	020000	001346	BIT	#PIP,RMSDI	: IS PIP SET??
2280	041104	001412			BEQ	110\$: NO!!
2281	041106	012737	000000	001140	MOV	#0,\$GDDAT	: EXPECTED STATUS
2282	041114	013737	001346	001142	MOV	RMSDI,\$BDDAT	: RECEIVED STATUS
2283	041122	042737	157777	001142	BIC	#^CPIP,\$BDDAT	
2284	041130	104261			EMT	261	
2285	041132			110\$:			
2286	041132	004737	044766		JSR	PC,TIMOUT	: WAIT FOR GO TO RESET
2287							
2288	041136	004737	044154		JSR	PC,GET	: GO READ REGISTER(S) WITH GET SUBROUTINE
	041142	000404			BR	120\$: GO TO 120\$ IF NO ERROR
	041144	000240			NOP		: RETURN HERE IF ERROR
	041146	104000			EMT		: ERROR # DEFINED BY GET SUBROUTINE
	041150	000137	041254		JMP	160\$: GO TO 160\$ IF ERROR
2289	041154			120\$:			
2290	041154	004737	045152		JSR	PC,PRIERR	: GO CHECK FOR PRIMARY ERRORS
	041160	000405			BR	130\$: GO TO 130\$ IF NO ERROR
	041162	000240			NOP		: RETURN HERE IF ERROR
	041164	104000			EMT		: ERROR # DEFINED BY PRIERR SUBROUTINE
	041166	004736			JSR	PC,@(SP)+	: GO BACK FOR MORE ERROR CHECKS
	041170	000137	041254		JMP	160\$: GO TO 160\$ IF ERROR
2291	041174			130\$:			
2292	041174	004737	061232		JSR	PC,STCDRVSTS	: GO CHECK FOR CHANGES IN DRIVE STATUS
	041200	000405			BR	140\$: GO TO 140\$ IF NO ERROR
	041202	000240			NOP		: RETURN HERE IF ERROR
	041204	104000			EMT		: ERROR # DEFINED BY STCDRVSTS SUBROUTINE
	041206	004736			JSR	PC,@(SP)+	: GO BACK FOR MORE ERROR CHECKS
	041210	000137	041254		JMP	160\$: GO TO 160\$ IF ERROR
2293	041214			140\$:			
2294	041214	004737	051644		JSR	PC,CMPEERRSTS	: CHECK ANY ERRORS NOT MASKED
	041220	155760			.WORD	NDIMSK!UNS	: MASK FOR RMER1
	041222	000010			.WORD	DPE	: MASK FOR RMER2
	041224	000405			BR	150\$: GO TO 150\$ IF NO ERROR
	041226	000240			NOP		: RETURN HERE IF ERROR
	041230	104000			EMT		: ERROR # DEFINED BY CMPEERRSTS SUBROUTINE
	041232	004736			JSR	PC,@(SP)+	: GO BACK FOR MORE ERROR CHECKS
	041234	000137	041254		JMP	160\$: GO TO 160\$ IF ERROR
2295	041240			150\$:			
2296	041240	004737	046004		JSR	PC,SECERR	: GO CHECK FOR SECONDARY ERRORS
	041244	000403			BR	160\$: GO TO 160\$ IF NO ERROR
	041246	000240			NOP		: RETURN HERE IF ERROR

```

041250 104000          EMT          ;ERROR # DEFINED BY SECERR SUBROUTINE
041252 004736          JSR          PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
2297 041254          160$:
2298
2299
*****
:*TEST 70          SEARCH AT OFFSET
*****
TST70:
041254          SCOPE          ;SCOPE CALL
041254 000004          NOP          ;START OF TEST
041256 000240          MOV          #STACK,SP ;INITIALIZE STACK POINTER
041260 012706 001100    MOV          $BASE,R0 ;RO = UNIBUS ADDRESS
041264 013700 001276    MOV          TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
041270 013701 001464    MOV          #70,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
041274 012737 000070 001226
2300
2301 041302 004737 043150 JSR          PC,TSTPRP ;PREPARE DEVICE FOR TEST
041306 054130          .WORD      054130 ;TASK DESCRIPTOR AS FOLLOWS:
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF "SKI" OR "PIP" IS SET
;VERIFY RECALIBRATION
041310 000404          BR          80$ ;GO TO 80$ IF NO ERROR
041312 000240          NOP          ;RETURN HERE IF ERROR
041314 104000          EMT          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
041316 000137 041612    JMP          180$ ;GO TO 180$ IF ERROR
2302 041322          80$:
2303 041322 012737 000000 001444    MOV          #0,RMDCO ;CYLINDER = 0
2304 041330 012737 000000 001416    MOV          #0,RMDAO ;TRACK = 0, SECTOR = 0
2305 041336 012737 010000 001442    MOV          #FMT16,RMOFO ;16 BIT FORMAT
2306 041344 012737 000015 001410    MOV          #OFFSET!GO,RMCS10 ;OFFSET COMMAND
2307 041352 012702 001543    MOV          #PUTINX,R2 ;SETUP REGISTER INDEX TABLE
2308 041356 112722 000006    MOVB        #RMDA,(R2)+ ;FOR SEEK TO CYLINDER 0
2309 041362 112722 000034    MOVB        #RMDC,(R2)+
2310 041366 112722 000032    MOVB        #RMOF,(R2)+
2311 041372 112722 000000    MOVB        #RMCS1,(R2)+
2312 041376 112722 000200    MOVB        #200,(R2)+
2313 041402 004737 044424    JSR          PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
041406 000404          BR          90$ ;GO TO 90$ IF NO ERROR
041410 000240          NOP          ;RETURN HERE IF ERROR
041412 104000          EMT          ;ERROR # DEFINED BY PUT SUBROUTINE
041414 000137 041612    JMP          180$ ;GO TO 180$ IF ERROR
2314 041420          90$:
2315 041420 004737 044070    JSR          PC,GETSTS ;SETUP FOR STATUS FETCH
2316 041424 004737 044766    JSR          PC,TIMOUT
2317 041430          120$:
2318 041430 112737 000000 001543    MOVB        #RMCS1,PUTINX ;LOAD REGISTER INDEX TABLE
2319 041436 112737 000200 001544    MOVB        #200,PUTINX+1
2320 041444 012737 000031 001410    MOV          #SEARCH!GO,RMCS10 ;STORE SEARCH COMMAND
2321 041452 004737 044424    JSR          PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
041456 000404          BR          130$ ;GO TO 130$ IF NO ERROR
041460 000240          NOP          ;RETURN HERE IF ERROR
041462 104000          EMT          ;ERROR # DEFINED BY PUT SUBROUTINE
041464 000137 041612    JMP          180$ ;GO TO 180$ IF ERROR
2322 041470          130$:
2323 041470 004737 044766    JSR          PC,TIMOUT ;WAIT FOR SEARCH TO COMPLETE

```

```

2324
2325 041474 004737 044154      JSR    PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      041500 000404          BR      140$          ;GO TO 140$ IF NO ERROR
      041502 000240          NOP                    ;RETURN HERE IF ERROR
      041504 104000          EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      041506 000137 041612      JMP      180$          ;GO TO 180$ IF ERROR
2326 041512
2327 041512 004737 045152      JSR    PC,PRIERR       ;GO CHECK FOR PRIMARY ERRORS
      041516 000405          BR      150$          ;GO TO 150$ IF NO ERROR
      041520 000240          NOP                    ;RETURN HERE IF ERROR
      041522 104000          EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      041524 004736          JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      041526 000137 041612      JMP      180$          ;GO TO 180$ IF ERROR
2328 041532
2329 041532 004737 057666      JSR    PC,SCHSTS       ;GO VERIFY SEARCH OPERATION
      041536 000405          BR      160$          ;GO TO 160$ IF NO ERROR
      041540 000240          NOP                    ;RETURN HERE IF ERROR
      041542 104000          EMT                    ;ERROR # DEFINED BY SCHSTS SUBROUTINE
      041544 004736          JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      041546 000137 041612      JMP      180$          ;GO TO 180$ IF ERROR
2330 041552
2331 041552 004737 051644      JSR    PC,CMPEERRSTS   ;CHECK ANY ERRORS NOT MASKED
      041556 115760          .WORD  NDTMSK         ;MASK FOR RMER1
      041560 000010          .WORD  DPE            ;MASK FOR RMER2
      041562 000405          BR      170$          ;GO TO 170$ IF NO ERROR
      041564 000240          NOP                    ;RETURN HERE IF ERROR
      041566 104000          EMT                    ;ERROR # DEFINED BY CMPEERRSTS SUBROUTINE
      041570 004736          JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      041572 000137 041612      JMP      180$          ;GO TO 180$ IF ERROR
2332 041576
2333 041576 004737 046004      JSR    PC,SECERR       ;GO CHECK FOR SECONDARY ERRORS
      041602 000403          BR      180$          ;GO TO 180$ IF NO ERROR
      041604 000240          NOP                    ;RETURN HERE IF ERROR
      041606 104000          EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      041610 004736          JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
2334 041612
2335
2343

```

```

*****
*TEST 71      HEAD ALIGNMENT SEEK
*
*HEAD ALIGNMENT FOR AN RM05/3/2 DRIVE, IS PERFORMED AT CYLINDER
*245 DECIMAL USING THE CE PACK. A 60 MINUTE WARMUP IS REQUIRED IF
*THE PACK WAS NOT IN THE DRIVE PRIOR TO HEAD ALIGNMENT OPERATION.
*
*****

```

```

041612
041612 000004
041614 000240
041616 012706 001100
041622 013700 001276
041626 013701 001464
041632 012737 000001 001206
041640 012737 000071 001226
2344
2345 041646
2346 041646 004737 043150      JSR    PC,TSTPRP       ;PREPARE DEVICE FOR TEST
      041652 054130          .WORD  054130         ;TASK DESCRIPTOR AS FOLLOWS:

```

```

TST71:
SCOPE          ;SCOPE CALL
NOP            ;START OF TEST
MOV #STACK,SP ;INITIALIZE STACK POINTER
MOV $BASE,R0  ;R0 = UNIBUS ADDRESS
MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
MOV #1,$TIMES ;DO 1 ITERATION
MOV #71,$TESTN ;SET TEST NUMBER IN APT MAIL BOX

```

```

                                ;CLEAR CONTROLLER & SELECT DEVICE
                                ;VERIFY CONTROLLER CLEAR OPERATION
                                ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                ;VERIFY PACK ACKNOWLEDGE
                                ;RECALIBRATE IF "SKI" OR "PIP" IS SET
                                ;VERIFY RECALIBRATION
                                ;GO TO 80$ IF NO ERROR
                                ;RETURN HERE IF ERROR
                                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                ;GO TO 140$ IF ERROR
041654 000404 BR 80$
041656 000240 NOP
041660 104000 EMT
041662 000137 042100 JMP 140$
2347 041666 80$:
2348 041666 012737 000365 001444 MOV #245.,RMDCO ;CYLINDER = 245.
2349 041674 012737 000000 001416 MOV #0,RMDAO ;TRACK = 0, SECTOR = 0
2350 041702 012737 000005 001410 MOV #SEEK!GO,RMCS10 ;LOAD SEEK COMMAND IN BUFFER
2351 041710 012702 001543 MOV #PUTINX,R2 ;R2 POINTS TO REGISTER TABLE
2352 041714 112722 000034 MOVB #RMDC,(R2)+ ;WRITE REGISTER INDEX TABLE
2353 041720 112722 000006 MOVB #RMDA,(R2)+
2354 041724 112722 000000 MOVB #RMCS1,(R2)+
2355 041730 112722 000200 MOVB #200,(R2)+ ;WRITE TERMINATOR
2356 041734 004737 044424 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
041740 000404 BR 90$ ;GO TO 90$ IF NO ERROR
041742 000240 NOP ;RETURN HERE IF ERROR
041744 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
041746 000137 042100 JMP 140$ ;GO TO 140$ IF ERROR
2357 041752 004737 044070 90$: JSR PC,GETSTS ;SETUP FOR STATUS FETCH
2358 041756 004737 044766 JSR PC,TIMOUT ;WAIT FOR SEEK TO COMPLETE
2359
2360 041762 004737 044154 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
041766 000404 BR 100$ ;GO TO 100$ IF NO ERROR
041770 000240 NOP ;RETURN HERE IF ERROR
041772 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
041774 000137 042100 JMP 140$ ;GO TO 140$ IF ERROR
2361 042000 100$:
2362 042000 004737 045152 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
042004 000405 BR 110$ ;GO TO 110$ IF NO ERROR
042006 000240 NOP ;RETURN HERE IF ERROR
042010 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
042012 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
042014 000137 042100 JMP 140$ ;GO TO 140$ IF ERROR
2363 042020 110$:
2364 042020 004737 052270 JSR PC,SEKSTS ;GO VERIFY RESULTS OF SEEK OPERATION
042024 000405 BR 120$ ;GO TO 120$ IF NO ERROR
042026 000240 NOP ;RETURN HERE IF ERROR
042030 104000 EMT ;ERROR # DEFINED BY SEKSTS SUBROUTINE
042032 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
042034 000137 042100 JMP 140$ ;GO TO 140$ IF ERROR
2365 042040 120$:
2366 042040 004737 051644 JSR PC,COMPERRSTS ;CHECK ANY ERRORS NOT MASKED
042044 115760 .WORD ND1MSK ;MASK FOR RMER1
042046 000010 .WORD DPE ;MASK FOR RMER2
042050 000405 BR 130$ ;GO TO 130$ IF NO ERROR
042052 000240 NOP ;RETURN HERE IF ERROR
042054 104000 EMT ;ERROR # DEFINED BY COMPERRSTS SUBROUTINE
042056 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
042060 000137 042100 JMP 140$ ;GO TO 140$ IF ERROR
2367 042064 130$:
2368 042064 004737 046004 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS

```

CZRMAO RM05/3/2 FCTNL TST 1
T71 HEAD ALIGNMENT SEEK

MACRO V03.01 11-APR-80 13:04:53 PAGE 12-89

E 14

SEQ 0173

042070 000403
042072 000240
042074 104000
042076 004736
2369 042100
2370
2374

140\$:

BR 140\$
NOP
EMT
JSR PC,@(SP)+

;GO TO 140\$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY SECERR SUBROUTINE
;GO BACK FOR MORE ERROR CHECKS

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20

.SBTTL END OF SUB-PASS ROUTINE

;THIS IS THE END OF SUB-PASS ROUTINE. THIS ROUTINE IS USED TO
;TERMINATE THE OPERATION OF THE CURRENT DEVICE UNDER TEST AND
;SELECT THE NEXT DEVICE FOR TEST. IF THERE ARE NO MORE DEVICES
;TO TEST, EXIT IS MADE TO '\$EOP' ROUTINE. OTHERWISE, RETURN
;IS MADE TO 'READY' ROUTINE.

```
$EOSP:  SCOPE
        NOP
        MOV     TSTQUE,RO      ;GET POINTER TO TSTQUE
        ADD     #2,RO         ;ADJUST POINTER TO NEXT DEVICE
        MOV     RO,TSTQUE     ;SAVE POINTER TO TSTQUE
        TST     (RO)         ;ANY MORE DEVICES FOR TEST ?
        BEQ     1$           ;BR IF NO
        JMP     READY        ;YES, JUMP TO READY
1$:     MOV     #TSTQUE+2,TSTQUE ;INITIALIZE POINTER TO FIRST DEVICE IN
        ;TEST QUE TABLE
```

.SBTTL END OF PASS ROUTINE

```
::*****
;*INCREMENT THE PASS NUMBER ($PASS)
;*TYPE 'END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYYY'
;*WHERE XXXXX AND YYYYYY ARE DECIMAL NUMBERS
;*IF THERES A MONITOR GO TO IT
;*IF THERE ISN'T JUMP TO READY
```

```
042136 042136 000240
042140 042140 005037 001116
042144 042144 005037 001206
042150 042150 005237 001230
042154 042154 042737 100000 001230
042162 042162 005327
042164 042164 000001
042166 042166 003063
042170 042170 012737
042172 042172 000001
042174 042174 042164
042176 042176 104401 042204
042202 042202 000407

042222 042222 013746 001230
042226 042226 104405
042230 042230 104401 042236
042234 042234 000421

042300 042300 013746 001126
042304 042304 104405
042306 042306 104401 001217
042312 042312 005037 001126
042316 042316 013700 000042

$EOP:  NOP
        CLR     $STSTM        ;;ZERO THE TEST NUMBER
        CLR     $TIMES        ;;ZERO THE NUMBER OF ITERATIONS
        INC     $PASS         ;;INCREMENT THE PASS NUMBER
        BIC     #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
        DEC     (PC)+         ;;LOOP?
$EOPCT: .WORD 1
        BGT     $DOAGN        ;;YES
        MOV     (PC)+,@(PC)+  ;;RESTORE COUNTER
$ENDCT: .WORD 1
        TYPE    ,65$          ;;TYPE ASCIZ STRING
        BR      64$          ;;GET OVER THE ASCIZ
;;65$: .ASCIZ <12><15>/END PASS #/
64$:   MOV     $PASS,-(SP)    ;;SAVE $PASS FOR TYPEOUT
        ;;TYPE PASS NUMBER
        TYPDS   ;;GO TYPE--DECIMAL ASCII WITH SIGN
        TYPE    ,67$          ;;TYPE ASCIZ STRING
        BR      66$          ;;GET OVER THE ASCIZ
;;67$: .ASCIZ / TOTAL ERRORS SINCE LAST REPORT /
66$:   MOV     $ERTTL,-(SP)   ;;SAVE $ERTTL FOR TYPEOUT
        ;;TOTAL NUMBER OF ERRORS
        TYPDS   ;;GO TYPE--DECIMAL ASCII WITH SIGN
        TYPE    , $CRLF       ;;TYPE CARRIAGE RETURN, LINE FEED
        CLR     $ERTTL        ;;CLEAR ERROR TOTAL
$GET42: MOV     @#42,RO      ;;GET MONITOR ADDRESS
```



```
042322 001405          BEQ      $DOAGN      ;;BRANCH IF NO MONITOR
042324 000005          RESET          ;;CLEAR THE WORLD
042326 004710          SENDAD: JSR     PC,(R0)  ;;GO TO MONITOR
042330 000240          NOP              ;;SAVE ROOM
042332 000240          NOP              ;;FOR
042334 000240          NOP              ;;ACT11
042336          SENDAD: JSR     @PC)+      ;;RETURN
042336 000137          $RTNAD: .WORD  READY
042340 007756          $ENULL: .BYTE  -1,-1,0
042342          377      000          .EVEN      ;;NULL CHARACTER STRING
```

21

```

1      .SBTTL SUBROUTINES
2      ;*****
3      .SBTTL ERROR TYPEOUT ROUTINE
4
5      ;*THE ERROR TYPEOUT ROUTINE ASSEMBLES AND PRINTS INFORMATION
6      ;*REGARDING THE DETECTION OF AN ERROR AS FOLLOWS:
7      ;*
8      ;*   .UNIT NUMBER, TEST NUMBER, ERROR NUMBER AND PROGRAM COUNTER ARE
9      ;*PRINTED ON THE FIRST LINE;
10     ;*   .ERROR MESSAGE IS ASSEMBLED, FORMATTED AND PRINTED ON
11     ;*ONE OR MORE SUCCEEDING LINES;
12     ;*   .PAIRED LINES OF ERROR HEADERS AND ERROR DATA
13     ;*ARE PRINTED AFTER THE ERROR MESSAGE.
14
15     042346
16     042346 104414
17     042350 032777 020000 136576
18     042356 001402
19     042360 000137 043076
20
21     042364 104401 001217
22     042370 104401 043112
23     042374 013746 001234
24
25     042400 104403
26     042402 003
27     042403 000
28     042404 005037 043102
29     042410 013737 001226 043102
30     042416 104401 043120
31     042422 013746 043102
32
33     042426 104403
34     042430 003
35     042431 000
36     042432 005037 043104
37     042436 113737 001130 043104
38     042444 001406
39     042446 104401 043130
40     042452 013746 043104
41
42     042456 104403
43     042460 003
44     042461 000
45     042462 104401 043137
46     042466 013746 001132
47
48     042472 104403
49     042474 006
50     042475 001
51
52     042476 005737 043104
53     042502 001575
54     042504 104401 001217
55     042510 105037 043110
56     042514 105037 043111
57     042520 013700 043104

```

```

ERRTYP:
    SAVREG
    BIT #SW13,@SWR ;INHIBIT TYPEOUTS??
    BEQ 1$ ;NO!!
    JMP 21$ ;YES!!
;TYPE UNIT NUMBER, TEST NUMBER, ERROR NUMBER, AND PROGRAM COUNTER
1$:
    TYPE , $CRLF
    TYPE , ERTY00 ;TYPE "UNT#"
    MOV $UNIT,-(SP) ;;SAVE $UNIT FOR TYPEOUT
    ;;TYPE UNIT NUMBER
    TYPOS ;GO TYPE--OCTAL ASCII
    .BYTE 3 ;TYPE 3 DIGIT(S)
    .BYTE 0 ;SUPPRESS LEADING ZEROS
    CLR TSTNMB ;LOAD TEST NUMBER FOR
    MOV $TESTN,TSTNMB
    TYPE , ERTY01 ;TYPE "TST#"
    MOV TSTNMB,-(SP) ;;SAVE TSTNMB FOR TYPEOUT
    ;;TYPE TEST NUMBER
    TYPOS ;GO TYPE--OCTAL ASCII
    .BYTE 3 ;TYPE 3 DIGIT(S)
    .BYTE 0 ;SUPPRESS LEADING ZEROS
    CLR ERRNMB ;LOAD ERROR NUMBER FOR
    MOVB $ITEMB,ERRNMB ;TYPEOUT
    BEQ 2$ ;SKIP IF NO ERROR CALLED
    TYPE , ERTY02 ;TYPE "ERR#"
    MOV ERRNMB,-(SP) ;;SAVE ERRNMB FOR TYPEOUT
    ;;TYPE ERROR NUMBER
    TYPOS ;GO TYPE--OCTAL ASCII
    .BYTE 3 ;TYPE 3 DIGIT(S)
    .BYTE 0 ;SUPPRESS LEADING ZEROS
    2$:
    TYPE , ERTY03 ;TYPE "PC="
    MOV $ERRPC,-(SP) ;;SAVE $ERRPC FOR TYPEOUT
    ;;TYPE PROGRAM COUNTER
    TYPOS ;GO TYPE--OCTAL ASCII
    .BYTE 6 ;TYPE 6 DIGIT(S)
    .BYTE 1 ;TYPE LEADING ZEROS
;GENERATE POINTER TO ERROR TABLE UNLESS ERROR NUMBER IS 0
3$:
    TST ERRNMB ;WAS AN ERROR CALLED?
    BEQ 21$ ;NO!!
    TYPE , $CRLF ;YES-TYPE CRLF
    CLRB BOTFLG ;CLEAR BOT FLAG
    CLRB CHRCNT ;CLEAR CHARACTER COUNTER
    MOV ERRNMB,R0 ;R0 POINTS TO FIRST OF

```

```

42 042524 006300          ASL      R0          ;FOUR ENTRIES IN ERROR
43 042526 006300          ASL      R0          ;TABLE
44 042530 006300          ASL      R0
45 042532 062700 001562  ADD      #SERRTB-8.,R0
46 042536 011001          MOV      (R0),R1      ;R1 POINTS TO ERROR MESSAGE
47                                     ;TABLE
48 042540 001507          BEQ      13$          ;BRANCH IF NO ERROR MESSAGE
49                                     ;TYPE THE ERROR MESSAGE
50 042542 012102 4$:      MOV      (R1)+,R2      ;R2 = ADDRESS OF MESSAGE STRING
51 042544 001505          BEQ      12$          ;BRANCH IF END OF MESSAGE
52 042546 010237 042714  MOV      R2,11$      ;LOAD ADDRESS OF STRING
53 042552 005037 043106  CLR      BOTADR      ;CLEAR BOT ADDRESS
54 042556 112203 5$:      MOVB     (R2)+,R3      ;END OF STRING??
55 042560 001454          BEQ      10$          ;YES!!
56 042562 122703 000015  CMPB     #CR,R3      ;CARRIAGE RETURN??
57 042566 001003          BNE      6$          ;NO!!
58 042570 105037 043111  CLRB     CHRCNT      ;YES-CLEAR CHAR COUNT
59 042574 000770          BR       5$          ;GET NEXT CHARACTER
60 042576 122703 000012 6$:      CMPB     #LF,R3      ;LINE FEED??
61 042602 001765          BEQ      5$          ;YES-GET NEXT CHARACTER
62 042604 122703 000011  CMPB     #HT,R3      ;HORIZONTAL TAB??
63 042610 001007          BNE      8$          ;NO!!
64 042612 105237 043111 7$:      INCB     CHRCNT      ;ADJUST CHARACTER COUNT
65 042616 132737 000007 043111 BITB     #7,CHRCNT
66 042624 001372          BNE      7$
67 042626 000407          BR       9$
68 042630 105237 043111 8$:      INCB     CHRCNT      ;INCREMENT CHARACTER COUNT
69 042634 122703 000040  CMPB     #' ,R3      ;SPACE??
70 042640 001002          BNE      9$          ;NO!!
71 042642 010237 043106  MOV      R2,BOTADR    ;SAVE ADDRESS OF SPACE
72 042646 122737 000100 043111 9$:      CMPB     #64.,CHRCNT ;END OF LINE??
73 042654 103340          BHIS     5$          ;NO!!
74 042656 013704 043106  MOV      BOTADR,R4    ;GET ADDRESS OF LAST SPACE
75 042662 001007          BNE      90$         ;BRANCH IF SPACE DETECTED
76 042664 104401 001217  TYPE     ,$CRLF      ;TYPE CRLF
77 042670 105037 043111  CLRB     CHRCNT      ;CLEAR CHARACTER COUNT
78 042674 013702 042714  MOV      11$,R2      ;SET UP R2 FOR TESTING
79 042700 000726          BR       5$
80 042702 105044 90$:     CLRB     -(R4)      ;REPLACE SPACE
81 042704 112737 177777 043110  MOVB     #-1,BOTFLG   ;SET BOT FLAG
82 042712 104401 10$:     TYPE     ;TYPE ERROR MESSAGE STRING
83 042714 000000 11$:     .WORD    ;STRING ADDRESS GOES HERE
84 042716 105737 043110  TSTB     BOTFLG      ;WAS STRING TRUNCATED??
85 042722 001707          BEQ      4$          ;NO!!
86 042724 104401 001217  TYPE     ,$CRLF      ;YES-TYPE CRLF
87 042730 105037 043110  CLRB     BOTFLG      ;CLEAR BOT FLAG
88 042734 105037 043111  CLRB     CHRCNT      ;CLEAR CHARACTER COUNT
89 042740 013702 043106  MOV      BOTADR,R2    ;SETUP R2 FOR TESTING
90 042744 010237 042714  MOV      R2,11$      ;SETUP 11$ FOR TYPING
91 042750 112742 000040  MOVB     #' ,-(R2)    ;RESTORE SPACE
92 042754 105722          TSTB     (R2)+      ;RESTORE R2
93 042756 000677          BR       5$          ;TYPE REST OF STRING
94 042760
95 12$:
;TYPE ERROR HEADER AND ERROR DATA
96 042760 016001 000002 13$:     MOV      2(R0),R1    ;R1 POINTS TO ERROR HEADER TABLE
97 042760 001444          BEQ      21$         ;BRANCH IF NO HEADER
98 042764

```

99	042766	104401	001217		TYPE	,\$CRLF	;(ASSUME NO DATA)
100	042772	016002	000004		MOV	4(R0),R2	;R2 POINTS TO DATA ADDRESS TABLE
101	042776	016003	000006		MOV	6(R0),R3	;R3 POINTS TO FORMAT TABLE
102	043002	012137	043012	14\$:	MOV	(R1)+,15\$;PUT HEADER ADDRESS FOR TYPE
103	043006	001433			BEQ	21\$;BRANCH IF END OF HEADERS
104							; (ASSUME END OF DATA)
105	043010	104401			TYPE		
106	043012	000000		15\$:	.WORD	0	;HEADER ADDRESS GOES HERE
107	043014	104401	001217		TYPE	,\$CRLF	
108	043020	005702			TST	R2	;DATA WITH HEADER??
109	043022	001767			BEQ	14\$;NO!!
110	043024	012204			MOV	(R2)+,R4	;R4 POINTS TO DATA ADDRESS
111	043026	012305			MOV	(R3)+,R5	;R5 POINTS TO FORMAT
112	043030	105725		16\$:	TSTB	(R5)+	;WHAT KIND OF DATA??
113	043032	100407			BMI	18\$;BINARY
114	043034	001403			BEQ	17\$;OCTAL
115	043036	013446			MOV	@(R4)+,-(SP)	;DECIMAL
116	043040	104405			TYPDS		
117	043042	000405			BR	19\$	
118	043044	013446		17\$:	MOV	@(R4)+,-(SP)	
119	043046	104402			TYPOC		
120	043050	000402			BR	19\$	
121	043052	013446		18\$:	MOV	@(R4)+,-(SP)	
122	043054	104406			TYPBN		
123	043056	005714		19\$:	TST	(R4)	;MORE DATA??
124	043060	001403			BEQ	20\$;NO!!
125	043062	104401	043145		TYPE	,ERTY04	;YES-TYPE 2 SPACES
126	043066	000760			BR	16\$;AND CONTINUE
127	043070	104401	001217	20\$:	TYPE	,\$CRLF	;TYPE ONE BLANK LINE
128	043074	000742			BR	14\$;BEFORE NEXT HEADER
129	043076	104415		21\$:	RESREG		
130	043100	000207			RTS	PC	
131							
132	043102	000000			TSTNMB: .WORD	0	;TEST NUMBER
133	043104	000000			ERRNMB: .WORD	0	;ERROR NUMBER
134	043106	000000			BOTADR: .WORD	0	;BEGINNING OF TEXT ADDRESS
135	043110	000			BOTFLG: .BYTE	0	;BOT FLAG
136	043111	000			CHRCNT: .BYTE	0	;CHARACTER COUNT
137							
138	043112	125	116	111	ERTY00: .ASCIZ	@UNIT#@	
139	043120	054	040	124	ERTY01: .ASCIZ	@, TEST#@	
140	043130	054	040	105	ERTY02: .ASCIZ	@, ERR#@	
141	043137	054	040	120	ERTY03: .ASCIZ	@, PC=@	
142	043145	040	040	000	ERTY04: .ASCIZ	@ @	
143					.EVEN		
144							

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

.SBTTL TEST PREPARATION MODULE

: THIS MODULE PREPARES THE SUBSYSTEM FOR THE EXECUTION OF A TEST,
: REPORTING AN ERROR TO THE USER IF AN ERROR IS DETECTED. THE USER
: SPECIFIES TASKS TO BE PERFORMED, WHICH THE MODULE EXECUTES
: USING SUBROUTINES.

:CALL:

JSR	PC,TSTPRP	TASK/VERIFY DESCRIPTOR
.WORD	NNNNNN	RETURN HERE IF NO ERROR
BR	??	RETURN HERE IF ERROR
NOP		ERROR DEFINED BY MODULE
ERROR		

:TASK/VERIFY DESCRIPTOR

BIT 15 = 1	SELECT DEVICE AND VERIFY DEVICE IS AVAILABLE

BIT 14 = 1	CLEAR CONTROLLER AND SELECT DEVICE
BIT 13	(RESERVED FOR DRIVE CLEAR)
BIT 12 = 1	PACK ACKNOWLEDGE IF VOLUME NOT VALID

BIT 11 = 1	RECALIBRATE IF POSITIONING IN PROGRESS
BIT 10	
BIT 9	

BIT 8	
BIT 7	
BIT 6 = 1	VERIFY CONTROLLER CLEAR OPERATION

BIT 5	(RESERVED FOR DRIVE CLEAR)
BIT 4 = 1	VERIFY PACK ACKNOWLEDGE
BIT 3 = 1	VERIFY RECALIBRATION

BIT 2	
BIT 1	
BIT 0	

TSTPRP:

:STORE TASK DESCRIPTOR AND CLEAR USER'S ERROR CALL

MOV	@(SP),500\$:STORE DESCRIPTOR
ADD	#6,(SP)	:MOVE SP TO USERS ERROR CALL
CLRB	@(SP)	:CLEAR ERROR CALL
SUB	#4,(SP)	:MOVE SP TO NO ERROR RETURN

JSR	PC,GETSTS	:SETUP TO READ ALL REGISTERS
JSR	PC,GET	:GET RMER2
BR	15\$:BR IF NO ERROR DETECTED
BR	10\$:GET OVER ERROR NUMBER
.WORD	0	:ERROR DEFINED BY GET SUBROUTINE
10\$:	ADD #4,(SP)	:XFER ERROR TO USER AND
MOV	10\$-2,@(SP)	:GET ERROR NUMBER.
JMP	400\$	

55	043226	013737	001376	044066	15\$:	MOV	RMER21,505\$:GET RMER2 AND SAVE FOR LATER
----	--------	--------	--------	--------	-------	-----	--------------	-------------------------------

:*****

```

58                                     ;SELECT DEVICE AND VERIFY DEVICE AVAILABLE IF BIT 15 SET IN TASK
59 043234 005737 044064                TST      500$      ;SELECT DEVICE??
60 043240 100014                        BPL      30$      ;NO!!
61
62 043242 004737 052056                JSR      PC,DEVSEL ;GO SELECT DEVICE
63 043246 000411                        BR       30$      ;NO ERROR - CONTINUE
64 043250 000401                        BR       20$
65 043252 000000                        .WORD   0         ;ERROR NUMBER FROM DEVSEL
66 043254 062716 000004                20$:    ADD     #4,(SP) ;TRANSFER ERROR TO USER
67 043260 113776 043252 000000        MOVB    20$,a(SP)
68 043266 000137 044054                JMP     400$
69
70                                     ;*****
71                                     ;CLEAR CONTROLLER IF BIT 14 IS SET IN TASK
72 043272                                30$:
73 043272 032737 040000 044064        BIT     #BIT14,500$ ;CLEAR CONTROLLER??
74 043300 001451                        BEQ     120$      ;NO!!
75
76 043302 004737 053530                JSR      PC,CNTCLR ;GO CLEAR CONTROLLER
77 043306 000411                        BR       60$      ;CONTINUE - NO ERROR
78 043310 000401                        BR       50$
79 043312 000000                        40$:    .WORD   0         ;ERROR NUMBER FROM CNTCLR
80 043314 062716 000004                50$:    ADD     #4,(SP) ;TRANSFER ERROR TO USER
81 043320 113776 043312 000000        MOVB    40$,a(SP)
82 043326 000137 044054                JMP     400$
83
84                                     ;*****
85                                     ;VERIFY CONTROLLER CLEAR IF BIT6 SET IN TASK
86 043332                                60$:
87 043332 032737 000100 044064        BIT     #BIT6,500$ ;VERIFY??
88 043340 001431                        BEQ     120$      ;NO!!
89
90 043342 004737 044154                JSR      PC,GET    ;GO GET STATUS
91 043346 000411                        BR       90$      ;NO ERROR GETTING STATUS
92 043350 000401                        BR       80$
93 043352 000000                        70$:    .WORD   0         ;ERROR FROM GETTING STATUS
94 043354 062716 000004                80$:    ADD     #4,(SP) ;TRANSFER ERROR TO USER
95 043360 113776 043352 000000        MOVB    70$,a(SP)
96 043366 000137 044054                JMP     400$
97
98 043372 004737 053646                90$:    JSR      PC,CLRSTS ;GO VERIFY STATUS CLEAR
99 043376 000412                        BR       120$     ;NO ERROR IN CLEAR
100 043400 000401                        BR       110$
101 043402 000000                        100$:   .WORD   0         ;ERROR IN STATUS CLEAR
102 043404 005726                        110$:   TST     (SP)+    ;STRIP RETURN ADDRESS TO
103 043406 062716 000004                ADD     #4,(SP)   ;SUBROUTINE AND TRANSFER
104 043412 113776 043402 000000        MOVB    100$,a(SP) ;ERROR TO USER
105 043420 000137 044054                JMP     400$
106
107                                     ;*****
108                                     ;EXECUTE PACK ACKNOWLEDGE IF BIT12 SET IN TASK AND VOLUME IS
109                                     ;NOT VALID
110 043424                                120$:
111 043424 032737 010000 044064        BIT     #BIT12,500$ ;PACK ACKNOWLEDGE??
112 043432 001501                        BEQ     240$      ;NO!!
113
114 043434 004737 044154                JSR      PC,GET

```

```

115 043440 000411          BR      150$          ;NO ERROR GETTING RMD5
116 043442 000401          BR      140$
117 043444 000000          130$: .WORD      0
118 043446 062716 000004    140$: ADD      #4,(SP)    ;TRANSFER ERROR TO USER
119 043452 113776 043444 000000  MOVB   130$,a(SP)
120 043460 000137 044054    JMP     400$
121
122 043464 032737 000100 001346 150$: BIT     #VV,RMDSI    ;IS VOLUME VALID??
123 043472 001061          BNE     240$          ;YES!!
124
125 043474 012737 000023 001410  MOV     #PAKACK!GO,RMCS10 ;LOAD PACK ACK COMMAND
126 043502 112737 000000 001543  MOVB   #RMCS1,PUTINX    ;SETUP REGISTER INDEX TABLE
127 043510 112737 000200 001544  MOVB   #200,PUTINX+1
128 043516 004737 044424    JSR     PC,PUT         ;GO WRITE COMMAND
129 043522 000410          BR      180$          ;NO ERROR LOADING REGISTER
130 043524 000401          BR      170$
131 043526 000000          160$: .WORD      0
132 043530 062716 000004    170$: ADD     #4,(SP)    ;ERROR FROM PUT SUB
133 043534 113776 043526 000000  MOVB   160$,a(SP)    ;TRANSFER ERROR TO USER
134 043542 000544          BR      400$
135
136 043544 004737 044766    180$: JSR     PC,TIMOUT  ;WAIT FOR COMMAND TO COMPLETE
137
138
139
140 043550 032737 000020 044064  ;*****
;VERIFY PACK ACKNOWLEDGE IF #BIT4 SET IN TASK
141 043556 001427          BIT     #BIT4,500$    ;VERIFY PACK ACKNOWLEDGE??
142          BEQ     240$          ;NO!!
143 043560 004737 044154    JSR     PC,GET         ;GO GET STATUS
144 043564 000410          BR      210$          ;NO ERROR GETTING STATUS
145 043566 000401          BR      200$
146 043570 000000          190$: .WORD      0
147 043572 062716 000004    200$: ADD     #4,(SP)    ;ERROR FROM GET SUB
148 043576 113776 043570 000000  MOVB   190$,a(SP)    ;TRANSFER ERROR TO USER
149 043604 000523          BR      400$
150
151 043606 004737 054526    210$: JSR     PC,ACKSTS  ;GO CHECK ACKNOWLEDGE
152 043612 000411          BR      240$          ;NO ERROR
153 043614 000401          BR      230$
154 043616 000000          220$: .WORD      0
155 043620 005726          230$: TST     (SP)+    ;PACK ACKNOWLEDGE ERROR
156 043622 062716 000004    ADD     #4,(SP)    ;STRIP RETURN TO SUB AND
157 043626 113776 043616 000000  MOVB   220$,a(SP)    ;TRANSFER ERROR TO USER
158 043634 000507          BR      400$
159
160
161
162
163 043636
164 043636 032737 004000 044064  ;*****
;RECALIBRATE DRIVE IF BIT 11 IS SET IN TASK AND "SKI" IS SET
;OR "PIP" IS ACTIVE.
165 043644 001505          240$: BIT     #BIT11,500$ ;RECALIBRATE??
166          BEQ     410$          ;NO!!
167 043646 004737 044154    JSR     PC,GET         ;GO GET RMD5
168 043652 000410          BR      270$          ;NO ERROR GETTING RMD5
169 043654 000401          BR      260$
170 043656 000000          250$: .WORD      0
171 043660 062716 000004    260$: ADD     #4,(SP)    ;ERROR FROM GET SUB
;TRANSFER ERROR TO USER

```

```

172 043664 113776 043656 000000      MOVB 250$,a(SP)
173 043672 000470      BR 400$
174
175 043674 032737 040000 044066 270$:  BIT #SKI,505$      ;WAS SKI SET ?
176 043702 001004      BNE 280$      ;YES, GO RECALIBRATE
177 043704 032737 020000 001346      BIT #PIP,RMDSI ;IS PIP ACTIVE??
178 043712 001462      BEQ 410$      ;NO!!
179
180 043714 012737 000007 001410 280$:  MOV #RECAL!GO,RMCS10 ;LOAD RECALIBRATE COMMAND
181 043722 112737 000000 001543      MOVB #RMCS1,PUTINX ;AND REGISTER INDEX
182 043730 112737 000200 001544      MOVB #200,PUTINX+1
183 043736 004737 044424      JSR PC,PUT      ;GO ISSUE RECALIBRATE
184 043742 000410      BR 300$      ;NO ERROR
185 043744 000401      BR 290$
186 043746 000000      .WORD 0      ;ERROR IN REGISTER TRANSFER
187 043750 062716 000004 000000 290$:  ADD #4,(SP)      ;TRANSFER ERROR TO USER
188 043754 113776 043746 000000      MOVB 290$-2,a(SP)
189 043762 000434      BR 400$
190
191 043764 004737 044766      300$:  JSR PC,TIMOUT ;WAIT FOR COMPLETION
192
193
194
195 043770 032737 000010 044064 ;:*****
196 043776 001430 ;:VERIFY RECALIBRATE IF BIT 3 SET IN TASK
197 ;:*****
198 044000 004737 044154      JSR PC,GET      ;GO GET STATUS
199 044004 000410      BR 330$      ;NO ERROR GETTING STATUS
200 044006 000401      BR 320$
201 044010 000000 310$:  .WORD 0      ;ERROR FROM GET
202 044012 062716 000004 000000 320$:  ADD #4,(SP)      ;TRANSFER ERROR TO USER
203 044016 113776 044010 000000      MOVB 310$,a(SP)
204 044024 000413      BR 400$
205
206 044026 004737 055322 330$:  JSR PC,RCLSTS ;GO CHECK RECALIBRATE
207 044032 000412      BR 410$      ;NO ERROR DURING RECALIBRATE
208 044034 000401      BR 350$
209 044036 000000 340$:  .WORD 0      ;ERROR DURING RECALIBRATE
210 044040 005726 350$:  TST (SP)+      ;STRIP RETURN TO SUB AND
211 044042 062716 000004 000000      ADD #4,(SP)      ;TRANSFER ERROR TO USER
212 044046 113776 044036 000000      MOVB 340$,a(SP)
213 044054 162716 000002 400$:  SUB #2,(SP)      ;MOVE SP BACK BEFORE ERROR
214 044060 000240 410$:  NOP
215 044062 000207      RTS PC      ;RETURN TO USER
216
217 044064 000000 500$:  .WORD 0      ;TASK/VERIFY DESCRIPTOR
218 044066 000000 505$:  .WORD 0      ;CONTAINS RMER2
219

```



```

1      .SBTTL  GET STATUS SUBROUTINE
2
3      ;THIS SUBROUTINE SETS UP THE "GET INDEX TABLE" AND THE "GET
4      ;BUFFER" FOR READING ALL SUBSYSTEM REGISTERS VIA THE GET SUBROUTINE
5      ;AND THEN RETURNS TO THE USER.
6
7      ;CALL:  JSR      PC,GETSTS
8      ;
9
10     GETSTS:
11     044070 010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
12     044072 010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
13     044074 010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
14     044076 012700 001514  MOV      #GETINX,R0    ;R0 = ADDRESS OF INDEX TABLE
15     044102 012701 001404  MOV      #RMEC21+2,R1  ;R1 = ADDRESS OF GET BUFFER
16     044106 012702 000046  MOV      #RMEC2,R2    ;R2 = REGISTER INDE
17     044112 110220      2$:  MOVB     R2,(R0)+    ;WRITE REGISTER INDEX IN TABLE
18     044114 005041      CLR      -(R1)        ;CLEAR CORRESPONDING LOCATION
19     044116 162702 000002  3$:  SUB      #2,R2      ;DECREMENT TO NEXT INDEX
20     044122 100405      BMI     4$           ;BRANCH OUT IF DONE
21     044124 022702 000022  CMP      #RMDB,R2    ;DONT WRITE RMDB INDEX
22     044130 001370      BNE     2$
23     044132 005041      CLR      -(R1)
24     044134 000770      BR      3$
25     044136 112720 000200  4$:  MOVB     #200,(R0)+  ;WRITE TERMINATOR
26     044142 012602      MOV     (SP)+,R2    ;;POP STACK INTO R2
27     044144 012601      MOV     (SP)+,R1    ;;POP STACK INTO R1
28     044146 012600      MOV     (SP)+,R0    ;;POP STACK INTO R0
29     044150 000240      NOP
30     044152 000207      RTS      PC

```

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 044154 000240
32 044156 062716 000004
33 044162 105076 000000
34 044166 162716 000004
35 044172 010046
044174 010146
044176 010246
044200 010346
044202 010446
044204 013746 000004
044210 013746 000006
36 044214 013700 001276
37 044220 012702 001334
38 044224 012704 001514
39 044230 012737 044336 000004
40 044236 012737 000300 000006
41 044244 016037 000010 001174
42 044252 016037 000000 001176
43 044260 032737 004000 001176
44 044266 001007
45 044270 062766 000004 000016
46 044276 112776 000112 000016
47 044304 000423
48 044306 105714
49 044310 100433
50 044312 111401
51 044314 042701 177700

.SBTTL GET SUBROUTINE

:THIS SUBROUTINE READS THE REGISTERS WHICH ARE LISTED IN THE
: "GET INDEX TABLE" AND STORES THEIR VALUES IN THE CORRESPONDING
: LOCATION IN THE "GET REGISTER BUFFER". FOR EXAMPLE, AN
: ENTRY OF 04 IN THE TABLE WILL CAUSE THE SUBROUTINE TO
: READ "RMB" AND STORE ITS CONTENTS AT THE LOCATION IN
: THE BUFFER ASSIGNED TO THAT REGISTER. THE NUMBER OF
: REGISTERS TO BE READ IS VARIABLE FROM 1 TO 22; THE INDEX
: TABLE MUST BE TERMINATED WITH A CONTROL BYTE (200)
: WHICH SHOULD FOLLOW THE LAST ENTRY.

:SUBROUTINE CALL:

- : (1) "GET INDEX TABLE" HAS BEEN LOADED WITH REGISTER INDEX
VALUES AND TERMINATED WITH A CONTROL BYTE
- : (2) "GET INPUT BUFFER" IS AVAILABLE FOR USE. (NOTE THAT
UNUSED LOCATIONS, I.E., ENTRIES IN BUFFER CORRESPONDING
TO REGISTERS NOT READ, ARE NOT CHANGED.)
- : (3) JSR PC,GET
BR ??? RETURN HERE IF NO ERROR FOUND
NOP RETURN HERE IF ANY ERROR FOUND
ERROR SUB DEFINES ERROR NUMBER
???

:R0 = REGISTER BASE ADDRESS
:R1 = REGISTER ADDRESS
:R2 = BUFFER BASE ADDRESS
:R3 = BUFFER ADDRESS
:R4 = POINTER TO REGISTER INDEX

```

GET:  NOP
      ADD #4,(SP) ;CLEAR ERROR NUMBER IN USER'S
      CLR @ (SP) ;ERROR CALL
      SUB #4,(SP)
      MOV R0,-(SP) ;;PUSH R0 ON STACK
      MOV R1,-(SP) ;;PUSH R1 ON STACK
      MOV R2,-(SP) ;;PUSH R2 ON STACK
      MOV R3,-(SP) ;;PUSH R3 ON STACK
      MOV R4,-(SP) ;;PUSH R4 ON STACK
      MOV ERRVEC,-(SP) ;;PUSH ERRVEC ON STACK
      MOV ERRVEC+2,-(SP) ;;PUSH ERRVEC+2 ON STACK
      MOV $BASE,R0
      MOV #GETBUF,R2
      MOV #GETINX,R4
      MOV #5$,ERRVEC ;SETUP FOR TIMEOUT
      MOV #PR6,ERRVEC+2
1$:  MOV RMCS2(R0),$TMP0 ;GET "NED" STATUS
      MOV RMCS1(R0),$TMP1 ;GET "DVA" STATUS
      BIT #DVA,$TMP1 ;DEVICE AVAILABLE??
      BNE 3$ ;YES!!
      ADD #4,16(SP) ;WRITE ERROR NUMBER IN USER'S
      MOV #112,@16(SP) ;ERROR CALL
      BR 7$
3$:  TSTB (R4) ;DONE??
      BMI 9$ ;YES!!
      MOV (R4),R1 ;R1 = REGISTER ADDRESS
      BIC #^CIDXMSK,R1 ;CLEAR ANY SIGN EXTENSION
    
```

52	044320	060001			ADD	R0,R1		
53	044322	112403			MOVB	(R4)+,R3	:R3 = STORAGE ADDRESS FOR REGISTER	
54	044324	042703	177700		BIC	#^CIDXMSK,R3	:CLEAR ANY SIGN EXTENSION	
55	044330	060203			ADD	R2,R3		
56	044332	011113			MOV	(R1),(R3)	:READ REGISTER	
57	044334	000764			BR	3\$		
58								
59	044336	022626			5\$: CMP	(SP)+,(SP)+	:RESTORE STACK	
60	044340	062766	000004	000016	ADD	#4,16(SP)	:WRITE ERROR NUMBER IN	
61	044346	112776	000007	000016	MOVB	#7,@16(SP)	:USER'S ERROR CALL	
62	044354	162766	000002	000016	7\$: SUB	#2,16(SP)		
63	044362	105714			8\$: TSTB	(R4)	:DONE CLEARING??	
64	044364	100405			BMI	9\$:YES!!	
65	044366	005003			CLR	R3	:CLEAR REMAINING STORAGE	
66	044370	112403			MOVB	(R4)+,R3	:LOCATIONS	
67	044372	060203			ADD	R2,R3		
68	044374	005013			CLR	(R3)		
69	044376	000771			BR	8\$		
70	044400				9\$: MOV	(SP)+,ERRVEC+2	::POP STACK INTO ERRVEC+2	
	044400	012637	000006		MOV	(SP)+,ERRVEC	::POP STACK INTO ERRVEC	
	044404	012637	000004		MOV	(SP)+,R4	::POP STACK INTO R4	
	044410	012604			MOV	(SP)+,R3	::POP STACK INTO R3	
	044412	012603			MOV	(SP)+,R2	::POP STACK INTO R2	
	044414	012602			MOV	(SP)+,R1	::POP STACK INTO R1	
	044416	012601			MOV	(SP)+,R0	::POP STACK INTO R0	
	044420	012600			MOV	(SP)+,R0	::POP STACK INTO R0	
71	044422	000207			RTS	PC	:RETURN	
72								

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

.SBTTL PUT SUBROUTINE

:THIS SUBROUTINE WRITES THE REGISTERS WHICH ARE LISTED IN THE
: 'PUT INDEX TABLE' WITH THE CONTENTS OF THE CORRESPONDING
: LOCATION IN THE 'PUT REGISTER BUFFER'. THE NUMBER OF
: REGISTERS WRITTEN IS VARIABLE; THE INDEX TABLE MUST
: BE TERMINATED WITH A CONTROL BYTE (200) WHICH SHOULD
: FOLLOW THE LAST ENTRY.

:SUBROUTINE CALL:

- : (1) 'PUT INDEX TABLE' HAS BEEN LOADED WITH INDEX VALUES
: OF REGISTERS TO BE WRITTEN.
- : (2) 'PUT REGISTER BUFFER' CONTAINS CONTENTS OF EACH
: REGISTER TO BE WRITTEN.
- : (3) JSR PC,PUT
: BR ??? RETURN HERE IF NO ERROR FOUND
: NOP RETURN HERE IF ANY ERROR FOUND
: ERROR SUB DEFINES ERROR NUMBER
: ???

:R0 = REGISTER BASE ADDRESS
:R1 = REGISTER ADDRESS
:R2 = BUFFER BASE ADDRESS
:R3 = BUFFER ADDRESS
:R4 = POINTER TO REGISTER INDEX

```

PUT:  NOP
      MOV R0,-(SP)      ;;PUSH R0 ON STACK
      MOV R1,-(SP)      ;;PUSH R1 ON STACK
      MOV R2,-(SP)      ;;PUSH R2 ON STACK
      MOV R3,-(SP)      ;;PUSH R3 ON STACK
      MOV R4,-(SP)      ;;PUSH R4 ON STACK
      MOV ERRVEC,-(SP)  ;;PUSH ERRVEC ON STACK
      MOV ERRVEC+2,-(SP) ;;PUSH ERRVEC+2 ON STACK
      MOV $BASE,R0
      MOV #PUTBUF,R2
      MOV #PUTINX,R4
      MOV #5$,ERRVEC    ;SETUP FOR TIMEOUT
      MOV #PR6,ERRVEC+2
1$:   MOV RMCS2(RO),$TMP0 ;GET 'NED' STATUS
      MOV RMCS1(RO),$TMP1 ;GET 'DVA' STATUS
      BIT #DVA,$TMP1     ;DEVICE AVAILABLE??
      BNE 3$            ;YES!!
      ADD #4,16(SP)     ;WRITE ERROR NUMBER IN
      MOVB #112,@16(SP) ;USER'S ERROR CALL
      BR 7$
3$:   TSTB (R4)         ;DONE??
      BMI 9$           ;YES!!
      MOVB (R4),R1     ;R1 = REGISTER ADDRESS
      BIC #^CIDXMSK,R1 ;CLEAR ANY SIGN EXTENSION
      ADD R0,R1
      MOVB (R4),R3     ;R3 = STORAGE ADDRESS
      BIC #^CIDXMSK,R3 ;CLEAR ANY SIGN EXTENSION
      ADD R2,R3
      MOV (R3),(R1)    ;WRITE REGISTER
      TSTB (R4)+      ;ADJUST REGISTER POINTER
    
```

```

044424 000240
044426 010046
044430 010146
044432 010246
044434 010346
044436 010446
044440 013746 000004
044444 013746 000006
044450 013700 001276
044454 012702 001410
044460 012704 001543
044464 012737 044574 000004
044472 012737 000300 000006
044500 016037 000010 001174
044506 016037 000000 001176
044514 032737 004000 001176
044522 001007
044524 062766 000004 000016
044532 112776 000112 000016
044540 000424
044542 105714
044544 100425
044546 111401
044550 042701 177700
044554 060001
044556 111403
044560 042703 177700
044564 060203
044566 011311
044570 105724
    
```

```
52 044572 000763          BR      3$
53
54 044574 022626          5$:    CMP      (SP)+,(SP)+      ;ADJUST STACK
55 044576 062766 000004 000016  ADD      #4,16(SP)      ;WRITE ERROR NUMBER IN
56 044604 112776 000007 000016  MOVB     #7,@16(SP)     ;USER'S ERROR CALL
57 044612 162766 000002 000016  7$:    SUB      #2,16(SP)
58
59 044620          9$:
   044620 012637 000006      MOV      (SP)+,ERRVEC+2      ;;POP STACK INTO ERRVEC+2
   044624 012637 000004      MOV      (SP)+,ERRVEC      ;;POP STACK INTO ERRVEC
   044630 012604      MOV      (SP)+,R4          ;;POP STACK INTO R4
   044632 012603      MOV      (SP)+,R3          ;;POP STACK INTO R3
   044634 012602      MOV      (SP)+,R2          ;;POP STACK INTO R2
   044636 012601      MOV      (SP)+,R1          ;;POP STACK INTO R1
   044640 012600      MOV      (SP)+,R0          ;;POP STACK INTO R0
60 044642 000207          RTS      PC
61
```

```

1      .SBTTL  SIZE CLOCK SUBROUTINE
2
3      044644      013746      000004      SIZCLK:
4      044650      013746      000006      MOV      ERRVEC,-(SP)      ;;PUSH ERRVEC ON STACK
5      044654      012737      044712      000004      MOV      ERRVEC+2,-(SP)      ;;PUSH ERRVEC+2 ON STACK
6      044662      012737      000300      000006      MOV      #1$,ERRVEC      ;SET UP FOR BUS TIMEOUT
7      044670      012737      177546      001510      MOV      #PR6,ERRVEC+2
8      044676      012737      000100      001512      MOV      #177546,CLKADR      ;LOAD ADDRESSES FOR KW11-L
9      044704      005777      134600      TST      @CLKADR      ;TEST FOR KW11-L PRESENT
10     044710      000421      BR      3$      ;YES - KW11-L IS PRESENT
11     044712      022626      1$:      CMP      (SP)+,(SP)+      ;RESTORE SP
12     044714      012737      044744      000004      MOV      #2$,ERRVEC      ;SET UP FOR BUS TIMEOUT
13     044722      012737      172540      001510      MOV      #172540,CLKADR      ;LOAD ADDRESSES FOR KW11-P CLOCK
14     044730      012737      000104      001512      MOV      #104,CLKVCT
15     044736      005777      134546      TST      @CLKADR      ;TEST FOR KW11-P PRESENT
16     044742      000404      BR      3$      ;YES - KW11-P IS PRESENT
17     044744      022626      2$:      CMP      (SP)+,(SP)+      ;RESTORE SP
18     044746      062766      000002      000004      ADD      #2,4(SP)      ;MOVE RETURN TO ERROR
19     044754      012637      000006      3$:      MOV      (SP)+,ERRVEC+2      ;;POP STACK INTO ERRVEC+2
20     044760      012637      000004      MOV      (SP)+,ERRVEC      ;;POP STACK INTO ERRVEC
      044764      000207      RTS      PC      ;RETURN TO USER
  
```

```

1      .SBTTL  TIMEOUT SUBROUTINE
2
3      ;THIS SUBROUTINE WAITS FOR RDY = 1 AND GO = 0 OR FOR A TIMEOUT
4      ;GREATER THAN APPROX. 500 MSEC AND THEN RETURNS.
5
6      ;CALL:  JSR      PC,TIMOUT
7      ;      ???
8      TIMOUT:
9      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
10     MOV      R1,-(SP)      ;;PUSH R1 ON STACK
11     MOV      R2,-(SP)      ;;PUSH R2 ON STACK
12     MOV      ERRVEC,-(SP)  ;;PUSH ERRVEC ON STACK
13     MOV      ERRVEC+2,-(SP) ;;PUSH ERRVEC+2 ON STACK
14     MOV      #4$,ERRVEC    ;SETUP FOR BUS TIMEOUT - 04 TRAP
15     MOV      #PR6,ERRVEC+2
16     MOV      $BASE,R0     ;R0=BASE ADDRESS
17     MOV      CLKADR,R1    ;R1=CLOCK ADDRESS
18     MOV      #30.,R2     ;R2=NUMBER OF CLOCK CYCLES
19     1$:      CMP      R1,#172540 ;KW11-P CLOCK??
20     BNE      2$          ;NO!!
21     MOV      #1,2(R1)    ;SET COUNTER
22     2$:      MOV      #BIT2!BIT0,(R1) ;START COUNTER
23     3$:      MOV      RMCS1(R0),-(SP) ;GET STATUS
24     BIC      #^C<RDY!GO>,(SP)
25     CMP      #RDY,(SP)+  ;RDY=1,GO=0??
26     BEQ      5$          ;YES!!
27     BIT      #BIT7,(R1)  ;TIMER DONE??
28     BEQ      3$          ;NO!!
29     DEC      R2          ;DEC NUMBER OF CYCLES
30     BNE      1$          ;CONTINUE IF NOT DONE
31     BR      5$          ;'RDY' DID NOT SET OR 'GO' DID NOT RESET
32     ;WITHIN 500 MSEC AFTER THE COMMAND WAS ISSUED.
33     4$:      CMP      (SP)+,(SP)+ ;ADJUST STACK
34     ADD      #4,12(SP)    ;MOVE SP TO USER'S CALL
35     MOVB     #7,@12(SP)  ;WRITE ERROR NUMBER
36     SUB      #2,12(SP)
37     5$:      MOV      (SP)+,ERRVEC+2 ;;POP STACK INTO ERRVEC+2
38     MOV      (SP)+,ERRVEC  ;;POP STACK INTO ERRVEC
39     MOV      (SP)+,R2     ;;POP STACK INTO R2
40     MOV      (SP)+,R1     ;;POP STACK INTO R1
41     MOV      (SP)+,R0     ;;POP STACK INTO R0
42     RTS      PC          ;RETURN TO USER

```

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

```

.SBTTL ERROR CHECK SUBROUTINES
:*****
.SBTTL PRIMARY ERROR CHECK SUBROUTINE

:THE PURPOSE OF THIS SUBROUTINE IS TO VERIFY THAT STATUS IS VALID AND
:THAT FURTHER ERROR AND STATUS CHECKING SHOULD BE PERFORMED. THE
:FOLLOWING CHECKS ARE MADE:
:
:   .CORRECT UNIT IS SELECTED, I.E., THE UNIT SELECT BITS OF RMCS2
:(BITS 0-2) EQUAL THE UNIT BEING TESTED;
:
:   .SELECTED UNIT IS AVAILABLE, I.E., DVA (BIT 11 OF RMCS1) IS SET
:AND NED (BIT 12 OF RMCS2) IS RESET;
:
:   .LAST COMMAND WAS COMPLETED, I.E., THE MASSBUS CONTROLLER IS
:READY (BIT 7 OF RMCS1) AND THE GO BIT IS RESET (BIT 0 OF RMCS1) OR THE
:DRIVE READY BIT (BIT 7 OF RMDS) IS SET.
:   .NO PARITY ERROR OCCURRED WHEN READING REMOTE REGISTERS,
:I.E., MCPE = 0.
:   .NO PARITY ERROR OCCURRED WHEN WRITING REMOTE REGISTERS,
:I.E., PAR = 0, OR, PAR = DPE = 1
:
:THE SUBROUTINE ASSUMES THAT:
:
:   .STATUS HAS BEEN STORED IN THE REGISTER INPUT BUFFER,
:IN PARTICULAR, RMCS1, RMCS2 AND RMDS HAVE BEEN STORED IN THEIR
:CORRESPONDING LOCATIONS OF THE "GET" BUFFER.
:
:   .($UNIT) CONTAINS THE DRIVE NUMBER
:
:THE SUBROUTINE IS CALLED AS FOLLOWS:
:
:(1)   JSR    PC,PRIERR           RETURN HERE IF NO ERROR
:      BR     ???                RETURN HERE TO REPORT AN ERROR
:      NOP
:      ERROR  ERROR NUMBER DEFINED BY SUB
:      JSR    PC,@(SP)+          GO BACK TO SUB FOR MORE ERROR CHECKS
:      ???                RETURN HERE IF NO MORE ERRORS

PRIERR:
:
:CLEAR USER'S ERROR CALL
:
:   ADD     #4,(SP)           ;MOVE (SP) TO ERROR CALL
:   CLRB   @ (SP)           ;CLEAR ERROR NUMBER
:   SUB    #4,(SP)           ;MOVE (SP) TO NO ERROR RETURN

:REPORT AN ERROR IF THE WRONG UNIT IS SELECTED
:
:   MOV    RMCS2I,$BDDAT     ;CORRECT UNIT SELECTED??
:   BIC   #^CUNTMSK,$BDDAT
:   MOV    $UNIT,$GDDAT     ;GOOD DATA FOR TYPEOUT
:   BIC   #^CUNTMSK,$GDDAT
:   CMPB  $GDDAT,$BDDAT     ;COMPARE EXPECTED AND RECEIVED
:                               ;DRIVE NUMBERS
:   BEQ   1$                ;YES!!
:   ADD   #4,(SP)
:   MOVB  #1,@(SP)         ;ERROR 1
  
```

045152				
045152	062716	000004		
045156	105076	000000		
045162	162716	000004		
045166	013737	001344	001142	
045174	042737	177770	001142	
045202	013737	001234	001140	
045210	042737	177770	001140	
045216	123737	001140	001142	
045224	001415			
045226	062716	000004		
045232	112776	000001	000000	


```

58 045240 162716 000002          SUB    #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
59 045244 004736                JSR    PC,@(SP)+       ;REPORT WRONG UNIT SELECTED
60 045246 162716 000010          SUB    #10,(SP)        ;RESTORE (SP)
61 045252 000240                NOP
62 045254 000137 045774          JMP    10$             ;SKIP OTHER CHECKS
63 045260
64
65                               ;REPORT AN ERROR IF THE DEVICE IS NOT AVAILABLE OR IF
66                               ;THE DEVICE IS NONEXISTANT
67 045260 032737 004000 001334    BIT    #DVA,RMCS11     ;DEVICE AVAILABLE??
68 045266 001045                BNE    5$              ;YES!!
69 045270 013737 001334 001140    MOV    RMCS11,$GDDAT   ;EXPECTED STATUS
70 045276 052737 004000 001140    BIS    #DVA,$GDDAT
71 045304 013737 001334 001142    MOV    RMCS11,$BDDAT   ;RECEIVED STATUS
72 045312 062716 000004                ADD    #4,(SP)
73 045316 112776 000002 000000    MOVB   #2,@(SP)        ;ERROR #2
74 045324 032737 010000 001344    BIT    #NED,RMCS21     ;WAS NED SET??
75 045332 001414                BEQ    2$              ;NO!!
76 045334 013737 001344 001140    MOV    RMCS21,$GDDAT   ;EXPECTED STATUS
77 045342 013737 001344 001142    MOV    RMCS21,$BDDAT   ;RECEIVED STATUS
78 045350 042737 010000 001140    BIC    #NED,$GDDAT
79 045356 112776 000003 000000    MOVB   #3,@(SP)        ;YES - CHANGE ERROR NUMBER
80 045364 162716 000002          SUB    #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
81 045370 004736                JSR    PC,@(SP)+       ;REPORT DEVICE NOT AVAILABLE
82 045372 162716 000010          SUB    #10,(SP)        ;RESTORE (SP)
83 045376 000240                NOP
84 045400 000575                BR     10$             ;SKIP OTHER CHECKS
85 045402
86
87                               ;REPORT AN ERROR IF MASSBUS CONTROLLER IS NOT READY
88 045402 032737 000200 001334    BIT    #RDY,RMCS11     ;CONTROLLER READY??
89 045410 001030                BNE    7$              ;YES!!
90 045412 013737 001334 001140    MOV    RMCS11,$GDDAT   ;EXPECTED STATUS
91 045420 052737 000200 001140    BIS    #RDY,$GDDAT
92 045426 042737 160001 001140    BIC    #SC!TRE!MCPE!GO,$GDDAT
93 045434 013737 001334 001142    MOV    RMCS11,$BDDAT   ;RECEIVED STATUS
94 045442 062716 000004                ADD    #4,(SP)
95 045446 112776 000004 000000    MOVB   #4,@(SP)        ;ERROR #4
96 045454 162716 000002          SUB    #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
97 045460 004736                JSR    PC,@(SP)+       ;REPORT CONTROLLER NOT READY
98 045462 162716 000010          SUB    #10,(SP)        ;RESTORE (SP)
99 045466 000240                NOP
100 045470 000541                BR     10$             ;SKIP OTHER CHECKS
101 045472
102
103                               ;REPORT AN ERROR IF GO IS NOT ZERO AND DRY IS NOT ONE
104 045472 032737 000001 001334    BIT    #GO,RMCS11      ;GO RESET??
105 045500 001431                BEQ    8$              ;YES!!
106 045502 032737 000200 001346    BIT    #DRY,RMDSI      ;DRIVE READY??
107 045510 001025                BNE    8$              ;YES!!
108 045512 013737 001334 001140    MOV    RMCS11,$GDDAT   ;EXPECTED STATUS
109 045520 042737 160001 001140    BIC    #SC!TRE!MCPE!GO,$GDDAT
110 045526 013737 001334 001142    MOV    RMCS11,$BDDAT   ;RECEIVED STATUS
111 045534 062716 000004                ADD    #4,(SP)
112 045540 112776 000005 000000    MOVB   #5,@(SP)        ;ERROR #5
113 045546 162716 000002          SUB    #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
114 045552 004736                JSR    PC,@(SP)+       ;REPORT DRIVE NOT READY
    
```

```

115 045554 162716 000010          SUB      #10,(SP)      ;RESTORE (SP)
116 045560 000240          NOP
117 045562 000504          BR       10$
118 045564          8$:
119
120          ;REPORT AN ERROR IF THE RH CONTROLLER DETECTED BAD
121          ;PARITY ON THE MASSBUS CONTROL BUS
122 045564 032737 020000 001334      BIT      #MCPE,RMCS11 ;PARITY ERROR ??
123 045572 001425          BEQ     9$           ;NO!!
124 045574 013737 001334 001140      MOV     RMCS11,$GDDAT ;EXPECTED STATUS
125 045602 042737 160001 001140      BIC     #SC!TR!MCPE!GO,$GDDAT
126 045610 013737 001334 001142      MOV     RMCS11,$BDDAT ;RECEIVED STATUS
127 045616 062716 000004          ADD     #4,(SP)      ;MOVE STACK TO USER'S ERROR
128 045622 112776 000013 000000      MOV     #13,@(SP)   ;ERROR #47
129 045630 162716 000002          SUB     #2,(SP)     ;MOVE SP TO RETURN FOR ERROR
130 045634 004736          JSR    PC,@(SP)+   ;REPORT ERROR VIA USER
131 045636 162716 000010          SUB     #10,(SP)   ;RESTORE STACK
132 045642 000240          NOP
133 045644 000453          BR       10$
134 045646          9$:
135
136          ;REPORT AN ERROR IF DETECTED A CONTROL BUS PARITY ERROR
137 045646 032737 000010 001350      BIT     #PAR,RMER11 ;WAS THERE A PARITY ERROR??
138 045654 001451          BEQ     11$         ;NO!!
139 045656 032737 000010 001376      BIT     #DPE,RMER21 ;WAS IT THE CONTROL BUS??
140 045664 001045          BNE     11$         ;NOT SURE!!
141 045666 032737 000010 001424      BIT     #PAR,RMER10 ;DID TEST SET PAR ??
142 045674 001413          BEQ     93$        ;NO!!
143 045676 010046          MOV     RO,-(SP)   ;:PUSH RO ON STACK
144 045700 012700 001543          MOV     #PUTINX,RO ;RO POINTS TO INDEX TABLE
145 045704 122710 000014          91$: CMP     #RMER1,(RO) ;SEARCH TABLE FOR RMER1
146 045710 001002          BNE     92$
147 045712 012600          MOV     (SP)+,RO  ;:POP STACK INTO RO
148 045714 000431          BR      11$       ;PAR WAS SET BY TEST
149 045716 105720          92$: TST     (RO)+    ;END OF TABLE??
150 045720 100371          BPL     91$       ;NO!!
151 045722 012600          MOV     (SP)+,RO  ;:POP STACK INTO RO
152 045724 013737 001350 001140      93$: MOV     RMER11,$GDDAT ;EXPECTED STATUS
153 045732 042737 000010 001140      BIC     #PAR,$GDDAT
154 045740 013737 001350 001142      MOV     RMER11,$BDDAT ;RECEIVED STATUS
155 045746 062716 000004          ADD     #4,(SP) ;MOVE SP TO USER'S ERROR CALL
156 045752 112776 000050 000000      MOV     #50,@(SP) ;WRITE THE ERROR NUMBER
157 045760 162716 000002          SUB     #2,(SP)   ;MOVE SP TO RETURN FOR ERROR
158 045764 004736          JSR    PC,@(SP)+ ;REPORT THE ERROR
159 045766 162716 000010          SUB     #10,(SP)  ;MOVE SP TO NO ERROR RETURN
160 045772 000240          NOP
161 045774 062716 000010          10$: ADD     #10,(SP)  ;RETURN TO ERROR
162 046000 000240          11$: NOP
163 046002 000207          RTS     PC
164

```

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

```
.SBTTL SECONDARY ERROR CHECK SUBROUTINE

;THE ERROR CHECK SUBROUTINE PROVIDES DETECTION OF SECONDARY ERRORS
;SUCH AS UNEXPECTED ERRORS AND UNEXPECTED REGISTER CONTENTS. THESE
;ERRORS ARE DEEMED SECONDARY IN THAT THEY ARE NOT NECESSARILY
;ASSOCIATED WITH THE OPERATION BEING PERFORMED.
;WHEN THE SUBROUTINE IDENTIFIES SUCH AN ERROR, IT MOVES THE ERROR
;NUMBER TO THE ERROR CALL IN THE TEST ROUTINE AND THEN RETURNS
;TO THE TEST ROUTINE WHICH MAKES THE ERROR CALL. AFTER THE TEST ROUTINE
;MAKES THE ERROR CALL, IT RETURNS TO THE SUBROUTINE WHICH THEN LOOKS FOR
;OTHER ERRORS. WHEN ALL ERRORS HAVE BEEN REPORTED, THE SUBROUTINE
;RETURNS TO THE ADDRESS FOLLOWING THE SUBROUTINE CALL.
```

```
:CALL: JSR PC,SECERR
:      BR ??? RETURN HERE IF NO ERROR
:      NOP RETURN HERE TO REPORT AN ERROR
:      ERROR ERROR NUMBER DEFINED BY SUB
:      JSR PC,@(SP)+ GO BACK TO SUB FOR MORE ERROR CHECKS
:      ??? RETURN HERE IF NO MORE ERRORS
```

```
:NOTE: THE SUBROUTINE ASSUMES THAT REGISTERS HAVE BEEN STORED AT THE
:INPUT REGISTER BUFFER.
```

046004

```
SECERR:
:*****
:STORE FUNCTION CODE AND CLEAR USER'S ERROR NUMBER
MOV RMCS10,515$ ;STORE FUNCTION CODE
BIC #^C<F0!F1!F2!F3!F4>,515$
ADD #4,(SP) ;MOVE (SP) TO ERROR CALL
CLRB @(SP) ;CLEAR ERROR NUMBER
SUB #4,(SP) ;MOVE (SP) TO NO ERROR RETURN
```

```
:*****
:CHECK SECONDARY ERRORS COMMON TO ALL COMMANDS
```

```
:REPORT ERROR IF DRIVE IS NOT READY, I.E., IF DRY = 0
BIT #DRY,RMDSI ;DRIVE READY??
BNE 5$ ;YES!!
MOV RMDSI,$BDDAT ;BAD DATA FOR TYPEOUT
BIC #^CDRY,$BDDAT
MOV #DRY,$GDDAT ;GOOD DATA FOR TYPEOUT
ADD #4,(SP)
MOVB #10,@(SP) ;ERROR NUMBER
SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
JSR PC,@(SP)+ ;REPORT NOT READY
SUB #10,(SP) ;RESTORE (SP) TO ERROR N
NOP
```

```
:REPORT ERROR IF GO BIT IS NOT RESET
5$: BIT #GO,RMCS11 ;GO BIT RESET??
BEQ 10$ ;YES!!
MOV RMCS11,$BDDAT ;BAD DATA FOR TYPEOUT
BIC #^CGO,$BDDAT
CLR $GDDAT ;GOOD DATA FOR TYPEOUT
ADD #4,(SP)
MOVB #11,@(SP) ;ERROR NUMBER
```

28	046004	013737	001410	051640
29	046012	042737	177701	051640
30	046020	062716	000004	
31	046024	105076	000000	
32	046030	162716	000004	
38	046034	032737	000200	001346
39	046042	001024		
40	046044	013737	001346	001142
41	046052	042737	177577	001142
42	046060	012737	000200	001140
43	046066	062716	000004	
44	046072	112776	000010	000000
45	046100	162716	000002	
46	046104	004736		
47	046106	162716	000010	
48	046112	000240		
51	046114	032737	000001	001334
52	046122	001423		
53	046124	013737	001334	001142
54	046132	042737	177776	001142
55	046140	005037	001140	
56	046144	062716	000004	
57	046150	112776	000011	000000


```

115 046460 162716 000010          SUB      #10,(SP)          ;RESTORE (SP)
116 046464 000240                NOP
117 046466                        45$:
118
119                                ;*****
120                                ;USING THE FUNCTION CODE TABLE, CHECK FOR THE FOLLOWING ERRORS:
121                                ;      .STATUS BITS NOT SET THAT SHOULD BE SET, E.G., ATA AND ILF
122                                ;      .STATUS BITS SET THAT SHOULD NOT BE SET, E.G., WCE AND ECH
123                                ;NOTE THAT SOME ERROR BITS ARE CONDITIONAL ON THE COMMAND AND OTHER
124                                ;STATUS CONDITIONS, E.G., WRITE LOCK ERROR SHOULD ONLY BE SET IF
125                                ;WRITE LOCK IS ON AND THE COMMAND IS A WRITE.
126
127                                ;GET AND STORE THE ENTRY FROM THE FUNCTION CODE TABLE
128 046466 010046          MOV      RO,-(SP)          ;;PUSH RO ON STACK
129 046470 013700 051640      MOV      515$,RO          ;GET FUNCTION CODE
130 046474 016037 067046 051632  MOV      FNCDTB(RO),500$ ;STORE ENTRY
131 046502 012600          MOV      (SP)+,RO          ;;POP STACK INTO RO
132
133                                ;REPORT AN ERROR IF AN UNEXPECTED ATTENTION OCCURRED OR IF
134                                ;ATA IS NOT SET AND SHOULD BE SET.
135 046504 013737 051632 001140      MOV      500$,SGDDAT      ;GET EXPECTED ATA STATUS
136 046512 032737 040000 001346      BIT      #ERR,RMDSI      ;IS COMPOSITE ERROR SET ??
137 046520 001403          BEQ      50$              ;NO !!
138 046522 052737 100000 001140      BIS      #ATA,$GDDAT      ;EXPECT AN ATTENTION
139 046530 042737 077777 001140 50$: BIC      #^CATA,$GDDAT      ;STRIP DONT CARES
140 046536 013737 001346 001142      MOV      RMDSI,$BDDAT      ;GET RECEIVED ATA
141 046544 042737 077777 001142      BIC      #^CATA,$BDDAT      ;STRIP DONT CARES
142 046552 023737 001140 001142      CMP      $GDDAT,$BDDAT      ;IS ATA OK ??
143 046560 001413          BEQ      55$              ;YES !!
144 046562 062716 000004          ADD      #4,(SP)          ;MOVE SP TO USERS ERROR CALL
145 046566 112776 000006 000000      MOVB    #6,@(SP)          ;LOAD ERROR # IN CALL
146 046574 162716 000002          SUB      #2,(SP)          ;MOVE SP TO ERROR RETURN
147 046600 004736          JSR      PC,@(SP)+        ;REPORT ERROR
148 046602 162716 000010          SUB      #10,(SP)         ;RESTORE SP
149 046606 000240          NOP
150 046610                        55$:
151
152                                ;REPORT ERROR IF ILF IS INCORRECT, I.E., IF ILF DOES NOT COMPARE
153                                ;WITH FUNCTION CODE TABLE
154 046610 013737 051632 001140      MOV      500$,SGDDAT      ;GET EXPECTED ILF
155 046616 042737 177776 001140      BIC      #^CILF,$GDDAT      ;CLEAR ALL OTHER BITS
156 046624 013737 001350 001142      MOV      RMER11,$BDDAT      ;GET RECEIVED ILF
157 046632 042737 177776 001142      BIC      #^CILF,$BDDAT      ;CLEAR ALL OTHER BITS
158 046640 023737 001140 001142      CMP      $GDDAT,$BDDAT      ;IS ILF OK ??
159 046646 001412          BEQ      60$              ;YES !!
160 046650 062716 000004          ADD      #4,(SP)          ;MOVE SP TO USERS ERROR CALL
161 046654 112776 000254 000000      MOVB    #254,@(SP)        ;WRITE ERROR NUMBER IN CALL
162 046662 162716 000002          SUB      #2,(SP)          ;MOVE SP TO ERROR RETURN
163 046666 004736          JSR      PC,@(SP)+        ;REPORT ERROR AND RETURN
164 046670 162716 000010          SUB      #10,(SP)         ;MOVE SP TO NO ERROR
165 046674 005037 001140 60$: CLR      $GDDAT          ;CLEAR EXPECTED STATUS
166
167                                ;REPORT AN ERROR IF WCE IS SET AND SHOULD NOT BE SET
168 046700 013746 051632          MOV      500$,-(SP)        ;GET WCE STATUS ENABLE
169 046704 052716 137777          BIS      #^CWCE,(SP)       ;SET ALL OTHER BITS
170 046710 013737 001344 001142      MOV      RMCS21,$BDDAT      ;RECEIVED STATUS
171 046716 042637 001142          BIC      (SP)+,$BDDAT      ;CLEAR WCE IF ENABLED
    
```

```

172 046722 001412          BEQ      90$          ;BRANCH IF WCE OK
173 046724 062716 000004  ADD      #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
174 046730 112776 000026 000000  MOVVB   #26,@(SP)    ;WRITE ERROR NUMBER
175 046736 162716 000002  SUB      #2,(SP)      ;MOVE SP TO ERROR RETURN
176 046742 004736          JSR      PC,@(SP)+    ;REPORT ERROR
177 046744 162716 000010  SUB      #10,(SP)     ;RESTORE ERROR
178 046750          90$:
179
180          ;REPORT ERROR IF OPI STATUS IS SET AND SHOULD NOT BE SET
181 046750 013746 051632  MOV      500$,-(SP)   ;GET OPI STATUS ENABLE
182 046754 052716 157777  BIS      #^COPI,(SP)  ;SET ALL OTHER BITS
183 046760 013737 001350 001142  MOV      RMER1I,$BDDAT ;GET RECEIVED STATUS
184 046766 042637 001142  BIC      (SP)+,$BDDAT ;CLEAR OPI IF ENABLED
185 046772 001412          BEQ      100$         ;BRANCH IF OPI OK
186 046774 062716 000004  ADD      #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
187 047000 112776 000164 000000  MOVVB   #164,@(SP)   ;WRITE ERROR NUMBER IN CALL
188 047006 162716 000002  SUB      #2,(SP)      ;MOVE SP TO ERROR RETURN
189 047012 004736          JSR      PC,@(SP)+    ;REPORT ERROR
190 047014 162716 000010  SUB      #10,(SP)     ;RESTORE SP
191 047020          100$:
192
193          ;REPORT ERROR IF IVC IS SET AND IS NOT ENABLED OR IF IVC IS
194          ;SET AND VV IS NOT RESET
195 047020 013746 051632  MOV      500$,-(SP)   ;GET IVC STATUS ENABLE
196 047024 032737 000100 001346  BIT      #VV,RMDSI    ;IS VV SET
197 047032 001402          BEQ      105$         ;NO !!
198 047034 042716 010000  BIC      #IVC,(SP)    ;YES - IVC SHOULD BE 0
199 047040 052716 167777 001142  BIS      #^CIVC,(SP)  ;SET ALL OTHER BITS
200 047044 013737 001376 001142  MOV      RMER2I,$BDDAT ;GET RECEIVED STATUS
201 047052 042637 001142  BIC      (SP)+,$BDDAT ;CLEAR IVC IF ENABLED
202 047056 001412          BEQ      110$         ;BRANCH IF IVC OK
203 047060 062716 000004  ADD      #4,(SP)      ;MOVE SP TO USERS ERROR CALL
204 047064 112776 000165 000000  MOVVB   #165,@(SP)   ;WRITE ERROR NUMBER IN CALL
205 047072 162716 000002  SUB      #2,(SP)      ;MOVE SP TO ERROR RETURN
206 047076 004736          JSR      PC,@(SP)+    ;REPORT ERROR
207 047100 162716 000010  SUB      #10,(SP)     ;RESTORE SP TO NO ERROR
208 047104          110$:
209
210          ;BIT 11 (WLE) OF THE FUNCTION CODE TABLE IS THE ENABLING BIT FOR
211          ; ALL WRITE ERRORS, I.E.,
212          ;       RMER1 - WLE, WCF
213          ;       RMER2 - DPE
214          ;       RMCS2 - UPE.
215          ;EACH OF THESE ERRORS IS CHECKED TO SEE IF AN ERROR IS SET WHEN THE
216          ;WRITE ERROR ENABLE BIT IS RESET.
217
218          ;REPORT AN ERROR IF WLE IS SET AND WRITE ERRORS ARE NOT ENABLED, OR IF
219          ;THE DRIVE IS NOT WRITE PROTECTED
220 047104 012746 177777  MOV      #-1,-(SP)    ;ASSUME WRITE ERRORS ENABLED
221 047110 032737 004000 051632  BIT      #WLE,500$    ;ARE WRITE ERRORS ENABLED ??
222 047116 001404          BEQ      115$         ;NO !!
223 047120 032737 004000 001346  BIT      #WRL,RMDSI   ;IS THE DRIVE WRITE PROTECTED ??
224 047126 001002          BNE      120$         ;YES !!
225 047130 042716 004000 115$: BIC      #WLE,(SP)    ;RESET WLE ENABLE
226 047134 013737 001350 001142 120$: MOV      RMER1I,$BDDAT ;GET RECEIVED STATUS
227 047142 042637 001142  BIC      (SP)+,$BDDAT ;CLEAR WLE IF ENABLED
228 047146 001412          BEQ      125$         ;BRANCH IF WLE OK
    
```

```

229 047150 062716 000004          ADD    #4,(SP)          ;MOVE SP TO USERS ERROR CALL
230 047154 112776 000023 000000  MOVB   #25,@(SP)       ;WRITE ERROR NUMBER IN CALL
231 047162 162716 000002          SUB    #2,(SP)         ;MOVE SP TO ERROR RETURN
232 047166 004736          JSR   PC,@(SP)+        ;REPORT ERROR AND RETURN
233 047170 162716 000010          SUB    #10,(SP)        ;RESTORE SP TO NO ERROR
234 047174          125$:
235
236          ;REPORT ERROR IF WCF IS SET AND WRITE ERRORS ARE NOT ENABLED
237 047174 012746 177777          MOV    #-1,-(SP)       ;ASSUME WRITE ERRORS ENABLED
238 047200 032737 004000 051632  BIT    #WLE,500$       ;ARE WRITE ERRORS ENABLED ??
239 047206 001002          BNE    130$           ;YES !!
240 047210 042716 000040          BIC    #WCF,(SP)       ;DISABLE WCF ERROR
241 047214 013737 001350 001142 130$:  MOV    RMER1I,$BDDAT   ;GET RECEIVED STATUS
242 047222 042637 001142          BIC    (SP)+,$BDDAT   ;RESET WCF IF ENABLED
243 047226 001412          BEQ    135$           ;BRANCH IF WCF OK
244 047230 062716 000004          ADD    #4,(SP)         ;MOVE SP TO USERS ERROR CALL
245 047234 112776 000025 000000  MOVB   #25,@(SP)       ;WRITE ERROR NUMBER IN CALL
246 047242 162716 000002          SUB    #2,(SP)         ;MOVE SP TO ERROR RETURN
247 047246 004736          JSR   PC,@(SP)+        ;REPORT ERROR
248 047250 162716 000010          SUB    #10,(SP)        ;RESTORE SP TO NO ERROR
249 047254          135$:
250
251          ;REPORT ERROR IF DPE IS SET AND WRITE ERRORS ARE NOT ENABLED
252 047254 012746 177777          MOV    #-1,-(SP)       ;ASSUME WRITE ERRORS ARE ENABLED
253 047260 032737 004000 051632  BIT    #WLE,500$       ;ARE WRITE ERRORS ENABLED ??
254 047266 001002          BNE    140$           ;YES !!
255 047270 042716 000010          BIC    #DPE,(SP)       ;RESET DPE ENABLE
256 047274 013737 001376 001142 140$:  MOV    RMER2I,$BDDAT   ;GET RECEIVED STATUS
257 047302 042637 001142          BIC    (SP)+,$BDDAT   ;RESET DPE IF ENABLED
258 047306 001412          BEQ    145$           ;BRANCH IF DPE OK
259 047310 062716 000004          ADD    #4,(SP)         ;MOVE SP TO USERS ERROR CALL
260 047314 112776 000040 000000  MOVB   #40,@(SP)       ;WRITE ERROR NUMBER IN CALL
261 047322 162716 000002          SUB    #2,(SP)         ;MOVE SP TO ERROR RETURN
262 047326 004736          JSR   PC,@(SP)+        ;REPORT ERROR
263 047330 162716 000010          SUB    #10,(SP)        ;RESTORE SP TO NO ERROR
264 047334          145$:
265
266          ;REPORT AN ERROR IF UPE IS SET AND WRITE ERRORS ARE NOT ENABLED
267 047334 012746 177777          MOV    #-1,-(SP)       ;ASSUME WRITE ERRORS ARE ENABLED
268 047340 032737 004000 051632  BIT    #WLE,500$       ;ARE WRITE ERRORS ENABLED ??
269 047346 001002          BNE    150$           ;YES !!
270 047350 042716 020000          BIC    #UPE,(SP)       ;DISABLE UPE ERROR
271 047354 013737 001344 001142 150$:  MOV    RMCS2I,$BDDAT   ;GET RECEIVED STATUS
272 047362 042637 001142          BIC    (SP)+,$BDDAT   ;RESET UPE IF ENABLED
273 047366 001412          BEQ    155$           ;BRANCH IF UPE OK
274 047370 062716 000004          ADD    #4,(SP)         ;MOVE SP TO USERS ERROR CALL
275 047374 112776 000024 000000  MOVB   #24,@(SP)       ;WRITE ERROR NUMBER IN CALL
276 047402 162716 000002          SUB    #2,(SP)         ;MOVE SP TO ERROR RETURN
277 047406 004736          JSR   PC,@(SP)+        ;REPORT ERROR AND RETURN
278 047410 162716 000010          SUB    #10,(SP)        ;MOVE SP TO NO ERROR
279 047414          155$:
280
281          ;REPORT AN ERROR IF IAE IS SET AND IS NOT ENABLED
282 047414 013746 051632          MOV    500$,-(SP)     ;GET IAE ENABLE
283 047420 052716 175777          BIS    #^CIAE,(SP)    ;SET ALL OTHER BITS
284 047424 013737 001350 001142  MOV    RMER1I,$BDDAT   ;GET RECEIVED STATUS
285 047432 042637 001142          BIC    (SP)+,$BDDAT   ;CLEAR IAE IF ENABLED

```

```

286 047436 001412          BEQ      160$          ;BRANCH IF IAE IS OK
287 047440 062716 000004    ADD      #4,(SP)        ;MOVE SP TO USERS ERROR CALL
288 047444 112776 000166 000000  MOVVB   #166,@(SP)     ;WRITE ERROR NUMBER
289 047452 162716 000002    SUB      #2,(SP)        ;MOVE SP TO ERROR RETURN
290 047456 004736          JSR      PC,@(SP)+     ;REPORT ERROR AND RETURN
291 047460 162716 000010    SUB      #10,(SP)       ;MOVE SP TO NO ERROR
292 047464          160$:
293
294          ;BIT 09 (AOE) OF THE FUNCTION CODE TABLE IS THE ENABLING BIT FOR
295          ; ALL READ/WRITE ERRORS, I.E.,
296          ;       RMCS1 - TRE
297          ;       RMCS2 - DLT,NEM,MXF
298          ;       RMDS  - LBT
299          ;       RMER1 - AOE
300          ;NOTE:
301          ;       LBT IS NOT CHECKED BECAUSE IT ONLY RESETS WHEN THE DESIRED
302          ;CYLINDER REGISTER IS WRITTEN
303          ;NOTE:
304          ;       AOE CANNOT BE SET IF LBT IS NOT ALSO SET
305          ;NOTE:
306          ;       TRE IS CHECKED AS A FUNCTION OF OTHER ERROR CONDITONS ABOVE
307
308
309          ;REPORT AN ERROR IF DLT IS SET AND READ/WRITE ERRORS ARE NOT ENABLED
310 047464 012746 177777    MOV      #-1,-(SP)     ;ASSUME ERRORS ARE ENABLED
311 047470 032737 001000 051632  BIT      #AOE,500$     ;ARE ERRORS ENABLED ??
312 047476 001002          BNE      165$          ;YES !!
313 047500 042716 100000    BIC      #DLT,(SP)     ;RESET DLT ENABLE
314 047504 013737 001344 001142 165$:  MOV      RMCS2I,$BDDAT ;GET RECEIVED STATUS
315 047512 042637 001142    BIC      (SP)+,$BDDAT ;CLEAR DLT IF ENABLED
316 047516 001412          BEQ      170$          ;BRANCH IF DLT IS OK
317 047520 062716 000004    ADD      #4,(SP)        ;MOVE SP TO USERS ERROR CALL
318 047524 112776 000032 000000  MOVVB   #32,@(SP)     ;WRITE ERROR NUMBER IN CALL
319 047532 162716 000002    SUB      #2,(SP)        ;MOVE SP TO ERROR RETURN
320 047536 004736          JSR      PC,@(SP)+     ;REPORT ERROR AND RETURN
321 047540 162716 000010    SUB      #10,(SP)       ;MOVE SP TO NO ERROR
322 047544          170$:
323
324          ;REPORT ERROR IF NEM IS SET AND READ/WRITE ERRORS ARE NOT ENABLED
325 047544 012746 177777    MOV      #-1,-(SP)     ;ASSUME ERRORS ARE ENABLED
326 047550 032737 001000 051632  BIT      #AOE,500$     ;ARE ERRORS ENABLED ??
327 047556 001002          BNE      175$          ;YES !!
328 047560 042716 004000    BIC      #NEM,(SP)     ;DISABLE NEM
329 047564 013737 001344 001142 175$:  MOV      RMCS2I,$BDDAT ;GET RECEIVED STATUS
330 047572 042637 001142    BIC      (SP)+,$BDDAT ;CLEAR NEM IF ENABLED
331 047576 001412          BEQ      180$          ;BRANCH IF NEM IS OK
332 047600 062716 000004    ADD      #4,(SP)        ;MOVE SP TO USERS ERROR CALL
333 047604 112776 000167 000000  MOVVB   #167,@(SP)     ;WRITE ERROR NUMBER IN CALL
334 047612 162716 000002    SUB      #2,(SP)        ;MOVE SP TO ERROR RETURN
335 047616 004736          JSR      PC,@(SP)+     ;REPORT ERROR AND RETURN
336 047620 162716 000010    SUB      #10,(SP)       ;MOVE SP TO NO ERROR
337 047624          180$:
338
339          ;REPORT ERROR IF MXF IS SET AND READ/WRITE ERRORS ARE NOT ENABLED
340 047624 012746 177777    MOV      #-1,-(SP)     ;ASSUME ERRORS ARE ENABLED
341 047630 032737 001000 051632  BIT      #AOE,500$     ;ARE DATA ERRORS ENABLED ??
342 047636 001002          BNE      185$          ;YES !!

```



```

343 047640 042716 001000          BIC      #MXF,(SP)      ;DISABLE MXF ERROR
344 047644 013737 001344 001142 185$: MOV      RMCS2I,$BDDAT  ;GET RECEIVED STATUS
345 047652 042637 001142          BIC      (SP)+,$BDDAT  ;CLEAR MXF IF ENABLED
346 047656 001412          BEQ      190$          ;BRANCH IF MXF IS OK
347 047660 062716 000004          ADD      #4,(SP)       ;MOVE SP TO USERS ERROR CALL
348 047664 112776 000033 000000  MOVB     #33,@(SP)     ;WRITE ERROR NUMBER IN CALL
349 047672 162716 000002          SUB      #2,(SP)       ;MOVE SP TO ERROR RETURN
350 047676 004736          JSR      PC,@(SP)+     ;REPORT ERROR AND RETURN
351 047700 162716 000010          SUB      #10,(SP)      ;MOVE SP TO NO ERROR
352 047704          190$:
353
354          ;REPORT ERROR IF AOE IS SET AND DATA ERRORS ARE NOT ENABLED
355 047704 012746 177777          MOV      #-1,-(SP)    ;ASSUME DATA ERRORS ARE ENABLED
356 047710 032737 001000 051632  BIT      #AOE,500$    ;ARE DATA ERRORS EANBLED ??
357 047716 001404          BEQ      191$          ;NO !!
358 047720 032737 002000 001346  BIT      #LBT,RMDSI   ;IS LBT ALSO SET ??
359 047726 001002          BNE      195$          ;YES !!
360 047730 042716 001000 191$: BIC      #AOE,(SP)    ;DISABLE AOE
361 047734 013737 001350 001142 195$: MOV      RMER1I,$BDDAT ;GET RECEIVED STATUS
362 047742 042637 001142          BIC      (SP)+,$BDDAT ;CLEAR AOE IF ENABLED
363 047746 001412          BEQ      200$          ;BRANCH IF AOE IS OK
364 047750 062716 000004          ADD      #4,(SP)       ;MOVE SP TO USERS ERROR CALL
365 047754 112776 000020 000000  MOVB     #20,@(SP)    ;WRITE ERROR NUMBER
366 047762 162716 000002          SUB      #2,(SP)       ;MOVE SP TO ERROR RETURN
367 047766 004736          JSR      PC,@(SP)+     ;REPORT ERROR AND RETURN
368 047770 162716 000010          SUB      #10,(SP)      ;MOVE SP TO NO ERROR
369 047774          200$:
370
371          ;BIT 07 (HCE) OF THE FUNCTION CODE TABLE IS THE ENABLING BIT FOR
372          ;HEADER ERRORS, I.E.,
373          ; RMER1 - HCRC,HCE,FER
374          ; RMER2 - BSE
375
376          ;RESET THE ENABLING BIT (HCE) IF HEADER COMPARE INHIBIT IS SET
377 047774 032737 002000 001366  BIT      #HCI,RMOFI   ;IS HCI SET ??
378 050002 001403          BEQ      201$          ;NO !!
379 050004 042737 000200 051632  BIC      #HCE,500$    ;YES - DISABLE ALL HEADER ERRORS
380 050012          201$:
381
382          ;REPORT AN ERROR IF HCRC IS SET AND HEADER ERRORS ARE NOT ENABLED
383 050012 012746 177777          MOV      #-1,-(SP)    ;ASSUME ERRORS ENABLED
384 050016 032737 000200 051632  BIT      #HCE,500$    ;ARE HEADER ERRORS ENABLED ??
385 050024 001002          BNE      205$          ;YES !!
386 050026 042716 000400          BIC      #HCRC,(SP)   ;DISABLE HCRC
387 050032 013737 001350 001142 205$: MOV      RMER1I,$BDDAT ;GET RECEIVED STATUS
388 050040 042637 001142          BIC      (SP)+,$BDDAT ;RESET HCRC IF ENABLED
389 050044 001412          BEQ      210$          ;BRANCH IF HCRC IS OK
390 050046 062716 000004          ADD      #4,(SP)       ;MOVE SP TO USERS ERROR CALL
391 050052 112776 000035 000000  MOVB     #35,@(SP)    ;WRITE ERROR NUMBER IN CALL
392 050060 162716 000002          SUB      #2,(SP)       ;MOVE SP TO ERROR RETURN
393 050064 004736          JSR      PC,@(SP)+     ;REPORT ERROR AND RETURN
394 050066 162716 000010          SUB      #10,(SP)      ;MOVE SP TO NO ERROR
395 050072          210$:
396
397          ;REPORT ERROR IF HCE IS SET AND HEADER ERRORS ARE NOT ENABLED
398 050072 012746 177777          MOV      #-1,-(SP)    ;ASSUME ERRORS ENABLED
399 050076 032737 000200 051632  BIT      #HCE,500$    ;ARE ERRORS ENABLED ??

```

```

SECONDARY ERROR CHECK SUBROUTINE

400 050104 001002          BNE      215$          ;YES !!
401 050106 042716 000200  BIC      #HCE,(SP)    ;DISABLE HCE
402 050112 013737 001350 001142 215$:  MOV      RMER1I,$BDDAT ;GET RECEIVED STATUS
403 050120 042637 001142    BIC      (SP)+,$BDDAT ;CLEAR HCE IF ENABLED
404 050124 001412          BEQ      220$          ;BRANCH IF HCE IS OK
405 050126 062716 000004    ADD      #4,(SP)      ;MOVE SP TO USERS ERROR CALL
406 050132 112776 000036 000000    MOVB    #36,@(SP)    ;WRITE ERROR NUMBER IN CALL
407 050140 162716 000002    SUB      #2,(SP)     ;MOVE SP TO ERROR RETURN
408 050144 004736          JSR      PC,@(SP)+   ;REPORT ERROR AND RETURN
409 050146 162716 000010    SUB      #10,(SP)   ;MOVE SP TO NO ERROR
410 050152          220$:
411
412          ;REPORT ERROR IF FER IS SET AND HEADER ERRORS ARE NOT ENABLED
413 050152 012746 177777    MOV      #-1,-(SP)   ;ASSUME FER IS ENABLED
414 050156 032737 000200 051632    BIT      #HCE,500$   ;ARE HEADER ERRORS ENABLED ??
415 050164 001002          BNE      225$          ;YES !!
416 050166 042716 000020    BIC      #FER,(SP)   ;DISABLE FER
417 050172 013737 001350 001142 225$:  MOV      RMER1I,$BDDAT ;GET RECEIVED STATUS
418 050200 042637 001142    BIC      (SP)+,$BDDAT ;RESET FER IF ENABLED
419 050204 001412          BEQ      230$          ;BRANCH IF FER OK
420 050206 062716 000004    ADD      #4,(SP)     ;MOVE SP TO USERS ERROR CALL
421 050212 112776 000037 000000    MOVB    #37,@(SP)   ;WRITE ERROR NUMBER IN CALL
422 050220 162716 000002    SUB      #2,(SP)     ;MOVE SP TO ERROR RETURN
423 050224 004736          JSR      PC,@(SP)+   ;REPORT ERROR AND RETURN
424 050226 162716 000010    SUB      #10,(SP)   ;MOVE SP TO NO ERROR
425 050232          230$:
426
427          ;REPORT ERROR IF BSE IS SET AND HEADER ERRORS ARE NOT ENABLED
428 050232 012746 177777    MOV      #-1,-(SP)   ;ASSUME ERRORS ENABLED
429 050236 032737 000200 051632    BIT      #HCE,500$   ;ARE THEY ENABLED ??
430 050244 001002          BNE      235$          ;YES !!
431 050246 042716 100000    BIC      #BSE,(SP)   ;DISABLE BSE
432 050252 013737 001376 001142 235$:  MOV      RMER2I,$BDDAT ;GET RECEIVED STATUS
433 050260 042637 001142    BIC      (SP)+,$BDDAT ;CLEAR BSE IF ENABLED
434 050264 001412          BEQ      240$          ;BRANCH IF BSE OK
435 050266 062716 000004    ADD      #4,(SP)     ;MOVE SP TO USERS ERROR CALL
436 050272 112776 000354 000000    MOVB    #35,@(SP)   ;WRITE ERROR NUMBER
437 050300 162716 000002    SUB      #2,(SP)     ;MOVE SP TO ERROR RETURN
438 050304 004736          JSR      PC,@(SP)+   ;REPORT ERROR AND RETURN
439 050306 162716 000010    SUB      #10,(SP)   ;MOVE SP TO NO ERROR
440 050312          240$:
441
442          ;BIT 06 OF THE FUNCTION CODE TABLE IS THE ENABLING BIT FOR DATA
443          ;FIELD ERRORS, I.E.,
444          ;      RMCS2 - MDPE
445          ;      RMER1 - DCK,ECH
446          ;NOTE:
447          ;      ECH CANNOT SET UNLESS IT IS ENABLED AND ECI IS RESET AND
448          ;      DCK IS SET.
449
450          ;REPORT ERROR IF MDPE IS SET AND IS NOT ENABLED
451 050312 012746 177777    MOV      #-1,-(SP)   ;ASSUME ENABLED
452 050316 032737 000100 051632    BIT      #ECH,500$   ;ARE DATA FIELD ERRORS ENABLED ??
453 050324 001002          BNE      245$          ;YES !!
454 050326 042716 000400    BIC      #MDPE,(SP)  ;DISBALE MDPE
455 050332 013737 001344 001142 245$:  MOV      RMCS2I,$BDDAT ;GET RECEIVED STATUS
456 050340 042637 001142    BIC      (SP)+,$BDDAT ;CLEAR MDPE IF ENABLED

```

```

SECONDARY ERROR CHECK SUBROUTINE

457 050344 001412 BEQ 250$ ;BRANCH IF MDPE OK
458 050346 062716 000004 ADD #4,(SP) ;MOVE SP TO USERS ERROR CALL
459 050352 112776 000027 000000 MOVB #27,@(SP) ;WRITE ERROR NUMBER IN CALL
460 050360 162716 000002 SUB #2,(SP) ;MOVE SP TO ERROR RETURN
461 050364 004736 JSR PC,@(SP)+ ;REPORT ERROR AND RETURN
462 050366 162716 000010 SUB #10,(SP) ;MOVE SP TO NO ERROR
463 050372 250$:
464
465 ;REPORT ERROR IF DCK IS SET AND DATA FIELD ERRORS ARE NOT ENABLED
466 050372 012746 177777 MOV #-1,-(SP) ;ASSUME ENABLED
467 050376 032737 000100 051632 BIT #ECH,500$ ;ARE THEY ENABLED ??
468 050404 001002 BNE 255$ ;YES !!
469 050406 042716 100000 BIC #DCK,(SP) ;DISABLE DCK
470 050412 013737 001350 001142 255$: MOV RMER11,$BDDAT ;GET RECEIVED STATUS
471 050420 042637 001142 BIC (SP)+,$BDDAT ;CLEAR DCK IF ENABLED
472 050424 001412 BEQ 260$ ;BRANCH IF DCK IS OK
473 050426 062716 000004 ADD #4,(SP) ;MOVE SP TO USERS ERROR CALL
474 050432 112776 000030 000000 MOVB #30,@(SP) ;WRITE ERROR NUMBER IN CALL
475 050440 162716 000002 SUB #2,(SP) ;MOVE SP TO ERROR RETURN
476 050444 004736 JSR PC,@(SP)+ ;REPORT ERROR AND RETURN
477 050446 162716 000010 SUB #10,(SP) ;MOVE SP TO NO ERROR
478 050452 260$:
479
480 ;REPORT ERROR IF ECH IS SET AND,
481 ; DATA FIELD ERRORS ARE NOT ENABLED, OR
482 ; ECI IS SET, OR
483 ; DCK IS NOT SET.
484 050452 012746 177777 MOV #-1,-(SP) ;ASSUME ENABLED
485 050456 032737 000100 051632 BIT #ECH,500$ ;ARE ERRORS ENABLED ??
486 050464 001410 BEQ 265$ ;NO !!
487 050466 032737 004000 001366 BIT #ECI,RMOF1 ;IS ECI SET ??
488 050474 001004 BNE 265$ ;YES !!
489 050476 032737 100000 001350 BIT #DCK,RMER11 ;IS DCK ALSO SET ??
490 050504 001002 BNE 270$ ;YES !!
491 050506 042716 000100 265$: BIC #ECH,(SP) ;DISABLE ECH
492 050512 013737 001350 001142 270$: MOV RMER11,$BDDAT ;GET RECEIVED STATUS
493 050520 042637 001142 BIC (SP)+,$BDDAT ;CLEAR ECH IF ENABLED
494 050524 001412 BEQ 275$ ;BRANCH IF ECH IS OK
495 050526 062716 000004 ADD #4,(SP) ;MOVE SP TO USERS ERROR CALL
496 050532 112776 000031 000000 MOVB #31,@(SP) ;WRITE ERROR NUMBER IN CALL
497 050540 162716 000002 SUB #2,(SP) ;MOVE SP TO ERROR RETURN
498 050544 004736 JSR PC,@(SP)+ ;REPORT ERROR AND RETURN
499 050546 162716 000010 SUB #10,(SP) ;MOVE SP TO NO ERROR
500 050552 275$:
501

```

```

1
2
3
4
5 050552 022737 000030 051640      CMP      #SEARCH,515$      ;WAS DATA TRANSFERRED ?
6 050560 103402                    BLO      280$              ;BR IF YES
7 050562 000137 051604            JMP      355$              ;NO - EXIT
8
9
10 050566 013737 001336 001142    ;REPORT ERROR IF RMWC NOT ZERO AND TRE IS ZERO
11 050574 001421                    280$:  MOV     RMWCI,$BDDAT    ;WORD COUNT ZERO??
12 050576 032737 040000 001334    BEQ     285$              ;YES
13 050604 001015                    BIT     #TRE,RMCS1I      ;TRANSFER ERROR DETECTED??
14 050606 062716 000004            BNE     285$              ;YES!!
15 050612 112776 000015 000000    ADD     #4,(SP)          ;
16 050620 005037 001140            MOV     #15,@(SP)        ;ERROR NUMBER
17 050624 162716 000002            CLR     $GDDAT           ;GOOD DATA FOR TYPEOUT
18 050630 004736                    SUB     #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
19 050632 162716 000010            JSR     PC,@(SP)+        ;REPORT WORD COUNT NOT ZERO
20 050636 000240                    SUB     #10,(SP)         ;RESTORE (SP)
21
22
23 050640 013737 001336 001140    ;REPORT ERROR IF RMBA IS NOT CORRECT
24 050646 163737 001412 001140    285$:  MOV     RMWCI,$GDDAT    ;GET WORD COUNT AT END OF TRANSFER AND
25 050654 006337 001140            SUB     RMWCO,$GDDAT     ;SUBTRACT STARTING WORD COUNT.
26 050660 063737 001414 001140    ASL     $GDDAT           ;* 2
27
28 050666 032737 000010 001344    ADD     RMBAO,$GDDAT     ;ADD STARTING BUS ADDRESS
29 050674 001403                    BIT     #BAI,RMCS2I      ;WAS BUS ADDRESS INHIBIT (BAI) SET ??
30 050676 013737 001414 001140    BEQ     290$              ;NO !!
31
32 050704 023737 001140 001340    MOV     RMBAO,$GDDAT     ;ADDRESS SHOULD NOT HAVE CHANGED
33 050712 001416                    290$:  CMP     $GDDAT,RMBAI     ;BUS ADDRESS OK??
34 050714 013737 001340 001142    BEQ     295$              ;YES!!
35 050722 062716 000004            MOV     RMBAI,$BDDAT     ;BAD DATA FOR TYPEOUT
36 050726 112776 000016 000000    ADD     #4,(SP)          ;
37 050734 162716 000002            MOV     #16,@(SP)        ;ERROR NUMBER
38 050740 004736                    SUB     #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
39 050742 162716 000010            JSR     PC,@(SP)+        ;REPORT UNEXPECTED ADDRESS
40 050746 000240                    SUB     #10,(SP)         ;RESTORE (SP)
41
42
43 050750 005046                    ;COMPUTE NUMBER OF SECTORS TRANSFERRED FROM WORD COUNT
44 050752 013746 001336 001140    295$:  CLR     -(SP)            ;NUMBER OF SECTORS TRANSFERRED
45 050756 163716 001412            MOV     RMWCI,-(SP)      ;GET WORD COUNT AT END OF TRANSFER AND
46
47 050762 012746 000400 001410    SUB     RMWCO,(SP)       ;SUBTRACT STARTING WORD COUNT.
48 050766 032737 000002 001410    MOV     #256,-(SP)      ;ASSUME 256. WORDS PER SECTOR
49 050774 001402                    BIT     #BIT1,RMCS10     ;HEADER & DATA COMMAND ??
50 050776 062716 000002            BEQ     300$              ;NO !!
51
52 051002 005266 000004            ADD     #2,(SP)          ;CHANGE TO 258. WORDS PER SECTOR
53 051006 161666 000002            300$:  INC     4(SP)            ;INCREMENT SECTOR COUNT
54 051012 003373                    SUB     (SP),2(SP)       ;SUBTRACT ONE SECTOR'S WORTH
55 051014 022626                    BGT     300$              ;CONTINUE IF NOT DONE
56
57
;COMPUTE EXPECTED SECTOR, TRACK AND CYLINDER ADDRESS FROM
    
```

```

58                                     ;NUMBER OF SECTORS
59 051016 013737 001444 051632      MOV     RMDCO,500$      ;STORE ORIGINAL CYLINDER
60 051024 013737 001416 051634      MOV     RMDAO,505$     ;STORE ORIGINAL TRACK
61 051032 013737 001416 051636      MOV     RMDAO,510$     ;STORE ORIGINAL SECTOR
62 051040 013737 001332 051642      MOV     LSTRK,520$     ;STORE LAST TRACK,
63 051046 000337 051642              SWAB    520$           ;GET TRACK ADDRESS TO LO BYTE AND
64 051052 005237 051642              INC     520$           ;INCREMENT TO GET TOTL # OF TRACKS.
65
66 051056 042737 000377 051634      BIC     #^C<TADMSK>,505$ ;SAVE TRACK ADDRESS BITS AND
67 051064 000337 051634              SWAB    505$           ;SWAP TRACK ADDRESS TO LOW BYTE.
68 051070 042737 177400 051636      BIC     #^C<SADMSK>,510$ ;SAVE SECTOR ADDRESS BITS
69 051076 062637 051636              ADD     (SP)+,510$
70
71 051102 023727 051636 000040 310$:  CMP     510$,#32.      ;SECTOR OVEFLOWED??
72 051110 103406                    BLO     315$           ;NO!!
73 051112 005237 051634              INC     505$           ;INCREMENT TRACK
74 051116 162737 000040 051636      SUB     #32.,510$     ;ADJUST SECTOR
75 051124 000766                    BR      310$           ;TRY AGAIN
76
77 051126 023737 051634 051642 315$:  CMP     505$,520$     ;TRACK OVERFLOWED??
78 051134 103407                    BLO     320$           ;NO!!
79 051136 005237 051632              INC     500$           ;INCREMENT CYLINDER
80 051142 163737 051642 051634      SUB     520$,505$     ;ADJUST TRACK
81 051150 000766                    BR      315$           ;TRY AGAIN
82 051152 000240                    NOP
83
84                                     ;REPORT ERROR IF "LBT" IS NOT CORRECT
85 051154                                320$:
86 051154 005037 001140              CLR     $GDDAT         ;SET GOOD DATA FOR LBT = 0
87 051160 023727 051632 001466      CMP     500$,#822.    ;SHOULD LBT BE SET??
88 051166 101407                    BLOS   325$           ;NO!!
89 051170 032737 002000 001350      BIT     #IAE,RMER11   ;WAS IAE SET ??
90 051176 001003                    BNE    325$           ;YES - LBT SHOULD NOT BE SET
91 051200 012737 002000 001140      MOV     #LBT,$GDDAT   ;SET GOOD DATA FOR LBT = 1
92 051206 013737 001346 001142 325$:  MOV     RMDSI,$BDDAT   ;BAD DATA FOR TYPEOUT
93 051214 042737 175777 001142      BIC     #^CLBT,$BDDAT
94 051222 023737 001140 001142      CMP     $GDDAT,$BDDAT ;IS LBT CORRECT??
95 051230 001413                    BEQ    330$           ;YES!!
96 051232 062716 000004              ADD     #4,(SP)
97 051236 112776 000017 000000      MOVB   #17,@(SP)     ;ERROR NUMBER
98 051244 162716 000002              SUB     #2,(SP)       ;MOVE SP TO RETURN FOR ERROR
99 051250 004736                    JSR    PC,@(SP)+     ;REPORT LBT IS WRONG
100 051252 162716 000010              SUB     #10,(SP)     ;RESTORE (SP)
101 051256 000240                    NOP
102
103                                     ;REPORT ERROR IF "AOE" IS INCORRECT
104 051260 005037 001140              330$: CLR     $GDDAT         ;SET FOR AOE = 0
105 051264 032737 002000 001350      BIT     #IAE,RMER11   ;WAS "IAE" DETECTED??
106 051272 001031                    BNE    340$           ;YES-"AOE" SHOULD BE ZERO
107 051274 023727 051632 001466      CMP     500$,#822.    ;SHOULD AOE BE SET??
108 051302 101425                    BLOS   340$           ;NO!!
109 051304 005737 051634              TST    505$           ;MAYBE
110 051310 001012                    BNE    335$           ;YES
111 051312 005737 051636              TST    510$
112 051316 001007                    BNE    335$           ;YES !!
113 051320 032737 000010 051640      BIT     #F2,515$     ;WAS THIS READ OR WRITE CHECK ??
114 051326 001413                    BEQ    340$           ;NO !!
    
```

```

115 051330 005737 001336          TST      RMWCI          ;WAS ALL DATA TRANSFERRED ??
116 051334 001410          BEQ      340$          ;YES !!
117 051336 012737 001000 001140 335$: MOV      #AOE,$GDDAT    ;SET FOR AOE = 1
118 051344 005037 051634          CLR      505$          ;CLEAR EXPECTED TRACK
119 051350 012737 000001 051636  MOV      #1,510$        ;EXPECT SECTOR = 1
120 051356 013737 001350 001142 340$: MOV      RMER11,$BDDAT  ;BAD DATA FOR TYPEOUT
121 051364 042737 176777 001142  BIC      #^CAOE,$BDDAT
122 051372 023737 001140 001142  CMP      $GDDAT,$BDDAT ;IS AOE CORRECT??
123 051400 001413          BEQ      345$          ;YES!!
124 051402 062716 000004          ADD      #4,(SP)
125 051406 112776 000020 000000  MOVVB   #20,@(SP)      ;ERROR NUMBER
126 051414 162716 000002          SUB      #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
127 051420 004736          JSR      PC,@(SP)+    ;REPORT AOE IS WRONG
128 051422 162716 000010          SUB      #10,(SP)     ;RESTORE (SP)
129 051426 000240          NOP
130
131
132 051430 032737 002000 001350 ;REPORT ERROR IF RMDA IS NOT CORRECT
133 051436 001062          345$: BIT      #IAE,RMER11 ;WAS THERE AN IAE ERROR ??
134 051440 013737 051634 001140  BNE      355$          ;YES - DONT CHECK RMDA,RMDC
135 051446 000337 001140          MOV      505,$GDDAT   ;SETUP EXPECTED DISK ADDRESS
136 051452 113737 051636 001140  SWAB    $GDDAT
137 051460 013737 001342 001142  MOVVB   510,$GDDAT
138 051466 023737 001140 001142  MOV      RMDAI,$BDDAT ;SETUP RECEIVED DISK ADDRESS
139 051474 001413          CMP      $GDDAT,$BDDAT ;COMPARE EXPECTED & RECEIVED
140 051476 062716 000004          BEQ      350$          ;BRANCH IF EQUAL
141 051502 112776 000021 000000  ADD      #4,(SP)
142 051510 162716 000002          MOVVB   #21,@(SP)     ;ERROR NUMBER
143 051514 004736          SUB      #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
144 051516 162716 000010          JSR      PC,@(SP)+    ;REPORT BAD DISK ADDRESS
145 051522 000240          SUB      #10,(SP)     ;RESTORE (SP)
146
147
148 051524 013737 051632 001140 ;REPORT ERROR IF RMDC IS INCORRECT
149 051532 042737 176000 001140 350$: MOV      500,$GDDAT   ;SETUP EXPECTED CYLINDER
150 051540 013737 001370 001142  BIC      #^C1777,$GDDAT
151 051546 023737 001140 001142  MOV      RMDCI,$BDDAT ;SETUP RECEIVED CYLINDER
152 051554 001413          CMP      $GDDAT,$BDDAT ;COMPARE CYLINDERS
153 051556 062716 000004          BEQ      355$          ;BRANCH IF EQUAL
154 051562 112776 000022 000000  ADD      #4,(SP)
155 051570 162716 000002          MOVVB   #22,@(SP)     ;ERROR NUMBER
156 051574 004736          SUB      #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
157 051576 162716 000010          JSR      PC,@(SP)+    ;REPORT BAD CYLINDER
158 051602 000240          SUB      #10,(SP)     ;RESTORE (SP)
159
160 051604 062716 000004          355$: ADD      #4,(SP)      ;MOVE (SP) TO ERROR CALL
161 051610 105776 000000          TSTB   @(SP)          ;WAS ERROR FOUND??
162 051614 001403          BEQ      360$
163 051616 062716 000004          ADD      #4,(SP)      ;MOVE (SP) TO ERROR RETURN
164 051622 000402          BR      365$
165 051624 162716 000004          360$: SUB      #4,(SP)      ;MOVE (SP) TO NO ERROR RETURN
166 051630 000207          365$: RTS      PC
167
168 051632 000000          500$: .WORD 0          ;CYLINDER
169 051634 000000          505$: .WORD 0          ;TRACK
170 051636 000000          510$: .WORD 0          ;SECTOR
171 051640 000000          515$: .WORD 0          ;FUNCTION CODE

```

172 051642 000000
173
174

520\$: .WORD 0

;# OF TRACKS FOR DEVICE UNDER TEST = LAST
;TRACK + 1 TRACK

```

1      .SBTTL  COMPOSITE ERROR CHECK SUBROUTINE
2
3      ;THIS SUBROUTINE CHECKS THE STORED CONTENTS OF RMER1 AND
4      ;RMER2 AFTER MASKING EACH REGISTER WORD WITH THE USER'S STATUS
5      ;MASKS AND REPORTS AN ERROR IF ANY BITS ARE LEFT ON AFTER
6      ;THE MASKS ARE APPLIED.
7
8      ;CALL:
9      ;(1)  JSR      PC,CMPERRSTS
10     ;      .WORD   MASK FOR ERROR REGISTER 1
11     ;      .WORD   MASK FOR ERROR REGISTER 2
12     ;      BR      ???      RETURN HERE IF NO ERROR
13     ;      NOP     RETURN HERE TO REPORT AN ERROR
14     ;      ERROR   ERROR NUMBER DEFINED BY SUB
15     ;      JSR     PC,@(SP)+ GO BACK TO SUB FOR MORE ERROR CHECKS
16     ;      ???     RETURN HERE IF NO MORE ERRORS
17
18     ;NOTE: BITS TO BE MASKED SHOULD BE ONE; BITS TO BE TESTED SHOULD
19     ;BE ZERO
20
21     051644  CMPERRSTS:
22
23     ;MASK AND STORE THE CONTENTS OF RMER1 AND RMER2
24     051644  013737  001350  001176  MOV     RMER1I,$TMP1  ;STORE RMER1 AT TEMP STORAGE
25     051652  047637  000000  001176  BIC     @ (SP),$TMP1  ;MASK RMER1
26     051660  062716  000002                ADD     #2,(SP)      ;MOVE SP TO NEXT MASK
27     051664  013737  001376  001200  MOV     RMER2I,$TMP2  ;STORE RMER2 AT TEMP STORAGE
28     051672  047637  000000  001200  BIC     @ (SP),$TMP2  ;MASK RMER2
29
30
31     ;CLEAR USER'S ERROR CALL
32     051700  062716  000006                ADD     #6,(SP)      ;MOVE SP TO USER'S ERROR CALL
33     051704  105076  000000                CLR     @ (SP)      ;CLEAR ERROR NUMBER
34     051710  162716  000004                SUB     #4,(SP)      ;LEAVE SP AT NO ERROR RETURN
35
36     ;SEE IF THERE WERE ANY ERRORS IN RMER1, I.E., $TMP1
37     051714  005737  001176                TST     $TMP1        ;ANY ERRORS TO REPORT??
38     051720  001420                BEQ     5$           ;NO !!
39     051722  013737  001176  001142  MOV     $TMP1,$BDDAT ;RECEIVED STATUS FOR TYPEOUT
40     051730  005037  001140                CLR     $GDDAT      ;EXPECTED STATUS FOR TYPEOUT
41     051734  062716  000004                ADD     #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
42     051740  112776  000066  000000  MOV     #66,@(SP)    ;CORRECTABLE DATA CHECK ERROR #
43     051746  162716  000002                SUB     #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
44     051752  004736                JSR     PC,@(SP)+    ;REPORT ERROR VIA USER
45     051754  162716  000010                SUB     #10,(SP)     ;MOVE SP BACK TO BRANCH
46     051760  000240                NOP
47     051762  5$:
48
49
50     ;SEE IF THERE ARE ANY ERRORS TO REPORT IN RMER2 ($TMP2)
51     051762  005737  001200                TST     $TMP2        ;ANY ERRORS IN RMER2?
52     051766  001420                BEQ     10$          ;NO!!
53
54     051770  013737  001200  001142  MOV     $TMP2,$BDDAT ;RECEIVED STATUS FOR TYPEOUT
55     051776  005037  001140                CLR     $GDDAT      ;EXPECTED STATUS FOR TYPEOUT
56     052002  062716  000004                ADD     #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
57     052006  112776  000067  000000  MOV     #67,@(SP)    ;WRITE ERROR NUMBER IN USER'S CALL
    
```



```
58 052014 16. 6 000002          SUB   #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
59 052020 004736          JSR   PC,@(SP)+        ;REPORT ERROR VIA USER
60 052022 162716 000010          SUB   #10,(SP)         ;MOVE SP TO NO ERROR RETURN
61 052026 000240          NOP
62 052030          10$:
63
64          ;AUGMENT THE RETURN ADDRESS IF ANY ERROR WAS DETECTED
65 052030 062716 000004          ADD   #4,(SP)         ;MOVE SP TO USER'S ERROR CALL
66 052034 105776 000000          TSTB @ (SP)          ;WAS THERE AN ERROR CALLED??
67 052040 001403          BEQ   20$            ;NO!!
68 052042 062716 000004          ADD   #4,(SP)         ;YES - MOVE SP TO ERROR RETURN
69 052046 000402          BR   30$
70 052050 162716 000004          20$: SUB   #4,(SP)         ;MOVE SP TO NO ERROR RETURN
71 052054 000207          30$: RTS   PC          ;RETURN TO USER
72
```

```

1      .SBTTL  DEVICE SELECT SUBROUTINE
2
3      ; THIS SUBROUTINE SELECTS THE DEVICE, GETTING THE DEVICE NUMBER FROM THE
4      ; TEST QUEUE.
5
6      ;CALL:
7      ;(1)   JSR      PC,DEVSEL
8      ;(2)   BR       ??          RETURN IF NO ERROR
9      ;(3)   NOP
10      ;(4)   ERROR          RETURN IF ERROR
11                                     ERROR DEFINED BY SUBROUTINE
12 052056  DEVSEL:
13
14      ;CLEAR USER'S ERROR CALL
15 052056 062716 000004  ADD      #4,(SP)          ;MOVE SP TO USER'S ERROR
16 052062 105076 000000  CLR      @ (SP)          ;CLEAR LOW ORDER BYTE OF CALL
17 052066 162716 000004  SUB      #4,(SP)          ;MOVE SP BACK
18
19      ;SAVE USER'S INFORMATION AND SETUP REGISTERS
20 052072 013746 000004  MOV      ERRVEC,-(SP)      ;;PUSH ERRVEC ON STACK
21 052076 013746 000006  MOV      ERRVEC+2,-(SP)   ;;PUSH ERRVEC+2 ON STACK
22 052102 010046  MOV      R0,-(SP)          ;;PUSH R0 ON STACK
23 052104 010146  MOV      R1,-(SP)          ;;PUSH R1 ON STACK
24 052106 012737 052226 000004  MOV      #20$,ERRVEC      ;SETUP FOR BUS TIMEOUT
25 052114 012737 000300 000006  MOV      #PR6,ERRVEC+2
26 052122 013700 001276  MOV      $BASE,R0          ;R0 = UNIBUS ADDRESS
27 052126 013701 001464  MOV      TSTQUE,R1        ;R1 POINTS TO DEVICE NUMBER
28
29      ;SELECT DEVICE AND VERIFY THAT DEVICE IS AVAILABLE
30
31 052132 111160 000010  MOV      (R1),RMCS2(R0)   ;WRITE UNIT SELECT BITS
32 052136 016037 000000 001176  MOV      RMCS1(R0),$TMP1  ;GET "DVA" STATUS
33 052144 016037 000010 001174  MOV      RMCS2(R0),$TMP0  ;GET "NED" STATUS
34
35 052152 032737 010000 001174  BIT      #NED,$TMP0      ;IS DEVICE NONEXISTENT ?
36 052160 001407  BEQ      10$             ;NO!!
37 052162 062766 000004 000010  ADD      #4,10(SP)        ;MOVE SP TO USERS ERROR CALL
38 052170 112776 000111 000010  MOV      #111,@10(SP)    ;WRITE ERROR NUMBER
39 052176 000422  BR       30$
40
41 052200 032737 004000 001176 10$:  BIT      #DVA,$TMP1      ;IS DEVICE AVAILABLE ?
42 052206 001021  BNE      35$             ;YES!!
43 052210 062766 000004 000010  ADD      #4,10(SP)        ;MOVE SP TO USERS ERROR CALL
44 052216 112776 000112 000010  MOV      #112,@10(SP)    ;WRITE ERROR NUMBER
45 052224 000407  BR       30$
46
47      ;HANDLE BUS TIMEOUT
48 052226 022626 20$:  CMP      (SP)+,(SP)+    ;ADJUST SP
49 052230 062766 000004 000010  ADD      #4,10(SP)        ;MOVE SP TO USERS ERROR CALL
50 052236 112776 000113 000010  MOV      #113,@10(SP)    ;WRITE BUS TIMEOUT ERROR NUMBER
51 052244 162766 000002 000010 30$:  SUB      #2,10(SP)      ;ADJUST RETURN TO 'NOP' PRECEDING
52                                     ;THE ERROR CALL
53
54      ;RESTORE USERS DATA AND RETURN TO ADDRESS ON STACK
55 052252 012601 35$:  MOV      (SP)+,R1          ;;POP STACK INTO R1
56 052254 012600  MOV      (SP)+,R0          ;;POP STACK INTO R0
57 052256 012637 000006  MOV      (SP)+,ERRVEC+2    ;;POP STACK INTO ERRVEC+2
    
```

52 052262 012637 000004
53 052266 000207

MOV (SP)+,ERRVEC ::POP STACK INTO ERRVEC
RTS PC :EXIT

```

1      .SBTTL  SEEK STATUS CHECK SUBROUTINE
2
3      ;THIS SUBROUTINE VERIFIES THE RESULTS OF SEEK TESTS USING STATUS
4      ;STORED IN THE GET BUFFER AND TEST PARAMETERS STORED IN THE PUT BUFFER.
5
6
7      ;THE SUBROUTINE RETURNS TO THE CALLING ROUTINE IF AN ERROR IS DETECTED
8      ;AFTER HAVING LOADED THE APPROPRIATE ERROR NUMBER IN THE "ERROR" TRAP
9      ;OF THE CALLING ROUTINE.  SEEK STATUS IS CHECKED AS FOLLOWS:
10
11      ;CALL:
12      ;(1)  JSR    PC,SEKSTS
13           BR     ???          RETURN HERE IF NO ERROR
14           NOP
15           ERROR          RETURN HERE TO REPORT AN ERROR
16           JSR    PC,@(SP)+    ERROR NUMBER DEFINED BY SUB
17           ???          GO BACK TO SUB FOR MORE ERROR CHECKS
18           RETURN HERE IF NO MORE ERRORS
19
20      SEKSTS:
21
22      ;CLEAR USERS' ERROR CALL
23      NOP
24      ADD     #4,(SP)          ;MOVE (SP) TO ERROR CALL
25      CLR    @ (SP)          ;CLEAR ERROR NUMBER
26      SUB     #4,(SP)          ;MOVE (SP) TO NO ERROR RETURN
27      CLR    300$          ;CLEAR ERROR FLAGS
28
29      ;TEST FOR MASSBUS CONTROL BUS PARITY ERROR WHEN WRITING
30      ;LOCAL REGISTERS, I.E., "PAR" = 1 AND "DPE" = 0
31      BIT     #PAR,RMER1I    ;WAS PARITY ERROR DETECTED??
32      BEQ    1$             ;NO!!
33      BIT     #DPE,RMER2I    ;WAS IT DUE TO CONTROL BUS??
34      BNE    1$             ;NOT SURE!!
35
36      ;REPORT REGISTER PARITY ERROR VIA USER'S ERROR CALL
37      CLR    $GDDAT          ;EXPECTED STATUS
38      MOV    RMER1I,$BDDAT   ;RECEIVED STATUS
39      ADD     #4,(SP)          ;MOVE STACK TO USER'S ERROR
40      MOV    #50,@(SP)       ;ERROR #50
41      SUB    #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
42      JSR    PC,@(SP)+
43      SUB    #10,(SP)        ;RESTORE STACK
44      BR     3$             ;IAE SHOULD BE ZERO
45
46      ;DETERMINE THE VALUE OF "IAE" STATUS BASED ON TRACK, SECTOR AND CYLINDER
47      ;ALSO, SET "SKI" IF CYLINDER ADDRESS IS TOO LARGE.
48      1$:  MOV    #IAE,$GDDAT  ;SETUP FOR IAE = 1
49           BIS    #SKI,300$    ;SETUP FOR SKI = 1
50           CMP    RMDCO,#822.  ;GREATER THAN LAST CYLINDER ?
51           BHI    3$           ;YES - CYLINDER IS INVALID
52           BIC    #SKI,300$    ;CLEAR SKI ERROR FLAG
53
54      CMP    RMDAO+1,LSTRK+1  ;GREATER THAN LAST TRACK ?
55      BHI    3$             ;YES - TRACK IS INVALID
56
57      CMP    RMDAO,#29.      ;SECTOR > 29. ?
58      BLOS  2$             ;BR IF NO

```

```

58 052444 032737 010000 001442      BIT      #FMT16,RMOFO      ;18 BIT FORMAT ?
59 052452 001406                    BEQ      3$              ;YES - SECTOR IS INVALID FOR 18 BIT MODE
60 052454 123727 001416 000037      CMPB    RMDAO,#31.      ;SECTOR > 31. ?
61 052462 101002                    BHI     3$              ;YES - SECTOR IS INVALID
62
63 052464 005037 001140      2$:    CLR      $GDDAT      ;"IAE" SHOULD = 0
64
65      ;COMPARE EXPECTED AND RECIEVED "IAE" STATUS
66 052470 013737 001350 001142      3$:    MOV      RMER11,$BDDAT ;IS IAE OK??
67 052476 042737 175777 001142      BIC     #^CIAE,$BDDAT   ;SAVE IAE BIT FOR COMPARE
68 052504 023737 001140 001142      CMP     $GDDAT,$BDDAT  ;CORRECT "IAE" STATUS ?
69 052512 001004                    BNE     35$            ;BR IF NO
70 052514 042737 040000 053526      BIC     #SKI,300$      ;CLEAR SKI FLAG
71 052522 000413                    BR      5$              ;GO CHECK NEXT ERROR
72 052524
73      35$:
74 052524 062716 000004                    ;REPORT INCORRECT "IAE" STATUS VIA USER'S ERROR CALL
75 052530 112776 000051 000000      ADD     #4,(SP)
76 052536 162716 000002                    MOVVB   #51,@(SP)      ;ERROR 51
77 052542 004736                    SUB     #2,(SP)        ;MOVE SP TO RETURN FOR ERROR
78 052544 162716 000010      JSR    PC,@(SP)+      ;REPORT INCORRECT IAE
79 052550 000240                    SUB     #10,(SP)      ;RESTORE (SP)
80 052552
81
82      5$:
83      ;REPORT ANY IVC ERROR AS
84      ; IVC ERROR WITH VOLUME VALID ZERO
85      ; ERRONEOUS IVC ERROR, VOLUME VALID IS SET
86 052552 032737 010000 001376      BIT     #IVC,RMER21    ;IVC ERROR??
87 052560 001427                    BEQ     52$            ;NO!!
88 052562 005037 001140      CLR     $GDDAT        ;EXPECTED STATUS
89 052566 013737 001376 001142      MOV     RMER21,$BDDAT ;RECEIVED STATUS
90 052574 062716 000004                    ADD     #4,(SP)        ;MOVE SP TO USER'S ERROR
91 052600 112776 000060 000000      MOVVB  #60,@(SP)      ;ERROR 60 IF VV = 0
92 052606 032737 000100 001346      BIT     #VV,RMDSI
93 052614 001403                    BEQ     51$            ;ERROR 61 IF VV = 1
94 052616 112776 000061 000000      MOVVB  #61,@(SP)      ;ERROR 61 IF VV = 1
95 052624 162716 000002      51$:    SUB     #2,(SP)        ;MOVE SP TO RETURN FOR ERROR
96 052630 004736                    JSR    PC,@(SP)+      ;REPORT ERROR VIA USER
97 052636 162716 000010      SUB     #10,(SP)      ;RESTORE SP
98 052636 000240                    NOP
99
100 052640 013737 001376 001142      52$:    MOV     RMER21,$BDDAT ;RECEIVED STATUS
101 052646 042737 137777 001142      BIC     #^CSKI,$BDDAT ;CLEAR DONT CARES
102 052654 013737 053526 001140      MOV     300$,$GDDAT   ;GET EXPECTED SKI STATUS
103 052662 042737 137777 001140      BIC     #^CSKI,$GDDAT ;CLEAR DONT CARES
104 052670 001417                    BEQ     53$            ;BRANCH IF 0 EXPECTED
105
106 052672 032737 040000 001142      ;REPORT ERROR IF SKI IS NOT SET (IAE WAS NOT DETECTED)
107 052700 001032                    BIT     #SKI,$BDDAT   ;WAS SKI DETECTED ??
108 052702 062716 000004                    BNE     54$            ;YES !!
109 052706 112776 000267 000000      ADD     #4,(SP)        ;MOVE SP TO USERS ERROR CALL
110 052714 162716 000002                    MOVVB  #267,@(SP)     ;WRITE ERROR NUMBER
111 052720 004736                    SUB     #2,(SP)        ;MOVE SP TO ERROR RETURN
112 052722 162716 000010      JSR    PC,@(SP)+      ;REPORT ERROR AND RETURN
113 052726 000443                    SUB     #10,(SP)      ;MOVE SP TO NO ERROR
114 052730      53$:    BR      6$              ;GO TO NEXT ERROR CHECK

```

```

115
116 ;REPORT ERROR IF SKI IS SET
117 052730 032737 040000 001142 BIT #SKI,$BDDAT ;IS SKI SET ??
118 052736 001413 BEQ 54$ ;NO - SKI IS OK
119 052740 062716 000004 ADD #4,(SP) ;MOVE (SP) TO ERROR
120 052744 112776 000054 000000 MOVB #54,@(SP) ;LOAD ERROR NUMBER
121 052752 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
122 052756 004736 JSR PC,@(SP)+ ;REPORT SEEK ERROR
123 052760 162716 000010 SUB #10,(SP) ;RESTORE (SP)
124 052764 000240 NOP
125
126 ;REPORT ANY DEVICE CHECK
127 052766 032737 000200 001376 54$: BIT #DVC,RMER2I ;WAS THERE DVC DURING SEEK??
128 052774 001420 BEQ 6$ ;NO!!
129 052776 005037 001140 CLR $GDDAT ;EXPECTED STATUS
130 053002 013737 001376 001142 MOV RMER2I,$BDDAT ;RECEIVED STATUS
131 053010 062716 000004 ADD #4,(SP)
132 053014 112776 000055 000000 MOVB #55,@(SP) ;ERROR #55
133 053022 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
134 053026 004736 JSR PC,@(SP)+ ;REPORT ERROR VIA USER
135 053030 162716 000010 SUB #10,(SP) ;RESTORE SP
136 053034 000240 NOP
137
138 ;REPORT ANY "OPI" ERROR AS OPI WITH MOL = 0, OR OPI
139 ;BECAUSE ON CYLINDER LATCH DIDN'T RESET
140 053036 032737 020000 001350 6$: BIT #OPI,RMER1I ;"OPI" ERROR??
141 053044 001427 BEQ 8$ ;NO!!
142 053046 005037 001140 CLR $GDDAT ;EXPECTED STATUS
143 053052 013737 001350 001142 MOV RMER1I,$BDDAT ;RECEIVED STATUS
144 053060 062716 000004 ADD #4,(SP) ;MOVE (SP) TO ERROR
145 053064 112776 000052 000000 MOVB #52,@(SP) ;LOAD ERROR NUMBER
146 053072 032737 010000 001346 BIT #MOL,RMDSI ;WAS MEDIUM ON LINE??
147 053100 001403 BEQ 7$ ;NO!!
148 053102 112776 000053 000000 MOVB #53,@(SP) ;YES - CHANGE ERROR NUMBER
149 053110 162716 000002 7$: SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
150 053114 004736 JSR PC,@(SP)+ ;REPORT "OPI" ERROR
151 053116 162716 000010 SUB #10,(SP) ;RESTORE (SP)
152 053122 000240 NOP
153
154 ;SEE IF "PIP" = 0, AND "ATA", "MOL" AND "VV" = 1
155 053124 013746 001346 8$: MOV RMDSI,-(SP)
156 053130 042716 047677 BIC #^C<ATA!PIP!MOL!VV>,(SP)
157 053134 022726 110100 CMP #ATA!MOL!VV,(SP)+
158 053140 001002 BNE 9$ ;ERROR IN RMDS
159 053142 000137 053476 JMP 14$ ;RMDS IS OK
160
161 ;REPORT ERROR IF MOL = 0 AND OPI = 0
162 053146 032737 010000 001346 9$: BIT #MOL,RMDSI ;IS MOL RESET??
163 053154 001030 BNE 10$ ;NO - MOL IS SET
164 053156 032737 020000 001350 BIT #OPI,RMER1I ;WAS OPI SET
165 053164 001024 BNE 10$ ;YES - DONT REPORT ERROR
166 053166 013737 001346 001140 MOV RMDSI,$GDDAT ;EXPECTED STATUS
167 053174 052737 010000 001140 BIS #MOL,$GDDAT
168 053202 013737 001346 001142 MOV RMDSI,$BDDAT ;RECEIVED STATUS
169 053210 062716 000004 ADD #4,(SP)
170 053214 112776 000062 000000 MOVB #62,@(SP)
171 053222 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR

```

```

172 053226 004736          JSR      PC,@(SP)+      ;REPORT ERROR VIA USER
173 053230 162716 000010  SUB      #10,(SP)
174 053234 000240          NOP
175
176          ;REPORT AN ERROR IF 'PIP' IS STIL SET AND SKI NOT SET
177 053236 032737 020000 001346 10$: BIT      #PIP,RMDSI      ;IS 'PIP' STILL SET??
178 053244 001430          BEQ      11$              ;NO!!
179 053246 032737 040000 001376  BIT      #SKI,RMER2I      ;WAS 'SKI' SET??
180 053254 001024          BNE      11$              ;YES-DONT REPORT PIP
181 053256 013737 001346 001140  MOV      RMDSI,$GDDAT      ;EXPECTED STATUS
182 053264 042737 020000 001142  BIC      #PIP,$BDDAT
183 053272 013737 001346 001142  MOV      RMDSI,$BDDAT      ;RECEIVED STATUS
184 053300 062716 000004          ADD      #4,(SP)          ;MOVE (SP) TO ERROR
185 053304 112776 000056 000000  MOVB    #56,@(SP)        ;LOAD ERROR NUMBER
186 053312 162716 000002          SUB      #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
187 053316 004736          JSR      PC,@(SP)+      ;REPORT 'PIP' SET AFTER SEEK
188 053320 162716 000010  SUB      #10,(SP)          ;RESTORE (SP)
189 053324 000240          NOP
190
191          ;REPORT AN ERROR IF 'ATA' IS NOT SET
192 053326 032737 100000 001346 11$: BIT      #ATA,RMDSI      ;WAS 'ATA' SET ??
193 053334 001024          BNE      13$              ;YES!!
194 053336 013737 001346 001140  MOV      RMDSI,$GDDAT      ;EXPECTED STATUS
195 053344 052737 110600 001140  BIS      #ATA!MOL!DPR!DRY,$GDDAT
196 053352 013737 001346 001142  MOV      RMDSI,$BDDAT      ;RECEIVED STATUS
197 053360 062716 000004          ADD      #4,(SP)          ;MOVE (SP) TO ERROR
198 053364 112776 000057 000000  MOVB    #57,@(SP)        ;LOAD ERROR NUMBER
199 053372 162716 000002          SUB      #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
200 053376 004736          JSR      PC,@(SP)+      ;REPORT ATTENTION NOT SET DURING
201          ;SEEK TEST
202 053400 162716 000010  SUB      #10,(SP)          ;RESTORE (SP)
203 053404 000240          NOP
204
205          ;REPORT ERROR IF VOLUME VALID IS RESET AND IVC IS ZERO
206 053406 032737 000100 001346 13$: BIT      #VV,RMDSI      ;IS VV = 0 ??
207 053414 001030          BNE      14$              ;NO!!
208 053416 032737 010000 001376  BIT      #IVC,RMER2I      ;IS IVC ALSO 0 ??
209 053424 001024          BNE      14$              ;NO - IVC IS SET
210 053426 013737 001346 001140  MOV      RMDSI,$GDDAT      ;EXPECTED STATUS
211 053434 052737 000100 001140  BIS      #VV,$GDDAT
212 053442 013737 001346 001142  MOV      RMDSI,$BDDAT      ;RECEIVED STATUS
213 053450 062716 000004          ADD      #4,(SP)
214 053454 112776 000064 000000  MOVB    #64,@(SP)        ;ERROR #64
215 053462 162716 000002          SUB      #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
216 053466 004736          JSR      PC,@(SP)+
217 053470 162716 000010  SUB      #10,(SP)
218 053474 000240          NOP
219 053476          14$:
220
221          ;MODIFY THE RETURN ADDRESS IF AN ERROR WAS DETECTED
222 053476 000240          NOP
223 053500 062716 000004          ADD      #4,(SP)          ;MOVE (SP) TO ERROR CALL
224 053504 105776 000000          TSTB    @(SP)            ;WAS ERROR CALLED??
225 053510 001403          BEQ      15$              ;NO!!
226 053512 062716 000004          ADD      #4,(SP)          ;MOVE TO ERROR RETURN
227 053516 000402          BR      16$
228

```

229 053520 162716 000004
230 053524 000207
231
232 053526 000000
233

15\$: SUB #4,(SP) ;MOVE (SP) TO NO ERROR RETURN
16\$: RTS PC ;RETURN
300\$: .WORD 0 ;ERROR FLAGS


```

1      .SBTTL  CONTROLLER CLEAR SUBROUTINE
2
3      ;THIS SUBROUTINE CLEARS THE MASSBUS CONTROLLER, MASSBUS ADAPTERS,
4      ;AND DRIVES, THEN SELECTS THE DRIVE.
5
6      ;CALL:  JSR      PC,CNTCLR
7              BR      ???
8              NOP
9              ERROR
10             ???
11
12     CNTCLR:
13     053530 010046          MOV      R0,-(SP)      ;;PUSH R0 ON STACK
14     053532 010146          MOV      R1,-(SP)      ;;PUSH R1 ON STACK
15     053534 013746 000004   MOV      ERRVEC,-(SP)  ;;PUSH ERRVEC ON STACK
16     053540 013746 000006   MOV      ERRVEC+2,-(SP) ;;PUSH ERRVEC+2 ON STACK
17     053544 012737 053604 000004 MOV      #10$,ERRVEC  ;SETUP FOR BUS TIMEOUT
18     053552 012737 000300 000006 MOV      #PR6,ERRVEC+2
19     053560 013700 001276          MOV      $BASE,R0      ;R0 = UNIBUS ADDRESS
20     053564 012760 000040 000010 MOV      #CLR,RMCS2(R0) ;CLEAR MASSBUS
21     053572 013701 001464          MOV      TSTQUE,R1     ;GET DEVICE UNDER TEST
22     053576 111160 000010          MOVB    (R1),RMCS2(R0) ;SELECT DEVICE
23     053602 000412          BR      20$
24
25     10$:  CMP      (SP)+,(SP)+  ;ADJUST STACK
26           ADD     #4,10(SP)    ;MOVE SP TO USER'S ERROR CALL
27           MOVB   #7,@10(SP)    ;WRITE THE ERROR NUMBER
28           SUB    #2,10(SP)    ;ADJUST SP TO RETURN TO ERROR
29
30     20$:  MOV      (SP)+,ERRVEC+2  ;;POP STACK INTO ERRVEC+2
31           MOV      (SP)+,ERRVEC  ;;POP STACK INTO ERRVEC
32           MOV      (SP)+,R1     ;;POP STACK INTO R1
33           MOV      (SP)+,R0     ;;POP STACK INTO R0
34           RTS      PC

```

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

```

.SBTTL STATUS CHECK SUBROUTINES
:*****
.SBTTL CONTROLLER CLEAR STATUS CHECK SUBROUTINE

:THIS SUBROUTINE VERIFIES THAT THE SUBSYSTEM IS INITIALIZED BASED ON
:STATUS STORED IN THE GET BUFFER. THIS SUBROUTINE SHOULD ONLY BE
:USED FOLLOWING A CONTROLLER CLEAR OPERATION, I.E., WRITING A 1 IN BIT
:5 OF RMCS2, BECAUSE THE ERROR MESSAGES ARE BASED ON THAT CONDITION.

:STATUS PERTINENT TO THE DEVICE IS NOT CHECKED. IN PARTICULAR, THE
:FOLLOWING STATUS BITS ARE NOT CHECKED:
:
:   ATA,ERR,PIP,MOL,WRL,LBT,PGM,VV,OM,UNS,SKI,DVC
:
:CALL:
:(1) JSR PC,CLRSTS
:     BR   ??? RETURN HERE IF NO ERROR
:     NOP RETURN HERE TO REPORT AN ERROR
:     ERROR ERROR NUMBER DEFINED BY SUB
:     JSR PC,@(SP)+ GO BACK TO SUB FOR MORE ERROR CHECKS
:     ??? RETURN HERE IF NO MORE ERRORS

CLRSTS:

:CLEAR USER'S ERROR CALL
ADD #4,(SP) ;MOVE SP TO ERROR
CLRB @(SP) ;CLEAR ERROR NUMBER
SUB #4,(SP) ;MOVE SP BACK TO NO ERROR

:REPORT ERROR IF RMCS1 NOT INITIALIZED
4$: MOV RMCS1I,$BDDAT ;VERIFY RMCS1
BIC #SC,$BDDAT ;IGNORE SPECIAL CONDITION
MOV #DVA!RDY,$GDDAT ;EXPECT DVA & RDY
CMP $GDDAT,$BDDAT ;COMPARE EXPECTED, RECEIVED
BEQ 5$ ;BRANCH IF EQUAL
ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
MOVB #126,@(SP) ;WRITE ERROR NUMBER IN CALL
SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
JSR PC,@(SP)+ ;REPORT ERROR VIA USER
SUB #10,(SP) ;MOVE SP BACK TO NO ERROR
NOP

:REPORT ERROR IF RMBA NOT RESET
5$: CLR $GDDAT ;VERIFY RMBA IS ZERO
MOV RMBAI,$BDDAT
BEQ 7$ ;BRANCH IF ZERO
ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
MOVB #127,@(SP) ;WRITE ERROR NUMBER IN CALL
SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
JSR PC,@(SP)+ ;REPORT ERROR VIA USER
SUB #10,(SP) ;MOVE SP BACK TO NO ERROR
NOP

:REPORT ERROR IF RMCS2 NOT INITIALIZED
7$: MOV RMCS2I,$BDDAT ;VERIFY RMCS2
MOV R1,-(SP) ;PUSH R1 ON STACK
CLR -(SP) ;EXPECT IR & UNIT NUMBER
MOV TSTQUE,R1 ;R1 = ADDRESS OF TEST QUE
MOVB (R1),(SP)

```

053646				
053646	062716	000004		
053652	105076	000000		
053656	162716	000004		
053662	013737	001334	001142	
053670	042737	100000	001142	
053676	012737	004200	001140	
053704	023737	001140	001142	
053712	001413			
053714	062716	000004		
053720	112776	000126	000000	
053726	162716	000002		
053732	004736			
053734	162716	000010		
053740	000240			
053742	005037	001140		
053746	013737	001340	001142	
053754	001413			
053756	062716	000004		
053762	112776	000127	000000	
053770	162716	000002		
053774	004736			
053776	162716	000010		
054002	000240			
054004	013737	001344	001142	
054012	010146			
054014	005046			
054016	013701	001464		
054022	111116			

```

58 054024 052716 000100      BIS      #1R,(SP)
59 054030 012637 001140      MOV      (SP)+,$GDDAT
60 054034 012601              MOV      (SP)+,R1      ;;POP STACK INTO R1
61 054036 023737 001140 001142      CMP      $GDDAT,$BDDAT ;COMPARE EXPECTED & RECEIVED
62 054044 001413              BEQ      9$           ;BRANCH IF EQUAL
63 054046 062716 000004      ADD      #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
64 054052 112776 000130 000000      MOVVB   #130,@(SP)    ;WRITE ERROR NUMBER IN CALL
65 054060 162716 000002      SUB      #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
66 054064 004736              JSR      PC,@(SP)+    ;REPORT ERROR VIA USER
67 054066 162716 000010      SUB      #10,(SP)     ;MOVE SP BACK TO NO ERROR
68 054072 000240              NOP
69                          ;REPORT ERROR IF RMER1 NOT RESET-IGNORE UNS
70 054074 005037 001140 9$:      CLR      $GDDAT      ;VERIFY RMER1
71 054100 013737 001350 001142      MOV      RMER1I,$BDDAT
72 054106 042737 040000 001142      BIC      #UNS,$BDDAT ;IGNORE UNSAFE
73 054114 001413              BEQ      13$         ;BRANCH IF ZERO
74 054116 062716 000004      ADD      #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
75 054122 112776 000131 000000      MOVVB   #131,@(SP)    ;WRITE ERROR NUMBER IN CALL
76 054130 162716 000002      SUB      #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
77 054134 004736              JSR      PC,@(SP)+    ;REPORT ERROR VIA USER
78 054136 162716 000010      SUB      #10,(SP)     ;MOVE SP BACK TO NO ERROR
79 054142 000240              NOP
80                          ;REPORT ERROR IF RMMR1 NOT INITIALIZED-IGNORE WC,LS,LST
81 054144 013737 001360 001142 13$:      MOV      RMMR1I,$BDDAT ;VERIFY RMMR1
82 054152 042737 000046 001142      BIC      #WC!LS!LST,$BDDAT ;IGNORE WORD CLOCK, SCT, TRK
83 054160 012737 000010 001140      MOV      #MWD,$GDDAT ;EXPECT WRITE DATA BIT
84 054166 023737 001140 001142      CMP      $GDDAT,$BDDAT ;COMPARE EXPECTED AND RECEIVED
85 054174 001413              BEQ      17$         ;BRANCH IF 0
86 054176 062716 000004      ADD      #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
87 054202 112776 000133 000000      MOVVB   #133,@(SP)    ;WRITE ERROR NUMBER IN CALL
88 054210 162716 000002      SUB      #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
89 054214 004736              JSR      PC,@(SP)+    ;REPORT ERROR VIA USER
90 054216 162716 000010      SUB      #10,(SP)     ;MOVE SP BACK TO NO ERROR
91 054222 000240              NOP
92                          ;REPORT AN ERROR IF RMEC2 IS NOT RESET
93 054224 005037 001140 17$:      CLR      $GDDAT      ;EXPECT ZEROS
94 054230 013737 001402 001142      MOV      RMEC2I,$BDDAT ;VERIFY RMEC2 = 0
95 054236 001413              BEQ      19$
96 054240 062716 000004      ADD      #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
97 054244 112776 000135 000000      MOVVB   #135,@(SP)    ;WRITE ERROR NUMBER IN CALL
98 054252 162716 000002      SUB      #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
99 054256 004736              JSR      PC,@(SP)+    ;REPORT ERROR VIA USER
100 054260 162716 000010      SUB      #10,(SP)     ;MOVE SP BACK TO NO ERROR
101 054264 000240              NOP
102                          ;REPORT ERROR IF RMMR2 NOT INITIALIZED-IGNORE RQA,RQB
103 054266 013737 001374 001142 19$:      MOV      RMMR2I,$BDDAT ;VERIFY RMMR2
104 054274 042737 140000 001142      BIC      #RQA!RQB,$BDDAT
105 054302 012737 011777 001140      MOV      #TST!1777,$GDDAT ;EXPECT TEST BIT ON
106 054310 023737 001140 001142      CMP      $GDDAT,$BDDAT
107 054316 001413              BEQ      21$
108 054320 062716 000004      ADD      #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
109 054324 112776 000136 000000      MOVVB   #136,@(SP)    ;WRITE ERROR NUMBER IN CALL
110 054332 162716 000002      SUB      #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
111 054336 004736              JSR      PC,@(SP)+    ;REPORT ERROR VIA USER
112 054340 162716 000010      SUB      #10,(SP)     ;MOVE SP BACK TO NO ERROR
113 054344 000240              NOP
114                          ;REPORT ERROR IF RMER2 NOT RESET-IGNORE SKI,DVC
    
```

```

115 054346 005037 001140 21$: CLR $GDDAT ;EXPECT ALL ZEROS
116 054352 013737 001376 001142 MOV RMER2I,$BDDAT ;VERIFY RMER2
117 054360 042737 040200 001142 BIC #SKI!DVC,$BDDAT ;IGNORE DEVICE ERRORS
118 054366 001413 BEQ 215$ ;BRANCH IF OTHER BITS 0
119 054370 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
120 054374 112776 000174 000000 MOVB #174,@(SP) ;WRITE ERROR NUMBER IN CALL
121 054402 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
122 054406 004736 JSR PC,@(SP)+ ;REPORT ERROR VIA USER
123 054410 162716 000010 SUB #10,(SP) ;MOVE SP BACK TO NO ERROR
124 054414 000240 NOP
125 ;REPORT ERROR IF RMD5 NOT INITIALIZED
126 054416 013737 001346 001142 215$: MOV RMD5I,$BDDAT ;TEST DRIVE STATUS REGISTER
127 054424 042737 177177 001142 BIC #^C<DRY!DPR>,$BDDAT
128 054432 012737 000600 001140 MOV #DPR!DRY,$GDDAT ;EXPECTED DRIVE STATUS
129 054440 023737 001140 001142 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED & RECEIVED
130 054446 001413 BEQ 22$ ;BRANCH IF EQUAL
131 054450 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
132 054454 112776 000134 000000 MOVB #134,@(SP) ;WRITE ERROR NUMBER
133 054462 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
134 054466 004736 JSR PC,@(SP)+ ;REPORT ERROR TO USER
135 054470 162716 000010 SUB #10,(SP) ;MOVE SP BACK TO NO ERROR
136 054474 000240 NOP
137 054476 062716 000004 22$: ADD #4,(SP) ;MOVE SP TO ERROR CALL
138 054502 105776 000000 TSTB @(SP) ;WAS AN ERROE DETECTED??
139 054506 001403 BEQ 23$ ;NO!!
140 054510 062716 000004 ADD #4,(SP) ;YES - MOVE TO ERROR RETURN
141 054514 000402 BR 24$
142 054516 162716 000004 23$: SUB #4,(SP) ;MOVE SP TO NO ERROR RETURN
143 054522 000240 24$: NOP
144 054524 000207 RTS
145 PC
    
```

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

054526
 054526 062716 000004
 054532 105076 000000
 054536 162716 000004
 054542 032737 000100 001346
 054550 001024
 054552 013737 001346 001140
 054560 052737 000100 001140
 054566 013737 001346 001142
 054574 062716 000004
 054600 112776 000155 000000
 054606 162716 000002
 054612 004736
 054614 162716 000010
 054620 000240
 054622
 054622 032737 010000 001346
 054630 001024
 054632 013737 001346 001140
 054640 052737 010000 001140
 054646 013737 001346 001142
 054654 062716 000004
 054660 112776 000041 000000
 054666 162716 000002
 054672 004736
 054674 162716 000010
 054700 000240
 054702
 054710 001570
 054712 032737 040000 001350
 054720 001424
 054722 013737 001350 001142

```

.SBTTL  PACK ACKNOWLEDGE STATUS CHECK

;THIS SUBROUTINE CHECKS THE RESULTS OF A PACK ACKNOWLEDGE
;COMMAND USING THE STATUS STORED IN THE GET BUFFER.  ERRORS ARE
;REPORTED TO THE USER VIA THE USER'S ERROR CALL.

;CALL:
;(1)  JSR    PC,ACKSTS
      BR    ???
      NOP
      ERROR
      JSR    PC,@(SP)+
      ???

RETURN HERE IF NO ERROR
RETURN HERE TO REPORT AN ERROR
ERROR NUMBER DEFINED BY SUB
GO BACK TO SUB FOR MORE ERROR CHECKS
RETURN HERE IF NO MORE ERRORS

ACKSTS:

;CLEAR USER'S ERROR CALL
ADD    #4,(SP)
CLR    @ (SP)
SUB    #4,(SP)
;MOVE SP TO ERROR CALL
;CLEAR LOW ORDER BYTE
;MOVE SP BACK

;REPORT AN ERROR IF 'VV' IS 0
BIT    #VV,RMDSI
BNE    1$
MOV    RMDSI,$GDDAT
BIS    #VV,$GDDAT
MOV    RMDSI,$BDDAT
ADD    #4,(SP)
MOVB   #155,@(SP)
SUB    #2,(SP)
JSR    PC,@(SP)+
SUB    #10,(SP)
NOP
;IS VOLUME VALID SET??
;YES!!
;EXPECTED STATUS
;RECEIVED STATUS
;MOVE SP TO ERROR CALL
;WRITE NUMBER IN ERROR CALL
;MOVE SP TO RETURN FOR ERROR
;REPORT THE ERROR
;MOVE SP BACK TO BRANCH

1$:
;REPORT AN ERROR IF 'MOL' IS 0
BIT    #MOL,RMDSI
BNE    2$
MOV    RMDSI,$GDDAT
BIS    #MOL,$GDDAT
MOV    RMDSI,$BDDAT
ADD    #4,(SP)
MOVB   #41,@(SP)
SUB    #2,(SP)
JSR    PC,@(SP)+
SUB    #10,(SP)
NOP
;IS MOL SET??
;YES!!
;EXPECTED STATUS
;RECEIVED STATUS
;MOVE SP TO ERROR CALL
;WRITE NUMBER OF ERROR IN CALL
;MOVE SP TO RETURN FOR ERROR
;REPORT TH ERROR
;MOVE SP TO BRANCH

2$:
;SEE IF 'UNS','OPI','RMR','ILR', OR 'ILF' IS SET
BIT    #UNS!OPI!RMR!ILR!ILF,RMER11
BEQ    7$

;REPORT AN ERROR IF 'UNS' IS SET
BIT    #UNS,RMER11
BEQ    3$
MOV    RMER11,$BDDAT
;WAS UNS SET??
;NO!!
;RECEIVED STATUS
  
```

58	054730	013737	001350	001140	MOV	RMER11,\$GDDAT	:EXPECTED STATUS
59	054736	042737	040000	001140	BIC	#UNS,\$GDDAT	
60	054744	062716	000004		ADD	#4,(SP)	:MOVE SP TO ERROR CALL
61	054750	112776	000042	000000	MOVB	#42,@(SP)	:WRITE NUMBER OF ERROR IN CALL
62	054756	162716	000002		SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
63	054762	004736			JSR	PC,@(SP)+	:REPORT THE ERROR VIA USER
64	054764	162716	000010		SUB	#10,(SP)	:MOVE SP TO NO ERROR RETURN
65	054770	000240			NOP		
66	054772						
67							
68							
69	054772	032737	020000	001350	:REPORT ANY OPI ERROR		
70	055000	001424			BIT	#OPI,RMER11	:WAS OPI SET ??
71	055002	013737	001350	001142	BEQ	4\$:NO!!
72	055010	013737	001350	001140	MOV	RMER11,\$BDDAT	:RECEIVED STATUS
73	055016	042737	020000	001140	MOV	RMER11,\$GDDAT	:EXPECTED STATUS
74	055024	062716	000004		BIC	#OPI,\$GDDAT	
75	055030	112776	000043	000000	ADD	#4,(SP)	:MOVE SP TO ERROR CALL
76	055036	162716	000002		MOVB	#43,@(SP)	:WRITE NUMBER OF ERROR IN CALL
77	055042	004736			SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
78	055044	162716	000010		JSR	PC,@(SP)+	:REPORT THE ERROR VIA USER
79	055050	000240			SUB	#10,(SP)	:MOVE SP TO NO ERROR RETURN
80	055052				NOP		
81							
82							
83	055052	032737	000004	001350	:REPORT ANY RMR ERROR		
84	055060	001424			BIT	#RMR,RMER11	:WAS RMR SET??
85	055062	013737	001350	001142	BEQ	5\$:NO!!
86	055070	013737	001350	001140	MOV	RMER11,\$BDDAT	:RECEIVED STATUS
87	055076	042737	000004	001140	MOV	RMER11,\$GDDAT	:EXPECTED STATUS
88	055104	062716	000004		BIC	#RMR,\$GDDAT	
89	055110	112776	000044	000000	ADD	#4,(SP)	:MOVE SP TO ERROR CALL
90	055116	162716	000002		MOVB	#44,@(SP)	:WRITE NUMBER OF ERROR IN CALL
91	055122	004736			SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
92	055124	162716	000010		JSR	PC,@(SP)+	:REPORT THE ERROR VIA USER
93	055130	000240			SUB	#10,(SP)	:MOVE SP TO NO ERROR RETURN
94	055132				NOP		
95							
96							
97	055132	032737	000002	001350	:REPORT ANY ILR ERROR		
98	055140	001424			BIT	#ILR,RMER11	:WAS ILR SET??
99	055142	013737	001350	001142	BEQ	6\$:NO!!
100	055150	013737	001350	001140	MOV	RMER11,\$BDDAT	:RECEIVED STATUS
101	055156	042737	000002	001140	MOV	RMER11,\$GDDAT	:EXPECTED STATUS
102	055164	062716	000004		BIC	#ILR,\$GDDAT	
103	055170	112776	000045	000000	ADD	#4,(SP)	:MOVE SP TO ERROR CALL
104	055176	162716	000002		MOVB	#45,@(SP)	:WRITE NUMBER OF ERROR IN CALL
105	055202	004736			SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
106	055204	162716	000010		JSR	PC,@(SP)+	:REPORT THE ERROR VIA USER
107	055210	000240			SUB	#10,(SP)	:MOVE SP TO NO ERROR RETURN
108	055212				NOP		
109							
110							
111	055212	032737	000001	001350	:REPORT ANY ILF ERROR		
112	055220	001424			BIT	#ILF,RMER11	:WAS ILF SET??
113	055222	013737	001350	001142	BEQ	7\$:NO!!
114	055230	013737	001350	001140	MOV	RMER11,\$BDDAT	:RECEIVED STATUS
					MOV	RMER11,\$GDDAT	:EXPECTED STATUS

115	055236	042737	000001	001140	BIC	#1LF,\$GDDAT	
116	055244	062716	000004		ADD	#4,(SP)	:MOVE SP TO ERROR CALL
117	055250	112776	000046	000000	MOVB	#46,@(SP)	:WRITE NUMBER OF ERROR IN CALL
118	055256	162716	000002		SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
119	055262	004736			JSR	PC,@(SP)+	:REPORT THE ERROR VIA USER
120	055264	162716	000010		SUB	#10,(SP)	:MOVE SP TO NO ERROR RETURN
121	055270	000240			NOP		
122	055272			7\$:			
123							
124							
125	055272	062716	000004		:AUGMENT RETURN ADDRESS IF ERROR WAS FOUND		
126	055276	105776	000000		ADD	#4,(SP)	:MOVE SP TO ERROR CALL
127	055302	001403			TSTB	@(SP)	:WAS ERROR FOUND??
128	055304	062716	000004		BEQ	8\$:NO!!
129	055310	000402			ADD	#4,(SP)	:YES - MOVE TO ERROR RETURN
130	055312	162716	000004	8\$:	BR	9\$	
131	055316	000240		9\$:	SUB	#4,(SP)	:MOVE SP TO NO ERROR RETURN
132	055320	000207			NOP		
133					RTS	PC	

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

```

.SBTTL RECALIBRATE STATUS CHECK SUBROUTINE
;THIS SUBROUTINE CHECKS THE RESULTS OF A RECALIBRATE OPERATION
;USING THE STATUS STORED IN THE GET BUFFER.

;CALL:
;(1) JSR PC,RCLSTS ;CALL SUBROUTINE
      BR ??? ;RETURN HERE IF NO ERROR
      NOP ;RETURN HERE TO REPORT AN ERROR
      ERROR ;ERROR NUMBER DEFINED BY SUB
      JSR PC,@(SP)+ ;GO BACK TO SUB FOR MORE ERROR CHECKS
      ??? ;RETURN HERE IF NO MORE ERRORS

RCLSTS:
;CLEAR USER'S ERROR NUMBER
      ADD #4,(SP)
      CLR @ (SP) ;CLEAR USER'S ERROR CALL
      SUB #4,(SP) ;MOVE SP BACK TO BRANCH

;SEE IF "PAR" OR "ILF" OR "OPI" OR "IAE" IS SET
      BIT #OPI!PAR!ILF!IAE,RMER1I
      BEQ 4$ ;NONE ARE SET - GO TO NEXT CHECK

;REPORT ANY MASSBUS CONTROL BUS PARITY ERROR, I.E.,
;"PAR" = 1 AND "DPE" = 0
      BIT #PAR,RMER1I ;WAS "PAR" SET??
      BEQ 1$ ;NO!!
      BIT #DPE,RMER2I ;WAS "DPE" SET??
      BNE 1$ ;YES - NOT A REGISTER ERROR
      MOV RMER1I,$GDDAT ;EXPECTED STATUS
      BIC #PAR,$GDDAT
      MOV RMER1I,$BDDAT ;RECEIVED STATUS
      ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
      MOV #50,@(SP) ;WRITE ERROR NUMBER IN CALL
      SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
      JSR PC,@(SP)+ ;GO REPORT ERROR
      SUB #10,(SP) ;MOVE SP BACK TO BRANCH
      NOP

1$:
;REPORT ANY "ILF" ERROR
      BIT #ILF,RMER1I ;WAS "ILF" SET??
      BEQ 2$ ;NO!!
      MOV RMER1I,$GDDAT ;EXPECTED STATUS
      BIC #ILF,$GDDAT
      MOV RMER1I,$BDDAT ;RECEIVED STATUS
      ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
      MOV #71,@(SP) ;WRITE ERROR NUMBER IN CALL
      SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
      JSR PC,@(SP)+ ;REPORT ERROR VIA USER
      SUB #10,(SP) ;MOVE SP BACK TO BRANCH
      NOP

2$:

```

```

055322
055322 062716 000004
055326 105076 000000
055332 162716 000004
055336 032737 022011 001350
055344 001553
055346 032737 000010 001350
055354 001430
055356 032737 000010 001376
055364 001024
055366 013737 001350 001140
055374 042737 000010 001140
055402 013737 001350 001142
055410 062716 000004
055414 112776 000050 000000
055422 162716 000002
055426 004736
055430 162716 000010
055434 000240
055436
055436 032737 000001 001350
055444 001424
055446 013737 001350 001140
055454 042737 000001 001140
055462 013737 001350 001142
055470 062716 000004
055474 112776 000071 000000
055502 162716 000002
055506 004736
055510 162716 000010
055514 000240
055516

```



```

58 ;REPORT ANY "OPI" ERROR AS
59 ; . OPI DUE TO "MOL" = 0
60 ; . OPI BECAUSE ON CYLINDER LATCH DIDN'T RESET
61 055516 032737 020000 001350 BIT #OPI,RMER11 ;WAS OPI SET??
62 055524 001433 BEQ 31$ ;NO!!
63 055526 013737 001350 001140 MOV RMER11,$GDDAT ;EXPECTED STATUS
64 055534 042737 020000 001140 BIC #OPI,$GDDAT
65 055542 013737 001350 001142 MOV RMER11,$BDDAT ;RECEIVED STATUS
66 055550 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
67 055554 112776 000072 000000 MOV #72,@(SP) ;WRITE ERROR NUMBER IN USER'S CALL
68 055562 032737 010000 001346 BIT #MOL,RMDSI ;WAS "MOL" = 0??
69 055570 001403 BEQ 3$ ;YES!!
70 055572 112776 000073 000000 MOV #73,@(SP) ;NO - CHANGE ERROR NUMBER
71 055600 162716 000002 3$: SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
72 055604 004736 JSR PC,@(SP)+ ;REPORT ERROR VIA USER
73 055606 162716 000010 SUB #10,(SP) ;MOVE SP BACK TO BRANCH
74 055612 000240 NOP
75 055614 31$:
76
77 ;REPORT AN ERROR IF "IAE" IS SET
78 055614 032737 002000 001350 BIT #IAE,RMER11 ;IS "IAE" SET??
79 055622 001424 BEQ 4$ ;NO!!
80 055624 013737 001350 001140 MOV RMER11,$GDDAT ;EXPECTED STATUS
81 055632 042737 002000 001140 BIC #IAE,$GDDAT
82 055640 013737 001350 001142 MOV RMER11,$BDDAT ;RECEIVED STATUS
83 055646 062716 000004 ADD #4,(SP) ;MOVE SP TO ERROR CALL
84 055652 112776 000070 000000 MOV #70,@(SP) ;WRITE ERROR NUMBER IN USER'S CALL
85 055660 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
86 055664 004736 JSR PC,@(SP)+ ;REPORT ERROR
87 055666 162716 000010 SUB #10,(SP) ;MOVE SP BACK TO NO ERROR RETURN
88 055672 000240 NOP
89 055674 4$:
90
91 ;SEE IF "SKI" OR "IVC" OR "DVC" IS SET
92 055674 032737 050200 001376 BIT #SKI!IVC!DVC,RMER21
93 055702 001517 BEQ 8$ ;NONE OF THE BITS ARE SET
94
95
96 ;REPORT ANY "IVC" ERROR AS
97 ; . IVC WITH VV = 0
98 ; . ERRONEOUS IVC ERROR
99 055704 032737 010000 001376 BIT #IVC,RMER21 ;WAS IVC SET??
100 055712 001433 BEQ 6$ ;NO!!
101 055714 013737 001376 001140 MOV RMER21,$GDDAT ;EXPECTED STATUS
102 055722 042737 010000 001140 BIC #IVC,$GDDAT
103 055730 013737 001376 001142 MOV RMER21,$BDDAT ;RECEIVED STATUS
104 055736 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
105 055742 112776 000074 000000 MOV #74,@(SP) ;WRITE ERROR NUMBER IN CALL
106 055750 032737 000100 001346 BIT #VV,RMDSI ;WAS VV = 0??
107 055756 001403 BEQ 5$ ;YES!!
108 055760 112776 000075 000000 MOV #75,@(SP) ;NO - CHANGE ERROR NUMBER
109 055766 162716 000002 5$: SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
110 055772 004736 JSR PC,@(SP)+ ;REPORT ERROR VIA USER
111 055774 162716 000010 SUB #10,(SP) ;MOVE SP BACK TO BRANCH
112 056000 000240 NOP
113 056002 6$:
114
    
```

```

115 ;REPORT ANY "SKI" ERROR
116 056002 032737 040000 001376 BIT #SKI,RMER2I ;WAS SKI SET??
117 056010 001424 BEQ 7$ ;NO!!
118 056012 013737 001376 001140 MOV RMER2I,$GDDAT ;EXPECTED STATUS
119 056020 042737 040000 001140 BIC #SKI,$GDDAT
120 056026 013737 001376 001142 MOV RMER2I,$BDDAT ;RECEIVED STATUS
121 056034 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
122 056040 112776 000076 000000 MOVVB #76,@(SP) ;WRITE ERROR NUMBER
123 056046 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
124 056052 004736 JSR PC,@(SP)+ ;REPORT ERROR VIA USER
125 056054 162716 000010 SUB #10,(SP) ;MOVE SP TO BRANCH
126 056060 000240 NOP
127 056062 7$:
128.
129 ;REPORT ANY "DVC" ERROR
130 056062 032737 000200 001376 BIT #DVC,RMER2I ;WAS "DVC" SET??
131 056070 001424 BEQ 8$ ;NO!!
132 056072 013737 001376 001140 MOV RMER2I,$GDDAT ;EXPECTED STATUS
133 056100 042737 000200 001140 BIC #DVC,$GDDAT
134 056106 013737 001376 001142 MOV RMER2I,$BDDAT ;RECEIVED STATUS
135 056114 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
136 056120 112776 000077 000000 MOVVB #77,@(SP) ;WRITE ERROR NUMBER
137 056126 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
138 056132 004736 JSR PC,@(SP)+ ;REPORT ERROR VIA USER
139 056134 162716 000010 SUB #10,(SP) ;MOVE SP TO USER'S BRANCH
140 056140 000240 NOP
141 056142 8$:
142
143 ;SEE IF "PIP" AND "OM" ARE 0, AND "ATA","MOL" AND "VV" ARE 1
144 056142 013746 001346 MOV RMDSI,-(SP) ;PUT RMDI ON STACK
145 056146 042716 047676 BIC #^C<PIP!MOL!VV!OM!ATA>,(SP)
146 056152 022726 110100 CMP #ATA!MOL!VV,(SP)+
147 056156 001002 BNE 85$
148 056160 000137 056574 JMP 13$
149 056164
150
151 ;REPORT AN ERROR IF MOL = 0 AND OPI = 0, I.E., MEDIUM WENT OFF
152 ;LINE AFTER RECALIBRATE WAS INITIATED
153 056164 032737 010000 001346 BIT #MOL,RMDSI ;DID MOL DROP??
154 056172 001030 BNE 9$ ;NO!!
155 056174 032737 020000 001350 BIT #OPI,RMER1I ;WAS OPI ERROR REPORTED??
156 056202 001024 BNE 9$ ;YES - DON'T REPORT MOL = 0
157 056204 013737 001346 001140 MOV RMDSI,$GDDAT ;EXPECTED STATUS
158 056212 052737 010000 001140 BIS #MOL,$GDDAT
159 056220 013737 001346 001142 MOV RMDSI,$BDDAT ;RECEIVED STATUS
160 056226 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
161 056232 112776 000100 000000 MOVVB #100,@(SP) ;WRITE ERROR NUMBER
162 056240 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
163 056244 004736 JSR PC,@(SP)+ ;REPORT ERROR VIA USER
164 056246 162716 000010 SUB #10,(SP) ;MOVE SP BACK TO USER'S BRANCH
165 056252 000240 NOP
166 056254 9$:
167
168 ;REPORT AN ERROR IF "VV" = 0 AND "IVC" = 0
169 056254 032737 000100 001346 BIT #VV,RMDSI ;DID "VV" DROP??
170 056262 001030 BNE 10$ ;NO!!
171 056264 032737 010000 001376 BIT #IVC,RMER2I ;WAS THERE A IVC ERROR??
    
```

```

172 056272 001024          BNE      10$          ;YES - DONT REPORT VV = 0
173 056274 013737 001346 001140  MOV      RMDSI,$GDDAT ;EXPECTED STATUS
174 056302 013737 001346 001142  MOV      RMDSI,$BDDAT ;RECEIVED STATUS
175 056310 052737 000100 001140  BIS      #VV,$GDDAT
176 056316 062716 000004          ADD      #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
177 056322 112776 000101 000000  MOVB    #101,@(SP)   ;WRITE ERROR NUMBER IN CALL
178 056330 162716 000002          SUB      #2,(SP)     ;MOVE SP TO RETURN FOR ERROR
179 056334 004736          JSR     PC,@(SP)+
180 056336 162716 000010          SUB      #10,(SP)   ;MOVE SP BACK TO USER'S BRANCH
181 056342 000240          NOP
182 056344          10$:
183
184          ;REPORT AN ERROR IF ATA IS NOT SET
185 056344 032737 100000 001346  BIT      #ATA,RMDSI  ;WAS ATA SET DURING RECALIBRATE??
186 056352 001024          BNE      11$          ;YES!!
187 056354 013737 001346 001140  MOV      RMDSI,$GDDAT ;EXPECTED STATUS
188 056362 052737 100000 001140  BIS      #ATA,$GDDAT
189 056370 013737 001346 001142  MOV      RMDSI,$BDDAT ;RECEIVED STATUS
190 056376 062716 000004          ADD      #4,(SP)     ;MOVE SP TO USER'S ERROR CALL
191 056402 112776 000102 000000  MOVB    #102,@(SP)   ;WRITE ERROR NUMBER IN CALL
192 056410 162716 000002          SUB      #2,(SP)
193 056414 004736          JSR     PC,@(SP)+
194 056416 162716 000010          SUB      #10,(SP)   ;MOVE SP TO USER'S BRANCH
195 056422 000240          NOP
196
197 056424          11$:
198
199          ;REPORT AN ERROR IF 'OM' IS NOT ZERO BECAUSE RECALIBRATE SHOULD
200          ;ALWAYS CLEAR OFFSET MODE
201 056424 032737 000001 001346  BIT      #OM,RMDSI   ;WAS 'OM' RESET??
202 056432 001424          BEQ     12$          ;YES!!
203 056434 013737 001346 001140  MOV      RMDSI,$GDDAT ;EXPECTED STATUS
204 056442 042737 000001 001140  BIC      #OM,$GDDAT
205 056450 013737 001346 001142  MOV      RMDSI,$BDDAT ;RECEIVED STATUS
206 056456 062716 000004          ADD      #4,(SP)     ;MOVE SP TO USER'S ERROR CALL
207 056462 112776 000103 000000  MOVB    #103,@(SP)   ;WRITE ERROR NUMBER
208 056470 162716 000002          SUB      #2,(SP)     ;MOVE SP TO RETURN FOR ERROR
209 056474 004736          JSR     PC,@(SP)+   ;REPORT ERROR VIA USER
210 056476 162716 000010          SUB      #10,(SP)   ;MOVE SP TO USER'S BRANCH
211 056502 000240          NOP
212 056504          12$:
213
214          ;REPORT AN ERROR IF 'PIP' IS STIL ON, I.E., DRIVE NOT ON
215          ;CYLINDER
216 056504 032737 020000 001346  BIT      #PIP,RMDSI  ;IS DRIVE OFF CYLINDER??
217 056512 001430          BEQ     13$          ;NO!!
218 056514 032737 040000 001376  BIT      #SKI,RMER2I ;WAS 'SKI' DETECTED??
219 056522 001024          BNE      13$          ;YES-DONT REPORT 'PIP'
220 056524 013737 001346 001140  MOV      RMDSI,$GDDAT ;EXPECTED STATUS
221 056532 042737 020000 001140  BIC      #PIP,$GDDAT
222 056540 013737 001346 001142  MOV      RMDSI,$BDDAT ;RECEIVED STATUS
223 056546 062716 000004          ADD      #4,(SP) ;MOVE SP TO USER'S ERROR CALL
224 056552 112776 000104 000000  MOVB    #104,@(SP)   ;WRITE ERROR NUMBER
225 056560 162716 000002          SUB      #2,(SP)     ;MOVE SP TO RETURN FOR ERROR
226 056564 004736          JSR     PC,@(SP)+
227 056566 162716 000010          SUB      #10,(SP)   ;MOVE SP BACK TO USER'S BRANCH
228 056572 000240          NOP

```

```

229 056574      13$:
230
231             ;SEE IF "ILR" OR "RMR" OR "UNS" IS SET
232 056574      032737 040006 001350      BIT      #ILR!RMR!UNS,RMER11
233 056602      001514             BEQ      16$
234
235             ;REPORT AN ERROR IF "ILR" IS SET
236 056604      032737 000002 001350      BIT      #ILR,RMER11      ;WAS ILR SET DURING RECALIBRATE??
237 056612      001424             BEQ      14$      ;NO!!
238 056614      013737 001350 001140      MOV      RMER11,$GDDAT      ;EXPECTED STATUS
239 056622      042737 000002 001140      BIC      #ILR,$GDDAT
240 056630      013737 001350 001142      MOV      RMER11,$BDDAT      ;RECEIVED STATUS
241 056636      062716 000004             ADD      #4,(SP) ;MOVE SP TO USER'S ERROR CALL
242 056642      112776 000105 000000      MOVVB   #105,@(SP) ;WRITE ERROR NUMBER IN CALL
243 056650      162716 000002             SUB      #2,(SP) ;MOVE SP TO RETURN FOR ERROR
244 056654      004736             JSR      PC,@(SP)+
245 056656      162716 000010             SUB      #10,(SP) ;MOVE SP TO USER'S BRANCH
246 056662      000240             NOP
247 056664
248
249             ;REPORT AN ERROR IF "RMR" IS SET
250 056664      032737 000004 001350      BIT      #RMR,RMER11      ;WAS RMR SET??
251 056672      001424             BEQ      15$      ;NO!!
252 056674      013737 001350 001140      MOV      RMER11,$GDDAT      ;EXPECTED STATUS
253 056702      042737 000004 001140      BIC      #RMR,$GDDAT
254 056710      013737 001350 001142      MOV      RMER11,$BDDAT      ;RECEIVED STATUS
255 056716      062716 000004             ADD      #4,(SP) ;MOVE SP TO USER'S ERROR CALL
256 056722      112776 000106 000000      MOVVB   #106,@(SP) ;WRITE ERROR NUMBER IN USER'S CALL
257 056730      162716 000002             SUB      #2,(SP) ;MOVE SP TO RETURN FOR ERROR
258 056734      004736             JSR      PC,@(SP)+ ;REPORT ERROR VIA USER
259 056736      162716 000010             SUB      #10,(SP) ;MOVE SP TO USER'S BRANCH
260 056742      000240             NOP
261 056744
262
263             ;REPORT AN ERROR IF "UNS" IS SET AND "DVC" IS 0
264 056744      032737 040000 001350      BIT      #UNS,RMER11      ;WAS UNSAFE ON??
265 056752      001430             BEQ      16$      ;NO!!
266 056754      032737 000200 001376      BIT      #DVC,RMER21      ;WAS THERE A DEVICE CHECK??
267 056762      001024             BNE     16$      ;YES - DON'T REPORT UNSAFE
268 056764      013737 001350 001140      MOV      RMER11,$GDDAT      ;EXPECTED STATUS
269 056772      042737 040000 001140      BIC      #UNS,$GDDAT
270 057000      013737 001350 001142      MOV      RMER11,$BDDAT      ;RECEIVED STATUS
271 057006      062716 000004             ADD      #4,(SP) ;MOVE SP TO USER'S ERROR CALL
272 057012      112776 000107 000000      MOVVB   #107,@(SP) ;WRITE ERROR NUMBER
273 057020      162716 000002             SUB      #2,(SP) ;MOVE SP TO RETURN FOR ERROR
274 057024      004736             JSR      PC,@(SP)+ ;REPORT ERROR VIA USER
275 057026      162716 000010             SUB      #10,(SP) ;MOVE SP BACK TO USER'S BRANCH
276 057032      000240             NOP
277 057034
278
279             ;AUGMENT THE RETURN ADDRESS IF ANY ERROR WAS DETECTED
280 057034      062716 000004             ADD      #4,(SP) ;MOVE SP TO USER'S ERROR CALL
281 057040      105776 000000             TSTB   @(SP) ;WAS AN ERROR REPORTED??
282 057044      001403             BEQ      17$      ;NO!!
283 057046      062716 000004             ADD      #4,(SP) ;YES - AUGMENT SP RETURN
284 057052      000402             BR      18$
285 057054      162716 000004             SUB      #4,(SP) ;NO ERROR - RETURN SP TO BRANCH
    
```

286 057060 000240
287 057062 000207
288

18\$: NOP
RTS PC

;STATUS CECK IS COMPLETE

```

1      .SBTTL  DRIVE CLEAR STATUS CHECK SUBROUTINE
2
3      :      BR      ???      RETURN HERE IF NO ERROR
4      :      NOP
5      :      ERROR      RETURN HERE TO REPORT AN ERROR
6      :      JSR      PC,@(SP)+  ERROR NUMBER DEFINED BY SUB
7      :      ???      GO BACK TO SUB FOR MORE ERROR CHECKS
8      :      RETURN HERE IF NO MORE ERRORS
9 057064  DRVSTS:
10
11      ;CLEAR USER'S ERROR CALL
12 057064 062716 000004      ADD      #4,(SP)      ;MOVE SP TO ERROR CALL
13 057070 105076 000000      CLR      @(SP)      ;CLEAR ERROR CALL
14 057074 162716 000004      SUB      #4,(SP)      ;MOVE SP TO USER'S BRANCH
15
16 057100 013737 001334 001142  ;REPORT ERROR IF RMCS1 NOT INITIALIZED
17 057106 042737 173700 001142  4$:  MOV      RMCS1,$BDDAT      ;CHECK RMCS1
18 057114 012737 004010 001140      BIC      #^C<DVA!FNCMSK>,$BDDAT ;CLEAR DONT CARES
19 057122 023737 001140 001142      MOV      #DVA!DRVCLR,$GDDAT      ;EXPECT DVA
20 057130 001443      CMP      $GDDAT,$BDDAT      ;COMPARE EXPECTED & RECEIVED
21 057132 062716 000004      BEQ      6$      ;BRANCH IF EQUAL
22 057136 112776 000141 000000      ADD      #4,(SP)      ;MOVE SP TO ERROR CALL
23 057144 162716 000002      MOV      #141,@(SP)      ;WRITE NUMBER OF ERROR IN CALL
24 057150 004736      SUB      #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
25 057152 162716 000010      JSR      PC,@(SP)+      ;REPORT THE ERROR VIA USER
26 057156 000240      SUB      #10,(SP)      ;MOVE SP TO NO ERROR RETURN
27      NOP
28 057160 013737 001346 001142  ;REPORT ERROR IF RMDS NOT INITIALIZED
29 057166 042737 021101 001142  5$:  MOV      RMDS1,$BDDAT      ;CHECK RMDS
30 057174 012737 010600 001140      BIC      #PGM!OM!VV!PIP,$BDDAT ;CLEAR DONT CARES
31 057202 023737 001140 001142      MOV      #MOL!DPR!DRY,$GDDAT      ;EXPECT DRY & DPR
32 057210 001413      CMP      $GDDAT,$BDDAT      ;COMPARE EXPECTED & RECEIVED
33 057212 062716 000004      BEQ      6$      ;BRANCH IF EQUAL
34 057216 112776 000142 000000      ADD      #4,(SP)      ;MOVE SP TO ERROR CALL
35 057224 162716 000002      MOV      #142,@(SP)      ;WRITE NUMBER OF ERROR IN CALL
36 057230 004736      SUB      #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
37 057232 162716 000010      JSR      PC,@(SP)+      ;REPORT THE ERROR VIA USER
38 057236 000240      SUB      #10,(SP)      ;MOVE SP TO NO ERROR RETURN
39      NOP
40 057240 005037 001140      ;REPORT ERROR IF RMER1 NOT INITIALIZED
41 057244 013737 001350 001142  6$:  CLR      $GDDAT      ;EXPECT 0'S
42 057252 001413      MOV      RMER1,$BDDAT      ;CHECK RMER1
43 057254 062716 000004      BEQ      8$      ;BRANCH IF EQUAL
44 057260 112776 000143 000000      ADD      #4,(SP)      ;MOVE SP TO ERROR CALL
45 057266 162716 000002      MOV      #143,@(SP)      ;WRITE NUMBER OF ERROR IN CALL
46 057272 004736      SUB      #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
47 057274 162716 000010      JSR      PC,@(SP)+      ;REPORT THE ERROR VIA USER
48 057300 000240      SUB      #10,(SP)      ;MOVE SP TO NO ERROR RETURN
49      NOP
50 057302 013737 001352 001142  ;REPORT ERROR IF ATA NOT INITIALIZED
51 057310 010146      8$:  MOV      RMAS1,$BDDAT      ;CHECK ATTENTION BIT
52 057312 010246      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
53 057314 013701 001464      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
54 057320 116102 000001      MOV      TSTQUE,R1
55 057324 042702 177400      MOV      1(R1),R2
56 057330 005102      BIC      #^CATNMSK,R2
57 057332 040237 001142      COM      R2
      BIC      R2,$BDDAT
    
```

58	057336	012602			MOV	(SP)+,R2	::POP STACK INTO R2
59	057340	012601			MOV	(SP)+,R1	::POP STACK INTO R1
60	057342	005737	001142		TST	\$BDDAT	:IS ATTENTION CLEARED??
61	057346	001413			BEQ	9\$:BRANCH IF ATTENTION CLEARED
62	057350	062716	000004		ADD	#4,(SP)	:MOVE SP TO ERROR CALL
63	057354	112776	000144	000000	MOVB	#144,@(SP)	:WRITE NUMBER OF ERROR IN CALL
64	057362	162716	000002		SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
65	057366	004736			JSR	PC,@(SP)+	:REPORT THE ERROR VIA USER
66	057370	162716	000010		SUB	#10,(SP)	:MOVE SP TO NO ERROR RETURN
67	057374	000240			NOP		
68							
69	057376	013737	001360	001142	9\$:	:REPORT ERROR IF RMMR1 NOT INITIALIZED	
70	057404	042737	000046	001142	MOV	RMMR1I,\$BDDAT	:CHECK RMMR
71	057412	012737	000010	001140	BIC	#WC!LS!LST,\$BDDAT	:CLEAR DONT CARES
72	057420	023737	001140	001142	MOV	#MWD,\$GDDAT	:EXPECT WRITE DATA ON
73	057426	001413			CMP	\$GDDAT,\$BDDAT	:COMPARE EXPECTED AND RECEIVED
74	057430	062716	000004		BEQ	11\$:BRANCH IF ZERO
75	057434	112776	000145	000000	ADD	#4,(SP)	:MOVE SP TO ERROR CALL
76	057442	162716	000002		MOVB	#145,@(SP)	:WRITE NUMBER OF ERROR IN CALL
77	057446	004736			SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
78	057450	162716	000010		JSR	PC,@(SP)+	:REPORT THE ERROR VIA USER
79	057454	000240			SUB	#10,(SP)	:MOVE SP TO NO ERROR RETURN
80					NOP		
81	057456	013737	001374	001142	11\$:	:REPORT ERROR IF RMMR2 NOT INITIALIZED	
82	057464	042737	140000	001142	MOV	RMMR2I,\$BDDAT	:CHECK RMMR2
83	057472	012737	011777	001140	BIC	#RQA!RQB,\$BDDAT	:CLEAR REQA, REQB
84	057500	023737	001140	001142	MOV	#TST!1777,\$GDDAT	:EXPECT TEST BIT ON
85	057506	001413			CMP	\$GDDAT,\$BDDAT	:COMPARE EXPECTED & RECEIVED
86	057510	062716	000004		BEQ	15\$:BRANCH IF EQUAL
87	057514	112776	000146	000000	ADD	#4,(SP)	:MOVE SP TO ERROR CALL
88	057522	162716	000002		MOVB	#146,@(SP)	:WRITE NUMBER OF ERROR IN CALL
89	057526	004736			SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
90	057530	162716	000010		JSR	PC,@(SP)+	:REPORT THE ERROR VIA USER
91	057534	000240			SUB	#10,(SP)	:MOVE SP TO NO ERROR RETURN
92	057536	005037	001140		NOP		
93					15\$:	CLR \$GDDAT	:EXPECT ZEROS
94	057542	013737	001402	001142	17\$:	:REPORT ERROR IF RMEC2 NOT RESET	
95	057550	001413			MOV	RMEC2I,\$BDDAT	:CHECK RMEC2
96	057552	062716	000004		BEQ	17\$:BRANCH IF 0
97	057556	112776	000150	000000	ADD	#4,(SP)	:MOVE SP TO ERROR CALL
98	057564	162716	000002		MOVB	#150,@(SP)	:WRITE NUMBER OF ERROR IN CALL
99	057570	004736			SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
100	057572	162716	000010		JSR	PC,@(SP)+	:REPORT THE ERROR VIA USER
101	057576	000240			SUB	#10,(SP)	:MOVE SP TO NO ERROR RETURN
102					NOP		
103	057600	013737	001376	001142	18\$:	:REPORT ERROR IF RMER2 NOT RESET	
104	057606	001413			MOV	RMER2I,\$BDDAT	:CHECK RMER2
105	057610	062716	000004		BEQ	18\$:BRANCH IF NO ERROR
106	057614	112776	000147	000000	ADD	#4,(SP)	:MOVE SP TO ERROR CALL
107	057622	162716	000002		MOVB	#147,@(SP)	:WRITE NUMBER OF ERROR IN CALL
108	057626	004736			SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
109	057630	162716	000010		JSR	PC,@(SP)+	:REPORT THE ERROR VIA USER
110	057634	000240			SUB	#10,(SP)	:MOVE SP TO NO ERROR RETURN
111	057636				NOP		
112					18\$:		
113	057636				19\$:		
114							

```
DRIVE CLEAR STATUS CHECK SUBROUTINE

115
116 057636 062716 000004
117 057642 105776 000000
118 057646 001403
119 057650 062716 000004
120 057654 000402
121 057656 162716 000004
122 057662 000240
123 057664 000207
124

;AUGMENT RETURN ADDRESS IF ANY ERROR WAS FOUND
ADD #4,(SP) ;MOVE SP TO ERROR CALL
TSTB @ (SP) ;WAS AN ERROR DETECTED??
BEQ 21$ ;NO!!
ADD #4,(SP) ;YES - MOVE SP TO ERROR RETURN
BR 23$
21$: SUB #4,(SP) ;MOVE SP BACK TO NO ERROR RETURN
23$: NOP
RTS PC ;RETURN TO USER
```


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

.SBTTL SEARCH STATUS CHECK SUBROUTINE

;THIS SUBROUTINE VERIFIES THE RESULTS OF SEARCH OPERATIONS USING
 ;STATUS STORED IN THE GET BUFFER AND TEST CONDITIONS STORED IN THE
 ;PUT BUFFER.

;THE SUBROUTINE RETURNS TO THE CALLING ROUTINE IF AN ERROR IS
 ;DETECTED AFTER HAVING LOADED THE APPROPRIATE ERROR NUMBER IN
 ;THE USER'S "ERROR" TRAP.

;THE FOLLOWING CONDITIONS ARE CHECKED:

.ANY ERROR WHICH OCCURRED WHILE READING OR WRITING REMOTE
 ;REGISTERS IS REPORTED, I.E., "MCPE" = 1 OR "PAR" = 1

;"IAE" STATUS IS CHECKED IN ACCORDANCE WITH THE VALUE DETERMINED
 ;BY THE PROGRAM, WHICH IS BASED ON FORMAT AND ADDRESS.

;"OPI", IF SET, IS REPORTED AS 1) "OPI" DUE TO MOL = 0, OR 2)
 ;"OPI" DUE TO ON CYLINDER LATCH.

;"IVC", IF SET, IS REPORTED AS 1) "IVC" ERROR WITH VOLUME
 ;VALID ZERO, OR 2) ERRONEOUS "IVC" ERROR WITH VOLUME VALID SET.

;"SKI" IS REPORTED IF SET

;"DVC" IS REPORTED IF SET

.AN ERROR IS REPORTED IF "MOL" = 0, OR "PIP" = 1, OR "ATA" = 0,
 ;OR "VV" = 0.

;CALL:

;(1)	JSR	PC,SCHSTS	
;(2)	BR	??	RETURN HERE IF NO ERROR
;(3)	NOP		RETURN HERE TO REPORT ERROR
;(4)	ERROR		SUBROUTINE WILL LOAD ERROR #
;(5)	JSR	PC,@(SP)+	GO BACK FOR MORE CHECKS
;(6)	??		RETURN AFTER ALL ERRORS REPORTED

SCHSTS:

;CLEAR USER'S ERROR CALL

ADD	#4,(SP)	;MOVE SP TO USER'S ERROR CALL
CLRB	@(SP)	;CLEAR ERROR NUMBER
SUB	#4,(SP)	;MOVE SP BACK TO NO ERROR BR
CLR	200\$;CLEAR STATUS FLAGS

;TEST FOR MASSBUS CONTROL BUS PARITY ERROR WHEN WRITING REMOTE
 ;REGISTERS, I.E., "PAR" = 1 AND "DPE" = 0.

BIT	#PAR,RMER11	;WAS PARITY ERROR DETECTED??
BEQ	10\$;NO!!
BIT	#DPE,RMER21	;WAS IT CONTROL BUS ERROR??
BNE	10\$;PROBABLY NOT!!

;REPORT CONTROL BUS PARITY ERROR VIA USER'S ERROR CALL

MOV	RMER11,\$GDDAT	;EXPECTED STATUS
BIC	#PAR,\$GDDAT	

057666

057666	062716	000004	
057672	105076	000000	
057676	162716	000004	
057702	005037	061230	

057706	032737	000010	001350
057714	001431		
057716	032737	000010	001376
057724	001025		

057726	013737	001350	001140
057734	042737	000010	001140

```

58 057742 013737 001350 001142      MOV      RMER11,$BDDAT      ;RECEIVED STATUS
59 057750 062716 000004                ADD      #4,(SP)           ;MOVE SP TO USER'S ERROR CALL
60 057754 112776 000050 000000      MOVVB   #50,@(SP)         ;WRITE ERROR NUMBER IN CALL
61 057762 162716 000002                SUB      #2,(SP)           ;MOVE SP TO RETURN IF ERROR
62 057766 004736                JSR     PC,@(SP)+         ;REPORT ERROR
63 057770 162716 000010      SUB      #10,(SP)        ;RESTORE SP TO NO ERROR
64 057774 000240                NOP
65 057776 000430                BR       15$              ;SKIP FURTHER ERROR CHECKS
66 060000
67
68
69
70 060000 032737 020000 001334      BIT     #MCPE,RMCS11     ;WAS PARITY ERROR DETECTED??
71 060006 001426                BEQ     20$              ;NO!!
72
73
74 060010 013737 001334 001140      MOV     RMCS11,$GDDAT    ;EXPECTED STATUS
75 060016 042737 020000 001140      BIC     #MCPE,$GDDAT
76 060024 013737 001334 001142      MOV     RMCS11,$BDDAT    ;RECEIVED STATUS
77 060032 062716 000004                ADD     #4,(SP)           ;MOVE SP TO USER'S ERROR CALL
78 060036 112776 000013 000000      MOVVB  #13,@(SP)        ;WRITE ERROR NUMBER
79 060044 162716 000002                SUB     #2,(SP)           ;MOVE SP TO RETURN IF ERROR
80 060050 004736                JSR    PC,@(SP)+         ;REPORT ERROR AND RETURN
81 060052 162716 000010      SUB     #10,(SP)        ;RESTORE SP TO NO ERROR
82 060056 000240                NOP
83 060060 000137 061202      15$:   JMP     150$           ;OMIT STATUS CHECKING
84 060064
85
86
87
88 060064 012737 002000 001140      MOV     #IAE,$GDDAT      ;SETUP FOR IAE = 1
89 060072 052737 040000 061230      BIS     #SKI,200$        ;SETUP FOR SKI = 1
90 060100 023727 001444 001466      CMP     RMDCO,#822.      ;GREATER THAN LAST CYLINDER ?
91 060106 101025                BHI    30$               ;YES - CYLINDER IS INVALID
92 060110 042737 040000 061230      BIC     #SKI,200$        ;"SKI" SHOULD BE ZERO
93
94 060116 123737 001417 001333      CMPB   RMDAO+1,LSTRK+1  ;GREATER THAN LAST TRACK ?
95 060124 101016                BHI    30$               ;YES - TRACK IS INVALID
96
97 060126 123727 001416 000035      CMPB   RMDAO,#29.        ;SECTOR > 29. ?
98 060134 101410                BLOS   25$               ;NO!!
99 060136 032737 010000 001442      BIT     #FMT16,RMOFO     ;18 BIT FORMAT ?
100 050144 001406                BEQ    30$               ;YES - SECTOR IS INVALID FOR 18 BIT MODE
101 060146 123727 001416 000037      CMPB   RMDAO,#31.        ;SECTOR > 31. ?
102 060154 101002                BHI    30$               ;YES - SECTOR IS INVALID
103
104 060156 005037 001140      25$:   CLR     $GDDAT         ;"IAE" SHOULD = 0
105
106
107 060162 013737 001350 001142      30$:   MOV     RMER11,$BDDAT    ;GET RECEIVED IAE
108 060170 042737 175777 001142      BIC     #^CIAE,$BDDAT
109 060176 023737 001140 001142      CMP     $GDDAT,$BDDAT    ;IS IAE CORRECT??
110 060204 001004                BNE    40$               ;NO!!
111 060206 042737 040000 061230      BIC     #SKI,200$        ;SKI SHOULD BE ZERO
112 060214 000413                BR      50$
113 060216
114
40$:

```

```

115 ;REPORT INCORRECT IAE STATUS VIA USER'S ERROR CALL
116 060216 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
117 060222 112776 000257 000000 MOVB #257,@(SP) ;WRITE ERROR NUMBER IN CALL
118 060230 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
119 060234 004736 JSR PC,@(SP)+ ;REPORT ERROR AND RETURN
120 060236 162716 000010 SUB #10,(SP) ;RETURN SP TO NO ERROR
121 060242 000240 NOP
122 060244 50$:
123
124 ;SEE IF "SKI" OR "DVC" OR "IVC" IS SET
125 060244 032737 050200 001376 BIT #SKI!DVC!IVC,RMER2I ;ARE ANY BITS SET??
126 060252 001531 BEQ 90$ ;NO!!
127
128 ;REPORT ERROR IF "SKI" IS SET AND "SKI" FLAG IS NOT SET
129 060254 013737 001376 001142 MOV RMER2I,$BDDAT ;RECEIVED "SKI" STATUS
130 060262 042737 137777 001142 BIC #^CSKI,$BDDAT
131 060270 013737 061230 001140 MOV 200$,$GDDAT ;EXPECTED "SKI" STATUS
132 060276 042737 137777 001140 BIC #^CSKI,$GDDAT
133 060304 023737 001140 001142 CMP $GDDAT,$BDDAT ;IS "SKI" OK??
134 060312 001422 BEQ 60$ ;YES-CHANGE ERROR NUMBER
135 060314 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
136 060320 112776 000263 000000 MOVB #263,@(SP) ;WRITE ERROR NUMBER
137 060326 032737 040000 001140 BIT #SKI,$GDDAT ;SHOULD "SKI" BE SET??
138 060334 001403 BEQ 55$ ;NO!!
139 060336 112776 000267 000000 MOVB #267,@(SP) ;YES-CHANGE ERROR NUMBER
140 060344 162716 000002 55$: SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
141 060350 004736 JSR PC,@(SP)+ ;REPORT ERROR AND RETURN
142 060352 162716 000010 SUB #10,(SP) ;RESTORE SP TO NO ERROR
143 060356 000240 NOP
144 060360 60$:
145
146 ;REPORT "IVC" ERROR AS
147 :
148 : .IVC DUE TO LOSS OF VOLUME VALID
149 : .ERRONEOUS IVC WITH VOLUME VALID SET
149 060360 032737 010000 001376 BIT #IVC,RMER2I ;IS IVC SET??
150 060366 001433 BEQ 80$ ;NO!!
151 060370 013737 001376 001140 MOV RMER2I,$GDDAT ;EXPECTED STATUS
152 060376 042737 010000 001140 BIC #IVC,$GDDAT
153 060404 013737 001376 001142 MOV RMER2I,$BDDAT ;RECEIVED STATUS
154 060412 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
155 060416 112776 000264 000000 MOVB #264,@(SP) ;WRITE ERROR NUMBER IN CALL
156 060424 032737 000100 001346 BIT #VV,RMSDI ;WAS VOLUME VALID??
157 060432 001403 BEQ 70$ ;NO!!
158 060434 112776 000265 000000 MOVB #265,@(SP) ;YES - CHANGE ERROR NUMBER
159 060442 162716 000002 70$: SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
160 060446 004736 JSR PC,@(SP)+ ;REPORT ERROR AND RETURN
161 060450 162716 000010 SUB #10,(SP) ;RESTORE SP TO NO ERROR
162 060454 000240 NOP
163 060456 80$:
164
165 ;REPORT ANY DEVICE FAULT, I.E., "DVC" = 1
166 060456 032737 000200 001376 BIT #DVC,RMER2I ;WAS THERE A DEVICE FAULT??
167 060464 001424 BEQ 90$ ;NO!!
168 060466 013737 001376 001140 MOV RMER2I,$GDDAT ;EXPECTED STATUS
169 060474 042737 000200 001140 BIC #DVC,$GDDAT
170 060502 013737 001376 001142 MOV RMER2I,$BDDAT ;RECEIVED STATUS
171 060510 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL

```

```

172 060514 112776 000266 000000      MOVB    #266,@(SP)      ;WRITE ERROR NUMBER IN CALL
173 060522 162716 000002              SUB     #2,(SP)         ;MOVE SP TO RETURN IF ERROR
174 060526 004736                      JSR    PC,@(SP)+       ;REPORT ERROR AND RETURN
175 060530 162716 000010              SUB     #10,(SP)        ;RESTORE SP TO NO ERROR
176 060534 000240                      NOP
177 060536                      90$:
178
179                      ;REPORT ANY "OPI" ERROR AS
180                      ;"OPI" BECAUSE MEDIUM IS NOT ONLINE
181                      ;"OPI" BECAUSE "ON CYLINDER" DIDN'T DROP
182 060536 032737 020000 001350      BIT     #OPI,RMER1I    ;WAS OPI SET??
183 060544 001433                      BEQ    110$            ;NO!!
184 060546 013737 001350 001140      MOV     RMER1I,$GDDAT  ;EXPECTED STATUS
185 060554 042737 020000 001140      BIC    #OPI,$GDDAT
186 060562 013737 001350 001142      MOV     RMER1I,$BDDAT ;RECEIVED STATUS
187 060570 062716 000004              ADD     #4,(SP)        ;MOVE SP TO USER'S ERROR CALL
188 060574 112776 000270 000000      MOVB   #270,@(SP)     ;SETUP ERROR FOR MOL = 0
189 060602 032737 010000 001346      BIT     #MOL,RMDSI    ;WAS MOL 0??
190 060610 001403                      BEQ    105$            ;YES!!
191 060612 112776 000271 000000      MOVB   #271,@(SP)     ;MOL WAS 1 - CHANGE ERROR
192 060620 162716 000002 105$:      SUB     #2,(SP)        ;MOVE SP TO RETURN IF ERROR
193 060624 004736                      JSR    PC,@(SP)+       ;REPORT ERROR AND RETURN
194 060626 162716 000010              SUB     #10,(SP)        ;RESTORE SP TO NO ERROR
195 060632 000240                      NOP
196 060634                      110$:
197
198                      ;SEE IF "ATA" = "MOL" = "VV" = 1 AND "PIP" = 0
199 060634 013746 001346              MOV     RMDSI,-(SP)    ;GET DRIVE STATUS
200 060640 042716 047677              BIC    #^C<ATA!PIP!MOL!VV>,(SP)
201 060644 022726 110100              CMP     #ATA!MOL!VV,(SP)+ ;IS DRIVE STATUS CORRECT??
202 060650 001554                      BEQ    150$            ;YES!!
203
204                      ;REPORT AN ERROR IF MOL = 0 AND OPI ERROR WAS NOT REPORTED, I.E., OPI = 0
205 060652 032737 010000 001346      BIT     #MOL,RMDSI    ;WAS MEDIUM OFF LINE??
206 060660 001030                      BNE    120$            ;NO!!
207 060662 013737 001346 001140      MOV     RMDSI,$GDDAT  ;EXPECTED STATUS
208 060670 052737 010000 001140      BIS    #MOL,$GDDAT
209 060676 013737 001346 001142      MOV     RMDSI,$BDDAT ;RECEIVED STATUS
210 060704 032737 020000 001350      BIT     #OPI,RMER1I    ;WAS OPI REPORTED BEFORE??
211 060712 001013                      BNE    120$            ;YES - DON'T REPORT MOL = 0
212 060714 062716 000004              ADD     #4,(SP)        ;MOVE SP TO USER'S ERROR CALL
213 060720 112776 000272 000000      MOVB   #272,@(SP)     ;WRITE ERROR NUMBER IN CALL
214 060726 162716 000002              SUB     #2,(SP)        ;MOVE SP TO RETURN IF ERROR
215 060732 004736                      JSR    PC,@(SP)+       ;REPORT ERROR AND RETURN
216 060734 162716 000010              SUB     #10,(SP)        ;RESTORE SP TO NO ERROR
217 060740 000240                      NOP
218 060742                      120$:
219
220                      ;REPORT AN ERROR IF PIP IS STIL SET AND SKI IS RESET
221 060742 032737 020000 001346      BIT     #PIP,RMDSI    ;IS POSITIONING IN PROGRESS??
222 060750 001430                      BEQ    130$            ;NO!!
223 060752 032737 040000 001376      BIT     #SKI,RMER2I    ;WAS "SKI" DETECTED??
224 060760 001024                      BNE    130$            ;YES-DONT REPORT PIP
225 060762 013737 001346 001140      MOV     RMDSI,$GDDAT  ;EXPECTED STATUS
226 060770 042737 020000 001140      BIC    #PIP,$GDDAT
227 060776 013737 001346 001142      MOV     RMDSI,$BDDAT ;RECEIVED STATUS
228 061004 062716 000004              ADD     #4,(SP)        ;MOVE SP TO USER'S ERROR CALL

```

```

229 061010 112776 000273 000000      MOVB    #273,@(SP)      ;WRITE ERROR NUMBER IN CALL
230 061016 162716 000002                SUB     #2,(SP)         ;MOVE SP TO RETURN IF ERROR
231 061022 004736                JSR    PC,@(SP)+       ;REPORT ERROR AND RETURN
232 061024 162716 000010                SUB     #10,(SP)       ;RESTORE SP TO NO ERROR
233 061030 000240                NOP
234 061032                130$:
235
236                ;REPORT AN ERROR IF VOLUME IS NOT VALID AND IVC = 0
237 061032 032737 000100 001346      BIT     #VV,RMDSI      ;IS VOLUME VALID??
238 061040 001030                BNE    140$           ;YES!!
239 061042 032737 010000 001376      BIT     #IVC,RMER2I    ;WAS IVC DETECTED??
240 061050 001024                BNE    140$           ;YES - DON'T REPORT VV = 0
241 061052 013737 001346 001140      MOV     RMDSI,$GDDAT   ;EXPECTED STATUS
242 061060 052737 000100 001140      BIS     #VV,$GDDAT
243 061066 013737 001346 001142      MOV     RMDSI,$BDDAT   ;RECEIVED STATUS
244 061074 062716 000004                ADD     #4,(SP)        ;MOVE SP TO USERS ERROR CALL
245 061100 112776 000350 000000      MOVB   #350,@(SP)     ;WRITE ERROR NUMBER IN CALL
246 061106 162716 000002                SUB     #2,(SP)        ;MOVE SP TO RETURN IF ERROR
247 061112 004736                JSR    PC,@(SP)+       ;REPORT ERROR AND RETURN
248 061114 162716 000010                SUB     #10,(SP)       ;RESTORE SP TO NO ERROR
249 061120 000240                NOP
250 061122                140$:
251
252                ;REPORT AN ERROR IF ATTENTION IS NOT SET
253 061122 032737 100000 001346      BIT     #ATA,RMDSI     ;IS ATA ON??
254 061130 001024                BNE    150$           ;YES!!
255 061132 013737 001346 001140      MOV     RMDSI,$GDDAT   ;EXPECTED STATUS
256 061140 052737 100000 001140      BIS     #ATA,$GDDAT
257 061146 013737 001346 001142      MOV     RMDSI,$BDDAT   ;RECEIVED STATUS
258 061154 062716 000004                ADD     #4,(SP)        ;MOVE SP TO USER'S ERROR CALL
259 061160 112776 000351 000000      MOVB   #351,@(SP)     ;WRITE ERROR NUMBER IN CALL
260 061166 162716 000002                SUB     #2,(SP)        ;MOVE SP TO RETURN IF ERROR
261 061172 004736                JSR    PC,@(SP)+       ;REPORT ERROR AND RETURN
262 061174 162716 000010                SUB     #10,(SP)       ;RESTORE SP TO NO ERROR
263 061200 000240                NOP
264 061202                150$:
265
266                ;ARGUMENT THE RETURN ADDRESS IF AN ERROR WAS DETECTED
267 061202 062716 000004                ADD     #4,(SP)        ;MOVE SP TO USER'S ERROR CALL
268 061206 105776 000000                TSTB   @(SP)          ;WAS ERROR FOUND??
269 061212 001403                BEQ    160$           ;NO!!
270 061214 062716 000004                ADD     #4,(SP)        ;YES - CHANGE RETURN
271 061220 000402                BR     170$
272 061222 162716 000004                160$: SUB     #4,(SP)     ;NO ERROR FOUND
273 061226 000207                170$: RTS    PC        ;RETURN TO USER
274
275 061230 000000                200$: .WORD          ;STORAGE FOR FLAGS
276

```

```

1      .SBTTL  STATIC DRIVE STATUS CHECK SUBROUTINE
2
3      ;THIS SUBROUTINE LOOKS FOR UNEXPECTED CHANGES IN DRIVE
4      ;STATUS, SUCH AS THE DRIVE LOSING VOLUME VALID.  THE SUBROUTINE
5      ;CAN BE USED BY HOUSEKEEPING AND OTHER COMMANDS DURING WHICH THERE
6      ;SHOULD NOT BE ANY DRIVE ERRORS OR CHANGES IN STATE.
7
8      ;THE FOLLOWING CONDITIONS ARE TESTED AND REPORTED AS ERRORS
9      ;IF TRUE:
10
11      ;      .MOL = 0, INDICATES DRIVE WENT OFFLINE, NOTE
12      ;THAT MOL IS ASSUMED TO HAVE BEEN SET
13      ;      .VV = 0, INDICATES THE DRIVE LOST VOLUME VALID
14      ;      .PIP = 1, INDICATES THAT THE DRIVE IS OFF CYLINDER
15      ;      .SKI = 1, INDICATES THE DRIVE HAS AN UNEXPECTED SKI ERROR
16      ;      .DVC = 1, INDICATES AN UNEXPECTED DEVICE FAULT
17
18      ;THE SUBROUTINE IS CALLED AFTER STORING STATUS IN THE GET BUFFER.
19
20      ;(1)  JSR      PC,STCDRVSTS
21      ;      BR      ???          RETURN HERE IF NO ERROR
22      ;      NOP          RETURN HERE TO REPORT AN ERROR
23      ;      ERROR     ERROR NUMBER DEFINED BY SUB
24      ;      JSR      PC,@(SP)+   GO BACK TO SUB FOR MORE ERROR CHECKS
25      ;      ???          RETURN HERE IF NO MORE ERRORS
26
27      STCDRVSTS:
28
29      ;CLEAR USER'S ERROR CALL
30      ADD      #4,(SP) ;MOVE SP TO USER'S ERROR CALL
31      CLRB    @ (SP)   ;CLEAR ERROR NUMBER
32      SUB      #4,(SP) ;MOVE SP BACK TO NO ERROR RETURN
33
34      ;SEE IF "MOL" = "VV" = 1, AND "PIP" = 0
35      MOV      RMDSI,-(SP) ;PUT DRIVE STATUS ON STACK
36      BIC     #^C<PIP!MOL!VV>,(SP)
37      CMP     #MOL!VV,(SP)+ ;ARE MOL,VV AND PIP O.K.??
38      BEQ     30$        ;YES!!
39
40      ;REPORT AN ERROR IF MOL = 0 AND "OPI" = 0
41      BIT     #MOL,RMDSI ;IS MOL ON ??
42      BNE     10$        ;YES!!
43      BIT     #OPI,RMER11 ;WAS "OPI" SET??
44      BNE     10$        ;YES-DONT REPORT "MOL" = 0
45      MOV     RMDSI,$GDDAT ;EXPECTED STATUS
46      BIS     #MOL,$GDDAT
47      MOV     RMDSI,$BDDAT ;RECEIVED STATUS
48      ADD     #4,(SP) ;MOVE SP TO USER'S ERROR CALL
49      MOVB   #207,@(SP) ;WRITE ERROR NUMBER IN CALL
50      SUB     #2,(SP) ;MOVE SP TO RETURN FOR ERROR
51      JSR     PC,@(SP)+ ;REPORT ERROR VIA USER
52      SUB     #10,(SP) ;MOVE SP BACK TO NO ERROR RETURN
53      NOP
54
55      10$:
56
57      ;REPORT AN ERROR IF VOLUME VALID IS NOW ZERO AND "IVC" = 0
58      BIT     #VV,RMDSI ;IS "VV" = 0??
59      BNE     20$        ;NO!!

```

```

27 061232
28
29
30 061232 062716 000004
31 061236 105076 000000
32 061242 162716 000004
33
34 061246 013746 001346
35 061252 042716 147677
36 061256 022726 010100
37 061262 001524
38
39
40 061264 032737 010000 001346
41 061272 001030
42 061274 032737 020000 001350
43 061302 001024
44 061304 013737 001346 001140
45 061312 052737 010000 001140
46 061320 013737 001346 001142
47 061326 062716 000004
48 061332 112776 000207 000000
49 061340 162716 000002
50 061344 004736
51 061346 162716 000010
52 061352 000240
53 061354
54
55
56 061354 032737 000100 001346
57 061362 001030

```

```

58 061364 032737 010000 001376      BIT      #IVC,RMER2I      ;WAS "IVC" SET??
59 061372 001024                      BNE      20$             ;YES-DONT REPORT "VV" = 0
60 061374 013737 001346 001140      MOV      RMDSI,$GDDAT    ;EXPECTED STATUS
61 061402 052737 000100 001346      BIS      #VV,RMDSI
62 061410 013737 001346 001142      MOV      RMDSI,$BDDAT    ;RECEIVED STATUS
63 061416 062716 000004                      ADD      #4,(SP) ;MOVE SP TO USER'S ERROR CALL
64 061422 112776 000210 000000      MOVVB   #210,@(SP)      ;WRITE ERROR NUMBER IN CALL
65 061430 162716 000002                      SUB      #2,(SP)         ;MOVE SP TO RETURN FOR ERROR
66 061434 004736                      JSR      PC,@(SP)+       ;REPORT ERROR VIA USER
67 061436 162716 000010                      SUB      #10,(SP)        ;MOVE SP BACK TO NO ERROR
68 061442 000240                      NOP
69 061444                                20$:
70
71                                ;REPORT AN ERROR IF DRIVE IS OFF CYLINDER AND "SKI" = 0
72 061444 032737 020000 001346      BIT      #PIP,RMDSI      ;IS DRIVE OFF CYLINDER??
73 061452 001430                      BEQ      30$             ;NO!!
74 061454 032737 040000 001376      BIT      #SKI,RMER2I     ;WAS "SKI" SET??
75 061462 001024                      BNE      30$             ;YES-DONT REPORT "PIP" = 1
76 061464 013737 001346 001140      MOV      RMDSI,$GDDAT    ;EXPECTED STATUS
77 061472 042737 020000 001140      BIC      #PIP,$GDDAT
78 061500 013737 001346 001142      MOV      RMDSI,$BDDAT    ;RECEIVED STATUS
79 061506 062716 000004                      ADD      #4,(SP)         ;MOVE SP TO USER'S ERROR CALL
80 061512 112776 000211 000000      MOVVB   #211,@(SP)      ;WRITE ERROR NUMBER IN USER'S CALL
81 061520 162716 000002                      SUB      #2,(SP)         ;MOVE SP TO RETURN FOR ERROR
82 061524 004736                      JSR      PC,@(SP)+       ;REPORT ERROR VIA USER
83 061526 162716 000010                      SUB      #10,(SP)        ;MOVE SP TO NO ERROR RETURN
84 061532 000240                      NOP
85 061534                                30$:
86
87                                ;SEE IF "SKI" = "DVC" = 0
88 061534 013746 001376                      MOV      RMER2I,-(SP)     ;PUT ERROR REG 2 ON STACK
89 061540 042726 137577                      BIC      #^C<SKI!DVC>,(SP)+
90 061544 001460                      BEQ      60$             ;BRANCH IF NO ERROR
91 061546                                40$:
92
93                                ;REPORT AN ERROR IF THERE IS A DEVICE FAULT
94 061546 032737 000200 001376      BIT      #DVC,RMER2I     ;ANY DEVICE FAULT??
95 061554 001424                      BEQ      50$             ;NO!!
96 061556 013737 001376 001140      MOV      RMER2I,$GDDAT    ;EXPECTED STATUS
97 061564 042737 000200 001140      BIC      #DVC,$GDDAT
98 061572 013737 001376 001142      MOV      RMER2I,$BDDAT    ;RECEIVED STATUS
99 061600 062716 000004                      ADD      #4,(SP)         ;MOVE SP TO USER'S CALL
100 061604 112776 000212 000000      MOVVB   #212,@(SP)      ;WRITE NUMBER OF ERROR IN CALL
101 061612 162716 000002                      SUB      #2,(SP)         ;MOVE SP TO RETURN FOR ERROR
102 061616 004736                      JSR      PC,@(SP)+       ;REPORT ERROR VIA USER
103 061620 162716 000010                      SUB      #10,(SP)        ;MOVE SP BACK TO NO ERROR
104 061624 000240                      NOP
105 061626                                50$:
106
107                                ;REPORT AN ERROR IF "SKI" = 1
108 061626 032737 040000 001376      BIT      #SKI,RMER2I     ;IS THERE A SEEK INCOMPLETE ERROR
109 061634 001424                      BEQ      60$             ;NO!!
110 061636 013737 001376 001140      MOV      RMER2I,$GDDAT    ;EXPECTED STATUS
111 061644 042737 040000 001140      BIC      #SKI,$GDDAT
112 061652 013737 001376 001142      MOV      RMER2I,$BDDAT    ;RECEIVED STATUS
113 061660 062716 000004                      ADD      #4,(SP)         ;MOVE SP TO USER'S ERROR CALL
114 061664 112776 000213 000000      MOVVB   #213,@(SP)      ;WRITE ERROR NUMBER IN USER'S ERROR CALL
    
```

115	061672	162716	000002		SUB	#2,(SP)		;MOVE SP TO RETURN FOR ERROR
116	061676	004736			JSR	PC,@(SP)+		;REPORT ERROR VIA USER
117	061700	162716	000010		SUB	#10,(SP)		;MOVE SP BACK TO NO ERROR
118	061704	000240			NOP			
119	061706			60\$:				
120								
121								
122	061706	062716	000004					
123	061712	105776	000000		ADD	#4,(SP)		;MOVE SP TO USER'S ERROR CALL
124	061716	001403			TSTB	@(SP)		;WAS AN ERROR DETECTED??
125	061720	062716	000004		BEQ	70\$;NO!!
126	061724	000402			ADD	#4,(SP)		;YES - MOVE SP TO USER'S ERROR RETURN
127	061726	162716	000004		BR	80\$		
128	061732	000240		70\$:	SUB	#4,(SP)		;NO - MOVE SP TO NO ERROR RETURN
129	061734	000207		80\$:	NOP			
130					RTS	PC		;RETURN TO USER

60\$:

;AUGMENT THE RETURN ADDRESS IF ANY ERROR WAS DETECTED

70\$:

80\$:


```

1      .SBTTL  STOP AND SHUTDOWN SUBROUTINES
2
3 061736  STOP:
4
5      ;DROP PRIORITY TO ALLOW CONSOLE INTERRUPT
6 061736 012746 000140      MOV    #PR3,-(SP)      ;;PUT NEW PS ON STACK
   061742 012746 061750      MOV    #64$,-(SP)     ;;PUT NEW PC ON STACK
   061746 000002              RTI                    ;;POP NEW PC AND PS
   061750
7 061750 000240      64$:  NOP
8
9      ;RAISE PRIORITY TO INHIBIT CONSOLE INTERRUPT
10 061752 012746 000300     MOV    #PR6,-(SP)     ;;PUT NEW PS ON STACK
   061756 012746 061764     MOV    #65$,-(SP)    ;;PUT NEW PC ON STACK
   061762 000002              RTI                    ;;POP NEW PC AND PS
   061764
11 061764 000207      65$:  RTS      PC      ;CONTINUE
12
13 061766 005737 000042     SHUT:  TST    @#42      ;ANY MONITOR PRESENT ?
14 061772 001015              BNE    10$           ;BR IF YES
15 061774 104401 062002     TYPE   ,65$         ;;TYPE ASCIZ STRING
   062000 000410              BR     64$           ;;GET OVER THE ASCIZ
   062022
16 062022 000137 005412     ;;65$: .ASCIZ <CRLF><07>/TEST HALTED/<CRLF>
17 062026 000137 042136     64$:  JMP    START    ;GO TO START
18              10$:  JMP    $EOP    ;RETURN CONTROL TO MONITOR

```

1

.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
    
```

```

062032
062032 010046
062034 010146
062036 010246
062040 010346
062042 010446
062044 010546
062046 016646 000022
062052 016646 000022
062056 016646 000022
062062 016646 000022
062066 000002
    
```

```

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

*RESTORE R0-R5

```

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

```

062070
062070 012666 000022
062074 012666 000022
062100 012666 000022
062104 012666 000022
062110 012605
062112 012604
062114 012603
062116 012602
062120 012601
062122 012600
062124 000002
    
```

2

.SBTTL BINARY TO ASCII AND TYPE ROUTINE

```

*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 16-BIT
*BINARY-ASCII NUMBER AND TYPE IT.
*CALL:
    
```

```

*   MOV NUMBER,-(SP)  ;;NUMBER TO BE TYPED
*   TYPBN              ;;TYPE IT
    
```

```

062126 010146
062130 016601 000006
062134 000261
    
```

```

$TYPBN: MOV R1,-(SP)      ;;SAVE R1 ON THE STACK
        MOV 6(SP),R1      ;;GET THE INPUT NUMBER
        SEC                ;;SET 'C' SO CAN KEEP TRACK OF THE NUMBER OF BITS
    
```

```

062136 112737 000060 062200 1$:   MOVB   #'0,$BIN      ;;SET CHARACTER TO AN ASCII '0'.
062144 006101                ROL    R1              ;;GET THIS BIT
062146 001406                BEQ    2$              ;;DONE?
062150 105537 062200        ADCB   $BIN      ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
062154 104401 062200        TYPE   , $BIN     ;;GO TYPE THIS BIT
062160 000241                CLC                    ;;CLEAR 'C' SO CAN KEEP TRACK OF BITS
062162 000765                BR     1$              ;;GO DO THE NEXT BIT
062164 012601                MOV    (SP)+,R1        ;;POP THE STACK INTO R1
062166 016666 000002 000004 2$:   MOV    2(SP),4(SP)    ;;ADJUST THE STACK
062174 012616                MOV    (SP)+,(SP)
062176 000002                RTI                    ;;RETURN TO USER
062200 000          000      $BIN:  .BYTE  0,0          ;;STORAGE FOR ASCII CHAR. AND TERMINATOR
3                                     .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
    
```

```

062202          $TYPDS:
062202 010046      MOV    R0,-(SP)      ;;PUSH R0 ON STACK
062204 010146      MOV    R1,-(SP)      ;;PUSH R1 ON STACK
062206 010246      MOV    R2,-(SP)      ;;PUSH R2 ON STACK
062210 010346      MOV    R3,-(SP)      ;;PUSH R3 ON STACK
062212 010546      MOV    R5,-(SP)      ;;PUSH R5 ON STACK
062214 012746 020200  MOV    #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
062220 016605 000020  MOV    20(SP),R5      ;;GET THE INPUT NUMBER
062224 100004      BPL    1$              ;;BR IF INPUT IS POS.
062226 005405      NEG    R5              ;;MAKE THE BINARY NUMBER POS.
062230 112766 000055 000001 1$:   MOVB   #'-,1(SP)      ;;MAKE THE ASCII NUMBER NEG.
062236 005000      CLR    R0              ;;ZERO THE CONSTANTS INDEX
062240 012703 062416      MOV    #$DBLK,R3      ;;SETUP THE OUTPUT POINTER
062244 112723 000040      MOVB   #' ,(R3)+      ;;SET THE FIRST CHARACTER TO A BLANK
062250 005002      CLR    R2              ;;CLEAR THE BCD NUMBER
062252 016001 062406      MOV    $DTBL(R0),R1   ;;GET THE CONSTANT
062256 160105      3$:   SUB    R1,R5          ;;FORM THIS BCD DIGIT
062260 002402      BLT    4$              ;;BR IF DONE
062262 005202      INC    R2              ;;INCREASE THE BCD DIGIT BY 1
062264 000774      BR     3$
062266 060105      4$:   ADD    R1,R5          ;;ADD BACK THE CONSTANT
062270 005702      TST    R2              ;;CHECK IF BCD DIGIT=0
062272 001002      BNE    5$              ;;FALL THROUGH IF 0
062274 105716      TSTB   (SP)          ;;STILL DOING LEADING 0'S?
062276 100407      BMI    7$              ;;BR IF YES
062300 106316      5$:   ASLB   (SP)          ;;MSD?
062302 103003      BCC    6$              ;;BR IF NO
062304 116663 000001 177777 6$:   MOVB   1(SP),-1(R3)   ;;YES--SET THE SIGN
062312 052702 000060      BIS    #'0,R2         ;;MAKE THE BCD DIGIT ASCII
062316 052702 000040      7$:   BIS    #' ,R2         ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
062322 110223      MOVB   R2,(R3)+      ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
062324 005720      TST   (R0)+          ;;JUST INCREMENTING
062326 020027 000010      CMP    R0,#10        ;;CHECK THE TABLE INDEX
    
```

```

062332 002746          BLT      2$          ;;GO DO THE NEXT DIGIT
062334 003002          BGT      8$          ;;GO TO EXIT
062336 010502          MOV      R5,R2        ;;GET THE LSD
062340 000764          BR       6$          ;;GO CHANGE TO ASCII
062342 105726          8$:    TSTB     (SP)+        ;;WAS THE LSD THE FIRST NON-ZERO?
062344 100003          BPL      9$          ;;BR IF NO
062346 116663 177777 177776 9$:    MOVB     -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
062354 105013          CLRB     (R3)        ;;SET THE TERMINATOR
062356 012605          MOV      (SP)+,R5    ;;POP STACK INTO R5
062360 012603          MOV      (SP)+,R3    ;;POP STACK INTO R3
062362 012602          MOV      (SP)+,R2    ;;POP STACK INTO R2
062364 012601          MOV      (SP)+,R1    ;;POP STACK INTO R1
062366 012600          MOV      (SP)+,R0    ;;POP STACK INTO R0
062370 104401 062416  TYPE     $DBLK        ;;NOW TYPE THE NUMBER
062374 016666 000002 000004  MOV      2(SP),4(SP)  ;;ADJUST THE STACK
062402 012616          MOV      (SP)+,(SP)
062404 000002          RTI                    ;;RETURN TO USER
062406 023420          $DTBL: 10000.
062410 001750          1000.
062412 000144          100.
062414 000012          10.
062416          $DBLK: .BLKW 4
          .SBTTL BINARY TO OCTAL (ASCII) AND TYPE
    
```

4

```

*****
*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
    
```

```

062426 017646 000000          $TYPOS: MOV      @ (SP),-(SP)    ;;PICKUP THE MODE
062432 116637 000001 062651  MOVB     1(SP),$OFILL  ;;LOAD ZERO FILL SWITCH
062440 112637 062653          MOVB     (SP)+,$OMODE+1 ;;NUMBER OF DIGITS TO TYPE
062444 062716 000002          ADD      #2,(SP)      ;;ADJUST RETURN ADDRESS
062450 000406          BR       $TYPON
062452 112737 000001 062651  $TYPOC: MOVB     #1,$OFILL  ;;SET THE ZERO FILL SWITCH
062460 112737 000006 062653  MOVB     #6,$OMODE+1  ;;SET FOR SIX(6) DIGITS
062466 112737 000005 062650  $TYPON: MOVB     #5,$OCNT  ;;SET THE ITERATION COUNT
062474 010346          MOV      R3,-(SP)    ;;SAVE R3
062476 010446          MOV      R4,-(SP)    ;;SAVE R4
    
```

```

062500 010546          MOV    R5, -(SP)      ;;SAVE R5
062502 113704 062653  MOVVB  $OMODE+1,R4    ;;GET THE NUMBER OF DIGITS TO TYPE
062506 005404          NEG    R4
062510 062704 000006  ADD    #6,R4          ;;SUBTRACT IT FOR MAX. ALLOWED
062514 110437 062652  MOVVB  R4,$OMODE      ;;SAVE IT FOR USE
062520 113704 062651  MOVVB  $OFILL,R4      ;;GET THE ZERO FILL SWITCH
062524 016605 000012  MOV    12(SP),R5     ;;PICKUP THE INPUT NUMBER
062530 005003          CLR    R3             ;;CLEAR THE OUTPUT WORD
062532 006105          1$:  ROL    R5             ;;ROTATE MSB INTO 'C'
062534 000404          BR     3$             ;;GO DO MSB
062536 006105          2$:  ROL    R5             ;;FORM THIS DIGIT
062540 006105          ROL    R5
062542 006105          ROL    R5
062544 010503          MOV    R5,R3
062546 006103          3$:  ROL    R3             ;;GET LSB OF THIS DIGIT
062550 105337 062652  DECB  $OMODE          ;;TYPE THIS DIGIT?
062554 100016          BPL   7$             ;;BR IF NO
062556 042703 177770  BIC   #177770,R3     ;;GET RID OF JUNK
062562 001002          BNE  4$             ;;TEST FOR 0
062564 005704          TST  R4             ;;SUPPRESS THIS 0?
062566 001403          BEQ  5$             ;;BR IF YES
062570 005204          4$:  INC   R4             ;;DON'T SUPPRESS ANYMORE 0'S
062572 052703 000060  BIS   #'0,R3         ;;MAKE THIS DIGIT ASCII
062576 052703 000040  5$:  BIS   #' ,R3         ;;MAKE ASCII IF NOT ALREADY
062602 110337 062646  MOVVB  R3,8$         ;;SAVE FOR TYPING
062606 104401 062646  TYPE  ,8$           ;;GO TYPE THIS DIGIT
062612 105337 062650  7$:  DECB  $OCNT        ;;COUNT BY 1
062616 003347          BGT  2$             ;;BR IF MORE TO DO
062620 002402          BLT  6$             ;;BR IF DONE
062622 005204          INC  R4             ;;INSURE LAST DIGIT ISN'T A BLANK
062624 000744          BR   2$             ;;GO DO THE LAST DIGIT
062626 012605          6$:  MOV   (SP)+,R5     ;;RESTORE R5
062630 012604          MOV   (SP)+,R4     ;;RESTORE R4
062632 012603          MOV   (SP)+,R3     ;;RESTORE R3
062634 016666 000002 000004  MOV   2(SP),4(SP)   ;;SET THE STACK FOR RETURNING
062642 012616          MOV   (SP)+,(SP)
062644 000002          RTI
062646 000          8$:  .BYTE  0             ;;RETURN
062647 000          .BYTE  0             ;;STORAGE FOR ASCII DIGIT
062650 000          $OCNT: .BYTE  0       ;;TERMINATOR FOR TYPE ROUTINE
062651 000          $OFILL: .BYTE  0      ;;OCTAL DIGIT COUNTER
062652 000000          $OMODE: .WORD  0     ;;ZERO FILL SWITCH
                                .SBTTL TYPE ROUTINE          ;;NUMBER OF DIGITS TO TYPE
    
```

5

```

*****
*ROUTINE TO TYPE ASCII MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
*
*CALL:
*1) USING A TRAP INSTRUCTION
*   TYPE ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCII STRING
*OR
*   TYPE
*   MESADR
    
```

```

;*
062654 105737 001173 $TYPE: TSTB $TPFLG ::IS THERE A TERMINAL?
062660 100002 BPL 1$ ::BR IF YES
062662 000000 HALT ::HALT HERE IF NO TERMINAL
062664 000430 BR 3$ ::LEAVE
062666 010046 1$: MOV R0,-(SP) ::SAVE R0
062670 017600 000002 MOV @2(SP),R0 ::GET ADDRESS OF ASCIZ STRING
062674 122737 000001 001242 CMPB #APTENV,$ENV ::RUNNING IN APT MODE
062702 001011 BNE 62$ ::NO,GO CHECK FOR APT CONSOLE
062704 132737 000100 001243 BITB #APTPOOL,$ENVM ::SPOOL MESSAGE TO APT
062712 001405 BEQ 62$ ::NO,GO CHECK FOR CONSOLE
062714 010037 062724 MOV R0,61$ ::SETUP MESSAGE ADDRESS FOR APT
062720 004737 065742 JSR PC,$ATY3 ::SPOOL MESSAGE TO APT
062724 000000 61$: .WORD 0 ::MESSAGE ADDRESS
062726 132737 000040 001243 62$: BITB #APTCSUP,$ENVM ::APT CONSOLE SUPPRESSED
062734 001003 BNE 60$ ::YES,SKIP TYPE OUT
062736 112046 2$: MOVB (R0)+,-(SP) ::PUSH CHARACTER TO BE TYPED ONTO STACK
062740 001005 BNE 4$ ::BR IF IT ISN'T THE TERMINATOR
062742 005726 TST (SP)+ ::IF TERMINATOR POP IT OFF THE STACK
062744 012600 60$: MOV (SP)+,R0 ::RESTORE R0
062746 062716 000002 3$: ADD #2,(SP) ::ADJUST RETURN PC
062752 000002 RTI ::RETURN
062754 122716 000011 4$: CMPB #HT,(SP) ::BRANCH IF <HT>
062760 001430 BEQ 8$
062762 122716 000200 CMPB #CRLF,(SP) ::BRANCH IF NOT <CRLF>
062766 001006 BNE 5$
062770 005726 TST (SP)+ ::POP <CR><LF> EQUIV
062772 104401 TYPE ::TYPE A CR AND LF
062774 001217 $CRLF
062776 105037 063204 CLRB $CHARCNT ::CLEAR CHARACTER COUNT
063002 000755 BR 2$ ::GET NEXT CHARACTER
063004 004737 063066 5$: JSR PC,$TYPEC ::GO TYPE THIS CHARACTER
063010 123726 001172 6$: CMPB $FILLC,(SP)+ ::IS IT TIME FOR FILLER CHARS.?
063014 001350 BNE 2$ ::IF NO GO GET NEXT CHAR.
063016 013746 001170 MOV $NULL,-(SP) ::GET # OF FILLER CHARS. NEEDED
::AND THE NULL CHAR.
063022 105366 000001 7$: DECB 1(SP) ::DOES A NULL NEED TO BE TYPED?
063026 002770 BLT 6$ ::BR IF NO--GO POP THE NULL OFF OF STACK
063030 004737 063066 JSR PC,$TYPEC ::GO TYPE A NULL
063034 105337 063204 DECB $CHARCNT ::DO NOT COUNT AS A COUNT
063040 000770 BR 7$ ::LOOP

```

;HORIZONTAL TAB PROCESSOR

```

063042 112716 000040 8$: MOVB #' ,(SP) ::REPLACE TAB WITH SPACE
063046 004737 063066 9$: JSR PC,$TYPEC ::TYPE A SPACE
063052 132737 000007 063204 BITB #7,$CHARCNT ::BRANCH IF NOT AT
063060 001372 BNE 9$ ::TAB STOP
063062 005726 TST (SP)+ ::POP SPACE OFF STACK
063064 000724 BR 2$ ::GET NEXT CHARACTER
063066 $TYPEC:
063066 105777 116066 TSTB @TKS ::CHAR IN KYBD BUFFER?
063072 100022 BPL 10$ ::BR IF NOT
063074 017746 116062 MOV @TKB,-(SP) ::GET CHAR
063100 042716 177600 BIC #177600,(SP) ::STRIP EXTRANEIOUS BITS
063104 122716 000023 CMPB #$XOFF,(SP) ::WAS CHAR XOFF

```

```

063110 001012
063112
063112 105777 116042
063116 100375
063120 117716 116036
063124 042716 177600
063130 122716 000021
063134 001366
063136
063136 005726
063140
063140 105777 116020
063144 100375
063146 116677 000002 116012
063154 122766 000015 000002
063162 001003
063164 105037 063204
063170 000406
063172 122766 000012 000002
063200 001402
063202 105227
063204 000000
063206 000207

101$: BNE 102$ ;;BR IF NOT
TSTB @STKS ;;WAIT FOR CHAR
BPL 101$
MOVB @STKB,(SP) ;;GET CHAR
BIC #177600,(SP) ;;STRIP IT
CMPB #$XON,(SP) ;;WAS IT XON?
BNE 101$ ;;BR IF NOT

102$: TST (SP)+ ;;FIX STACK

10$: TSTB @STPS ;;WAIT UNTIL PRINTER IS READY
BPL 10$
MOVB 2(SP),@STPB ;;LOAD CHAR TO BE TYPED INTO DATA REG.
CMPB #CR,2(SP) ;;IS CHARACTER A CARRIAGE RETURN?
BNE 1$ ;;BRANCH IF NO
CLRB $CHARCNT ;;YES--CLEAR CHARACTER COUNT
BR $TYPEX ;;EXIT
1$: CMPB #LF,2(SP) ;;IS CHARACTER A LINE FEED?
BEQ $TYPEX ;;BRANCH IF YES
INCB (PC)+ ;;COUNT THE CHARACTER
$CHARCNT: .WORD 0 ;;CHARACTER COUNT STORAGE
$TYPEX: RTS PC
    
```

6

.SBTTL SCOPE HANDLER ROUTINE

```

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

```

063210
063210 104410
063212 004737 061736
063216 032777 040000 115730
063224 001131

063226 000416

063230 013746 000004
063234 012737 063254 000004
063242 005737 177060
063246 012637 000004
063252 000500
063254 022626
063256 012637 000004
063262 000440
063264
063264 032777 000400 115662
063272 001421

$SCOPE: CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
JSR PC,STOP
1$: BIT #BIT14,@SWR ;;LOOP ON PRESENT TEST?
BNE $OVER ;;YES IF SW14=1
;#####START OF CODE FOR THE XOR TESTER#####
$XTSTR: BR 6$ ;;IF RUNNING ON THE "XOR" TESTER CHANGE
;;THIS INSTRUCTION TO A "NOP" (NOP=240)
MOV @#ERRVEC,-(SP) ;;SAVE THE CONTENTS OF THE ERROR VECTOR
MOV #5,@#ERRVEC ;;SET FOR TIMEOUT
TST @#177060 ;;TIME OUT ON XOR?
MOV (SP)+,@#ERRVEC ;;RESTORE THE ERROR VECTOR
BR $SVLAD ;;GO TO THE NEXT TEST
5$: CMP (SP)+,(SP)+ ;;CLEAR THE STACK AFTER A TIME OUT
MOV (SP)+,@#ERRVEC ;;RESTORE THE ERROR VECTOR
BR 7$ ;;LOOP ON THE PRESENT TEST
6$:;#####END OF CODE FOR THE XOR TESTER#####
BIT #BIT08,@SWR ;;LOOP ON SPEC. TEST?
BEQ 2$ ;;BR IF NO
    
```

063274	005046			CLR	-(SP)	::CLEAR A TEMP. LOCATION	
063276	117716	115652		MOVB	@SWR,(SP)	::PICKUP THE DESIRED TEST NUMBER	
063302	001414			BEQ	8\$::BRANCH IF BAD TEST NUMBER IN SWR	
063304	022716	000071		CMP	#71,(SP)	::CHECK THE NUMBER IN THE SWR	
063310	002411			BLT	8\$::BRANCH IF TEST NUMBER IS OUT OF RANGE	
063312	011637	001116		MOV	(SP),\$TSTNM	::UPDATE THE TEST NUMBER	
063316	005316			DEC	(SP)	::BACKUP BY ONE	
063320	006316			ASL	(SP)	::SCALE THE TEST NUMBER AS AN INDEX	
063322	062716	063526		ADD	#\$SW08TBL,(SP)	::FORM THE ADDRESS OF TEST POINTER	
063326	013637	001122		MOV	@(SP)+,\$LPADR	::SET LOOP ADDRESS TO DESIRED TEST	
063332	000466			BR	\$OVER	::GO LOOP ON THE TEST	
063334	005726		8\$:	TST	(SP)+	::CLEAN THE BAD TEST NUMBER OFF OF THE STACK	
063336	105737	001117	2\$:	TSTB	\$ERFLG	::HAS AN ERROR OCCURRED?	
063342	001421			BEQ	3\$::BR IF NO	
063344	123737	001131	001117	CMPB	\$ERMAX,\$ERFLG	::MAX. ERRORS FOR THIS TEST OCCURRED?	
063352	101015			BHI	3\$::BR IF NO	
063354	032777	001000	115572	BIT	#BIT09,@SWR	::LOOP ON ERROR?	
063362	001404			BEQ	4\$::BR IF NO	
063364	013737	001124	001122	7\$:	MOV	\$LPERR,\$LPADR	::SET LOOP ADDRESS TO LAST SCOPE
063372	000446			BR	\$OVER		
063374	105037	001117		4\$:	CLRB	\$ERFLG	::ZERO THE ERROR FLAG
063400	005037	001206		CLR	\$TIMES	::CLEAR THE NUMBER OF ITERATIONS TO MAKE	
063404	000415			BR	1\$::ESCAPE TO THE NEXT TEST	
063406	032777	004000	115540	3\$:	BIT	#BIT11,@SWR	::INHIBIT ITERATIONS?
063414	001011			BNE	1\$::BR IF YES	
063416	005737	001230		TST	\$PASS	::IF FIRST PASS OF PROGRAM	
063422	001406			BEQ	1\$::INHIBIT ITERATIONS	
063424	005237	001120		INC	\$ICNT	::INCREMENT ITERATION COUNT	
063430	023737	001206	001120	CMP	\$TIMES,\$ICNT	::CHECK THE NUMBER OF ITERATIONS MADE	
063436	002024			BGE	\$OVER	::BR IF MORE ITERATION REQUIRED	
063440	012737	000001	001120	1\$:	MOV	#1,\$ICNT	::REINITIALIZE THE ITERATION COUNTER
063446	013737	063524	001206	MOV	\$MXCNT,\$TIMES	::SET NUMBER OF ITERATIONS TO DO	
063454	105237	001116		\$SVLAD:	INCB	\$TSTNM	::COUNT TEST NUMBERS
063460	113737	001116	001226	MOVB	\$TSTNM,\$TESTN	::SET TEST NUMBER IN APT MAILBOX	
063466	011637	001122		MOV	(SP),\$LPADR	::SAVE SCOPE LOOP ADDRESS	
063472	011637	001124		MOV	(SP),\$LPERR	::SAVE ERROR LOOP ADDRESS	
063476	005037	001210		CLR	\$ESCAPE	::CLEAR THE ESCAPE FROM ERROR ADDRESS	
063502	112737	000001	001131	MOVB	#1,\$ERMAX	::ONLY ALLOW ONE(1) ERROR ON NEXT TEST	
063510	013777	001116	115440	\$OVER:	MOV	\$TSTNM,@DISPLAY	::DISPLAY TEST NUMBER
063516	013716	001122		MOV	\$LPADR,(SP)	::FUDGE RETURN ADDRESS	
063522	000002			RTI		::FIXES PS	
063524	000012			\$MXCNT:	10.	::MAX. NUMBER OF ITERATIONS	
063526				\$SW08TBL:			
063526	010120			.WORD	TST1+2	::STARTING ADDRESS OF TEST 1	
063530	010304			.WORD	TST2+2	::STARTING ADDRESS OF TEST 2	
063532	010470			.WORD	TST3+2	::STARTING ADDRESS OF TEST 3	
063534	010672			.WORD	TST4+2	::STARTING ADDRESS OF TEST 4	
063536	011542			.WORD	TST5+2	::STARTING ADDRESS OF TEST 5	
063540	012100			.WORD	TST6+2	::STARTING ADDRESS OF TEST 6	
063542	012304			.WORD	TST7+2	::STARTING ADDRESS OF TEST 7	
063544	012620			.WORD	TST10+2	::STARTING ADDRESS OF TEST 10	
063546	013000			.WORD	TST11+2	::STARTING ADDRESS OF TEST 11	
063550	014052			.WORD	TST12+2	::STARTING ADDRESS OF TEST 12	
063552	014352			.WORD	TST13+2	::STARTING ADDRESS OF TEST 13	
063554	014570			.WORD	TST14+2	::STARTING ADDRESS OF TEST 14	
063556	015076			.WORD	TST15+2	::STARTING ADDRESS OF TEST 15	
063560	015406			.WORD	TST16+2	::STARTING ADDRESS OF TEST 16	

063562	015650	.WORD	TST17+2	:: STARTING ADDRESS OF TEST 17
063564	016126	.WORD	TST20+2	:: STARTING ADDRESS OF TEST 20
063566	016376	.WORD	TST21+2	:: STARTING ADDRESS OF TEST 21
063570	016626	.WORD	TST22+2	:: STARTING ADDRESS OF TEST 22
063572	017152	.WORD	TST23+2	:: STARTING ADDRESS OF TEST 23
063574	017572	.WORD	TST24+2	:: STARTING ADDRESS OF TEST 24
063576	020136	.WORD	TST25+2	:: STARTING ADDRESS OF TEST 25
063600	020530	.WORD	TST26+2	:: STARTING ADDRESS OF TEST 26
063602	021220	.WORD	TST27+2	:: STARTING ADDRESS OF TEST 27
063604	021566	.WORD	TST30+2	:: STARTING ADDRESS OF TEST 30
063606	022110	.WORD	TST31+2	:: STARTING ADDRESS OF TEST 31
063610	022472	.WORD	TST32+2	:: STARTING ADDRESS OF TEST 32
063612	023010	.WORD	TST33+2	:: STARTING ADDRESS OF TEST 33
063614	023574	.WORD	TST34+2	:: STARTING ADDRESS OF TEST 34
063616	024320	.WORD	TST35+2	:: STARTING ADDRESS OF TEST 35
063620	025110	.WORD	TST36+2	:: STARTING ADDRESS OF TEST 36
063622	025674	.WORD	TST37+2	:: STARTING ADDRESS OF TEST 37
063624	026226	.WORD	TST40+2	:: STARTING ADDRESS OF TEST 40
063626	026666	.WORD	TST41+2	:: STARTING ADDRESS OF TEST 41
063630	027106	.WORD	TST42+2	:: STARTING ADDRESS OF TEST 42
063632	027444	.WORD	TST43+2	:: STARTING ADDRESS OF TEST 43
063634	027732	.WORD	TST44+2	:: STARTING ADDRESS OF TEST 44
063636	030220	.WORD	TST45+2	:: STARTING ADDRESS OF TEST 45
063640	030532	.WORD	TST46+2	:: STARTING ADDRESS OF TEST 46
063642	031040	.WORD	TST47+2	:: STARTING ADDRESS OF TEST 47
063644	031352	.WORD	TST50+2	:: STARTING ADDRESS OF TEST 50
063646	031662	.WORD	TST51+2	:: STARTING ADDRESS OF TEST 51
063650	032174	.WORD	TST52+2	:: STARTING ADDRESS OF TEST 52
063652	032612	.WORD	TST53+2	:: STARTING ADDRESS OF TEST 53
063654	033240	.WORD	TST54+2	:: STARTING ADDRESS OF TEST 54
063656	033656	.WORD	TST55+2	:: STARTING ADDRESS OF TEST 55
063660	034216	.WORD	TST56+2	:: STARTING ADDRESS OF TEST 56
063662	034560	.WORD	TST57+2	:: STARTING ADDRESS OF TEST 57
063664	035112	.WORD	TST60+2	:: STARTING ADDRESS OF TEST 60
063666	035540	.WORD	TST61+2	:: STARTING ADDRESS OF TEST 61
063670	036216	.WORD	TST62+2	:: STARTING ADDRESS OF TEST 62
063672	036722	.WORD	TST63+2	:: STARTING ADDRESS OF TEST 63
063674	037322	.WORD	TST64+2	:: STARTING ADDRESS OF TEST 64
063676	037732	.WORD	TST65+2	:: STARTING ADDRESS OF TEST 65
063700	040332	.WORD	TST66+2	:: STARTING ADDRESS OF TEST 66
063702	040712	.WORD	TST67+2	:: STARTING ADDRESS OF TEST 67
063704	041256	.WORD	TST70+2	:: STARTING ADDRESS OF TEST 70
063706	041614	.WORD	TST71+2	:: STARTING ADDRESS OF TEST 71

.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 ERRTP 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
```

```

063710 $ERROR:
063710 104410 7$: CKSWR ::TEST FOR CHANGE IN SOFT-SWR
063712 105237 001117 INCB $ERFLG ::SET THE ERROR FLAG
063716 001775 BEQ 7$ ::DON'T LET THE FLAG GO TO ZERO
063720 013777 001116 115230 MOV $TSTNM,@DISPLAY ::DISPLAY TEST NUMBER AND ERROR FLAG
063726 032777 002000 115220 BIT #BIT10,@SWR ::BELL ON ERROR?
063734 001402 BEQ 1$ ::NO - SKIP
063736 104401 001212 TYPE $BELL ::RING BELL
063742 005237 001126 1$: INC $ERTTL ::COUNT THE NUMBER OF ERRORS
063746 011637 001132 MOV (SP),$ERRPC ::GET ADDRESS OF ERROR INSTRUCTION
063752 162737 000002 001132 SUB #2,$ERRPC
063760 117737 115146 001130 MOV @ERRPC,$ITEMB ::STRIP AND SAVE THE ERROR ITEM CODE
063766 032777 020000 115160 BIT #BIT13,@SWR ::SKIP TYPEOUT IF SET
063774 001004 BNE 20$ ::SKIP TYPEOUTS
063776 004737 042346 JSR PC,ERRTP ::GO TO USER ERROR ROUTINE
064002 104401 001217 TYPE $CRLF
064006 20$: CMPB #APTENV,$ENV ::RUNNING IN APT MODE
064006 122737 000001 001242 BNE 2$ ::NO,SKIP APT ERROR REPORT
064014 001007 MOV @ITEMB,21$ ::SET ITEM NUMBER AS ERROR NUMBER
064016 113737 001130 064030 JSR PC,$ATY4 ::REPORT FATAL ERROR TO APT
064024 004737 065752 21$: .BYTE 0
064030 000 .BYTE 0
064031 000 22$: BR 22$ ::APT ERROR LOOP
064032 000777 2$: TST @SWR ::HALT ON ERROR
064034 005777 115114 BPL 3$ ::SKIP IF CONTINUE
064040 100002 HALT ::HALT ON ERROR!
064042 000000 CKSWR ::TEST FOR CHANGE IN SOFT-SWR
064044 104410 3$: BIT #BIT09,@SWR ::LOOP ON ERROR SWITCH SET?
064046 032777 001000 115100 BEQ 4$ ::BR IF NO
064054 001402 MOV $LPERR,(SP) ::FUDGE RETURN FOR LOOPING
064056 013716 001124 4$: TST $ESCAPE ::CHECK FOR AN ESCAPE ADDRESS
064062 005737 001210 BEQ 5$ ::BR IF NONE
064066 001402 MOV $ESCAPE,(SP) ::FUDGE RETURN ADDRESS FOR ESCAPE
064070 013716 001210 5$: CMP #ENDAD,@#42 ::ACT-11 AUTO-ACCEPT?
064074 022737 042326 000042 BNE 6$ ::BRANCH IF NO
064102 001001 HALT ::YES
064104 000000 6$: RTI ::RETURN
064106 000002 .SBTTL TTY INPUT ROUTINE
8

```

```

*****
064110 000000 .ENABL LSB
064112 000000 $TKCNT: .WORD 0 ::NUMBER OF ITEMS IN QUEUE
064114 000000 $TKQIN: .WORD 0 ::INPUT POINTER
064116 064117 $TKQOUT: .WORD 0 ::OUTPUT POINTER
$TKQSRT: .BLKB 1 ::TTY KEYBOARD QUEUE
$TKQEND=.
.EVEN

;*TK INITIALIZE ROUTINE
;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
:
;*CALL:
;* JSR PC,$TKINT

```

```

;*      RETURN
064120 005037 064110      $TKINT: CLR      $TKCNT      ;;CLEAR COUNT OF ITEMS IN QUEUE
064124 012737 064116 064112  MOV      #$TKQSR, $TKQIN  ;;MOVE THE STARTING ADDRESS OF THE
064132 013737 064112 064114  MOV      $TKQIN, $TKQOUT  ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
064140 012737 064170 000060  MOV      #$TKSRV, @#TKVEC  ;;INITIALIZE THE KEYBOARD VECTOR
064146 012737 000200 000062  MOV      #200, @#TKVEC+2  ;;'BR' LEVEL 4
064154 005777 115002      TST      @#TKB           ;;CLEAR DONE FLAG
064160 012777 000100 114772  MOV      #100, @#TKS     ;;ENABLE TTY KEYBOARD INTERRUPT
064166 000207      RTS      PC           ;;RETURN TO CALLER

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

064170 117746 114766      $TKSRV: MOVB     @#TKB, -(SP)  ;;PICKUP THE CHARACTER
064174 042716 177600      BIC      #^C177, (SP)    ;;STRIP THE JUNK
064200 021627 000003      CMP      (SP), #3       ;;IS IT A CONTROL C?
064204 001007      BNE      1$            ;;BRANCH IF NO
064206 104401 065304      TYPE     , $CNTLC      ;;TYPE A CONTROL-C (^C)
064212 004737 064120      JSR      PC, $TKINT    ;;INIT THE KEYBOARD
064216 005726      TST      (SP)+         ;;CLEAN UP STACK
064220 000137 061766      JMP      SHUT          ;;CONTROL C RESTART
064224 021627 000007 1$:  CMP      (SP), #7       ;;IS IT A CONTROL G?
064230 001004      BNE      2$            ;;BRANCH IF NO
064232 022737 000176 001154  CMP      #SWREG, SWR    ;;IS SOFT-SWR SELECTED?
064240 001500      BEQ      6$            ;;GO TO SWR CHANGE

064242      2$:
064242 022737 000001 064110  CMP      #1, $TKCNT    ;;IS THE QUEUE FULL?
064250 001004      BNE      3$            ;;BRANCH IF NO
064252 104401 001212      TYPE     , $BELL      ;;RING THE TTY BELL
064256 005726      TST      (SP)+         ;;CLEAN CHARACTER OFF OF STACK
064260 000451      BR       5$            ;;EXIT
064262 021627 000023 3$:  CMP      (SP), #23     ;;IS IT A CONTROL-S?
064266 001021      BNE      32$          ;;BRANCH IF NO
064270 005077 114664      CLR      @#TKS         ;;DISABLE TTY KEYBOARD INTERRUPTS
064274 005726      TST      (SP)+         ;;CLEAN CHAR OFF STACK
064276 105777 114656 31$:  TSTB     @#TKS         ;;WAIT FOR A CHAR
064302 100375      BPL      31$          ;;LOOP UNTIL ITS THERE
064304 117746 114652      MOVB     @#TKB, -(SP)  ;;GET THE CHARACTER
064310 042716 177600      BIC      #^C177, (SP)  ;;MAKE IT 7-BIT ASCII
064314 022627 000021      CMP      (SP)+, #21    ;;IS IT A CONTROL-Q?
064320 001366      BNE      31$          ;;BRANCH IF NO
064322 012777 000100 114630  MOV      #100, @#TKS   ;;REENABLE TTY KEYBOARD INTERRUPTS
064330 000002      RTI              ;;RETURN
064332 005237 064110 32$:  INC      $TKCNT        ;;COUNT THIS CHARACTER
064336 021627 000140      CMP      (SP), #140    ;;IS IT UPPER CASE?
064342 002405      BLT      4$            ;;BRANCH IF YES
064344 021627 000175      CMP      (SP), #175    ;;IS IT A SPECIAL CHAR?
064350 003002      BGT      4$            ;;BRANCH IF YES
064352 042716 000040      BIC      #40, (SP)     ;;MAKE IT UPPER CASE
064356 112677 177530 4$:  MOVB     (SP)+, @#TKQIN ;;AND PUT IT IN QUEUE
064362 005237 064112      INC      $TKQIN       ;;UPDATE THE POINTER

```

```
064366 023727 064112 064117      CMP      $TKQIN,#$TKQEND  ;;GO OFF THE END?
064374 001003                    BNE      5$              ;;BRANCH IF NO
064376 012737 064116 064112      MOV      #$TKQSRT,$TKQIN ;;RESET THE POINTER
064404 000002                    RTI                      ;;RETURN
5$:
```

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

```
064406 022737 000176 001154 $CKSWR: CMP      #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED
064414 001124                    BNE      15$              ;;EXIT IF NOT
064416 105777 114536            TSTB     @TKS              ;;IS A CHAR WAITING?
064422 100121                    BPL      15$              ;;IF NOT, EXIT
064424 117746 114532            MOV      @TKB,-(SP)       ;;YES
064430 042716 177600            BIC     #^C177,(SP)      ;;MAKE IT 7-BIT ASCII
064434 021627 000007            CMP      (SP),#7         ;;IS IT A CONTROL-G?
064440 001300                    BNE      2$              ;;IF NOT, PUT IT IN THE TTY QUEUE
                                ;;AND EXIT
```

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

```
064442 123727 001150 000001 6$:  CMPB     $AUTOB,#1      ;;ARE WE RUNNING IN AUTO-MODE?
064450 001674                    BEQ      2$              ;;BRANCH IF YES
064452 005726                    TST      (SP)+           ;;CLEAR CONTROL-G OFF STACK
064454 004737 064120            JSR     PC,$TKINT       ;;FLUSH THE TTY INPUT QUEUE
064460 005077 114474            CLR     @TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
064464 112737 000001 001151            MOV      #1,$INTAG     ;;SET INTERRUPT MODE INDICATOR
```

```
064472 104401 065316            TYPE     ,$CNTLG        ;;ECHO THE CONTROL-G (^G)
064476 104401 065323            TYPE     ,$MSWR        ;;TYPE CURRENT CONTENTS
064502 013746 000176            MOV      SWREG,-(SP)   ;;SAVE SWREG FOR TYPEOUT
064506 104402                    TYPOC                    ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
064510 104401 065334            TYPE     ,$MNEW        ;;PROMPT FOR NEW SWR
064514 005046                    CLR      -(SP)          ;;CLEAR COUNTER
064516 005046                    CLR      -(SP)          ;;THE NEW SWR
064520 105777 114434            7$:  TSTB     @TKS           ;;CHAR THERE?
064524 100375                    BPL      7$             ;;IF NOT TRY AGAIN
```

```
064526 117746 114430            MOV      @TKB,-(SP)    ;;PICK UP CHAR
064532 042716 177600            BIC     #^C177,(SP)   ;;MAKE IT 7-BIT ASCII
```

```
064536 021627 000003            CMP      (SP),#3       ;;IS IT A CONTROL-C?
064542 001015                    BNE      9$            ;;BRANCH IF NOT
064544 104401 065304            TYPE     ,$CNTLC       ;;YES, ECHO CONTROL-C (^C)
064550 062706 000006            ADD     #6,SP          ;;CLEAN UP STACK
064554 123727 001151 000001            CMPB     $INTAG,#1     ;;REENABLE TTY KEYBOARD INTERRUPTS?
064562 001003                    BNE      8$            ;;BRANCH IF NO
064564 012777 000100 114366            MOV      #100,@TKS     ;;ALLOW TTY KEYBOARD INTERRUPTS
064572 000137 061766            8$:  JMP      SHUT         ;;CONTROL-C RESTART
```

```
064576 021627 000025            9$:  CMP      (SP),#25     ;;IS IT A CONTROL-U?
064602 001005                    BNE      10$           ;;BRANCH IF NOT
064604 104401 065311            TYPE     ,$CNTLU       ;;YES, ECHO CONTROL-U (^U)
```

```

064610 062706 000006      20$:  ADD    #6,SP      ;;IGNORE PREVIOUS INPUT
064614 000737              BR      19$      ;;LET'S TRY IT AGAIN

064616 021627 000015      10$:  CMP    (SP),#15   ;;IS IT A <CR>?
064622 001022              BNE    16$      ;;BRANCH IF NO
064624 005766 000004      TST    4(SP)     ;;YES, IS IT THE FIRST CHAR?
064630 001403              BEQ    11$      ;;BRANCH IF YES
064632 016677 000002 114314  MOV    2(SP),@SWR ;;SAVE NEW SWR
064640 062706 000006      11$:  ADD    #6,SP      ;;CLEAR UP STACK
064644 104401 001217      14$:  TYPE   , $CRLF   ;;ECHO <CR> AND <LF>
064650 123727 001151 000001  CMPB   $INTAG,#1 ;;RE-ENABLE TTY KBD INTERRUPTS?
064656 001003              BNE    15$      ;;BRANCH IF NOT
064660 012777 000100 114272  MOV    #100,@$TKS ;;RE-ENABLE TTY KBD INTERRUPTS
064666 000002              RTI                    ;;RETURN
064670 004737 063066      15$:  JSR    PC,$TYPEC  ;;ECHO CHAR
064674 021627 000060      16$:  CMP    (SP),#60   ;;CHAR < 0?
064700 002420              BLT    18$      ;;BRANCH IF YES
064702 021627 000067      CMP    (SP),#67   ;;CHAR > 7?
064706 003015              BGT    18$      ;;BRANCH IF YES
064710 042726 000060      BIC    #60,(SP)+  ;;STRIP-OFF ASCII
064714 005766 000002      TST    2(SP)     ;;IS THIS THE FIRST CHAR
064720 001403              BEQ    17$      ;;BRANCH IF YES
064722 006316              ASL    (SP)      ;;NO, SHIFT PRESENT
064724 006316              ASL    (SP)      ;;CHAR OVER TO MAKE
064726 006316              ASL    (SP)      ;;ROOM FOR NEW ONE.
064730 005266 000002      17$:  INC    2(SP)     ;;KEEP COUNT OF CHAR
064734 056616 177776      BIS    -2(SP),(SP) ;;SET IN NEW CHAR
064740 000667              BR      7$      ;;GET THE NEXT ONE
064742 104401 001216      18$:  TYPE   , $QUES   ;;TYPE ?<CR><LF>
064746 000720              BR      20$     ;;SIMULATE CONTROL-U
.DSABL  LSB

```

```

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

```

```

064750 011646 000004 000002  $RDCHR: MOV    (SP),-(SP)  ;;PUSH DOWN THE PC AND
064752 016666 000004 000002  MOV    4(SP),2(SP) ;;THE PS
064760 005066 000004              CLR    4(SP)     ;;GET READY FOR A CHARACTER
064764 005046              CLR    -(SP)    ;;PUT NEW PS ON STACK
064766 012746 064774              MOV    #64$,-(SP) ;;PUT NEW PC ON STACK
064772 000002              RTI                    ;;POP NEW PC AND PS
064774              64$:
064774 005737 064110      1$:  TST    $TKCNT    ;;WAIT ON A CHARACTER
065000 001775              BEQ    1$      ;;
065002 005337 064110      DEC    $TKCNT    ;;DECREMENT THE COUNTER
065006 117766 177102 000004  MOVB   @ $TKQOUT,4(SP) ;;GET ONE CHARACTER
065014 005237 064114      INC    $TKQOUT   ;;UPDATE THE POINTER
065020 023727 064114 064117  CMP    $TKQOUT,#$TKQEND ;;DID IT GO OFF OF THE END?
065026 001003              BNE    2$      ;;BRANCH IF NO
065030 012737 064116 064114  MOV    #$TKQSRT,$TKQOUT ;;RESET THE POINTER

```

065036 000002

```
2$: RTI ;;RETURN
*****
*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
*CALL:
* RDLIN ;;INPUT A STRING FROM THE TTY
* RETURN HERE ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
* ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
```

065040 010346
065042 005046
065044 012703 065274
065050 022703 065304
065054 101456
065056 104411
065060 112613
065062 122713 000177
065066 001022
065070 005716
065072 001007
065074 112737 000134 065272
065102 104401 065272
065106 012716 177777
065112 005303
065114 020327 065274
065120 103434
065122 111337 065272
065126 104401 065272
065132 000746
065134 005716
065136 001406
065140 112737 000134 065272
065146 104401 065272
065152 005016
065154 122713 000025
065160 001003
065162 104401 065311
065166 000726
065170 122713 000022
065174 001011
065176 105013
065200 104401 001217
065204 104401 065274
065210 000717
065212 104401 001216
065216 000712
065220 111337 065272
065224 104401 065272
065230 122723 000015
065234 001305
065236 105063 177777
065242 104401 001220
065246 005726
065250 012603
065252 011646
065254 016666 000004 000002
065262 012766 065274 000004
065270 000002

```
$RDLIN: MOV R3,-(SP) ;;SAVE R3
CLR -(SP) ;;CLEAR THE RUBOUT KEY
1$: MOV #TTYIN,R3 ;;GET ADDRESS
2$: CMP #TTYIN+8.,R3 ;;BUFFER FULL?
BLOS 4$ ;;BR IF YES
RDCHR ;;GO READ ONE CHARACTER FROM THE TTY
MOV (SP)+,(R3) ;;GET CHARACTER
10$: CMPB #177,(R3) ;;IS IT A RUBOUT
BNE 5$ ;;BR IF NO
TST (SP) ;;IS THIS THE FIRST RUBOUT?
BNE 6$ ;;BR IF NO
MOV #'\,9$ ;;TYPE A BACK SLASH
TYPE ,9$
MOV #-1,(SP) ;;SET THE RUBOUT KEY
6$: DEC R3 ;;BACKUP BY ONE
CMP R3,#TTYIN ;;STACK EMPTY?
BLO 4$ ;;BR IF YES
MOV (R3),9$ ;;SETUP TO TYPEOUT THE DELETED CHAR.
TYPE ,9$ ;;GO TYPE
BR 2$ ;;GO READ ANOTHER CHAR.
5$: TST (SP) ;;RUBOUT KEY SET?
BEQ 7$ ;;BR IF NO
MOV #'\,9$ ;;TYPE A BACK SLASH
TYPE ,9$
CLR (SP) ;;CLEAR THE RUBOUT KEY
7$: CMPB #25,(R3) ;;IS CHARACTER A CTRL U?
BNE 8$ ;;BR IF NO
TYPE ,%CNTLU ;;TYPE A CONTROL 'U'
BR 1$ ;;GO START OVER
8$: CMPB #22,(R3) ;;IS CHARACTER A '^R'?
BNE 3$ ;;BRANCH IF NO
CLRB (R3) ;;CLEAR THE CHARACTER
TYPE ,%SCLF ;;TYPE A 'CR' & 'LF'
TYPE ,TTYIN ;;TYPE THE INPUT STRING
BR 2$ ;;GO PICKUP ANOTHER CHACTER
4$: TYPE ,%QUES ;;TYPE A '?'
BR 1$ ;;CLEAR THE BUFFER AND LOOP
3$: MOV (R3),9$ ;;ECHO THE CHARACTER
TYPE ,9$
CMPB #15,(R3)+ ;;CHECK FOR RETURN
BNE 2$ ;;LOOP IF NOT RETURN
CLRB -1(R3) ;;CLEAR RETURN (THE 15)
TYPE ,%LF ;;TYPE A LINE FEED
TST (SP)+ ;;CLEAN RUBOUT KEY FROM THE STACK
MOV (SP)+,R3 ;;RESTORE R3
MOV (SP),-(SP) ;;ADJUST THE STACK AND PUT ADDRESS OF THE
MOV 4(SP),2(SP) ;; FIRST ASCII CHARACTER ON IT
MOV #TTYIN,4(SP)
RTI ;;RETURN
```

065272 000
065273 000
065274
065304 136 103
065311 136 125
065316 136 107
065323 015 012
065334 040 040

9\$: .BYTE 0 ;: STORAGE FOR ASCII CHAR. TO TYPE
.BYTE 0 ;: TERMINATOR
\$TTYIN: .BLKB 8. ;: RESERVE 8 BYTES FOR TTY INPUT
\$CNTLC: .ASCIZ /[^]C/<15><12> ;: CONTROL 'C'
\$CNTLU: .ASCIZ /[^]U/<15><12> ;: CONTROL 'U'
\$CNTLG: .ASCIZ /[^]G/<15><12> ;: CONTROL 'G'
\$MSWR: .ASCIZ <15><12>/SWR = /
\$MNEW: .ASCIZ / NEW = /
.EVEN
.SBTTL READ AN OCTAL NUMBER FROM THE TTY

9
:*****
: *THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
: *CHANGE IT TO BINARY.
: *CALL:

: * RDOCT ;: READ AN OCTAL NUMBER
: * RETURN HERE ;: LOW ORDER BITS ARE ON TOP OF THE STACK
: * ;: HIGH ORDER BITS ARE IN \$HIOCT

065346 011646
065350 016666 000004 000002
065356 010046
065360 010146
065362 010246
065364 104412
065366 012600
065370 005001
065372 005002
065374 112046
065376 001412
065400 006301
065402 006102
065404 006301
065406 006102
065410 006301
065412 006102
065414 042716 177770
065420 062601
065422 000764
065424 005726
065426 010166 000012
065432 010237 065446
065436 012602
065440 012601
065442 012600
065444 000002
065446 000000

\$RDOCT: MOV (SP),-(SP) ;: PROVIDE SPACE FOR THE
MOV 4(SP),2(SP) ;: INPUT NUMBER
MOV R0,-(SP) ;: PUSH R0 ON STACK
MOV R1,-(SP) ;: PUSH R1 ON STACK
MOV R2,-(SP) ;: PUSH R2 ON STACK
1\$: RDLIN ;: READ AN ASCII LINE
MOV (SP)+,R0 ;: GET ADDRESS OF 1ST CHARACTER
CLR R1 ;: CLEAR DATA WORD
CLR R2
2\$: MOVB (R0)+,-(SP) ;: PICKUP THIS CHARACTER
BEQ 3\$;: IF ZERO GET OUT
ASL R1 ;: *2
ROL R2 ;: *4
ASL R1 ;: *8
ROL R2
065414 042716 177770 BIC #[^]C7,(SP) ;: STRIP THE ASCII JUNK
065420 062601 ADD (SP)+,R1 ;: ADD IN THIS DIGIT
065422 000764 BR 2\$;: LOOP
065424 005726 3\$: TST (SP)+ ;: CLEAN TERMINATOR FROM STACK
065426 010166 000012 MOV R1,12(SP) ;: SAVE THE RESULT
065432 010237 065446 MOV R2,\$HIOCT
065436 012602 MOV (SP)+,R2 ;: POP STACK INTO R2
065440 012601 MOV (SP)+,R1 ;: POP STACK INTO R1
065442 012600 MOV (SP)+,R0 ;: POP STACK INTO R0
065444 000002 RTI ;: RETURN
065446 000000 \$HIOCT: .WORD 0 ;: HIGH ORDER BITS GO HERE
.SBTTL TRAP DECODER

10
:*****
: *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.

065450 016646 000002
065454 042716 000020

\$TRAP: MOV 2(SP),-(SP) ;: ASSUME THE STATUS OF
BIC #20,(SP) ;: THE CALLER--DO NOT ALLOW

```

065460 012746 065466      MOV    #1$,-(SP)      ;; T-BIT TRAPS
065464 000002              RTI                      ;; SET THE NEW STATUS
065466 010046              1$: MOV    RO,-(SP)      ;; SAVE RO
065470 016600 000002      MOV    2(SP),RO      ;; GET TRAP ADDRESS
065474 005740              TST    -(RO)          ;; BACKUP BY 2
065476 111000              MOVB   (RO),RO       ;; GET RIGHT BYTE OF TRAP
065500 006300              ASL    RO            ;; POSITION FOR INDEXING
065502 016000 065522      MOV    $TRPAD(RO),RO ;; INDEX TO TABLE
065506 000200              RTS    RO            ;; GO TO ROUTINE

```

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

```

065510 011646              $TRAP2: MOV   (SP),-(SP)  ;; MOVE THE PC DOWN
065512 016666 000004 000002  MOV   4(SP),2(SP)  ;; MOVE THE PSW DOWN
065520 000002              RTI                      ;; RESTORE THE PSW

```

.SBTTL TRAP TABLE

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

```

:          ROUTINE
:          -----
065522 065510  $TRPAD: .WORD  $TRAP2
065524 062654  $TYPE   ;;CALL=TYPE      TRAP+1(104401)  TTY TYPEOUT ROUTINE
065526 062452  $TYPOC  ;;CALL=TYPOC      TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
065530 062426  $TYPOS  ;;CALL=TYPOS      TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
065532 062466  $TYPON  ;;CALL=TYPON      TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
065534 062202  $TYPDS  ;;CALL=TYPDS      TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
065536 062126  $TYPBN  ;;CALL=TYPBN      TRAP+6(104406)  TYPE BINARY (ASCII) NUMBER

065540 064476  $GTSWR  ;;CALL=GTSWR      TRAP+7(104407)  GET SOFT-SWR SETTING

065542 064406  $CKSWR  ;;CALL=CKSWR      TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
065544 064750  $RDCHR  ;;CALL=RDCHR      TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
065546 065040  $RDLIN  ;;CALL=RDLIN      TRAP+12(104412) TTY TYPEIN STRING ROUTINE
065550 065346  $RDOCT  ;;CALL=RDOCT      TRAP+13(104413) READ AN OCTAL NUMBER FROM TTY
065552 062032  $SAVREG ;;CALL=SAVREG      TRAP+14(104414) SAVE RO-R5 ROUTINE
065554 062070  $RESREG ;;CALL=RESREG      TRAP+15(104415) RESTORE RO-R5 ROUTINE

```

11

.SBTTL POWER DOWN AND UP ROUTINES

```

:*****
:POWER DOWN ROUTINE
065556 012737 065716 000024 $PWRDN: MOV    # $ILLUP,@#PWRVEC ;;SET FOR FAST UP
065564 012737 000340 000026  MOV    #340,@#PWRVEC+2 ;;PRIO:7
065572 010046              MOV    RO,-(SP)      ;;PUSH RO ON STACK
065574 010146              MOV    R1,-(SP)      ;;PUSH R1 ON STACK
065576 010246              MOV    R2,-(SP)      ;;PUSH R2 ON STACK
065600 010346              MOV    R3,-(SP)      ;;PUSH R3 ON STACK
065602 010446              MOV    R4,-(SP)      ;;PUSH R4 ON STACK
065604 010546              MOV    R5,-(SP)      ;;PUSH R5 ON STACK
065606 017746 113342  MOV    @SWR,-(SP)     ;;PUSH @SWR ON STACK
065612 010637 065722  MOV    SP,$SAVR6     ;;SAVE SP
065616 012737 065630 000024  MOV    # $PWRUP,@#PWRVEC ;;SET UP VECTOR
065624 000000              HALT
065626 000776              BR    -2             ;;HANG UP

```



```

*****
:POWER UP ROUTINE
065630 012737 065716 000024 $PWRUP: MOV    $ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
065636 013706 065722          MOV    $$SAVR6,SP      ;;GET SP
065642 005037 065722          CLR    $$SAVR6          ;;WAIT LOOP FOR THE TTY
065646 005237 065722          1$:  INC    $$SAVR6          ;;WAIT FOR THE INC
065652 001375          BNE    1$              ;;OF WORD
065654 012677 113274          MOV    (SP)+,@SWR      ;;POP STACK INTO @SWR
065660 012605          MOV    (SP)+,R5        ;;POP STACK INTO R5
065662 012604          MOV    (SP)+,R4        ;;POP STACK INTO R4
065664 012603          MOV    (SP)+,R3        ;;POP STACK INTO R3
065666 012602          MOV    (SP)+,R2        ;;POP STACK INTO R2
065670 012601          MOV    (SP)+,R1        ;;POP STACK INTO R1
065672 012600          MOV    (SP)+,R0        ;;POP STACK INTO R0
065674 012737 065556 000024 MOV    #PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
065702 012737 000340 000026 MOV    #340,@#PWRVEC+2 ;;PRIO:7
065710 104401          TYPE          ;;REPORT THE POWER FAILURE
065712 065724          $PWRMG: .WORD $POWER ;;POWER FAIL MESSAGE POINTER
065714 000002          RTI
065716 000000          $ILLUP: HALT          ;;THE POWER UP SEQUENCE WAS STARTED
065720 000776          BR    .-2            ;; BEFORE THE POWER DOWN WAS COMPLETE
065722 000000          $SAVR6: 0            ;;PUT THE SP HERE
065724 015 012 120 $POWER: .ASCIZ <15><12>'POWER'
          .EVEN
          .SBTTL APT COMMUNICATIONS ROUTINE

```

12

```

*****
065734 112737 000001 066200 $ATY1: MOV    #1,$FFLG ;;TO REPORT FATAL ERROR
065742 112737 000001 066176 $ATY3: MOV    #1,$MFLG ;;TO TYPE A MESSAGE
065750 000403          BR    $ATYC
065752 112737 000001 066200 $ATY4: MOV    #1,$FFLG ;;TO ONLY REPORT FATAL ERROR
065760          $ATYC:
065760 010046          MOV    R0,-(SP)      ;;PUSH R0 ON STACK
065762 010146          MOV    R1,-(SP)      ;;PUSH R1 ON STACK
065764 105737 066176          TST    $MFLG        ;;SHOULD TYPE A MESSAGE?
065770 001450          BEQ    5$           ;;IF NOT: BR
065772 122737 000001 001242 CMPB   #APTENV,$ENV  ;;OPERATING UNDER APT?
066000 001031          BNE    3$           ;;IF NOT: BR
066002 132737 000100 001243 BITB   #APTPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
066010 001425          BEQ    3$           ;;IF NOT: BR
066012 017600 000004          MOV    @4(SP),R0    ;;GET MESSAGE ADDR.
066016 062766 000002 000004 ADD    #2,4(SP)      ;;BUMP RETURN ADDR.
066024 005737 001222          1$:  TST    $MSGTYPE   ;;SEE IF DONE W/ LAST XMISSION?
066030 001375          BNE    1$           ;;IF NOT: WAIT
066032 010037 001236          MOV    R0,$MSGAD    ;;PUT ADDR IN MAILBOX
066036 105720          2$:  TSTB   (R0)+      ;;FIND END OF MESSAGE
066040 001376          BNE    2$
066042 163700 001236          SUB    $MSGAD,R0    ;;SUB START OF MESSAGE
066046 006200          ASR    R0           ;;GET MESSAGE LNTH IN WORDS
066050 010037 001240          MOV    R0,$MSGGLT   ;;PUT LENGTH IN MAILBOX
066054 012737 000004 001222 MOV    #4,$MSGTYPE   ;;TELL APT TO TAKE MSG.
066062 000413          BR    5$
066064 017637 000004 066110 3$:  MOV    @4(SP),4$    ;;PUT MSG ADDR IN JSR LINKAGE
066072 062766 000002 000004 ADD    #2,4(SP)      ;;BUMP RETURN ADDRESS
066100 013746 177776          MOV    177776,-(SP) ;;PUSH 177776 ON STACK
066104 004737 062654          JSR    PC,$TYPE     ;;CALL TYPE MACRO

```

```
066110 000000          4$: .WORD 0
066112          5$:
066112 105737 066200 10$: TSTB $FFLG      ;; SHOULD REPORT FATAL ERROR?
066116 001416          BEQ 12$      ;; IF NOT: BR
066120 005737 001242  TST $ENV      ;; RUNNING UNDER APT?
066124 001413          BEQ 12$      ;; IF NOT: BR
066126 005737 001222 11$: TST $MSGTYPE  ;; FINISHED LAST MESSAGE?
066132 001375          BNE 11$      ;; IF NOT: WAIT
066134 017637 000004 001224 MOV @4(SP), $FATAL  ;; GET ERROR #
066142 062766 000002 000004 ADD #2, 4(SP)      ;; BUMP RETURN ADDR.
066150 005237 001222  INC $MSGTYPE  ;; TELL APT TO TAKE ERROR
066154 105037 066200 12$: CLRB $FFLG  ;; CLEAR FATAL FLAG
066160 105037 066177  CLRB $LFLG  ;; CLEAR LOG FLAG
066164 105037 066176  CLRB $MFLG  ;; CLEAR MESSAGE FLAG
066170 012601          MOV (SP)+, R1  ;; POP STACK INTO R1
066172 012600          MOV (SP)+, R0  ;; POP STACK INTO R0
066174 000207          RTS PC      ;; RETURN
066176 000          $MFLG: .BYTE 0  ;; MESSG. FLAG
066177 000          $LFLG: .BYTE 0  ;; LOG FLAG
066200 000          $FFLG: .BYTE 0  ;; FATAL FLAG
                                .EVEN
                                APTSIZE = 200
                                APTENV = 001
                                APTSPool = 100
                                APTCSUP = 040

000200
000001
000100
000040
```

```
1
2
3 066202 075 000
4 066204 101 114
5 066211 040 077
6 066215 054 040
7 066220 200 124
8 066251
9 066251 200 103
10 066304 200 125
11 066337 200 102
12 066355 040 114
13 066417 126 105
14 066437 040 114
15 066473 102 122
16 066505 040 114
17 066536 200
18 066537 200 124
19 066624 200 101
20 066701 200
21 066702 040 077
22 066723 200 104
23 066737 104 122
24 066745 040 111
25 066765 040 116
26 067002 040 116
27 067021 040 117
28 067033 040 117
29
30
31
```

.SBTTL CONSOLE MESSAGES

```
EQUALS: .ASCIZ @=@
ALL: .ASCIZ @ALL@<CRLF>
QUES: .ASCIZ @ ? @
COMMA: .ASCIZ @, @
MSHELP: .ASCIZ <CRLF>@TYPE HELP TEXT (Y/N) ? @
UBUSQST:
      .ASCIZ <CRLF>@CHANGE ADDRESSES (Y/N) ? @
CNSLO0: .ASCIZ <CRLF>@USE SAME DEVICES (Y/N) ? @
CNSLO1: .ASCIZ <CRLF>@BUS ADDRESS @
CNSLO2: .ASCIZ @ LIMITS - LO= 160000, HI= 17XXXX@<CRLF>
CNSLO3: .ASCIZ @VECTOR ADDRESS @
CNSLO4: .ASCIZ @ LIMITS - LO= 0, HI= 1000@<CRLF><LF>
CNSLO5: .ASCIZ @BR LEVEL @
CNSLO6: .ASCIZ @ LIMITS - LO= 0, HI= 7@<CRLF><LF>
CNSLO7: .ASCII <CRLF>
      .ASCII <CRLF>@TYPE 'A' TO TEST ALL DRIVES, OR TYPE DRIVE NUMBER(S)@
      .ASCII <CRLF>@AND TERMINATE INPUT WITH A CARRIAGE RETURN.@
CNSLO8: .ASCII <CRLF>
CNSLO9: .ASCIZ @ ?ILLEGAL INPUT@<CRLF>
MSDRVS: .ASCIZ <CRLF>/DRIVE(S): /
MSGDRV: .ASCIZ /DRIVE/
LODEV: .ASCIZ / IS LOAD DEVICE/
NOTPRS: .ASCIZ / NOT PRESENT/
NOTAVL: .ASCIZ / NOT AVAILABLE/
OFFLIN: .ASCIZ / OFF LINE/
ONLINE: .ASCIZ / ON LINE /

.EVEN
```

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

.SBTTL FUNCTION CODE TABLE

;THE FUNCTION CODE TABLE IS USED TO DEFINE STATUS CONDITIONS FOR
;EACH FUNCTION CODE. BIT USAGE IS AS FOLLOWS:

; ATA - BIT 15 IS SET IN THE ENTRY FOR A GIVEN FUNCTION CODE
;IF ATA SHOULD BE SET WHEN THE FUNCTION CODE IS EXECUTED, OTHERWISE,
;BIT 15 IS ZERO, INDICATING THAT ATA SHOULD NOT NORMALLY BE SET.
;NOTE THAT ATA MAY BE SET WHEN A COMMAND IS EXECUTED EVEN THOUGH
;IT IS NOT EXPECTED AS A RESULT OF THE COMMAND.

; WCE - BIT 14 IS SET IN THE ENTRY FOR A GIVEN FUNCTION CODE
;IF WRITE CHECK ERRORS ARE ENABLED AS A FUNCTION OF THE COMMAND.

; OPI - BIT 13 IS SET IN THE ENTRY FOR A GIVEN FUNCTION CODE
;IF OPI ERRORS ARE ENABLED DURING THE EXECUTION OF THAT COMMAND.

; IVC - BIT 12 IS SET IN THE ENTRY FOR A GIVEN FUNCTION CODE
;IF IVC ERRORS ARE ENABLED DURING THE EXECUTION OF THAT COMMAND.

; WLE - BIT 11 IS SET IN THE ENTRY FOR A GIVEN FUNCTION CODE
;IF WRITE ERRORS ARE ENABLED DURING THE EXECUTION OF THAT COMMAND.
;THE WRITE ERRORS WHICH ARE ENABLED ARE 'WLE', 'WCF', 'DPE', 'UPE'.

; IAE - BIT 10 IS SET IN THE ENTRY FOR A GIVEN FUNCTION CODE
;IF INVALID ADDRESS ERROR IS ENABLED FOR THAT COMMAND.

; AOE - BIT 09 IS SET IN THE ENTRY FOR A GIVEN FUNCTION CODE
;IF READ AND WRITE ERRORS ARE ENABLED DURING THE EXECUTION OF THE
;COMMAND. THE ERRORS ENABLED BY THIS BIT ARE 'TRE', 'DLT', 'NEM',
;'MXF', 'LBT', AND 'AOE'.

; BIT 08 IS NOT USED.

; HCE - BIT 07 IS SET IN THE ENTRY FOR A GIVEN FUNCTION CODE
;IF HEADER ERRORS ARE ENABLED DURING THE EXECUTION OF THAT COMMAND.
;HEADER ERRORS INCLUDE 'HCRC', 'HCE', 'FER' AND 'BSE'.

; ECH - BIT 06 IS SET IN THE ENTRY FOR A GIVEN FUNCTION CODE
;IF DATA FIELD ERRORS ARE ENABLED DURING THE EXECUTION OF THAT
;COMMAND. THESE ERRORS INCLUDE 'MDPE', 'DCK', AND 'ECH'.

; BIT 05 IS NOT USED.

; BIT 04 IS NOT USED.

; BIT 03 IS NOT USED.

; BIT 02 IS NOT USED.

; BIT 01 IS NOT USED.

; ILF - BIT 00 IS SET IF THE FUNCTION CODE IS ILLEGAL.

FNCDTB: ;FUNCTION CODE TABLE

.WORD OPI ;NCP

067046

067046 020000

58	067050	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (2)
59	067052	132000	.WORD	ATA!OPI!IVC!IAE	:SEEK
60	067054	130000	.WORD	ATA!OPI!IVC	:RECALIBRATE
61	067056	020000	.WORD	OPI	:DRIVE CLEAR
62	067060	030000	.WORD	OPI!IVC	:RELEASE
63	067062	130000	.WORD	OPI!ATA!IVC	:OFFSET
64	067064	130000	.WORD	OPI!ATA!IVC	:RETURN TO CENTERLINE
65	067066	020000	.WORD	OPI	:READ IN PRESET
66	067070	020000	.WORD	OPI	:PACK ACKNOWLEDGE
67	067072	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (24)
68	067074	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (26)
69	067076	132000	.WORD	ATA!OPI!IVC!IAE	:SEARCH
70	067100	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (32)
71	067102	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (34)
72	067104	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (36)
73	067106	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (40)
74	067110	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (42)
75	067112	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (44)
76	067114	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (46)
77	067116	073300	.WORD	WCE!OPI!IVC!IAE!AOE!HCE!ECH	:WRITE CHECK DATA
78	067120	073300	.WORD	WCE!OPI!IVC!IAE!AOE!HCE!ECH	:WRITE CHECK HEADER AND DATA
79	067122	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (54)
80	067124	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (56)
81	067126	037200	.WORD	OPI!IVC!WLE!IAE!AOE!HCE	:WRITE DATA
82	067130	037000	.WORD	OPI!IVC!WLE!IAE!AOE	:WRITE HEADER AND DATA
83	067132	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (64)
84	067134	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (66)
85	067136	033300	.WORD	OPI!IVC!IAE!AOE!HCE!ECH	:READ DATA
86	067140	033300	.WORD	OPI!IVC!IAE!AOE!HCE!ECH	:READ HEADER AND DATA
87	067142	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (74)
88	067144	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (76)
89					

1
2
3 067146 001
4 067147 002
5 067150 004
6 067151 010
7 067152 020
8 067153 040
9 067154 100
10 067155 200
11

.SBTTL ATTENTION (ATA) TABLE

ATNTBL: .BYTE 1.
 .BYTE 2.
 .BYTE 4.
 .BYTE 8.
 .BYTE 16.
 .BYTE 32.
 .BYTE 64.
 .BYTE 128.

Line	Address	Pattern	Category	Count
1				
2				
3	067156			
4	067156			
5	067156	000000	MIXED:	
6	067160	000001	.WORD	0.
7	067162	000003	.WORD	1.
8	067164	000007	.WORD	3.
9	067166	000017	.WORD	7.
10	067170	000037	.WORD	15.
11	067172	000077	.WORD	31.
12	067174	000177	.WORD	63.
13	067176	000377	.WORD	127.
14	067200	000777	.WORD	255.
15	067202	001777	.WORD	511.
16	067204	003777	.WORD	1023.
17	067206	007777	.WORD	2047.
18	067210	017777	.WORD	4095.
19	067212	037777	.WORD	8191.
20	067214	077777	.WORD	16383.
21	067216	177777	.WORD	32767.
22	067220	177777	ONES:	65535.
23	067222	077777	.WORD	65535.
24	067224	037777	.WORD	32767.
25	067226	017777	.WORD	16383.
26	067230	007777	.WORD	8191.
27	067232	003777	.WORD	4095.
28	067234	001777	.WORD	2047.
29	067236	000777	.WORD	1023.
30	067240	000377	.WORD	511.
31	067242	000177	.WORD	255.
32	067244	000077	.WORD	127.
33	067246	000037	.WORD	63.
34	067250	000017	.WORD	31.
35	067252	000007	.WORD	15.
36	067254	000003	.WORD	7.
37	067256	000001	.WORD	3.
38	067260	000000	ZEROS:	1.
39	067262	000000	.WORD	0.
40	067264	000001	.WORD	0.
41	067266	000002	.WORD	1.
42	067270	000004	.WORD	2.
43	067272	000010	.WORD	4.
44	067274	000020	.WORD	8.
45	067276	000040	.WORD	16.
46	067300	000100	.WORD	32.
47	067302	000200	.WORD	64.
48	067304	000400	.WORD	128.
49	067306	001000	.WORD	256.
50	067310	002000	.WORD	512.
51	067312	004000	.WORD	1024.
52	067314	010000	.WORD	2048.
53	067316	020000	.WORD	4096.
54	067320	040000	.WORD	8192.
55	067322	100000	.WORD	16384.
56	067324	100000	.WORD	32768.
57	067326	040000	.WORD	32768.
			.WORD	16384.

Line	Address	Value	Description
58	067330	020000	.WORD 8192.
59	067332	010000	.WORD 4096.
60	067334	004000	.WORD 2048.
61	067336	002000	.WORD 1024.
62	067340	001000	.WORD 512.
63	067342	000400	.WORD 256.
64	067344	000200	.WORD 128.
65	067346	000100	.WORD 64.
66	067350	000040	.WORD 32.
67	067352	000020	.WORD 16.
68	067354	000010	.WORD 8.
69	067356	000004	.WORD 4.
70	067360	000002	.WORD 2.
71	067362	000001	.WORD 1.
72	067364	000000	.WORD 0.
73	067366	177777	.WORD 65535.
74	067370	177776	.WORD 65534.
75	067372	177774	.WORD 65532.
76	067374	177770	.WORD 65528.
77	067376	177760	.WORD 65520.
78	067400	177740	.WORD 65504.
79	067402	177700	.WORD 65472.
80	067404	177600	.WORD 65408.
81	067406	177400	.WORD 65280.
82	067410	177000	.WORD 65024.
83	067412	176000	.WORD 64512.
84	067414	174000	.WORD 63488.
85	067416	170000	.WORD 61440.
86	067420	160000	.WORD 57344.
87	067422	140000	.WORD 49152.
88	067424	100000	.WORD 32768.
89	067426	000000	.WORD 0.
90	067430	000000	.WORD 0.
91	067432	100000	.WORD 32768.
92	067434	140000	.WORD 49152.
93	067436	160000	.WORD 57344.
94	067440	170000	.WORD 61440.
95	067442	174000	.WORD 63488.
96	067444	176000	.WORD 64512.
97	067446	177000	.WORD 65024.
98	067450	177400	.WORD 65280.
99	067452	177600	.WORD 65408.
100	067454	177700	.WORD 65472.
101	067456	177740	.WORD 65504.
102	067460	177760	.WORD 65520.
103	067462	177770	.WORD 65528.
104	067464	177774	.WORD 65532.
105	067466	177776	.WORD 65534.
106	067470	177777	.WORD 65535.
107	067472	125252	.WORD 43690.
108	067474	152525	.WORD 43690. /2
109	067476	125252	.WORD 43690.
110	067500	177777	.WORD 65535.
111	067502	177776	.WORD 65534.
112	067504	177775	.WORD 65533.
113	067506	177773	.WORD 65531.
114	067510	177767	.WORD 65527.

EARLY:

Line	Code	Value	Label	Address
115	067512	177757	.WORD	65519.
116	067514	177737	.WORD	65503.
117	067516	177677	.WORD	65471.
118	067520	177577	.WORD	65407.
119	067522	177377	.WORD	65279.
120	067524	176777	.WORD	65023.
121	067526	175777	.WORD	64511.
122	067530	173777	.WORD	63487.
123	067532	167777	.WORD	61439.
124	067534	157777	.WORD	57343.
125	067536	137777	.WORD	49151.
126	067540	077777	.WORD	32767.
127	067542	077777	.WORD	32767.
128	067544	137777	.WORD	49151.
129	067546	157777	.WORD	57343.
130	067550	167777	.WORD	61439.
131	067552	173777	.WORD	63487.
132	067554	175777	.WORD	64511.
133	067556	176777	.WORD	65023.
134	067560	177377	.WORD	65279.
135	067562	177577	.WORD	65407.
136	067564	177677	.WORD	65471.
137	067566	177737	.WORD	65503.
138	067570	177757	.WORD	65519.
139	067572	177767	.WORD	65527.
140	067574	177773	.WORD	65531.
141	067576	177775	.WORD	65533.
142	067600	177776	.WORD	65534.
143	067602	177777	.WORD	65535.
144	067604			
145				

ENRGDT:

				.SBTTL	ERROR MESSAGE TABLE	
1						
2						
3	067604	074200	000000	EMT1:	.WORD	EMS1,0
4	067610	074247	074264 000000	EMT2:	.WORD	EMS2,EMS3,0
5	067616	074247	074327 000000	EMT3:	.WORD	EMS2,EMS4,0
6	067624	074372	074422 000000	EMT4:	.WORD	EMS5,EMS6,0
7	067632	074372	074534 000000	EMT5:	.WORD	EMS5,EMS10,0
8	067640	101175	076413 000000	EMT6:	.WORD	EMS167,EMS64,0
9	067646	077161	101222 000000	EMT7:	.WORD	EMS110,EMS170,0
10	067654	074467	000000	EMT10:	.WORD	EMS7,0
11	067660	074534	000000	EMT11:	.WORD	EMS10,0
12	067664	074576	074607 000000	EMT12:	.WORD	EMS11,EMS12,0
13	067672	074650	074661 074672	EMT13:	.WORD	EMS13,EMS14,EMS15,EMS16,0
14	067704	074744	076413 000000	EMT14:	.WORD	EMS17,EMS64,0
15	067712	074576	075027 000000	EMT15:	.WORD	EMS11,EMS21,0
16	067720	074576	075052 075201	EMT16:	.WORD	EMS11,EMS22,EMS27,0
17	067730	074576	075066 075212	EMT17:	.WORD	EMS11,EMS23,EMS30,0
18	067740	074576	075114 075212	EMT20:	.WORD	EMS11,EMS24,EMS30,0
19	067750	074576	075143 075201	EMT21:	.WORD	EMS11,EMS25,EMS27,0
20	067760	074576	075160 075201	EMT22:	.WORD	EMS11,EMS26,EMS27,0
21	067770	074576	075222 075212	EMT23:	.WORD	EMS11,EMS31,EMS30,0
22	070000	074576	075251 075212	EMT24:	.WORD	EMS11,EMS32,EMS30,0
23	070010	074576	075300 075212	EMT25:	.WORD	EMS11,EMS33,EMS30,0
24	070020	074576	075326 075212	EMT26:	.WORD	EMS11,EMS34,EMS30,0
25	070030	074576	075377 075212	EMT27:	.WORD	EMS11,EMS35,EMS30,0
26	070040	074576	075426 075212	EMT30:	.WORD	EMS11,EMS36,EMS30,0
27	070050	074576	075455 075212	EMT31:	.WORD	EMS11,EMS37,EMS30,0
28	070060	074576	075503 075212	EMT32:	.WORD	EMS11,EMS40,EMS30,0
29	070070	074576	075532 075212	EMT33:	.WORD	EMS11,EMS41,EMS30,0
30	070100	074576	075560 075212	EMT34:	.WORD	EMS11,EMS42,EMS30,0
31	070110	074576	075607 075212	EMT35:	.WORD	EMS11,EMS43,EMS30,0
32	070120	074576	075636 075212	EMT36:	.WORD	EMS11,EMS44,EMS30,0
33	070130	074576	075711 075212	EMT37:	.WORD	EMS11,EMS45,EMS30,0
34	070140	076477	075004 000000	EMT40:	.WORD	EMS66,EMS20,0
35	070146	076665	100376 076673	EMT41:	.WORD	EMS75,EMS141,EMS76,0
36	070156	100467	100477 076621	EMT42:	.WORD	EMS144,EMS145,EMS72,EMS76,0
37	070170	076003	076117 076673	EMT43:	.WORD	EMS47,EMS53,EMS76,0
38	070200	076724	076117 076673	EMT44:	.WORD	EMS77,EMS53,EMS76,0
39	070210	076752	076117 076673	EMT45:	.WORD	EMS100,EMS53,EMS76,0
40	070220	077000	076117 076673	EMT46:	.WORD	EMS101,EMS53,EMS76,0
41	070230	076431	076413 000000	EMT47:	.WORD	EMS65,EMS64,0
42	070236	074650	074672 076365	EMT50:	.WORD	EMS13,EMS15,EMS63,0
43	070246	074576	075754 075212	EMT51:	.WORD	EMS11,EMS46,EMS30,EMS67,0
44	070260	076003	076117 076547	EMT52:	.WORD	EMS47,EMS53,EMS67,EMS115,EMS140,EMS141,0
45	070276	076003	076117 076547	EMT53:	.WORD	EMS47,EMS53,EMS67,EMS115,EMS141,EMS164,0
46	070314	076032	076117 076547	EMT54:	.WORD	EMS50,EMS53,EMS67,0
47	070324	100424	100441 076117	EMT55:	.WORD	EMS142,EMS143,EMS53,EMS67,0
48	070336	076061	076621 076547	EMT56:	.WORD	EMS51,EMS72,EMS67,EMS115,EMS50,EMS70,0
49	070354	101175	076631 076547	EMT57:	.WORD	EMS167,EMS73,EMS67,0
50	070364	076202	076547 077361	EMT60:	.WORD	EMS56,EMS67,EMS115,EMS150,EMS152,EMS70,0
51	070402	076606	076202 076547	EMT61:	.WORD	EMS71,EMS56,EMS67,EMS115,EMS150,EMS152,EMS72,0
52	070422	100355	076547 077361	EMT62:	.WORD	EMS140,EMS67,EMS115,EMS47,EMS70,0
53	070436	000000		EMT63:	.WORD	
54	070440	100562	100625 076574	EMT64:	.WORD	EMS150,EMS152,EMS70,EMS115,EMS56,EMS73,EMS67,0
55	070460	076107	101622 102003	EMT65:	.WORD	EMS52,EMS205,EMS214,EMS206,EMS115,EMS51,EMS72,0
56	070500	101134	077055 076621	EMT66:	.WORD	EMS165,EMS103,EMS72,EMS124,0
57	070512	101134	077055 076621	EMT67:	.WORD	EMS165,EMS103,EMS72,EMS171,0

58	070524	075754	075004	101027	EMT70:	.WORD	EMS46,EMS20,EMS163,0
59	070534	076606	077000	101027	EMT71:	.WORD	EMS71,EMS101,EMS163,0
60	070544	076003	076621	101027	EMT72:	.WORD	EMS47,EMS72,EMS163,EMS115,EMS140,EMS141,0
61	070562	076003	076621	101027	EMT73:	.WORD	EMS47,EMS72,EMS163,EMS115,EMS141,EMS72,0
62	070600	076202	076117	101027	EMT74:	.WORD	EMS56,EMS53,EMS163,0
63	070610	076606	076202	101027	EMT75:	.WORD	EMS71,EMS56,EMS163,EMS115,EMS150,EMS152,EMS72,0
64	070630	076032	076117	101027	EMT76:	.WORD	EMS50,EMS53,EMS163,0
65	070640	100424	100441	076117	EMT77:	.WORD	EMS142,EMS143,EMS53,EMS163,0
66	070652	076665	100376	101027	EMT100:	.WORD	EMS75,EMS141,EMS163,EMS115,EMS47,EMS70,0
67	070670	076665	100562	101027	EMT101:	.WORD	EMS75,EMS150,EMS163,EMS115,EMS56,EMS73,0
68	070706	101175	076631	101027	EMT102:	.WORD	EMS167,EMS73,EMS163,0
69	070716	100545	100601	076646	EMT103:	.WORD	EMS147,EMS151,EMS74,EMS163,0
70	070730	076061	076646	101027	EMT104:	.WORD	EMS51,EMS74,EMS163,EMS115,EMS50,EMS70,0
71	070746	101134	076752	101027	EMT105:	.WORD	EMS165,EMS100,EMS163,0
72	070756	101134	076724	101027	EMT106:	.WORD	EMS165,EMS77,EMS163,0
73	070766	100467	100477	076117	EMT107:	.WORD	EMS144,EMS145,EMS53,EMS163,EMS115,EMS143,EMS70,0
74	071006	077161	077221	077364	EMT110:	.WORD	EMS110,EMS112,EMS116,EMS111,0
75	071020	077305	074327	000000	EMT111:	.WORD	EMS113,EMS4,0
76	071026	077305	074264	000000	EMT112:	.WORD	EMS113,EMS3,0
77	071034	077161	077361	077364	EMT113:	.WORD	EMS110,EMS115,EMS116,EMS117,EMS114,0
78	071050	077305	077436	000000	EMT114:	.WORD	EMS113,EMS120,0
79	071056	077457	100117	100143	EMT115:	.WORD	EMS121,EMS132,EMS133,0
80	071066	077522	100117	100143	EMT116:	.WORD	EMS122,EMS132,EMS133,0
81	071076	077557	100117	100143	EMT117:	.WORD	EMS123,EMS132,EMS133,0
82	071106	077622	100117	100143	EMT120:	.WORD	EMS124,EMS132,EMS133,0
83	071116	077654	100117	100143	EMT121:	.WORD	EMS125,EMS132,EMS133,0
84	071126	077717	100117	100143	EMT122:	.WORD	EMS126,EMS132,EMS133,0
85	071136	100320	100117	100143	EMT123:	.WORD	EMS137,EMS132,EMS133,0
86	071146	100021	100117	100143	EMT124:	.WORD	EMS130,EMS132,EMS133,0
87	071156	100057	100117	100143	EMT125:	.WORD	EMS131,EMS132,EMS133,0
88	071166	077457	100117	100166	EMT126:	.WORD	EMS121,EMS132,EMS134,EMS123,0
89	071200	077522	100117	100166	EMT127:	.WORD	EMS122,EMS132,EMS134,EMS123,0
90	071212	077557	100117	100166	EMT130:	.WORD	EMS123,EMS132,EMS134,EMS123,0
91	071224	077622	100117	100166	EMT131:	.WORD	EMS124,EMS132,EMS134,EMS123,0
92	071236	077654	100117	100166	EMT132:	.WORD	EMS125,EMS132,EMS134,EMS123,0
93	071250	077717	100117	100166	EMT133:	.WORD	EMS126,EMS132,EMS134,EMS123,0
94	071262	100320	100117	100166	EMT134:	.WORD	EMS137,EMS132,EMS134,EMS123,0
95	071274	100021	100117	100166	EMT135:	.WORD	EMS130,EMS132,EMS134,EMS123,0
96	071306	100057	100117	100166	EMT136:	.WORD	EMS131,EMS132,EMS134,EMS123,0
97	071320	077457	100117	100230	EMT137:	.WORD	EMS121,EMS132,EMS135,0
98	071330	077557	100117	100230	EMT140:	.WORD	EMS123,EMS132,EMS135,0
99	071340	077457	100117	100273	EMT141:	.WORD	EMS121,EMS132,EMS136,0
100	071350	100320	100117	100273	EMT142:	.WORD	EMS137,EMS132,EMS136,0
101	071360	077622	100117	100273	EMT143:	.WORD	EMS124,EMS132,EMS136,0
102	071370	077654	100117	100273	EMT144:	.WORD	EMS125,EMS132,EMS136,0
103	071400	077717	100117	100273	EMT145:	.WORD	EMS126,EMS132,EMS136,0
104	071410	100057	100117	100273	EMT146:	.WORD	EMS131,EMS132,EMS136,0
105	071420	101251	100117	100273	EMT147:	.WORD	EMS171,EMS132,EMS136,0
106	071430	100021	100117	100273	EMT150:	.WORD	EMS130,EMS132,EMS136,0
107	071440	100355	077361	100376	EMT151:	.WORD	EMS140,EMS115,EMS141,EMS70,0
108	071452	100424	077361	100441	EMT152:	.WORD	EMS142,EMS115,EMS143,EMS72,0
109	071464	100467	100477	100526	EMT153:	.WORD	EMS144,EMS145,EMS146,EMS115,EMS143,EMS72,0
110	071502	100467	100477	075004	EMT154:	.WORD	EMS144,EMS145,EMS20,EMS115,EMS143,EMS70,0
111	071520	100562	100625	100673	EMT155:	.WORD	EMS150,EMS152,EMS154,EMS153,0
112	071532	100545	100601	100673	EMT156:	.WORD	EMS147,EMS151,EMS154,EMS155,0
113	071544	100545	100601	100727	EMT157:	.WORD	EMS147,EMS151,EMS156,EMS157,0
114	071556	100777	100727	100762	EMT160:	.WORD	EMS161,EMS156,EMS160,0

115	071566	075143	075201	100727	EMT161: .WORD	EMS25,EMS27,EMS156,EMS160,0
116	071600	075160	075201	100727	EMT162: .WORD	EMS26,EMS27,EMS156,EMS160,0
117	071612	076003	101027	100355	EMT163: .WORD	EMS47,EMS163,EMS140,0
118	071622	076003	075004	000000	EMT164: .WORD	EMS47,EMS20,0
119	071630	076202	075004	000000	EMT165: .WORD	EMS56,EMS20,0
120	071636	075754	075004	000000	EMT166: .WORD	EMS46,EMS20,0
121	071644	102323	075004	000000	EMT167: .WORD	EMS224,EMS20,0
122	071652	101175	100673	101150	EMT170: .WORD	EMS167,EMS154,EMS166,0
123	071662	101175	100673	100745	EMT171: .WORD	EMS167,EMS154,EMS157,0
124	071672	101175	100673	100707	EMT172: .WORD	EMS167,EMS154,EMS155,0
125	071702	101251	100117	100143	EMT173: .WORD	EMS171,EMS132,EMS133,0
126	071712	101251	100117	100166	EMT174: .WORD	EMS171,EMS132,EMS134,EMS123,0
127	071724	077305	101424	000000	EMT175: .WORD	EMS113,EMS177,0
128	071732	101446	101463	000000	EMT176: .WORD	EMS200,EMS201,0
129	071740	101561	100376	075212	EMT177: .WORD	EMS203,EMS141,EMS30,EMS202,0
130	071752	101561	076061	075212	EMT200: .WORD	EMS203,EMS51,EMS30,EMS202,0
131	071764	101561	101574	075212	EMT201: .WORD	EMS203,EMS204,EMS30,EMS202,0
132	071776	101561	076032	075212	EMT202: .WORD	EMS203,EMS50,EMS30,EMS202,0
133	072010	101561	100441	075212	EMT203: .WORD	EMS203,EMS143,EMS30,EMS202,0
134	072022	100562	076646	101531	EMT204: .WORD	EMS150,EMS74,EMS202,EMS115,EMS152,EMS72,0
135	072040	076032	077055	076621	EMT205: .WORD	EMS50,EMS103,EMS72,0
136	072050	077064	076631	101531	EMT206: .WORD	EMS104,EMS73,EMS202,0
137	072060	076665	100376	077361	EMT207: .WORD	EMS75,EMS141,EMS115,EMS140,0
138	072072	076665	100562	000000	EMT210: .WORD	EMS75,EMS150,0
139	072100	076061	076621	077361	EMT211: .WORD	EMS51,EMS72,EMS115,EMS50,EMS70,0
140	072114	100424	076117	077361	EMT212: .WORD	EMS142,EMS53,EMS115,EMS143,EMS72,0
141	072130	076032	077055	076117	EMT213: .WORD	EMS50,EMS103,EMS53,0
142	072140	076107	101622	101150	EMT214: .WORD	EMS52,EMS205,EMS166,EMS206,EMS115,EMS51,EMS72,0
143	072160	076107	077414	076202	EMT215: .WORD	EMS52,EMS117,EMS56,EMS163,0
144	072172	076202	075212	076413	EMT216: .WORD	EMS56,EMS30,EMS64,0
145	072202	075754	075212	076413	EMT217: .WORD	EMS46,EMS30,EMS64,0
146	072212	075222	075212	076413	EMT220: .WORD	EMS31,EMS30,EMS64,0
147	072222	076003	075212	076413	EMT221: .WORD	EMS47,EMS30,EMS64,0
148	072232	076107	077414	076724	EMT222: .WORD	EMS52,EMS117,EMS77,0
149	072242	076107	077414	076365	EMT223: .WORD	EMS52,EMS117,EMS63,0
150	072252	000000			EMT224: .WORD	
151	072254	000000			EMT225: .WORD	
152	072256	000000			EMT226: .WORD	
153	072260	000000			EMT227: .WORD	
154	072262	000000			EMT230: .WORD	
155	072264	000000			EMT231: .WORD	
156	072266	000000			EMT232: .WORD	
157	072270	000000			EMT233: .WORD	
158	072272	000000			EMT234: .WORD	
159	072274	000000			EMT235: .WORD	
160	072276	000000			EMT236: .WORD	
161	072300	000000			EMT237: .WORD	
162	072302	000000			EMT240: .WORD	
163	072304	000000			EMT241: .WORD	
164	072306	000000			EMT242: .WORD	
165	072310	000000			EMT243: .WORD	
166	072312	000000			EMT244: .WORD	
167	072314	000000			EMT245: .WORD	
168	072316	101175	100117	101661	EMT246: .WORD	EMS167,EMS132,EMS207,0
169	072326	101175	100117	101706	EMT247: .WORD	EMS167,EMS132,EMS210,EMS125,0
170	072340	101175	101717	101706	EMT250: .WORD	EMS167,EMS211,EMS210,EMS207,EMS206,0
171	072354	101735	101760	000000	EMT251: .WORD	EMS212,EMS213,0

172	072362	101134	101735	000000	EMT252: .WORD	EMS165,EMS212,0
173	072370	100545	100601	100727	EMT253: .WORD	EMS147,EMS151,EMS156,EMS210,EMS26,EMS27,0
174	072406	101561	077000	075212	EMT254: .WORD	EMS203,EMS101,EMS30,0
175	072416	101561	101175	075212	EMT255: .WORD	EMS203,EMS167,EMS30,0
176	072426	101561	076752	075212	EMT256: .WORD	EMS203,EMS100,EMS30,0
177	072436	074576	075754	075212	EMT257: .WORD	EMS11,EMS46,EMS30,EMS102,0
178	072450	076107	077414	076202	EMT260: .WORD	EMS52,EMS117,EMS56,EMS102,0
179	072462	076107	101622	102125	EMT261: .WORD	EMS52,EMS205,EMS220,EMS206,EMS115,EMS51,EMS72,0
180	072502	077064	076631	101531	EMT262: .WORD	EMS104,EMS73,EMS202,0
181	072512	076032	076117	077026	EMT263: .WORD	EMS50,EMS53,EMS102,0
182	072522	076202	076117	077026	EMT264: .WORD	EMS56,EMS53,EMS102,EMS115,EMS150,EMS152,EMS70,0
183	072542	076606	076202	077026	EMT265: .WORD	EMS71,EMS56,EMS102,EMS115,EMS150,EMS152,EMS72,0
184	072562	100424	100441	076117	EMT266: .WORD	EMS142,EMS143,EMS53,EMS102,0
185	072574	076032	100526	077361	EMT267: .WORD	EMS50,EMS146,EMS115,EMS52,EMS117,EMS46,0
186	072612	076003	076117	077026	EMT270: .WORD	EMS47,EMS53,EMS102,EMS115,EMS140,0
187	072626	076003	076117	077026	EMT271: .WORD	EMS47,EMS53,EMS102,EMS115,EMS141,EMS72,0
188	072644	076665	100376	077026	EMT272: .WORD	EMS75,EMS141,EMS102,EMS115,EMS47,EMS73,0
189	072662	076061	076646	077026	EMT273: .WORD	EMS51,EMS74,EMS102,EMS115,EMS50,EMS70,0
190	072700	076365	076117	076257	EMT274: .WORD	EMS63,EMS53,EMS57,EMS115,EMS41,EMS146,0
191	072716	077364	076117	075532	EMT275: .WORD	EMS116,EMS53,EMS41,EMS57,0
192	072730	076003	076117	076257	EMT276: .WORD	EMS47,EMS53,EMS57,EMS115,EMS140,0
193	072744	076003	076117	076257	EMT277: .WORD	EMS47,EMS53,EMS57,EMS115,EMS141,EMS72,0
194	072762	076202	076117	076257	EMT300: .WORD	EMS56,EMS53,EMS57,EMS115,EMS150,EMS152,EMS70,0
195	073002	076606	076202	076117	EMT301: .WORD	EMS71,EMS56,EMS53,EMS57,EMS115,EMS150,EMS152,EMS72,0
196	073024	101134	076752	077055	EMT302: .WORD	EMS165,EMS100,EMS103,EMS57,0
197	073036	101134	077000	077055	EMT303: .WORD	EMS165,EMS101,EMS103,EMS57,0
198	073050	101134	076724	077055	EMT304: .WORD	EMS165,EMS77,EMS103,EMS57,0
199	073062	075754	075212	076413	EMT305: .WORD	EMS46,EMS30,EMS64,EMS57,0
200	073074	076032	076117	076257	EMT306: .WORD	EMS50,EMS53,EMS57,0
201	073104	076032	100526	077361	EMT307: .WORD	EMS50,EMS146,EMS115,EMS52,EMS117,EMS46,EMS57,0
202	073124	100424	100441	076117	EMT310: .WORD	EMS142,EMS143,EMS53,EMS57,0
203	073136	077064	077055	076117	EMT311: .WORD	EMS104,EMS103,EMS53,EMS57,0
204	073150	077113	077055	076117	EMT312: .WORD	EMS105,EMS103,EMS53,EMS57,0
205	073162	100467	100477	077055	EMT313: .WORD	EMS144,EMS145,EMS103,EMS57,EMS115,EMS143,EMS70,0
206	073202	075560	077055	076117	EMT314: .WORD	EMS42,EMS103,EMS53,EMS57,0
207	073214	075222	077055	076117	EMT315: .WORD	EMS31,EMS103,EMS53,EMS57,0
208	073226	076606	075222	077055	EMT316: .WORD	EMS71,EMS31,EMS103,EMS57,0
209	073240	075607	077055	076257	EMT317: .WORD	EMS43,EMS103,EMS57,0
210	073250	075711	077055	076257	EMT320: .WORD	EMS45,EMS103,EMS57,0
211	073260	075636	077055	076257	EMT321: .WORD	EMS44,EMS103,EMS57,0
212	073270	077142	075004	000000	EMT322: .WORD	EMS106,EMS20,0
213	073276	075426	077055	076257	EMT323: .WORD	EMS36,EMS103,EMS57,0
214	073306	101324	075426	077055	EMT324: .WORD	EMS173,EMS36,EMS103,EMS57,0
215	073320	101304	075426	077055	EMT325: .WORD	EMS172,EMS36,EMS103,EMS57,0
216	073332	074650	101341	074672	EMT326: .WORD	EMS13,EMS174,EMS15,EMS35,EMS53,EMS175,0
217	073350	100545	100601	076646	EMT327: .WORD	EMS147,EMS151,EMS74,EMS175,0
218	073362	076477	076117	101347	EMT330: .WORD	EMS66,EMS53,EMS175,0
219	073372	075300	077055	076117	EMT331: .WORD	EMS33,EMS103,EMS53,EMS175,0
220	073404	075503	077055	076117	EMT332: .WORD	EMS40,EMS103,EMS53,EMS57,0
221	073416	076061	076646	076257	EMT333: .WORD	EMS51,EMS74,EMS57,EMS115,EMS50,EMS70,0
222	073434	076665	100376	076257	EMT334: .WORD	EMS75,EMS141,EMS57,EMS115,EMS47,EMS73,0
223	073452	076665	100562	100625	EMT335: .WORD	EMS75,EMS150,EMS152,EMS57,EMS115,EMS56,EMS73,0
224	073472	076305	076320	076347	EMT336: .WORD	EMS60,EMS61,EMS62,0
225	073502	077364	077414	075326	EMT337: .WORD	EMS116,EMS117,EMS34,0
226	073512	075326	076117	076131	EMT340: .WORD	EMS34,EMS53,EMS54,EMS111,0
227	073524	076153	075326	000000	EMT341: .WORD	EMS55,EMS34,0
228	073532	076107	077414	076202	EMT342: .WORD	EMS52,EMS117,EMS56,EMS57,0

229	073544	076107	077414	075711	EMT343: .WORD	EMS52,EMS117,EMS45,EMS57,0
230	073556	076107	077414	075636	EMT344: .WORD	EMS52,EMS117,EMS44,EMS57,0
231	073570	076107	077414	102145	EMT345: .WORD	EMS52,EMS117,EMS221,0
232	073600	077142	075004	077361	EMT346: .WORD	EMS106,EMS20,EMS115,EMS223,EMS72,0
233	073614	076107	101622	102215	EMT347: .WORD	EMS52,EMS205,EMS222,EMS206,0
234	073626	076665	100562	077026	EMT350: .WORD	EMS75,EMS150,EMS102,EMS115,EMS56,EMS73,0
235	073644	101175	076631	077026	EMT351: .WORD	EMS167,EMS73,EMS102,0
236	073654	102021	000000		EMT352: .WORD	EMS215,0
237	073660	102072	101561	102042	EMT353: .WORD	EMS217,EMS203,EMS216,0
238	073670	102145	075004	000000	EMT354: .WORD	EMS221,EMS20,0
239						

1	073676	102375	103173	103250	EHT1: .WORD	EH1,STSH1,STSH2,STSH4,0
2	073710	103173	103250	103375	EHT2: .WORD	STSH1,STSH2,STSH4,0
3						
4	073720	102414	000000		EHT110: .WORD	EH110,0
5	073724	102423	000000		EHT111: .WORD	EH111,0
6						
7	073730	102442	000000		EHT114: .WORD	EH114,0
8	073734	102471	103173	103250	EHT223: .WORD	EH223,STSH1,STSH2,STSH4,0
9	073746	102517	103173	103250	EHT256: .WORD	EH256,STSH1,STSH2,STSH4,0
10						
11	073760	102573	103173	103250	EHT336: .WORD	EH336,STSH1,STSH2,STSH4,0
12	073772	102632	103173	103250	EHT337: .WORD	EH337,STSH1,STSH2,STSH4,0
13	074004	102767	103173	103250	EHT344: .WORD	EH344,STSH1,STSH2,STSH4,0
14						
15	074016	103125	000000		EHT353: .WORD	EH353,0
16						

1	074022	103434	103530	103546	EDT1:	.WORD	ED1,STSD1,STSD2,STSD4
2	074032	103530	103546	103600	EDT2:	.WORD	STSD1,STSD2,STSD4
3							
4	074040	103442			EDT110:	.WORD	ED110
5	074042	103446			EDT111:	.WORD	ED111
6							
7	074044	103454			EDT114:	.WORD	ED114
8	074046	103464	103530	103546	EDT223:	.WORD	ED223,STSD1,STSD2,STSD4
9							
10	074056	103474	103530	103546	EDT336:	.WORD	ED336,STSD1,STSD2,STSD4
11	074066	103506	103530	103546	EDT337:	.WORD	ED337,STSD1,STSD2,STSD4
12	074076	103506	103530	103546	EDT344:	.WORD	ED337,STSD1,STSD2,STSD4,0
13							
14	074110	103520			EDT353:	.WORD	ED353
15							

Line	Job	Unit	Job	Job	Code	Type	Message
1	074112	103613	103631	103631	EFT1:	.WORD	EF111,STSF,STSF,STSF
2	074122	103631	103631	103631	EFT2:	.WORD	STSF,STSF,STSF
3							
4	074130	103612			EFT110:	.WORD	EF110
5	074132	103613			EFT111:	.WORD	EF111
6							
7	074134	103615			EFT114:	.WORD	EF114
8	074136	103615	103631	103631	EFT223:	.WORD	EF114,STSF,STSF,STSF
9							
10	074146	103620	103631	103631	EFT336:	.WORD	EF336,STSF,STSF,STSF
11	074156	103620	103631	103631	EFT337:	.WORD	EF336,STSF,STSF,STSF
12	074166	103620	103631	103631	EFT344:	.WORD	EF336,STSF,STSF,STSF
13							
14	074176	103615			EFT353:	.WORD	EF114
15							

Line	Code	Offset	Address	Label	Description
1				.SBTTL	ERROR MESSAGE STRINGS
2					
3	074200	127	122	117 EMS1:	.ASCIZ @WRONG UNIT SELECTED (RMCS2, BITS 0-2) @
4	074247	104	105	126 EMS2:	.ASCIZ @DEVICE WENT @
5	074264	125	116	101 EMS3:	.ASCIZ @UNAVAILABLE "DVA" (RMCS1, BIT 11) @
6	074327	116	117	116 EMS4:	.ASCIZ @NONEXISTENT "NED" (RMCS2, BIT 12) @
7	074372	103	117	115 EMS5:	.ASCIZ @COMMAND NOT COMPLETED, @
8	074422	103	117	116 EMS6:	.ASCIZ @CONTROLLER NOT READY (RMCS1, BIT 7) @
9	074467	104	122	111 EMS7:	.ASCIZ @DRIVE NOT READY "DRY" (RMDS, BIT 7) @
10	074534	107	117	040 EMS10:	.ASCIZ @GO NOT RESET "GO" (RMCS1, BIT 0) @
11	074576	111	116	126 EMS11:	.ASCIZ @INVALID @
12	074607	106	125	116 EMS12:	.ASCIZ @FUNCTION CODE (RMCS1, BITS 1-5) @
13	074650	115	101	123 EMS13:	.ASCIZ @MASSBUS @
14	074661	103	117	116 EMS14:	.ASCIZ @CONTROL @
15	074672	102	125	123 EMS15:	.ASCIZ @BUS PARITY ERROR @
16	074714	042	115	103 EMS16:	.ASCIZ @"MCPE" (RMCS1, BIT 13) @
17	074744	124	122	101 EMS17:	.ASCIZ @TRANSFER ERROR (RMCS1, BIT 14) @
18	075004	123	110	117 EMS20:	.ASCIZ @SHOULD NOT BE SET @
19	075027	127	117	122 EMS21:	.ASCIZ @WORD COUNT (RMWC) @
20	075052	102	125	123 EMS22:	.ASCIZ @BUS (RMB) @
21	075066	042	114	102 EMS23:	.ASCIZ @"LBT" (RMDS, BIT 10) @
22	075114	042	101	117 EMS24:	.ASCIZ @"AOE" (RMER1, BIT 09) @
23	075143	104	111	123 EMS25:	.ASCIZ @DISK (RMDA) @
24	075160	103	131	114 EMS26:	.ASCIZ @CYLINDER (RMDC) @
25	075201	101	104	104 EMS27:	.ASCIZ @ADDRESS @
26	075212	123	124	101 EMS30:	.ASCIZ @STATUS @
27	075222	042	127	114 EMS31:	.ASCIZ @"WLE" (RMER1, BIT 11) @
28	075251	042	125	120 EMS32:	.ASCIZ @"UPE" (RMCS2, BIT 13) @
29	075300	042	127	103 EMS33:	.ASCIZ @"WCF" (RMER1, BIT 5) @
30	075326	127	122	111 EMS34:	.ASCIZ @WRITE CHECK ERROR-"WCE" (RMCS2, BIT 14) @
31	075377	042	115	104 EMS35:	.ASCIZ @"MDPE" (RMCS2, BIT 8) @
32	075426	042	104	103 EMS36:	.ASCIZ @"DCK" (RMER1, BIT 15) @
33	075455	042	105	103 EMS37:	.ASCIZ @"ECH" (RMER1, BIT 6) @
34	075503	042	104	114 EMS40:	.ASCIZ @"DLT" (RMCS2, BIT 15) @
35	075532	042	115	130 EMS41:	.ASCIZ @"MXF" (RMCS2, BIT 9) @
36	075560	042	104	124 EMS42:	.ASCIZ @"DTE" (RMER1, BIT 12) @
37	075607	042	110	103 EMS43:	.ASCIZ @"HCRC" (RMER1, BIT 8) @
38	075636	110	105	101 EMS44:	.ASCIZ @HEADER COMPARE ERROR "HCE" (RMER1, BIT 7) @
39	075711	106	117	122 EMS45:	.ASCIZ @FORMAT ERROR "FER" (RMER1, BIT 4) @
40	075754	042	111	101 EMS46:	.ASCIZ @"IAE" (RMER1, BIT 10) @
41	076003	042	117	120 EMS47:	.ASCIZ @"OPI" (RMER1, BIT 13) @
42	076032	042	123	113 EMS50:	.ASCIZ @"SKI" (RMER2, BIT 14) @
43	076061	042	120	111 EMS51:	.ASCIZ @"PIP" (RMDS, BIT 13) @
44	076107	124	110	105 EMS52:	.ASCIZ @THE RM @
45	076117	104	105	124 EMS53:	.ASCIZ @DETECTED @
46	076131	101	124	040 EMS54:	.ASCIZ @AT AN UNEXPECTED @
47	076153	111	116	103 EMS55:	.ASCIZ @INCORRECT DATA DURING @
48	076202	111	116	126 EMS56:	.ASCIZ @INVALID COMMAND ERROR "IVC" (RMER2, BIT 12) @
49	076257	104	125	122 EMS57:	.ASCIZ @DURING DATA TRANSFER @
50	076305	104	101	124 EMS60:	.ASCIZ @DATA READ @
51	076320	104	117	105 EMS61:	.ASCIZ @DOES NOT COMPARE WITH @
52	076347	104	101	124 EMS62:	.ASCIZ @DATA WRITTEN @
53	076365	042	120	101 EMS63:	.ASCIZ @"PAR" (RMER1, BIT 3) @
54	076413	111	123	040 EMS64:	.ASCIZ @IS INCORRECT @
55	076431	103	117	115 EMS65:	.ASCIZ @COMPOSITE ERROR "ERR" (RMDS, BIT 14) @
56	076477	104	101	124 EMS66:	.ASCIZ @DATA PARITY ERROR "DPE" (RMER2, BIT 3) @
57	076547	104	125	122 EMS67:	.ASCIZ @DURING SEEK COMMAND @

58	076574	111	123	040	EMS70:	.ASCIZ	@IS RESET @
59	076606	105	122	122	EMS71:	.ASCIZ	@ERRONEOUS @
60	076621	111	123	040	EMS72:	.ASCIZ	@IS SET @
61	076631	104	111	104	EMS73:	.ASCIZ	@DID NOT SET @
62	076646	104	111	104	EMS74:	.ASCIZ	@DID NOT RESET @
63	076665	114	117	123	EMS75:	.ASCIZ	@LOST @
64	076673	104	125	122	EMS76:	.ASCIZ	@DURING PACK ACK COMMAND @
65	076724	042	122	115	EMS77:	.ASCIZ	@'RMR' (RMER1, BIT 2) @
66	076752	042	111	114	EMS100:	.ASCIZ	@'ILR' (RMER1, BIT 1) @
67	077000	042	111	114	EMS101:	.ASCIZ	@'ILF' (RMER1, BIT 0) @
68	077026	104	125	122	EMS102:	.ASCIZ	@DURING SEARCH COMMAND @
69	077055	105	122	122	EMS103:	.ASCIZ	@ERROR @
70	077064	042	114	102	EMS104:	.ASCIZ	@'LBC' (RMER2, BIT 10) @
71	077113	042	114	123	EMS105:	.ASCIZ	@'LSC' (RMER2, BIT 11) @
72	077142	110	105	101	EMS106:	.ASCIZ	@HEADER ERRORS @
73	077161	102	125	123	EMS110:	.ASCIZ	@BUS TIMEOUT (04 TRAP) @
74	077210	101	104	104	EMS111:	.ASCIZ	@ADDRESS @
75	077221	127	110	105	EMS112:	.ASCIZ	@WHEN READING/WRITING RH REGISTERS @
76	077263	101	124	040		.ASCIZ	@AT THE FOLLOWING @
77	077305	124	110	105	EMS113:	.ASCIZ	@THE SELECTED DEVICE IS @
78	077335	116	117	116	EMS114:	.ASCIZ	@NONEXISTENT DEVICE @
79	077361	015	012	000	EMS115:	.ASCIZ	<CR><LF>
80	077364	124	110	105	EMS116:	.ASCIZ	@THE MASSBUS CONTROLLER @
81	077414	106	101	111	EMS117:	.ASCIZ	@FAILED TO DETECT @
82	077436	116	117	124	EMS120:	.ASCIZ	@NOT AN RM05/3/2 @
83	077457	103	117	116	EMS121:	.ASCIZ	@CONTROL STATUS REGISTER 1, RMCS1, @
84	077522	102	125	123	EMS122:	.ASCIZ	@BUS ADDRESS REGISTER, RMBA, @
85	077557	103	117	116	EMS123:	.ASCIZ	@CONTROL STATUS REGISTER 2, RMCS2, @
86	077622	105	122	122	EMS124:	.ASCIZ	@ERROR REGISTER 1, RMER1, @
87	077654	101	124	124	EMS125:	.ASCIZ	@ATTENTION SUMMARY REGISTER, RMAS, @
88	077717	115	101	111	EMS126:	.ASCIZ	@MAINTENANCE REGISTER #1, RMMR #1, @
89	077762	105	103	103	EMS127:	.ASCIZ	@ECC POSITION REGISTER, RMEC1, @
90	100021	105	103	103	EMS130:	.ASCIZ	@ECC PATTERN REGISTER, RMEC2, @
91	100057	115	101	111	EMS131:	.ASCIZ	@MAINTENANCE REGISTER 2, RMMR2, @
92	100117	116	117	124	EMS132:	.ASCIZ	@NOT INITIALIZED BY @
93	100143	125	116	111	EMS133:	.ASCIZ	@UNIBUS INITIALIZE @
94	100166	103	117	116	EMS134:	.ASCIZ	@CONTROLLER CLEAR, I.E. BIT 5 OF @
95	100230	122	110	057	EMS135:	.ASCIZ	@RH/RM ERROR CLEAR (RMCS1, BIT 14) @
96	100273	104	122	111	EMS136:	.ASCIZ	@DRIVE CLEAR COMMAND @
97	100320	104	122	111	EMS137:	.ASCIZ	@DRIVE STATUS REGISTER, RMDS @
98	100355	115	105	104	EMS140:	.ASCIZ	@MEDIUM OFF LINE @
99	100376	042	115	117	EMS141:	.ASCIZ	@'MOL' (RMDS, BIT 12) @
100	100424	104	122	111	EMS142:	.ASCIZ	@DRIVE FAULT @
101	100441	042	104	126	EMS143:	.ASCIZ	@'DVC' (RMER2, BIT 7) @
102	100467	125	116	123	EMS144:	.ASCIZ	@UNSAFE @
103	100477	042	125	116	EMS145:	.ASCIZ	@'UNS' (RMER1, BIT 14) @
104	100526	123	110	117	EMS146:	.ASCIZ	@SHOULD BE SET @
105	100545	117	106	106	EMS147:	.ASCIZ	@OFFSET MODE @
106	100562	040	126	117	EMS150:	.ASCIZ	@ VOLUME VALID @
107	100601	042	117	115	EMS151:	.ASCIZ	@'OM' (RMDS, BIT 0) @
108	100625	042	126	126	EMS152:	.ASCIZ	@'VV' (RMDS, BIT 6) @
109	100651	120	101	103	EMS153:	.ASCIZ	@PACK ACK COMMAND @
110	100673	116	117	124	EMS154:	.ASCIZ	@NOT SET BY @
111	100707	117	106	106	EMS155:	.ASCIZ	@OFFSET COMMAND @
112	100727	116	117	124	EMS156:	.ASCIZ	@NOT RESET BY @
113	100745	122	124	103	EMS157:	.ASCIZ	@RTC COMMAND @
114	100762	122	111	120	EMS160:	.ASCIZ	@RIP COMMAND @

115	100777	117	106	106	EMS161: .ASCIZ @OFFSET REGISTER (RMOF) @
116	101027				EMS162: ;<UNUSED>
117	101027	104	125	122	EMS163: .ASCIZ @DURING RECALIBRATE @
118	101053	111	123	040	EMS164: .ASCII @IS INTERMITTENT OR DRIVE DIDNT DROP ON @
119	101122	103	131	114	.ASCIZ @CYLINDER @
120	101134	125	116	105	EMS165: .ASCIZ @UNEXPECTED @
121	101150	122	105	103	EMS166: .ASCIZ @RECALIBRATE COMMAND @
122	101175	042	101	124	EMS167: .ASCIZ @'ATA' (RMDS, BIT15) @
123	101222	127	110	105	EMS170: .ASCIZ @WHEN READING REGISTER @
124	101251	105	122	122	EMS171: .ASCIZ @ERROR REGISTER #2, RMER2, @
125	101304	116	117	116	EMS172: .ASCIZ @NONRECOVERABLE @
126	101324	122	105	103	EMS173: .ASCIZ @RECOVERABLE @
127	101341	104	101	124	EMS174: .ASCIZ @DATA @
128	101347	104	125	122	EMS175: .ASCIZ @DURING WRITE COMMAND @
129	101375	042	117	120	EMS176: .ASCIZ @'OPE' (RMER2, BIT 13) @
130	101424	111	116	040	EMS177: .ASCIZ @IN WRITE PROTECT @
131	101446	103	101	116	EMS200: .ASCIZ @CAN NOT SET @
132	101463	104	111	101	EMS201: .ASCIZ @DIAGNOSTIC MODE 'DMD' (RMMR1, BIT 0) @
133	101531	104	125	122	EMS202: .ASCIZ @DURING DIAGNOSTIC MODE @
134	101561	111	116	103	EMS203: .ASCIZ @INCORRECT @
135	101574	042	127	122	EMS204: .ASCIZ @'WRL' (RMDS, BIT 11) @
136	101622	105	130	105	EMS205: .ASCIZ @EXECUTED @
137	101634	127	111	124	EMS206: .ASCIZ @WITH COMP ERROR SET @
138	101661	042	107	117	EMS207: .ASCIZ @'GO' (RMCS1, BIT 0) @
139	101706	127	122	111	EMS210: .ASCIZ @WRITING @
140	101717	127	101	123	EMS211: .ASCIZ @WAS RESET BY @
141	101735	120	122	117	EMS212: .ASCIZ @PROGRAM INTERRUPT @
142	101760	127	101	123	EMS213: .ASCIZ @WAS NOT GENERATED @
143	102003	123	105	105	EMS214: .ASCIZ @SEEK COMMAND @
144	102021	120	122	117	EMS215: .ASCIZ @PROGRAM TIMEOUT @
145	102042	104	125	122	EMS216: .ASCIZ @DURING LOOK AHEAD TEST @
146	102072	114	117	117	EMS217: .ASCIZ @LOOK AHEAD REGISTER,RMLA, @
147	102125	123	105	101	EMS220: .ASCIZ @SEARCH COMMAND @
148	102145	102	101	104	EMS221: .ASCIZ @BAD SECTOR ERROR 'BSE' (RMER2, BIT 15) @
149	102215	101	040	104	EMS222: .ASCIZ @A DATA TRANSFER COMMAND @
150	102246	110	105	101	EMS223: .ASCIZ @HEADER COMPARE INHIBIT 'HCI' (RMOF, BIT 10) @
151	102323	116	117	116	EMS224: .ASCIZ @NONEXISTENT MEMORY 'NEM' (RMCS2, BIT 11) @
152					

1	102375	105	130	120	EH1:	.ASCIZ	@EXPCTD	RECEVD@			
2	102414	102	125	123	EH110:	.ASCIZ	@BUSADRA				
3	102423	040	122	115	EH111:	.ASCIZ	@ RMCS2	RMCS1@			
4											
5	102442	122	105	103	EH114:	.ASCIZ	@RECEVD	SNGPRT	DULPRT@		
6	102471	105	130	120	EH223:	.ASCIZ	@EXPCTD	RECEVD	DATA@		
7	102517	105	130	120	EH256:	.ASCII	@EXPCTD	RECEVD	RGSTR@<CRLF>		
8	102545	123	124	101		.ASCIZ	@STATUS	STATUS	INDEX@		
9											
10	102573	107	104	101	EH336:	.ASCIZ	@GDADRS	GDDATA	BDADRS	BDDATA@	
11	102632	122	115	103	EH337:	.ASCII	@RMCS2	STATUS	FAILING	DATA@<CRLF>	
12	102671	137	137	137		.ASCII	@			@<CRLF>	
13	102730	105	130	120		.ASCIZ	@EXPCTD	RECEVD	--BIT--	ADRESS@	
14											
15	102767	122	115	105	EH344:	.ASCII	@RMER1	STATUS	HEADER	FAILING@<CRLF>	
16	103027	137	137	137		.ASCII	@		WORD	BIT@<CRLF>	
17	103065	105	130	120		.ASCIZ	@EXPCTD	RECEVD	NUMBER	POSITION@	
18	103125	105	130	120	EH353:	.ASCII	@EXPCTD	RECEVD@<CRLF>			
19	103144	040	122	115		.ASCIZ	@ RMLA	RMLA	RMOF @		
20											
21	103173	040	122	115	STSH1:	.ASCII	@ RMCS1	RMCS2	RMDS	RMER1	RMER2@
22	103240	040	040	040		.ASCIZ	@	RMAS@			
23	103250	040	122	115	STSH2:	.ASCII	@ RMWC	RMBA	RMDA	RMOF	RMDC@
24	103315	040	040	040		.ASCIZ	@	RMEC1	RMEC2@		
25	103337	040	122	115	STSH3:	.ASCIZ	@ RMDA	RMDC	RMOF	RMLA@	
26	103375	040	122	115	STSH4:	.ASCIZ	@ RMMR1	RMMR2	RMDT	RMSNA@	
27											
28					.EVEN						

1	103434	001140	001142	000000	ED1:	.WORD	\$GDDAT,\$BDDAT,0
2	103442	001276	000000		ED110:	.WORD	\$BASE,0
3	103446	001174	001176	000000	ED111:	.WORD	\$TMP0,\$TMP1,0
4							
5	103454	001362	001176	001200	ED114:	.WORD	RMDI1,\$TMP1,\$TMP2,0
6	103464	001140	001142	001174	ED223:	.WORD	\$GDDAT,\$BDDAT,\$TMP0,0
7							
8	103474	001134	001140	001136	ED336:	.WORD	\$GDADR,\$GDDAT,\$BDADR,\$BDDAT,0
9							
10	103506	001140	001142	001174	ED337:	.WORD	\$GDDAT,\$BDDAT,\$TMP0,\$TMP1,0
11	103520	001140	001142	001442	ED353:	.WORD	\$GDDAT,\$BDDAT,RMOFO,0
12							
13	103530	001334	001344	001346	STSD1:	.WORD	RMCS1I,RMCS2I,RMDSI,RMER1I,RMER2I,RMASI,0
14	103546	001336	001340	001342	STSD2:	.WORD	RMWCI,RMBAI,RMDAI,RMOFI,RMDCI,RMECI
15	103562	001402	000000			.WORD	RMEC2I,0
16	103566	001342	001370	001366	STSD3:	.WORD	RMDAI,RMDCI,RMOFI,RMLAI,0
17	103600	001360	001374	001362	STSD4:	.WORD	RMMR1I,RMMR2I,RMDTI,RMSNI,0
18							

1	103612	000			EF110:	.BYTE	0
2	103613	000	000		EF111:	.BYTE	0,0
3	103615	000	000	000	EF114:	.BYTE	0,0,0
4	103620	000	000	000	EF336:	.BYTE	0,0,0,0
5	103624	000	000	000	EF337:	.BYTE	0,0,0,0,0
6							
7	103631	000	000	000	STSF:	.BYTE	0,0,0,0,0,0,0
8							

1	103640				111	123	066	066	.ASCII	@T56	ABORT SEEK TEST@<CRLF>
2	103640				065	065	065	065	.ASCII	@T55	IVC SEEK TEST@<CRLF>
3	104644				065	065	065	065	.ASCII	@T54	SEEK INVALID CYLINDER@<CRLF>
4					065	065	065	065	.ASCII	@T53	SEEK INVALID TRACK@<CRLF>
5	103640				065	065	065	065	.ASCII	@T52	SEEK INVALID SECTOR@<CRLF>
6					065	065	065	065	.ASCII	@T51	SEEK ADJACENT REVERSE@<CRLF>
7	103640				064	064	064	064	.ASCII	@T50	SEEK ADJACENT FORWARD@<CRLF>
8	103640	200			064	064	064	064	.ASCII	@T47	SEEK MAXIMUM DIFFERENCE FORWARD@<CRLF>
9	103641	200			064	064	064	064	.ASCII	@T46	SEEK ZERO DIFFERENCE@<CRLF>
10	103642	114	111	123	064	064	064	064	.ASCII	@T45	SEEK PRIME CYLINDERS@<CRLF>
11	103660	055	055	055	064	064	064	064	.ASCII	@T44	SEEK FIRST CYLINDER@<CRLF>
12	103676	124	061	011	064	064	064	064	.ASCII	@T43	SEEK LAST CYLINDER@<CRLF>
13	103730	124	062	011	064	064	064	064	.ASCII	@T42	ILLEGAL REGISTER TEST@<CRLF>
14	103761	124	063	011	064	064	064	064	.ASCII	@T41	PARITY ERROR TEST@<CRLF>
15	104004	124	064	011	064	064	064	064	.ASCII	@T40	RMR TEST@<CRLF>
16	104036	124	065	011	064	064	064	064	.ASCII	@T37	ERROR ABORT TESTS@<CRLF>
17	104067	124	066	011	064	064	064	064	.ASCII	@T36	WRITE LOCK ERROR TEST@<CRLF>
18	104113	124	067	011	064	064	064	064	.ASCII	@T35	INVALID ADDRESS ERROR TEST@<CRLF>
19	104140	124	061	060	064	064	064	064	.ASCII	@T34	INVALID COMMAND TEST@<CRLF>
20	104201	124	061	061	064	064	064	064	.ASCII	@T33	ILLEGAL FUNCTION TEST@<CRLF>
21	104232	124	061	062	064	064	064	064	.ASCII	@T32	RMDC CLEAR OFFSET TEST@<CRLF>
22	104264	124	061	063	064	064	064	064	.ASCII	@T31	READ IN PRESET TEST@<CRLF>
23	104311	124	061	064	064	064	064	064	.ASCII	@T30	RETURN TO CENTERLINE TEST@<CRLF>
24	104344	124	061	065	064	064	064	064	.ASCII	@T27	INHIBIT INTERRUPT TEST@<CRLF>
25	104375	124	061	066	064	064	064	064	.ASCII	@T26	PROGRAM INTERRUPT TEST@<CRLF>
26	104426	124	061	067	064	064	064	064	.ASCII	@T25	ERROR/ATA TEST@<CRLF>
27	104460	124	062	060	064	064	064	064	.ASCII	@T24	WRITE ATA TEST@<CRLF>
28	104505	124	062	061	064	064	064	064	.ASCII	@T23	GO/ATA TEST@<CRLF>
29	104522	124	062	062	064	064	064	064	.ASCII	@T22	OFFSET TEST@<CRLF>
30	104542	124	062	063	064	064	064	064	.ASCII	@T21	NOP TEST@<CRLF>
31	104562	124	062	064	064	064	064	064	.ASCII	@T20	DRIVE CLEAR TEST@<CRLF>
32	104605	124	062	065	064	064	064	064	.ASCII	@T17	RECALIBRATE AT OFFSET@<CRLF>
33	104630	124	062	066	064	064	064	064	.ASCII	@T16	IAE RECALIBRATE TEST@<CRLF>
34	104663	124	062	067	064	064	064	064	.ASCII	@T15	IVC RECALIBRATE TEST@<CRLF>
35	104716	124	063	060	064	064	064	064	.ASCII	@T14	ABORT RECALIBRATE TEST@<CRLF>
36	104754	124	063	061	064	064	064	064	.ASCII	@T13	RECALIBRATE TEST@<CRLF>
37	105004	124	063	062	064	064	064	064	.ASCII	@T12	PACK ACKNOWLEDGE TEST@<CRLF>
38	105037	124	063	063	064	064	064	064	.ASCII	@T11	DIAGNOSTIC MODE TEST@<CRLF>
39	105071	124	063	064	064	064	064	064	.ASCII	@T10	PRIMARY/SECONDARY ERROR TEST@<CRLF>
40	105122	124	063	065	064	064	064	064	.ASCII	@T7	DRIVE STATUS TEST@<CRLF>
41	105161	124	063	066	064	064	064	064	.ASCII	@T6	ERROR CLEAR TEST@<CRLF>
42	105213	124	063	067	064	064	064	064	.ASCII	@T5	CONTROLLER CLEAR TEST@<CRLF>
43	105241	124	064	060	064	064	064	064	.ASCII	@T4	UNIBUS INITIALIZE TEST@<CRLF>
44	105256	124	064	061	064	064	064	064	.ASCII	@T3	DRIVE TYPE TEST@<CRLF>
45	105304	124	064	062	064	064	064	064	.ASCII	@T2	DEVICE AVAILABLE TEST@<CRLF>
46	105336	124	064	063	064	064	064	064	.ASCII	@T1	CONTROLLER ACCESS TEST@<CRLF>
47	105365	124	064	064	064	064	064	064	.ASCII	@	LIST OF TESTS@<CRLF>
48	105415	124	064	065	064	064	064	064	.ASCII	@	-----@<CRLF>
49	105446	124	064	066	064	064	064	064	.ASCII	@	<CRLF>
50	105477	124	064	067	064	064	064	064	.ASCII	@	<CRLF>
51	105543	124	065	060	064	064	064	064	.ASCII	@	<CRLF>
52	105575	124	065	061	064	064	064	064	.ASCII	@	<CRLF>
53	105627	124	065	062	064	064	064	064	.ASCII	@	<CRLF>
54	105657	124	065	063	064	064	064	064	.ASCII	@	<CRLF>
55	105706	124	065	064	064	064	064	064	.ASCII	@	<CRLF>
56	105740	124	065	065	064	064	064	064	.ASCII	@	<CRLF>
57	105762	124	065	066	064	064	064	064	.ASCII	@	<CRLF>

BUFFER:
 BUFONE: .BLKW 258.
 BUFTWO: .BLKW 258.

.=BUFFER

HELP:
 .ASCII <CRLF>
 .ASCII <CRLF>
 @LIST OF TESTS@<CRLF>
 @-----@<CRLF>
 @T1 CONTROLLER ACCESS TEST@<CRLF>
 @T2 DEVICE AVAILABLE TEST@<CRLF>
 @T3 DRIVE TYPE TEST@<CRLF>
 @T4 UNIBUS INITIALIZE TEST@<CRLF>
 @T5 CONTROLLER CLEAR TEST@<CRLF>
 @T6 ERROR CLEAR TEST@<CRLF>
 @T7 DRIVE STATUS TEST@<CRLF>
 @T10 PRIMARY/SECONDARY ERROR TEST@<CRLF>
 @T11 DIAGNOSTIC MODE TEST@<CRLF>
 @T12 PACK ACKNOWLEDGE TEST@<CRLF>
 @T13 RECALIBRATE TEST@<CRLF>
 @T14 ABORT RECALIBRATE TEST@<CRLF>
 @T15 IVC RECALIBRATE TEST@<CRLF>
 @T16 IAE RECALIBRATE TEST@<CRLF>
 @T17 RECALIBRATE AT OFFSET@<CRLF>
 @T20 DRIVE CLEAR TEST@<CRLF>
 @T21 NOP TEST@<CRLF>
 @T22 OFFSET TEST@<CRLF>
 @T23 GO/ATA TEST@<CRLF>
 @T24 WRITE ATA TEST@<CRLF>
 @T25 ERROR/ATA TEST@<CRLF>
 @T26 PROGRAM INTERRUPT TEST@<CRLF>
 @T27 INHIBIT INTERRUPT TEST@<CRLF>
 @T30 RETURN TO CENTERLINE TEST@<CRLF>
 @T31 READ IN PRESET TEST@<CRLF>
 @T32 RMDC CLEAR OFFSET TEST@<CRLF>
 @T33 ILLEGAL FUNCTION TEST@<CRLF>
 @T34 INVALID COMMAND TEST@<CRLF>
 @T35 INVALID ADDRESS ERROR TEST@<CRLF>
 @T36 WRITE LOCK ERROR TEST@<CRLF>
 @T37 ERROR ABORT TESTS@<CRLF>
 @T40 RMR TEST@<CRLF>
 @T41 PARITY ERROR TEST@<CRLF>
 @T42 ILLEGAL REGISTER TEST@<CRLF>
 @T43 SEEK LAST CYLINDER@<CRLF>
 @T44 SEEK FIRST CYLINDER@<CRLF>
 @T45 SEEK PRIME CYLINDERS@<CRLF>
 @T46 SEEK ZERO DIFFERENCE@<CRLF>
 @T47 SEEK MAXIMUM DIFFERENCE FORWARD@<CRLF>
 @T50 SEEK ADJACENT FORWARD@<CRLF>
 @T51 SEEK ADJACENT REVERSE@<CRLF>
 @T52 SEEK INVALID SECTOR@<CRLF>
 @T53 SEEK INVALID TRACK@<CRLF>
 @T54 SEEK INVALID CYLINDER@<CRLF>
 @T55 IVC SEEK TEST@<CRLF>
 @T56 ABORT SEEK TEST@<CRLF>


```
58 106006      124      065      067  .ASCII @T57  SEEK AT OFFSET@<CRLF>
59 106031      124      066      060  .ASCII @T60  LOOK AHEAD TEST@<CRLF>
60 106055      124      066      061  .ASCII @T61  SEARCH ON CYLINDER@<CRLF>
61 106104      124      066      062  .ASCII @T62  SEARCH OFF CYLINDER@<CRLF>
62 106134      124      066      063  .ASCII @T63  SEARCH INVALID SECTOR@<CRLF>
63 106166      124      066      064  .ASCII @T64  SEARCH INVALID TRACK@<CRLF>
64 106217      124      066      065  .ASCII @T65  SEARCH INVALID CYLINDER@<CRLF>
65 106253      124      066      066  .ASCII @T66  IVC SEARCH TEST@<CRLF>
66 106277      124      066      067  .ASCII @T67  ABORT SEARCH TEST@<CRLF>
67 106325      124      067      060  .ASCII @T70  SEARCH AT OFFSET@<CRLF>
68 106352      124      067      061  .ASCII @T71  HEAD ALIGNMENT SEEK@<CRLF>
69 106402      200
70 106403      117      120      105  .ASCII @OPERATIONAL SWITCH SETTINGS@<CRLF>
71 106437      055      055      055  .ASCII @-----@<CRLF>
72 106473      123      127      111  .ASCII @SWITCH                               USE@<CRLF>
73 106510      055      055      055  .ASCII @-----@<CRLF>
74 106545      040      040      061  .ASCII @ 15  HALT ON ERROR@<CRLF>
75 106571      040      040      061  .ASCII @ 14  LOOP ON TEST@<CRLF>
76 106614      040      040      061  .ASCII @ 13  INHIBIT ERROR TYPEOUTS@<CRLF>
77 106651      040      040      061  .ASCII @ 12  @<CRLF>
78 106660      040      040      061  .ASCII @ 11  INHIBIT ITERATIONS@<CRLF>
79 106711      040      040      061  .ASCII @ 10  BELL ON ERROR@<CRLF>
80 106735      040      040      040  .ASCII @ 9  LOOP ON ERROR@<CRLF>
81 106761      040      040      040  .ASCII @ 8  LOOP ON TEST IN SWR<7:0>@<CRLF>
82 107020      040      040      040  .ASCII @ 7  TN128@<CRLF>
83 107034      040      040      040  .ASCII @ 6  TN64@<CRLF>
84 107047      040      040      040  .ASCII @ 5  TN32@<CRLF>
85 107062      040      040      040  .ASCII @ 4  TN16@<CRLF>
86 107075      040      040      040  .ASCII @ 3  TN8@<CRLF>
87 107107      040      040      040  .ASCII @ 2  TN4@<CRLF>
88 107121      040      040      040  .ASCII @ 1  TN2@<CRLF>
89 107133      040      040      040  .ASCII @ 0  TN1@<CRLF>
90
91              000200      .END      200
```

ABASE = 176700	AUNIT = 000000	CLKVCT 001512	EDT223 074046	EMS105 077113
ACDW1 = 000000	AUSWR = 000000	CLR = 000040	EDT336 074056	EMS106 077142
ACDW2 = 000000	AVECT1= 120254	CLRSTS 053646	EDT337 074066	EMS11 074576
ACKSTS 054526	AVECT2= 000000	CMNSTA 007604	EDT344 074076	EMS110 077161
ACPUOP= 000000	A16 = 000400	CMPERR 051644	EDT353 074110	EMS111 077210
ADDW0 = 000000	A17 = 001000	CNSL00 066304	ED1 103434	EMS112 077221
ADDW1 = 000000	BACK = 000001	CNSL01 066337	ED110 103442	EMS113 077305
ADDW10= 000000	BADTMO 005332	CNSL02 066355	ED111 103446	EMS114 077335
ADDW11= 000000	BAI = 000010	CNSL03 066417	ED114 103454	EMS115 077361
ADDW12= 000000	BB00 = 000001	CNSL04 066437	ED223 103464	EMS116 077364
ADDW13= 000000	BB01 = 000002	CNSL05 066473	ED336 103474	EMS117 077414
ADDW14= 000000	BB02 = 000004	CNSL06 066505	ED337 103506	EMS12 074607
ADDW15= 000000	BB03 = 000010	CNSL07 066536	ED353 103520	EMS120 077436
ADDW2 = 000000	BB04 = 000020	CNSL08 066701	EECC = 000020	EMS121 077457
ADDW3 = 000000	BB05 = 000040	CNSL09 066702	EFT1 074112	EMS122 077522
ADDW4 = 000000	BB06 = 000100	CNTCLR 053530	EFT110 074130	EMS123 077557
ADDW5 = 000000	BB07 = 000200	COMMA 066215	EFT111 074132	EMS124 077622
ADDW6 = 000000	BB08 = 000400	CONT = 000100	EFT114 074134	EMS125 077654
ADDW7 = 000000	BB09 = 001000	CR = 000015	EFT2 074122	EMS126 077717
ADDW8 = 000000	BIT0 = 000001	CRLF = 000200	EFT223 074136	EMS127 077762
ADDW9 = 000000	BIT00 = 000001	CTFLG 001326	EFT336 074146	EMS13 074650
ADEVCT= 000000	BIT01 = 000002	CYLSK= 001777	EFT337 074156	EMS130 100021
ADEVM = 000000	BIT02 = 000004	DBCK = 100000	EFT344 074166	EMS131 100057
ADR = 000001	BIT03 = 000010	DBEN = 040000	EFT353 074176	EMS132 100117
AENV = 000000	BIT04 = 000020	DBL = 002000	EF110 103612	EMS133 100143
AENVM = 000000	BIT05 = 000040	DCK = 100000	EF111 103613	EMS134 100166
AFATAL= 000000	BIT06 = 000100	DDISP = 177570	EF114 103615	EMS135 100230
ALL 066204	BIT07 = 000200	DEBL = 020000	EF336 103620	EMS136 100273
AMADR1= 000000	BIT08 = 000400	DEVSEL 052056	EF337 103624	EMS137 100320
AMADR2= 000000	BIT09 = 001000	DISPLA 001156	EHT1 073676	EMS14 074661
AMADR3= 000000	BIT1 = 000002	DISPRE 000174	EHT110 073720	EMS140 100355
AMADR4= 000000	BIT10 = 002000	DLT = 100000	EHT111 073724	EMS141 100376
AMAMS1= 000000	BIT11 = 004000	DMD = 000001	EHT114 073730	EMS142 100424
AMAMS2= 000000	BIT12 = 010000	DPE = 000010	EHT2 073710	EMS143 100441
AMAMS3= 000000	BIT13 = 020000	DPEHI = 040000	EHT223 073734	EMS144 100467
AMAMS4= 000000	BIT14 = 040000	DPELO = 020000	EHT256 073746	EMS145 100477
AMSGAD= 000000	BIT15 = 100000	DPR = 000400	EHT336 073760	EMS146 100526
AMSGLG= 000000	BIT2 = 000004	DRQ = 004000	EHT337 073772	EMS147 100545
AMSGTY= 000000	BIT3 = 000010	DRVCLR= 000010	EHT344 074004	EMS15 074672
AMTYP1= 000000	BIT4 = 000020	DRVSTS 057064	EHT353 074016	EMS150 100562
AMTYP2= 000000	BIT5 = 000040	DRY = 000200	EH1 102375	EMS151 100601
AMTYP3= 000000	BIT6 = 000100	DSWR = 177570	EH110 102414	EMS152 100625
AMTYP4= 000000	BIT7 = 000200	DTE = 010000	EH111 102423	EMS153 100651
AOE = 001000	BIT8 = 000400	DTO = 010000	EH114 102442	EMS154 100673
APASS = 000000	BIT9 = 001000	DULPRT= 024024	EH223 102471	EMS155 100707
APE = 100000	BOTADR 043106	DVA = 004000	EH256 102517	EMS156 100727
APRIOR= 000000	BOTFLG 043110	DVC = 000200	EH336 102573	EMS157 100745
APTCSU= 000040	BPTVEC= 000014	EARLY 067472	EH337 102632	EMS16 074714
APTENV= 000001	BSE = 100000	EBL = 020000	EH344 102767	EMS160 100762
APTSIZ= 000200	BUFFER 103640	ECH = 000100	EH353 103125	EMS161 100777
APTSPO= 000100	BUFONE 103640	ECI = 004000	EMS1 074200	EMS162 101027
ARGS = 000002	BUFTWO 104644	ECRC = 001000	EMS10 074534	EMS163 101027
ASWREG= 000000	CC = 004000	EDT1 074022	EMS100 076752	EMS164 101053
ATA = 100000	CH = 002000	EDT110 074040	EMS101 077000	EMS165 101134
ATESTN= 000000	CHRCNT 043111	EDT111 074042	EMS102 077026	EMS166 101150
ATNSK= 000377	CKSWR = 104410	EDT114 074044	EMS103 077055	EMS167 101175
ATNTBL 067146	CLKADR 001510	EDT2 074032	EMS104 077064	EMS17 074744

EMS170	101222	EMS50	076032	EMT131	071224	EMT213	072130	EMT276	072730
EMS171	101251	EMS51	076061	EMT132	071236	EMT214	072140	EMT277	072744
EMS172	101304	EMS52	076107	EMT133	071250	EMT215	072160	EMT3	067616
EMS173	101324	EMS53	076117	EMT134	071262	EMT216	072172	EMT30	070040
EMS174	101341	EMS54	076131	EMT135	071274	EMT217	072202	EMT300	072762
EMS175	101347	EMS55	076153	EMT136	071306	EMT22	067760	EMT301	073002
EMS176	101375	EMS56	076202	EMT137	071320	EMT220	072212	EMT302	073024
EMS177	101424	EMS57	076257	EMT14	067704	EMT221	072222	EMT303	073036
EMS2	074247	EMS6	074422	EMT140	071330	EMT222	072232	EMT304	073050
EMS20	075004	EMS60	076305	EMT141	071340	EMT223	072242	EMT305	073062
EMS200	101446	EMS61	076320	EMT142	071350	EMT224	072252	EMT306	073074
EMS201	101463	EMS62	076347	EMT143	071360	EMT225	072254	EMT307	073104
EMS202	101531	EMS63	076365	EMT144	071370	EMT226	072256	EMT31	070050
EMS203	101561	EMS64	076413	EMT145	071400	EMT227	072260	EMT310	073124
EMS204	101574	EMS65	076431	EMT146	071410	EMT23	067770	EMT311	073136
EMS205	101622	EMS66	076477	EMT147	071420	EMT230	072262	EMT312	073150
EMS206	101634	EMS67	076547	EMT15	067712	EMT231	072264	EMT313	073162
EMS207	101661	EMS7	074467	EMT150	071430	EMT232	072266	EMT314	073202
EMS21	075027	EMS70	076574	EMT151	071440	EMT233	072270	EMT315	073214
EMS210	101706	EMS71	076606	EMT152	071452	EMT234	072272	EMT316	073226
EMS211	101717	EMS72	076621	EMT153	071464	EMT235	072274	EMT317	073240
EMS212	101735	EMS73	076631	EMT154	071502	EMT236	072276	EMT32	070060
EMS213	101760	EMS74	076646	EMT155	071520	EMT237	072300	EMT320	073250
EMS214	102003	EMS75	076665	EMT156	071532	EMT24	070000	EMT321	073260
EMS215	102021	EMS76	076673	EMT157	071544	EMT240	072302	EMT322	073270
EMS216	102042	EMS77	076724	EMT16	067720	EMT241	072304	EMT323	073276
EMS217	102072	EMTVEC=	000030	EMT160	071556	EMT242	072306	EMT324	073306
EMS22	075052	EMT1	067604	EMT161	071566	EMT243	072310	EMT325	073320
EMS220	102125	EMT10	067654	EMT162	071600	EMT244	072312	EMT326	073332
EMS221	102145	EMT100	070652	EMT163	071612	EMT245	072314	EMT327	073350
EMS222	102215	EMT101	070670	EMT164	071622	EMT246	072316	EMT33	070070
EMS223	102246	EMT102	070706	EMT165	071630	EMT247	072326	EMT330	073362
EMS224	102323	EMT103	070716	EMT166	071636	EMT25	070010	EMT331	073372
EMS23	075066	EMT104	070730	EMT167	071644	EMT250	072340	EMT332	073404
EMS24	075114	EMT105	070746	EMT17	067730	EMT251	072354	EMT333	073416
EMS25	075143	EMT106	070756	EMT170	071652	EMT252	072362	EMT334	073434
EMS26	075160	EMT107	070766	EMT171	071662	EMT253	072370	EMT335	073452
EMS27	075201	EMT11	067660	EMT172	071672	EMT254	072406	EMT336	073472
EMS3	074264	EMT110	071006	EMT173	071702	EMT255	072416	EMT337	073502
EMS30	075212	EMT111	071020	EMT174	071712	EMT256	072426	EMT34	070100
EMS31	075222	EMT112	071026	EMT175	071724	EMT257	072436	EMT340	073512
EMS32	075251	EMT113	071034	EMT176	071732	EMT26	070020	EMT341	073524
EMS33	075300	EMT114	071050	EMT177	071740	EMT260	072450	EMT342	073532
EMS34	075326	EMT115	071056	EMT2	067610	EMT261	072462	EMT343	073544
EMS35	075377	EMT116	071066	EMT20	067740	EMT262	072502	EMT344	073556
EMS36	075426	EMT117	071076	EMT200	071752	EMT263	072512	EMT345	073570
EMS37	075455	EMT12	067664	EMT201	071764	EMT264	072522	EMT346	073600
EMS4	074327	EMT120	071106	EMT202	071776	EMT265	072542	EMT347	073614
EMS40	075503	EMT121	071116	EMT203	072010	EMT266	072562	EMT35	070110
EMS41	075532	EMT122	071126	EMT204	072022	EMT267	072574	EMT350	073626
EMS42	075560	EMT123	071136	EMT205	072040	EMT27	070030	EMT351	073644
EMS43	075607	EMT124	071146	EMT206	072050	EMT270	072612	EMT352	073654
EMS44	075636	EMT125	071156	EMT207	072060	EMT271	072626	EMT353	073660
EMS45	075711	EMT126	071166	EMT21	067750	EMT272	072644	EMT354	073670
EMS46	075754	EMT127	071200	EMT210	072072	EMT273	072662	EMT36	070120
EMS47	076003	EMT13	067672	EMT211	072100	EMT274	072700	EMT37	070130
EMS5	074372	EMT130	071212	EMT212	072114	EMT275	072716	EMT4	067624

EMT40 070140
 EMT41 070146
 EMT42 070156
 EMT43 070170
 EMT44 070200
 EMT45 070210
 EMT46 070220
 EMT47 070230
 EMT5 067632
 EMT50 070236
 EMT51 070246
 EMT52 070260
 EMT53 070276
 EMT54 070314
 EMT55 070324
 EMT56 070336
 EMT57 070354
 EMT6 067640
 EMT60 070364
 EMT61 070402
 EMT62 070422
 EMT63 070436
 EMT64 070440
 EMT65 070460
 EMT66 070500
 EMT67 070512
 EMT7 067646
 EMT70 070524
 EMT71 070534
 EMT72 070544
 EMT73 070562
 EMT74 070600
 EMT75 070610
 EMT76 070630
 EMT77 070640
 ENRGDT 067604
 EQUALS 066202
 ERR = 040000
 ERRNMB 043104
 ERROR = 104000
 ERRYP 042346
 ERRVEC = 000004
 ERTY00 043112
 ERTY01 043120
 ERTY02 043130
 ERTY03 043137
 ERTY04 043145
 ESRC = 004000
 FER = 000020
 FIND = 000001
 FMT16 = 010000
 FNCDTB 067046
 FNCMSK = 000077
 FO = 000002
 F1 = 000004
 F2 = 000010
 F3 = 000020

F4 = 000040
 GET = 044154
 GETBUF 001334
 GETINX 001514
 GETSTS 044070
 GO = 000001
 GTSWR = 104407
 HCE = 000200
 HCI = 002000
 HCRC = 000400
 HELP = 103640
 HT = 000011
 IAE = 002000
 IDXMSK = 000077
 IE = 000100
 ILF = 000001
 ILF02 = 000002
 ILF24 = 000024
 ILF26 = 000026
 ILF30 = 000030
 ILF32 = 000032
 ILF34 = 000034
 ILF36 = 000036
 ILF40 = 000040
 ILF42 = 000042
 ILF44 = 000044
 ILF46 = 000046
 ILF54 = 000054
 ILF56 = 000056
 ILF64 = 000064
 ILF66 = 000066
 ILF74 = 000074
 ILF76 = 000076
 ILR = 000002
 ILRG50 = 000050
 ILRG52 = 000052
 ILRG54 = 000054
 ILRG56 = 000056
 ILRG60 = 000060
 ILRG62 = 000062
 ILRG64 = 000064
 ILRG66 = 000066
 ILRG70 = 000070
 ILRG72 = 000072
 ILRG74 = 000074
 ILRG76 = 000076
 IOTVEC = 000020
 IPCK0 = 000001
 IPCK1 = 000002
 IPCK2 = 000004
 IPCK3 = 000010
 IR = 000100
 IVC = 010000
 LBC = 002000
 LBT = 002000
 LF = 000012
 LODEV 066745

LS = 000004
 LSC = 004000
 LST = 000002
 LSTRK = 001332
 MCLK = 004000
 MCPE = 020000
 MDF = 000100
 MDPE = 000400
 MI = 000004
 MIXED = 067156
 MOC = 000400
 MOH = 020000
 MOL = 010000
 MRD = 002000
 MS = 000040
 MSC = 000002
 MSDRVS = 066723
 MSE = 100000
 MSER = 000200
 MSGDRV = 066737
 MSHELP = 066220
 MUR = 001000
 MWD = 000010
 MWP = 000010
 MXF = 001000
 NDTMSK = 115760
 NED = 010000
 NEM = 004000
 NOP = 000000
 NOTAVL = 067002
 NOTPRS = 066765
 NSA = 100000
 OCC = 100000
 OFD = 000200
 OFFLIN = 067021
 OFFSET = 000014
 OM = 000001
 ONES = 067216
 ONLINE = 067033
 OPE = 020000
 OPI = 020000
 OR = 000200
 PACACK = 000022
 PAKACK = 000022
 PAR = 000010
 PAT = 000020
 PDA = 000400
 PGE = 002000
 PGM = 001000
 PHA = 000200
 PIP = 020000
 PIRQ = 177772
 PIRQVE = 000240
 PLFS = 002000
 PRIERR = 045152
 PRO = 000000
 PR1 = 000040

PR2 = 000100
 PR3 = 000140
 PR4 = 000200
 PR5 = 000240
 PR6 = 000300
 PR7 = 000340
 PS = 177776
 PSEL = 002000
 PSW = 177776
 PUT = 044424
 PUTBUF = 001410
 PUTINX = 001543
 PWRVEC = 000024
 QUES = 066211
 RCLSTS = 055322
 RD = 000070
 RDCHR = 104411
 RDLIN = 104412
 RDOCT = 104413
 RDY = 000200
 READY = 007756
 RECAL = 000006
 RESREG = 104415
 RESVEC = 000010
 REX = 010000
 RG = 040000
 RGDTPT = 067156
 RH = 000072
 RIP = 000020
 RELEASE = 000012
 RMAS = 000016
 RMASI = 001352
 RMASO = 001426
 RMBA = 000004
 RMBAE = 000050
 RMBAE1 = 001404
 RMBAE0 = 001460
 RMBAI = 001340
 RMBAO = 001414
 RMCS1 = 000000
 RMCS11 = 001334
 RMCS10 = 001410
 RMCS2 = 000010
 RMCS21 = 001344
 RMCS20 = 001420
 RMCS3 = 000052
 RMCS31 = 001406
 RMCS30 = 001462
 RMDA = 000006
 RMDAI = 001342
 RMDAO = 001416
 RMDB = 000022
 RMDB1 = 001356
 RMDB0 = 001432
 RMDC = 000034
 RMDC1 = 001370
 RMDCO = 001444

RMDS = 000012
 RMDS1 = 001346
 RMDSO = 001422
 RMDT = 000026
 RMDT1 = 001362
 RMDT0 = 001436
 RMEC1 = 000044
 RMEC11 = 001400
 RMEC10 = 001454
 RMEC2 = 000046
 RMEC21 = 001402
 RMEC20 = 001456
 RMER1 = 000014
 RMER11 = 001350
 RMER10 = 001424
 RMER2 = 000042
 RMER21 = 001376
 RMER20 = 001452
 RMHR = 000036
 RMHRI = 001372
 RMHRO = 001446
 RMLA = 000020
 RMLAI = 001354
 RMLAO = 001430
 RMMR1 = 000024
 RMMR11 = 001360
 RMMR10 = 001434
 RMMR2 = 000040
 RMMR21 = 001374
 RMMR20 = 001450
 RMOF = 000032
 RMOF1 = 001366
 RMOFO = 001442
 RMR = 000004
 RMSN = 000030
 RMSNI = 001364
 RMSNO = 001440
 RMWC = 000002
 RMWC1 = 001336
 RMWCO = 001412
 RQA = 100000
 RQB = 040000
 RTC = 000016
 R6 = %000006
 R7 = %000007
 SADMSK = 000377
 SAVREG = 104414
 SA1 = 000001
 SA16 = 000020
 SA2 = 000002
 SA4 = 000004
 SAB = 000010
 SC = 100000
 SCHSTS = 057666
 SCOPE = 000004
 SCTMSK = 003700
 SCO = 000100

SC1 = 000200
 SC2 = 000400
 SC3 = 001000
 SC4 = 002000
 SEARCH= 000030
 SECERR 046004
 SEEK = 000004
 SEKSTS 052270
 SHUT 061766
 SIZCLK 044644
 SKI = 040000
 SNGPRT= 020024
 STACK = 001100
 STANDA 006700
 START 005412
 STCDRV 061232
 STKLMT= 177774
 STOP 061736
 STSD1 103530
 STSD2 103546
 STSD3 103566
 STSD4 103600
 STSF 103631
 STSH1 103173
 STSH2 103250
 STSH3 103337
 STSH4 103375
 SWR 001154
 SWREG 000176
 SW0 = 000001
 SW00 = 000001
 SW01 = 000002
 SW02 = 000004
 SW03 = 000010
 SW04 = 000020
 SW05 = 000040
 SW06 = 000100
 SW07 = 000200
 SW08 = 000400
 SW09 = 001000
 SW1 = 000002
 SW10 = 002000
 SW11 = 004000
 SW12 = 010000
 SW13 = 020000
 SW14 = 040000
 SW15 = 100000
 SW2 = 000004
 SW3 = 000010
 SW4 = 000020
 SW5 = 000040
 SW6 = 000100
 SW7 = 000200
 SW8 = 000400
 SW9 = 001000
 TADMSK= 177400
 TAG = 020000

TAGADR= 001114
 TAP = 040000
 TA1 = 000400
 TA16 = 010000
 TA2 = 001000
 TA4 = 002000
 TAB = 004000
 TBITVE= 000014
 TIMEOUT 044766
 TKVEC = 000060
 TPVEC = 000064
 TRAPVE= 000034
 TRE = 040000
 TRTVEC= 000014
 TST = 010000
 TSTNMB 043102
 TSTPRP 043150
 TSTQUE 001464
 TST1 010116
 TST10 012616
 TST11 012776
 TST12 014050
 TST13 014350
 TST14 014566
 TST15 015074
 TST16 015404
 TST17 015646
 TST2 010302
 TST20 016124
 TST21 016374
 TST22 016624
 TST23 017150
 TST24 017570
 TST25 020134
 TST26 020526
 TST27 021216
 TST3 010466
 TST30 021564
 TST31 022106
 TST32 022470
 TST33 023006
 TST34 023572
 TST35 024316
 TST36 025106
 TST37 025672
 TST4 010670
 TST40 026224
 TST41 026664
 TST42 027104
 TST43 027442
 TST44 027730
 TST45 030216
 TST46 030530
 TST47 031036
 TST5 011540
 TST50 031350
 TST51 031660

TST52 032172
 TST53 032610
 TST54 033236
 TST55 033654
 TST56 034214
 TST57 034556
 TST6 012076
 TST60 035110
 TST61 035536
 TST62 036214
 TST63 036720
 TST64 037320
 TST65 037730
 TST66 040330
 TST67 040710
 TST7 012302
 TST70 041254
 TST71 041612
 TYPBN = 104406
 TYPDS = 104405
 TYPE = 104401
 TYPOC = 104402
 TYPON = 104404
 TYPOS = 104403
 UBUSQS 066251
 UNS = 040000
 UNTMSK= 000007
 UPE = 020000
 USE = 040000
 U0 = 000001
 U1 = 000002
 U2 = 000004
 VV = 000100
 WC = 000040
 WCD = 000050
 WCE = 040000
 WCEHI = 010000
 WCELO = 004000
 WCF = 000040
 WCH = 000052
 WD = 000060
 WH = 000062
 WLE = 004000
 WRL = 004000
 XSIZ 006454
 XXDP 001330
 ZEROS 067260
 \$APTHD 001100
 \$ATYC 065760
 \$ATY1 065734
 \$ATY3 065742
 \$ATY4 065752
 \$AUTOB 001150
 \$BASE 001276
 \$BDADR 001136
 \$BDDAT 001142
 \$BELL 001212

\$BIN 062200
 \$CDW1 001302
 \$CDW2 001304
 \$CHARC 063204
 \$CKSWR 064406
 \$CMTAG 001114
 \$CM3 = 000000
 \$CM4 = 000005
 \$CNTLC 065304
 \$CNTLG 065316
 \$CNTLU 065311
 \$CPUOP 001250
 \$CRLF 001217
 \$DBLK 062416
 \$DDW0 001306
 \$DDW1 001310
 \$DDW2 001312
 \$DDW3 001314
 \$DDW4 001316
 \$DDW5 001320
 \$DDW6 001322
 \$DDW7 001324
 \$DEVCT 001232
 \$DEVM 001300
 \$DOAGN 042336
 \$DTBL 062406
 \$ENDAD 042326
 \$ENDCT 042172
 \$ENULL 042342
 \$ENV 001242
 \$ENVM 001243
 \$EOP 042136
 \$EOPCT 042164
 \$EOSP 042100
 \$ERFLG 001117
 \$ERMAX 001131
 \$ERROR 063710
 \$ERRPC 001132
 \$ERRTB 001572
 \$ERTTL 001126
 \$ESCAP 001210
 \$ETABL 001242
 \$ETEND 001326
 \$FATAL 001224
 \$FFLG 066200
 \$FILLC 001172
 \$FILLS 001171
 \$GDADR 001134
 \$GDDAT 001140
 \$GET42 042316
 \$GTSWR 064476
 \$HD = 000000
 \$HIBTS 001100
 \$HIOCT 065446
 \$ICNT 001120
 \$ILLUP 065716
 \$INTAG 001151

\$ITEMB 001130
 \$LF 001220
 \$LFLG 066177
 \$LPADR 001122
 \$LPERR 001124
 \$MADR1 001254
 \$MADR2 001260
 \$MADR3 001264
 \$MADR4 001270
 \$MAIL 001222
 \$MAMS1 001252
 \$MAMS2 001256
 \$MAMS3 001262
 \$MAMS4 001266
 \$MBADR 001102
 \$MFLG 066176
 \$MNEW 065334
 \$MSGAD 001236
 \$MSGLG 001240
 \$MSGTY 001222
 \$MSWR 065323
 \$MTYP1 001253
 \$MTYP2 001257
 \$MTYP3 001263
 \$MTYP4 001267
 \$MXCNT 063524
 \$NULL 001170
 \$NWTST= 000001
 \$OCNT 062650
 \$OMODE 062652
 \$OVER 063510
 \$PASS 001230
 \$PASTM 001106
 \$POWER 065724
 \$PWRDN 065556
 \$PWRMG 065712
 \$PWRUP 065630
 \$QUES 001216
 \$RDCHR 064750
 \$RDLIN 065040
 \$RDOCT 065346
 \$RDSZ = 000010
 \$RESRE 062070
 \$RTNAD 042340
 \$SAVRE 062032
 \$SAVR6 065722
 \$SCOPE 063210
 \$SETUP= 000137
 \$STUP = 177777
 \$SVLAD 063454
 \$SVPC = 000204
 \$SWR = 167400
 \$SWREG 001244
 \$SWRMK= 000000
 \$SWOBT 063526
 \$TESTN 001226
 \$TIMES 001206

\$TKB	001162	\$TMP0	001174	\$TRAP	065450	\$TYPE	062654	\$VECT1	001272
\$TKCNT	064110	\$TMP1	001176	\$TRAP2	065510	\$TYPEC	063066	\$VECT2	001274
\$TKINT	064120	\$TMP2	001200	\$TRP =	000016	\$TYPEX	063206	\$XOFF =	000023
\$TKQEN=	064117	\$TMP3	001202	\$TRPAD	065522	\$TYPOC	062452	\$XON =	000021
\$TKQIN	064112	\$TMP4	001204	\$TSTM	001104	\$TYPON	062466	\$XTSTR	063226
\$TKQOU	064114	\$TN =	000072	\$TSTNM	001116	\$TYPOS	062426	\$\$GET4=	000000
\$TKQSR	064116	\$TPB	001166	\$TTYIN	065274	\$UNIT	001234	\$\$SW08=	000072
\$TKS	001160	\$TPFLG	001173	\$TYPBN	062126	\$UNITM	001110	\$OFILL	062651
\$TKSRV	064170	\$TPS	001164	\$TYPDS	062202	\$USWR	001246	.\$X =	001100

. ABS. 107146 000
 000000 001

ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 55208 WORDS (216 PAGES)
DYNAMIC MEMORY AVAILABLE FOR 69 PAGES
CZRMMA.BIC,CZRMMA/C=CZRMMA.DOC,CZRMMA,SYSMAC/M

\$CKSWR	35-8#	35-10	35-10											
\$CM3	5-0	5-0#												
\$CM4	5-0	5-0	5-0	5-0	5-0	5-0	5-0	5-0	5-0	5-0	5-0#	5-0#	5-0#	5-0#
	5-0#	5-0#												
\$CMTAG	5-0#	9-19	9-19	9-19	9-19	9-19	9-19	9-19						
\$CNTLC	35-8	35-8	35-8	35-8	35-8#									
\$CNTLG	35-8	35-8#												
\$CNTLU	35-8	35-8	35-8#											
\$CPUOP	5-0#													
\$CRLF	5-0#	9-45	9-97	9-107	10-17	10-41	10-114	10-152	13-20	14-21	14-38	14-76	14-86	14-99
	14-107	14-127	35-5	35-5	35-5	35-7	35-7	35-7	35-8	35-8	35-8	35-8		
\$DBLK	35-3	35-3	35-3#											
\$DDW0	5-0#													
\$DDW1	5-0#													
\$DDW2	5-0#													
\$DDW3	5-0#													
\$DDW4	5-0#													
\$DDW5	5-0#													
\$DDW6	5-0#													
\$DDW7	5-0#													
\$DEVCT	5-0#													
\$DEVN	5-0#	9-64*	9-74	9-96*	10-115*	10-122*	10-149*	10-150	11-3	11-30				
\$DOAGN	13-20	13-20	13-20#											
\$DTBL	35-3	35-3#												
\$ENDAD	4-778	9-25	13-20#	35-7										
\$ENDCT	9-19	13-20#												
\$ENULL	13-20#													
\$ENV	5-0#	9-25	11-27	35-5	35-7	35-12	35-12							
\$ENVN	5-0#	9-19	9-68	35-5	35-5	35-12								
\$EOP	11-35	12-31	13-20#	34-17										
\$EOPCT	9-19*	13-20	13-20#											
\$EOSP	12-60	12-88	12-298	13-9#										
\$ERFLG	5-0#	35-6	35-6	35-6	35-6	35-6	35-6*	35-7	35-7	35-7*				
\$ERMAX	5-0#	9-19*	35-6	35-6	35-6	35-6*								
\$ERROR	9-19	35-7#												
\$ERRPC	5-0#	14-34	35-7	35-7	35-7	35-7*	35-7*							
\$ERRTB	7-0#	14-45												
\$ERTTL	5-0#	13-20	13-20*	35-7	35-7	35-7*								
\$ESCAP	5-0#	9-19*	35-6*	35-7	35-7	35-7	35-7							
\$ETABL	5-0#													
\$ETEND	4-781	5-0#												
\$FATAL	5-0#	35-12*												
\$FFLG	35-12	35-12#	35-12*	35-12*	35-12*									
\$FILLC	5-0#	35-5	35-5	35-5										
\$FILLS	5-0#	35-5	35-5											
\$GDADR	5-0#	46-8												
\$GDDAT	5-0#	12-129*	12-130	12-134*	12-143*	12-144	12-148*	12-166*	12-167	12-171*	12-178*	12-179	12-183*	12-190*
	12-191	12-251*	12-273*	12-274*	12-275*	12-283*	12-289*	12-296*	12-302*	12-303*	12-310*	12-342*	12-350*	12-356*
	12-362*	12-368*	12-374*	12-382*	12-390*	12-396*	12-402*	12-408*	12-414*	12-420*	12-429*	12-632*	12-638*	12-658*
	12-674*	12-696*	12-713*	12-731*	12-741*	12-883*	12-891*	12-925*	12-930*	12-935*	12-959*	12-970*	12-:15*	12-:16*
	12-:19	12-:23*	12-:26*	12-:27*	12-:30	12-:82*	12-:83*	12-:86	12-:90*	12-:93*	12-:94*	12-:97	12-:48*	12-:49*
	12-:52	12-:56*	12-:59*	12-:60*	12-:63	12-<14*	12-<15*	12-<18	12-<22*	12-<25*	12-<26*	12-<29	12-=08*	12-=37*
	12-=67*	12-=70	12-B12*	12-B44*	12-C55*	12-F50*	12-F81*	21-51*	21-52*	21-53	21-69*	21-70*	21-76*	21-78*
	21-90*	21-91*	21-92*	21-108*	21-109*	21-124*	21-125*	21-152*	21-153*	22-42*	22-55*	22-66*	22-67	22-79*
	22-84*	22-87	22-98*	22-106*	22-109	22-135*	22-138*	22-139*	22-142	22-154*	22-155*	22-158	22-165*	23-16*
	23-23*	23-24*	23-25*	23-26*	23-30*	23-32	23-86*	23-91*	23-94	23-104*	23-117*	23-122	23-134*	23-135*

23-136* 23-138 23-148* 23-149* 23-151 24-40* 24-55*^{D 7} 26-36* 26-47* 26-63* 26-68 26-87* 26-101* 26-102*
SEQ 0287

	26-129*	26-142*	26-166*	26-167*	26-181*	26-194*	26-195*	26-210*	26-211*	28-33*	28-34	28-43*	28-59*	28-61
	28-70*	28-83*	28-84	28-93*	28-105*	28-106	28-115*	28-128*	28-129	29-25*	29-26*	29-39*	29-40*	29-58*
	29-59*	29-72*	29-73*	29-86*	29-87*	29-100*	29-101*	29-114*	29-115*	30-33*	30-34*	30-47*	30-48*	30-63*
	30-64*	30-80*	30-81*	30-101*	30-102*	30-118*	30-119*	30-132*	30-133*	30-157*	30-158*	30-173*	30-175*	30-187*
	30-188*	30-203*	30-204*	30-220*	30-221*	30-238*	30-239*	30-252*	30-253*	30-268*	30-269*	31-18*	31-19	31-30*
	31-31	31-40*	31-71*	31-72	31-83*	31-84	31-92*	32-56*	32-57*	32-74*	32-75*	32-88*	32-104*	32-109
	32-131*	32-132*	32-133	32-137	32-151*	32-152*	32-168*	32-169*	32-184*	32-185*	32-207*	32-208*	32-225*	32-226*
	32-241*	32-242*	32-255*	32-256*	33-44*	33-45*	33-60*	33-76*	33-77*	33-96*	33-97*	33-110*	33-111*	46-1
	46-6	46-8	46-10	46-11										
\$GET42	13-20#													
\$GTSWR	35-8#	35-10	35-10											
\$HD	4-479	4-479	4-479											
\$HIBTS	4-781#													
\$HIOCT	35-9#	35-9*												
\$ICNT	5-0#	35-6	35-6	35-6	35-6*	35-6*								
\$ILLUP	35-11	35-11	35-11#											
\$INTAG	5-0#	35-8	35-8	35-8	35-8	35-8*								
\$ITEMB	5-0#	14-29	35-7	35-7	35-7	35-7*								
\$LF	5-0#	35-5	35-5	35-7	35-7	35-8	35-8	35-8						
\$LFLG	35-12#	35-12*												
\$LPADR	5-0#	9-19*	35-6	35-6	35-6	35-6*	35-6*	35-6*						
\$LPERR	5-0#	9-19*	35-6	35-6	35-6	35-6*	35-7							
\$MADR1	5-0#													
\$MADR2	5-0#													
\$MADR3	5-0#													
\$MADR4	5-0#													
\$MAIL	4-781	4-781	5-0#	9-19	9-25	11-27	12-2	12-34	12-63	12-91	12-196	12-238	12-263	12-314
	12-332	12-434	12-459	12-476	12-500	12-522	12-547	12-569	12-599	12-617	12-648	12-684	12-719	12-751
	12-816	12-870	12-901	12-945	12-974	12-:40	12-:07	12-:73	12-<39	12-<81	12-=13	12-=46	12->13	12->41
	12->69	12-?03	12-?37	12-?72	12-@06	12-@40	12-@89	12-A40	12-A88	12-B22	12-B59	12-B95	12-C72	12-D23
	12-D83	12-E28	12-E76	12-F22	12-F60	12-F99	12-G43	35-5	35-6	35-7				
\$MAMS1	5-0#													
\$MAMS2	5-0#													
\$MAMS3	5-0#													
\$MAMS4	5-0#													
\$MBADR	4-781#													
\$MFLG	35-12	35-12#	35-12*	35-12*										
\$MNEW	35-8	35-8#												
\$MSGAD	5-0#	35-12	35-12*											
\$MSGIG	5-0#	35-12*												
\$MSGTY	5-0#	35-12	35-12	35-12*	35-12*									
\$MSWR	35-8	35-8#												
\$MTYP1	5-0#													
\$MTYP2	5-0#													
\$MTYP3	5-0#													
\$MTYP4	5-0#													
\$MXCNT	35-6	35-6	35-6	35-6#										
\$NULL	5-0#	35-5	35-5	35-5										
\$NWTST	12-2	12-2#	12-2#	12-34	12-34#	12-34#	12-63	12-63#	12-63#	12-91	12-91#	12-91#	12-196	12-196#
	12-196#	12-238	12-238#	12-238#	12-263	12-263#	12-263#	12-314	12-314#	12-314#	12-332	12-332#	12-332#	12-434
	12-434#	12-434#	12-459	12-459#	12-459#	12-476	12-476#	12-476#	12-500	12-500#	12-500#	12-522	12-522#	12-522#
	12-547	12-547#	12-547#	12-569	12-569#	12-569#	12-599	12-599#	12-599#	12-617	12-617#	12-617#	12-648	12-648#
	12-648#	12-684	12-684#	12-684#	12-719	12-719#	12-719#	12-751	12-751#	12-751#	12-816	12-816#	12-816#	12-870
	12-870#	12-870#	12-901	12-901#	12-901#	12-945	12-945#	12-945#	12-974	12-974#	12-974#	12-:40	12-:40#	12-:40#
	12-:07	12-:07#	12-:07#	12-:73	12-:73#	12-:73#	12-<39	12-<39#	12-<39#	12-<81	12-<81#	12-<81#	12-=13	12-=13#
	12-=13#	12-=46	12-=46#	12-=46#	12->13	12->13#	12->13#	12->4	12->41#	12->41#	12->69	12->69#	12->69#	12-?03

12-?03# 12-?03# 12-?37 12-?37# 12-?37# 12-?72 12-?72#^{F 7} 12-?72# 12-a06 12-a06# 12-a06# 12-a40 12-a40# 12-a40#
SEQ 0289

	12-a89	12-a89#	12-a89#	12-A40	12-A40#	12-A40#	12-A88	12-A88#	12-A88#	12-B22	12-B22#	12-B22#	12-B59	12-B59#
	12-B59#	12-B95	12-B95#	12-B95#	12-C72	12-C72#	12-C72#	12-D23	12-D23#	12-D23#	12-D83	12-D83#	12-D83#	12-E28
	12-E28#	12-E28#	12-E76	12-E76#	12-E76#	12-F22	12-F22#	12-F22#	12-F60	12-F60#	12-F60#	12-F99	12-F99#	12-F99#
	12-G43	12-G43	12-G43#	12-G43#										
\$OCNT	35-4#	35-4*	35-4*											
\$OMODE	35-4	35-4#	35-4*	35-4*	35-4*	35-4*								
\$OVER	35-6	35-6	35-6	35-6	35-6#									
\$PASS	5-0#	9-19*	13-20	13-20	13-20	13-20*	13-20*	35-6	35-6	35-6				
\$PASTM	4-781#													
\$POWER	35-11	35-11#												
\$PWDRN	9-19	35-11	35-11#											
\$PWRMG	35-11#													
\$PWRUP	35-11	35-11#												
\$QUES	5-0#	35-5	35-5	35-7	35-7	35-8	35-8	35-8	35-8					
\$R2A	35-10													
\$RDCHR	35-8#	35-10	35-10											
\$RDDEC	35-10													
\$RDLIN	35-8#	35-10	35-10											
\$RDOCT	35-9#	35-10	35-10											
\$RDSZ	35-8	35-8#												
\$RESRE	35-1#	35-10												
\$RTNAD	13-20#													
\$SAVR6	35-11	35-11#	35-11*	35-11*	35-11*									
\$SAVRE	35-1#	35-10	35-10											
\$SCOPE	9-19	35-6#												
\$SETUP	4-774	4-774	4-774	4-774	4-774	4-774	4-774#	4-774#	4-774#	4-774#	4-774#	4-774#	4-774#	9-19
	9-19	9-19	9-19	9-19	9-19	9-19	9-19	9-19	9-19	9-19	9-19	9-25	9-25	9-25
\$STUP	11-27	13-20	13-20	35-6	35-7	35-7	35-7	35-7	35-8	35-8	35-8	35-8	35-8	4-774#
	4-774	4-774	4-774	4-774	4-774	4-774	4-774#	4-774#	4-774#	4-774#	4-774#	4-774#	4-774#	4-774#
	4-774#	4-774#	4-774#	4-774#										
\$SVLAD	35-6	35-6#												
\$SVPC	4-778	4-778#												
\$SWOBT	35-6	35-6#												
\$SWR	4-468#	4-479	4-480	4-480	4-480	4-480	4-480	4-480	4-480	4-480	5-0	5-0	5-0	9-19
	9-19	9-19	9-19	9-19	12-2	12-34	12-63	12-91	12-196	12-238	12-263	12-314	12-332	12-434
	12-459	12-476	12-500	12-522	12-547	12-569	12-599	12-617	12-648	12-684	12-719	12-751	12-816	12-870
	12-901	12-945	12-974	12-:40	12-:07	12-:73	12-<39	12-<81	12-=13	12-=46	12->13	12->41	12->69	12-?03
	12-?37	12-?72	12-a06	12-a40	12-a89	12-A40	12-A88	12-B22	12-B59	12-B95	12-C72	12-D23	12-D83	12-E28
	12-E76	12-F22	12-F60	12-F99	12-G43	13-20	13-20	13-20	13-20	13-20	35-6	35-6	35-6	35-6
	35-6	35-6	35-6	35-6	35-6	35-6	35-6	35-6	35-6	35-6	35-6	35-6	35-6	35-6
	35-6	35-6	35-7	35-7	35-7	35-7	35-7	35-7	35-7	35-7	35-7	35-7	35-7	35-11
\$SWREG	5-0#	9-19												
\$SWRMK	4-480	4-480	4-480	4-480	4-480	4-480	4-480	4-480	4-480	35-6	35-6	35-6	35-6	35-6
	35-6	35-6	35-6	35-6	35-6									
\$TESTN	5-0#	12-2*	12-34*	12-63*	12-91*	12-196*	12-238*	12-263*	12-314*	12-332*	12-434*	12-459*	12-476*	12-500*
	12-522*	12-547*	12-569*	12-599*	12-617*	12-648*	12-684*	12-719*	12-751*	12-816*	12-870*	12-901*	12-945*	12-974*
	12-:40*	12-:07*	12-:73*	12-<39*	12-<81*	12-=13*	12-=46*	12->13*	12->41*	12->69*	12-?03*	12-?37*	12-?72*	12-a06*
	12-a40*	12-a89*	12-A40*	12-A88*	12-B22*	12-B59*	12-B95*	12-C72*	12-D23*	12-D83*	12-E28*	12-E76*	12-F22*	12-F60*
	12-F99*	12-G43*	14-25	35-6*										
\$TIMES	5-0#	9-19*	11-38*	12-91*	12-459*	12-476*	12-500*	12-522*	12-547*	12->13*	12->41*	12->69*	12-?03*	12-?37*
	12-?72*	12-a06*	12-B59*	12-G43*	13-20*	35-6	35-6	35-6	35-6*	35-6*				
\$TKB	5-0#	35-5	35-5	35-8	35-8	35-8	35-8	35-8	35-8	35-8				
\$TKCNT	35-8	35-8	35-8#	35-8*	35-8*	35-8*								
\$TKINT	10-4	11-40	12-120	35-8	35-8	35-8#								
\$TKQEN	35-8	35-8	35-8#											
\$TKQIN	35-8	35-8	35-8#	35-8*	35-8*	35-8*	35-8*							

STKQOU 35-8 35-8 35-8# 35-8* 35-8* 35-8*

H 7

SEQ 0291

8XON 35-5 35-5

J 7

SEQ 0293

AMTYP1 5-0 5-0

L 7

SEQ 0295

AMTYP2	5-0	5-0												
AMTYP3	5-0	5-0												
AMTYP4	5-0	5-0												
AOE	4-571# 37-86	4-582	22-311	22-326	22-341	22-356	22-360	23-117	23-121	37-77	37-78	37-81	37-82	37-85
APASS	5-0	5-0												
APE	4-749#													
APRIOR	5-0													
APTCSU	35-5	35-12#												
APTENV	35-5	35-7	35-12	35-12#										
APTSIZ	9-19	35-12#												
APTSPO	35-5	35-12	35-12#											
ARGS	12-100#	12-118#	12-124#	12-205#	12-222#	12-229#	12-231#	12-235#	12-248#	12-269#	12-323#	12-326#	12-329#	12-339#
	12-379#	12-425#	12-446#	12-449#	12-452#	12-456#	12-461	12-461#	12-467#	12-469#	12-471#	12-473#	12-478	12-478#
	12-485#	12-491#	12-493#	12-495#	12-497#	12-502	12-502#	12-512#	12-514#	12-519#	12-524	12-524#	12-534#	12-538#
	12-540#	12-542#	12-544#	12-549	12-549#	12-556#	12-558#	12-560#	12-562#	12-564#	12-566#	12-571	12-571#	12-584#
	12-589#	12-592#	12-594#	12-596#	12-601	12-601#	12-606#	12-608#	12-610	12-610#	12-612#	12-614#	12-619	12-619#
	12-625#	12-627#	12-641	12-641#	12-643#	12-645#	12-650	12-650#	12-655#	12-663	12-663#	12-667#	12-669#	12-677
	12-677#	12-679#	12-681#	12-686	12-686#	12-691#	12-704#	12-706#	12-708#	12-716	12-716#	12-721	12-721#	12-726#
	12-738#	12-746#	12-748#	12-753	12-753#	12-777#	12-792#	12-794#	12-802#	12-804#	12-806	12-806#	12-808#	12-810#
	12-818	12-818#	12-863#	12-872	12-872#	12-878#	12-880#	12-894	12-894#	12-896#	12-898#	12-903	12-903#	12-915#
	12-919#	12-921#	12-938	12-938#	12-940#	12-942#	12-947	12-947#	12-952#	12-954#	12-965#	12-976	12-976#	12-:09#
	12-:12#	12-:14#	12-:42	12-:42#	12-:76#	12-:79#	12-:81#	12-:09	12-:09#	12-:42#	12-:45#	12-:47#	12-:75	12-:75#
	12-:08#	12-:11#	12-:13#	12-:41	12-:41#	12-:44#	12-:47#	12-:66#	12-:69#	12-:71#	12-:73#	12-:83	12-:83#	12-:86#
	12-:94#	12-:03#	12-:15	12-:15#	12-:19#	12-:31#	12-:33#	12-:48	12-:48#	12-:62#	12-:64#	12-:66#	12-:16	12-:16#
	12->26#	12->30#	12->32#	12->34#	12->36	12->36#	12->38#	12->44	12->44#	12->54#	12->58#	12->60#	12->62#	12->64
	12->64#	12->66#	12->73	12->73#	12->82#	12->87#	12->89#	12->91#	12->93	12->93#	12->95#	12->?06	12->?06#	12->?18#
	12-?22#	12-?24#	12-?26#	12-?28	12-?28#	12-?30#	12-?40	12-?40#	12-?51#	12-?56#	12-?58#	12-?60#	12-?62	12-?62#
	12-?64#	12-?75	12-?75#	12-?86#	12-?90#	12-?92#	12-?94#	12-?96	12-?96#	12-?98#	12-?09	12-?09#	12-?20#	12-?24#
	12-@26#	12-@28#	12-@30	12-@30#	12-@32#	12-@43	12-@43#	12-@59#	12-@61#	12-@63#	12-@67#	12-@69#	12-@71#	12-@74
	12-@74#	12-@76#	12-@91	12-@91#	12-A08#	12-A10#	12-A12#	12-A16#	12-A18#	12-A20#	12-A23	12-A23#	12-A25#	12-A42
	12-A42#	12-A58#	12-A60#	12-A62#	12-A66#	12-A68#	12-A70#	12-A72	12-A72#	12-A74#	12-B05#	12-B07#	12-B09#	12-B17
	12-B17#	12-B19#	12-B24	12-B24#	12-B37#	12-B39#	12-B48#	12-B50#	12-B52#	12-B54	12-B54#	12-B56#	12-B62	12-B62#
	12-B72#	12-B79#	12-B82#	12-B84#	12-B86#	12-B88#	12-B90	12-B90#	12-B92#	12-C12#	12-C15#	12-C17#	12-C74	12-C74#
	12-C88#	12-C91#	12-C93#	12-C95#	12-C97#	12-D01#	12-D03#	12-D05#	12-D07	12-D07#	12-D09#	12-D25	12-D25#	12-D41#
	12-D45#	12-D47#	12-D49#	12-D53#	12-D57#	12-D59#	12-D61#	12-D63	12-D63#	12-D65#	12-D85	12-D85#	12-E01#	12-E05#
	12-E07#	12-E09#	12-E11#	12-E13	12-E13#	12-E15#	12-E30	12-E30#	12-E47#	12-E51#	12-E53#	12-E55#	12-E57#	12-E59
	12-E59#	12-E61#	12-E78	12-E78#	12-E94#	12-E98#	12-F00#	12-F02#	12-F04#	12-F06	12-F06#	12-F08#	12-F24	12-F24#
	12-F40#	12-F44#	12-F46#	12-F55	12-F55#	12-F57#	12-F62	12-F62#	12-F75#	12-F77#	12-F88#	12-F90#	12-F92#	12-F94
	12-F94#	12-F96#	12-G01	12-G01#	12-G13#	12-G21#	12-G25#	12-G27#	12-G29#	12-G31	12-G31#	12-G33#	12-G46	12-G46#
	12-G56#	12-G60#	12-G62#	12-G64#	12-G66	12-G66#	12-G68#							
ASWREG	5-0	5-0												
ATA	4-551#	12-274	12-634	12-637	12-638	12-656	12-658	12-660	12-670	12-673	12-692	12-695	12-696	12-709
	12-712	12-727	12-730	12-731	12-739	12-741	12-743	12-887	12-890	12-891	12-:26	12-:27	12-:29	12-:93
	12-:94	12-:96	12-:59	12-:60	12-:62	12-:25	12-:26	12-:28	22-138	22-139	22-141	26-156	26-157	26-192
	26-195	30-145	30-146	30-185	30-188	32-200	32-201	32-253	32-256	37-58	37-59	37-60	37-63	37-64
	37-67	37-68	37-69	37-70	37-71	37-72	37-73	37-74	37-75	37-76	37-79	37-80	37-83	37-84
	37-87	37-88												
ATESTN	5-0	5-0												
ATNSK	4-588#	12-699	31-55											
ATNTBL	9-74	9-96	10-149	11-11	38-3#									
AUNIT	5-0	5-0												
AUSWR	5-0	5-0												
AVECT1	4-772#	5-0	5-0											
AVECT2	5-0	5-0												
BACK	12-100	12-100#	12-118	12-118#	12-124	12-124#	12-205	12-205#	12-222	12-222#	12-229	12-229#	12-231	12-231#

12-231# 12-235 12-235# 12-235# 12-248 12-248# 12-269^N 7 12-269# 12-323 12-323# 12-326 12-326# 12-326# 12-329
SEQ 0297

BADTMO 9-3# 9-21

C 8

SEQ 0299

12-G31 12-G66 24-21#

E 8

SEQ 0301

EBL

4-622#

G 8

SEQ 0303

EFT110 7-222 43-4#

1 8

SEQ 0305

EMS111 40-74 40-226 44-74#

K 8

SEQ 0307

EMS112	40-74	44-75#												
EMS113	40-75	40-76	40-78	40-127	44-77#									
EMS114	40-77	44-78#												
EMS115	40-44	40-45	40-48	40-50	40-51	40-52	40-54	40-55	40-60	40-61	40-63	40-66	40-67	40-70
	40-73	40-77	40-107	40-108	40-109	40-110	40-134	40-137	40-139	40-140	40-142	40-179	40-182	40-183
	40-185	40-186	40-187	40-188	40-189	40-190	40-192	40-193	40-194	40-195	40-201	40-205	40-221	40-222
	40-223	40-232	40-234	44-79#										
EMS116	40-74	40-77	40-191	40-225	44-80#									
EMS117	40-77	40-143	40-148	40-149	40-178	40-185	40-201	40-225	40-228	40-229	40-230	40-231	44-81#	
EMS12	40-12	44-12#												
EMS120	40-78	44-82#												
EMS121	40-79	40-88	40-97	40-99	44-83#									
EMS122	40-80	40-89	44-84#											
EMS123	40-81	40-88	40-89	40-90	40-90	40-91	40-92	40-93	40-94	40-95	40-96	40-98	40-126	44-85#
EMS124	40-56	40-82	40-91	40-101	44-86#									
EMS125	40-83	40-92	40-102	40-169	44-87#									
EMS126	40-84	40-93	40-103	44-88#										
EMS127	44-89#													
EMS13	40-13	40-42	40-216	44-13#										
EMS130	40-86	40-95	40-106	44-90#										
EMS131	40-87	40-96	40-104	44-91#										
EMS132	40-79	40-80	40-81	40-82	40-83	40-84	40-85	40-86	40-87	40-88	40-89	40-90	40-91	40-92
	40-93	40-94	40-95	40-96	40-97	40-98	40-99	40-100	40-101	40-102	40-103	40-104	40-105	40-106
	40-125	40-126	40-168	40-169	44-92#									
EMS133	40-79	40-80	40-81	40-82	40-83	40-84	40-85	40-86	40-87	40-125	44-93#			
EMS134	40-88	40-89	40-90	40-91	40-92	40-93	40-94	40-95	40-96	40-126	44-94#			
EMS135	40-97	40-98	44-95#											
EMS136	40-99	40-100	40-101	40-102	40-103	40-104	40-105	40-106	44-96#					
EMS137	40-85	40-94	40-100	44-97#										
EMS14	40-13	44-14#												
EMS140	40-44	40-52	40-60	40-107	40-117	40-137	40-186	40-192	44-98#					
EMS141	40-35	40-44	40-45	40-60	40-61	40-66	40-107	40-129	40-137	40-187	40-188	40-193	40-222	44-99#
EMS142	40-47	40-65	40-108	40-140	40-184	40-202	44-100#							
EMS143	40-47	40-65	40-73	40-108	40-109	40-110	40-133	40-140	40-184	40-202	40-205	44-101#		
EMS144	40-36	40-73	40-109	40-110	40-205	44-102#								
EMS145	40-36	40-73	40-109	40-110	40-205	44-103#								
EMS146	40-109	40-185	40-190	40-201	44-104#									
EMS147	40-69	40-112	40-113	40-173	40-217	44-105#								
EMS15	40-13	40-42	40-216	44-15#										
EMS150	40-50	40-51	40-54	40-63	40-67	40-111	40-134	40-138	40-182	40-183	40-194	40-195	40-223	40-234
	44-106#													
EMS151	40-69	40-112	40-113	40-173	40-217	44-107#								
EMS152	40-50	40-51	40-54	40-63	40-111	40-134	40-182	40-183	40-194	40-195	40-223	44-108#		
EMS153	40-111	44-109#												
EMS154	40-111	40-112	40-122	40-123	40-124	44-110#								
EMS155	40-112	40-124	44-111#											
EMS156	40-113	40-114	40-115	40-116	40-173	44-112#								
EMS157	40-113	40-123	44-113#											
EMS16	40-13	44-16#												
EMS160	40-114	40-115	40-116	44-114#										
EMS161	40-114	44-115#												
EMS162	44-116#													
EMS163	40-58	40-59	40-60	40-61	40-62	40-63	40-64	40-65	40-66	40-67	40-68	40-69	40-70	40-71
	40-72	40-73	40-117	40-143	44-117#									
EMS164	40-45	44-118#												
EMS165	40-56	40-57	40-71	40-72	40-172	40-196	40-197	40-198	44-120#					

EMS166 40-122 40-142 44-121#

M 8

SEQ 0309

EMS44 40-32 40-211 40-230 44-38#

B 9

SEQ 0311

EMT115 7-237 40-79#

D 9

SEQ 0313

EMT116	7-240	40-80#
EMT117	7-243	40-81#
EMT12	7-31	40-12#
EMT120	7-246	40-82#
EMT121	7-249	40-83#
EMT122	7-252	40-84#
EMT123	7-255	40-85#
EMT124	7-258	40-86#
EMT125	7-261	40-87#
EMT126	7-264	40-88#
EMT127	7-267	40-89#
EMT13	7-34	40-13#
EMT130	7-270	40-90#
EMT131	7-273	40-91#
EMT132	7-276	40-92#
EMT133	7-279	40-93#
EMT134	7-282	40-94#
EMT135	7-285	40-95#
EMT136	7-288	40-96#
EMT137	7-291	40-97#
EMT14	7-37	40-14#
EMT140	7-294	40-98#
EMT141	7-297	40-99#
EMT142	7-300	40-100#
EMT143	7-303	40-101#
EMT144	7-306	40-102#
EMT145	7-309	40-103#
EMT146	7-312	40-104#
EMT147	7-315	40-105#
EMT15	7-40	40-15#
EMT150	7-318	40-106#
EMT151	7-321	40-107#
EMT152	7-324	40-108#
EMT153	7-327	40-109#
EMT154	7-330	40-110#
EMT155	7-333	40-111#
EMT156	7-336	40-112#
EMT157	7-339	40-113#
EMT16	7-43	40-16#
EMT160	7-342	40-114#
EMT161	7-345	40-115#
EMT162	7-348	40-116#
EMT163	40-117#	
EMT164	7-355	40-118#
EMT165	7-358	40-119#
EMT166	7-361	40-120#
EMT167	7-364	40-121#
EMT17	7-46	40-17#
EMT170	40-122#	
EMT171	7-370	40-123#
EMT172	7-373	40-124#
EMT173	7-376	40-125#
EMT174	7-379	40-126#
EMT175	7-382	40-127#
EMT176	7-385	40-128#
EMT177	7-388	40-129#

EMT2 7-7 40-4#

F 9

SEQ 0315

EMT20	7-49	40-18#
EMT200	7-391	40-130#
EMT201	7-394	40-131#
EMT202	7-397	40-132#
EMT203	7-400	40-133#
EMT204	7-403	40-134#
EMT205	7-406	40-135#
EMT206	7-409	40-136#
EMT207	7-412	40-137#
EMT21	7-52	40-19#
EMT210	7-415	40-138#
EMT211	7-418	40-139#
EMT212	7-421	40-140#
EMT213	7-424	40-141#
EMT214	7-427	40-142#
EMT215	7-430	40-143#
EMT216	7-433	40-144#
EMT217	7-436	40-145#
EMT22	7-55	40-20#
EMT220	7-439	40-146#
EMT221	7-442	40-147#
EMT222	7-445	40-148#
EMT223	7-448	40-149#
EMT224	40-150#	
EMT225	40-151#	
EMT226	40-152#	
EMT227	40-153#	
EMT23	7-58	40-21#
EMT230	40-154#	
EMT231	40-155#	
EMT232	40-156#	
EMT233	40-157#	
EMT234	40-158#	
EMT235	40-159#	
EMT236	40-160#	
EMT237	40-161#	
EMT24	7-61	40-22#
EMT240	40-162#	
EMT241	40-163#	
EMT242	40-164#	
EMT243	40-165#	
EMT244	40-166#	
EMT245	40-167#	
EMT246	7-505	40-168#
EMT247	7-508	40-169#
EMT25	7-64	40-23#
EMT250	7-511	40-170#
EMT251	7-514	40-171#
EMT252	7-517	40-172#
EMT253	7-520	40-173#
EMT254	7-523	40-174#
EMT255	7-526	40-175#
EMT256	7-529	40-176#
EMT257	7-532	40-177#
EMT26	7-67	40-24#
EMT260	7-535	40-178#

EMT261 7-538 40-179#

H 9

SEQ 0317

EMT262	7-541	40-180#
EMT263	7-544	40-181#
EMT264	7-547	40-182#
EMT265	7-550	40-183#
EMT266	7-553	40-184#
EMT267	7-557	40-185#
EMT27	7-70	40-25#
EMT270	7-560	40-186#
EMT271	7-564	40-187#
EMT272	7-567	40-188#
EMT273	7-570	40-189#
EMT274	7-574	40-190#
EMT275	7-578	40-191#
EMT276	7-582	40-192#
EMT277	7-587	40-193#
EMT3	7-10	40-5#
EMT30	7-73	40-26#
EMT300	7-591	40-194#
EMT301	7-595	40-195#
EMT302	7-598	40-196#
EMT303	7-601	40-197#
EMT304	7-604	40-198#
EMT305	7-607	40-199#
EMT306	7-610	40-200#
EMT307	7-613	40-201#
EMT31	7-76	40-27#
EMT310	7-616	40-202#
EMT311	7-619	40-203#
EMT312	7-622	40-204#
EMT313	7-625	40-205#
EMT314	7-628	40-206#
EMT315	7-631	40-207#
EMT316	7-634	40-208#
EMT317	7-637	40-209#
EMT32	7-79	40-28#
EMT320	7-640	40-210#
EMT321	7-643	40-211#
EMT322	7-646	40-212#
EMT323	7-649	40-213#
EMT324	7-652	40-214#
EMT325	7-655	40-215#
EMT326	7-658	40-216#
EMT327	7-661	40-217#
EMT33	7-82	40-29#
EMT330	7-664	40-218#
EMT331	7-667	40-219#
EMT332	7-670	40-220#
EMT333	7-673	40-221#
EMT334	7-676	40-222#
EMT335	7-679	40-223#
EMT336	7-352	7-682 40-224#
EMT337	7-685	40-225#
EMT34	7-85	40-30#
EMT340	7-688	40-226#
EMT341	7-691	40-227#
EMT342	7-694	40-228#

EMT343 7-697 40-229#

J 9

SEQ 0319

12-38 12-39 12-40* 12-41* 12-45* 12-46* 12-57*^L 12-58*⁹ 12--52 12--53 12--54* 12--55* 12->10* 12->11*
SEQ 0321

	12-C68*	12-C69*	17-35	17-35	17-39*	17-40*	17-70*	17-70*	18-29	18-29	18-33*	18-34*	18-59*	18-59*
	19-3	19-3	19-4*	19-5*	19-11*	19-18*	19-18*	20-8	20-8	20-9*	20-10*	20-34*	20-34*	25-20
	25-20	25-21*	25-22*	25-51*	25-51*	27-12	27-12	27-13*	27-14*	27-25*	27-25*	35-6	35-6*	35-6*
	35-6*													
ERTY00	14-22	14-138#												
ERTY01	14-26	14-139#												
ERTY02	14-31	14-140#												
ERTY03	14-33	14-141#												
ERTY04	14-125	14-142#												
ESRC	4-624#													
FO	4-494#	22-29												
F1	4-493#	22-29												
F2	4-492#	22-29	23-113											
F3	4-491#	22-29												
F4	4-490#	22-29												
FER	4-576#	4-582	22-416											
FIND	12-100	12-100#	12-100#	12-118	12-118#	12-118#	12-124	12-124#	12-124#	12-205	12-205#	12-205#	12-222	12-222#
	12-222#	12-229	12-229#	12-229#	12-231	12-231#	12-231#	12-235	12-235#	12-235#	12-248	12-248#	12-248#	12-269
	12-269#	12-269#	12-323	12-323#	12-323#	12-326	12-326#	12-326#	12-329	12-329#	12-329#	12-339	12-339#	12-339#
	12-379	12-379#	12-379#	12-425	12-425#	12-425#	12-446	12-446#	12-446#	12-449	12-449#	12-449#	12-452	12-452#
	12-452#	12-456	12-456#	12-456#	12-461	12-461#	12-461#	12-467	12-467#	12-467#	12-469	12-469#	12-469#	12-471
	12-471#	12-471#	12-473	12-473#	12-473#	12-478	12-478#	12-478#	12-485	12-485#	12-485#	12-491	12-491#	12-491#
	12-493	12-493#	12-493#	12-495	12-495#	12-495#	12-497	12-497#	12-497#	12-502	12-502#	12-502#	12-512	12-512#
	12-512#	12-514	12-514#	12-514#	12-519	12-519#	12-519#	12-524	12-524#	12-524#	12-534	12-534#	12-534#	12-538
	12-538#	12-538#	12-540	12-540#	12-540#	12-542	12-542#	12-542#	12-544	12-544#	12-544#	12-549	12-549#	12-549#
	12-556	12-556#	12-558	12-558#	12-560	12-560#	12-560#	12-562	12-562#	12-562#	12-564	12-564#	12-564#	12-566
	12-566#	12-566#	12-571	12-571#	12-571#	12-584	12-584#	12-584#	12-589	12-589#	12-589#	12-592	12-592#	12-592#
	12-594	12-594#	12-594#	12-596	12-596#	12-596#	12-601	12-601#	12-601#	12-606	12-606#	12-606#	12-608	12-608#
	12-608#	12-610	12-610#	12-610#	12-612	12-612#	12-612#	12-614	12-614#	12-614#	12-619	12-619#	12-619#	12-625
	12-625#	12-625#	12-627	12-627#	12-627#	12-641	12-641#	12-641#	12-643	12-643#	12-643#	12-645	12-645#	12-645#
	12-650	12-650#	12-650#	12-655	12-655#	12-655#	12-663	12-663#	12-663#	12-667	12-667#	12-667#	12-669	12-669#
	12-669#	12-677	12-677#	12-677#	12-679	12-679#	12-679#	12-681	12-681#	12-681#	12-686	12-686#	12-686#	12-691
	12-691#	12-691#	12-704	12-704#	12-704#	12-706	12-706#	12-706#	12-708	12-708#	12-708#	12-716	12-716#	12-716#
	12-721	12-721#	12-721#	12-726	12-726#	12-726#	12-738	12-738#	12-738#	12-746	12-746#	12-746#	12-748	12-748#
	12-748#	12-753	12-753#	12-753#	12-777	12-777#	12-777#	12-792	12-792#	12-792#	12-794	12-794#	12-794#	12-802
	12-802#	12-802#	12-804	12-804#	12-804#	12-806	12-806#	12-806#	12-808	12-808#	12-808#	12-810	12-810#	12-810#
	12-818	12-818#	12-818#	12-863	12-863#	12-863#	12-872	12-872#	12-872#	12-878	12-878#	12-878#	12-880	12-880#
	12-880#	12-894	12-894#	12-894#	12-896	12-896#	12-896#	12-898	12-898#	12-898#	12-903	12-903#	12-903#	12-915
	12-915#	12-915#	12-919	12-919#	12-919#	12-921	12-921#	12-921#	12-938	12-938#	12-938#	12-940	12-940#	12-940#
	12-942	12-942#	12-942#	12-947	12-947#	12-947#	12-952	12-952#	12-952#	12-954	12-954#	12-954#	12-965	12-965#
	12-965#	12-976	12-976#	12-976#	12-:09	12-:09#	12-:09#	12-:12	12-:12#	12-:12#	12-:14	12-:14#	12-:14#	12-:42
	12-:42#	12-:42#	12-:76	12-:76#	12-:76#	12-:79	12-:79#	12-:79#	12-:81	12-:81#	12-:81#	12-:09	12-:09#	12-:09#
	12-:42	12-:42#	12-:42#	12-:45	12-:45#	12-:45#	12-:47	12-:47#	12-:47#	12-:75	12-:75#	12-:75#	12-<08	12-<08#
	12-<08#	12-<11	12-<11#	12-<11#	12-<13	12-<13#	12-<13#	12-<41	12-<41#	12-<41#	12-<44	12-<44#	12-<47	12-<47#
	12-<47#	12-<66	12-<66#	12-<66#	12-<69	12-<69#	12-<71	12-<71#	12-<71#	12-<73	12-<73#	12-<73#	12-<83	12-<83#
	12-<83#	12-<86	12-<86#	12-<94	12-<94#	12-<94#	12-=03	12-=03#	12-=03#	12-=15	12-=15#	12-=15#	12-=19	12-=19#
	12-=19#	12-=31	12-=31#	12-=31#	12-=33	12-=33#	12-=33#	12-=48	12-=48#	12-=48#	12-=62	12-=62#	12-=62#	12-=64
	12-=64#	12-=64#	12-=66	12-=66#	12-=66#	12->16	12->16#	12->16#	12->26	12->26#	12->26#	12->30	12->30#	12->30#
	12->32	12->32#	12->32#	12->34	12->34#	12->34#	12->36	12->36#	12->36#	12->38	12->38#	12->38#	12->44	12->44#
	12->44#	12->54	12->54#	12->54#	12->58	12->58#	12->58#	12->60	12->60#	12->60#	12->62	12->62#	12->62#	12->64
	12->64#	12->64#	12->66	12->66#	12->66#	12->73	12->73#	12->73#	12->82	12->82#	12->82#	12->87	12->87#	12->87#
	12->89	12->89#	12->89#	12->91	12->91#	12->91#	12->93	12->93#	12->93#	12->95	12->95#	12->95#	12-?06	12-?06#
	12-?06#	12-?18	12-?18#	12-?18#	12-?22	12-?22#	12-?22#	12-?24	12-?24#	12-?24#	12-?26	12-?26#	12-?26#	12-?28
	12-?28#	12-?28#	12-?30	12-?30#	12-?30#	12-?40	12-?40#	12-?40#	12-?51	12-?51#	12-?51#	12-?56	12-?56#	12-?56#
	12-?58	12-?58#	12-?58#	12-?60	12-?60#	12-?60#	12-?62	12-?62#	12-?62#	12-?64	12-?64#	12-?64#	12-?75	12-?75#
	12-?75#	12-?86	12-?86#	12-?86#	12-?90	12-?90#	12-?90#	12-?92	12-?92#	12-?92#	12-?94	12-?94#	12-?94#	12-?96

12-?96# 12-?96# 12-?98 12-?98# 12-?98# 12-a09 12-a09#^{N 9} 12-a09# 12-a20 12-a20# 12-a20# 12-a24 12-a24# 12-a24#
SEQ 0323

HCRC 4-572# 4-582 22-386

C 10

SEQ 0325

HELP	10-37	48-7#												
HT	4-483#	14-62	35-5	35-5										
IAE	4-570#	12-:49	12-:51	12-@74	12-A23	12-A72	12-E13	12-E59	12-F06	22-283	23-89	23-105	23-132	26-47
IDXMSK	26-67	30-24	30-78	30-81	32-88	32-108	37-59	37-69	37-77	37-78	37-81	37-82	37-85	37-86
IE	4-709#	17-51	17-54	18-45	18-48									
ILF	4-722#	4-755#	12-779	12-840										
ILF02	4-580#	12-:16	12-:18	22-155	22-157	29-51	29-111	29-115	30-24	30-45	30-48	37-58	37-67	37-68
ILF24	37-70	37-71	37-72	37-73	37-74	37-75	37-76	37-79	37-80	37-83	37-84	37-87	37-88	
ILF26	4-500#													
ILF30	4-510#													
ILF32	4-511#													
ILF34	4-515#													
ILF36	4-515#													
ILF40	4-515#													
ILF42	4-515#													
ILF44	4-515#													
ILF46	4-515#													
ILF54	4-518#													
ILF56	4-519#													
ILF64	4-522#													
ILF66	4-523#													
ILF74	4-526#													
ILF76	4-527#	12-:35	12-:02	12-:68	12-<34	12-<77								
ILR	4-579#	12-=69	12-=97	29-51	29-97	29-101	30-232	30-236	30-239					
ILRG50	4-707#													
ILRG52	4-707#													
ILRG54	4-707#													
ILRG56	4-707#													
ILRG60	4-707#													
ILRG62	4-707#													
ILRG64	4-707#													
ILRG66	4-707#													
ILRG70	4-707#													
ILRG72	4-707#													
ILRG74	4-707#													
ILRG76	4-707#													
IOTVEC	4-483#	9-19*	9-19*											
IPCK0	4-759#													
IPCK1	4-758#													
IPCK2	4-757#													
IPCK3	4-756#													
IR	4-735#	12-142	28-58											
IVC	4-676#	12-515	12-:83	12-:85	12-B10	12-B12	12-B14	12-B17	12-F48	12-F50	12-F52	12-F55	22-198	22-199
	26-85	26-208	30-92	30-99	30-102	30-171	32-125	32-149	32-152	32-239	33-58	37-58	37-59	37-60
	37-62	37-63	37-64	37-67	37-68	37-69	37-70	37-71	37-72	37-73	37-74	37-75	37-76	37-77
	37-78	37-79	37-80	37-81	37-82	37-83	37-84	37-85	37-86	37-87	37-88			
LBC	4-678#	12-426	12-429	12-430										
LBT	4-556#	12-274	22-358	23-91	23-93									
LF	4-483#	14-60	35-5	35-5	36-14	36-16	44-79							
LODEV	9-76	36-24#												
LS	4-633#	12-165	28-82	31-70										
LSC	4-677#	12-577												
LST	4-634#	12-165	28-82	31-70										
LSTRK	6-0#	11-46*	11-54*	12-@95	12-A34	12-E34	12-E70	23-62	26-53	32-94				

MCLK

4-607#

E 10

SEQ 0327

PIRQVE 4-483#

G 10

SEQ 0329

RMAS 4-693# 12-702 12-774

I 10

SEQ 0331

22-358 23-92 26-91 26-146 26-155 26-162 26-166^{K 10} 26-168 26-177 26-181 26-183 26-192 26-194 26-196
SEQ 0333

	26-206	26-210	26-212	28-126	29-23	29-25	29-27	29-37	29-39	29-41	30-68	30-106	30-144	30-153
	30-157	30-159	30-169	30-173	30-174	30-185	30-187	30-189	30-201	30-203	30-205	30-216	30-220	30-222
	31-28	32-156	32-189	32-199	32-205	32-207	32-209	32-221	32-225	32-227	32-237	32-241	32-243	32-253
	32-255	32-257	33-34	33-40	33-44	33-46	33-56	33-60	33-61*	33-62	33-72	33-76	33-78	46-13
RMDSO	6-0#													
RMDT	4-696#	11-50	12-67											
RMDT1	6-0#	12-67*	12-69	12-71	12-74	12-76	12-79	12-81	46-5	46-17				
RMDTO	6-0#													
RMEC1	4-703#													
RMEC11	6-0#	46-14												
RMEC10	6-0#													
RMEC2	4-704#	16-15												
RMEC21	6-0#	12-172	16-14	28-94	31-94	46-15								
RMEC20	6-0#													
RMER1	4-692#	12-112	12-216	12-480	12-581	12-734	12-<49	12-B33	12-F71	21-145				
RMER11	6-0#	12-149	12-286	12-288	12-293	12-295	12-:17	12-:50	12-<16	12-=06	12-=09	12-=35	12-=38	12-=68
	21-137	21-152	21-154	22-80	22-156	22-183	22-226	22-241	22-284	22-361	22-387	22-402	22-417	22-470
	22-489	22-492	23-89	23-105	23-120	23-132	24-24	26-30	26-37	26-66	26-140	26-143	26-164	28-71
	29-51	29-55	29-57	29-58	29-69	29-71	29-72	29-83	29-85	29-86	29-97	29-99	29-100	29-111
	29-113	29-114	30-24	30-29	30-33	30-35	30-45	30-47	30-49	30-61	30-63	30-65	30-78	30-80
	30-82	30-155	30-232	30-236	30-238	30-240	30-250	30-252	30-254	30-264	30-268	30-270	31-41	32-50
	32-56	32-58	32-107	32-182	32-184	32-186	32-210	33-42	46-13					
RMER10	6-0#	12-104*	12-209*	12-480*	12-575*	12-734*	12-<49*	12-B26*	12-F64*	21-141				
RMER2	4-702#	12-113	12-217	12-580										
RMER21	6-0#	12-184	12-280	12-282	12-307	12-309	12-364	12-366	12-370	12-372	12-404	12-406	12-410	12-412
	12-426	12-428	12-515	12-:84	12-B10	12-B13	12-F48	12-F51	15-55	21-139	22-82	22-200	22-256	22-432
	24-27	26-32	26-85	26-88	26-99	26-127	26-130	26-179	26-208	28-116	30-31	30-92	30-99	30-101
	30-103	30-116	30-118	30-120	30-130	30-132	30-134	30-171	30-218	30-266	31-103	32-52	32-125	32-129
	32-149	32-151	32-153	32-166	32-168	32-170	32-223	32-239	33-58	33-74	33-88	33-94	33-96	33-98
	33-108	33-110	33-112	46-13										
RMER20	6-0#	12-105*	12-210*	12-576*	12-577*									
RMHR	4-700#													
RMHR1	6-0#													
RMHRO	6-0#													
RMLA	4-694#	12-C21	12-C22	12-C31										
RMLA1	6-0#	46-16												
RMLAO	6-0#													
RMMR1	4-695#	12-114	12-218	12-336	12-377	12-438	12-440	12-504	12-506	12-986	12-988	12-:52	12-:54	12-:19
	12-:21	12-:85	12-:87	12-<88	12-<90	12-<97	12-A92	12-A99	12-F26	12-F33				
RMMR11	6-0#	12-164	12-340	12-343	12-380	12-383	28-81	31-69	46-17					
RMMR10	6-0#	12-106*	12-211*	12-336*	12-377*	12-438*	12-440*	12-504*	12-506*	12-986*	12-988*	12-:52*	12-:54*	12-:19*
	12-:21*	12-:85*	12-:87*	12-<88*	12-<90*	12-<97*	12-A92*	12-A95*	12-F26*	12-F27*				
RMMR2	4-701#													
RMMR21	6-0#	12-176	28-103	31-81	46-17									
RMMR20	6-0#													
RMOF	4-698#	12-908	12-:04	12-:71	12-:37	12-<03	12-<61	12-a50	12-a99	12-A51	12-C07	12-C84	12-D35	12-D93
	12-E40	12-E85	12-F36	12-G10										
RMOF1	6-0#	12-922	22-377	22-487	46-14	46-16								
RMOFO	6-0#	12-913*	12-:00*	12-:67*	12-:33*	12-:99*	12-<57*	12-a46*	12-a82	12-a84*	12-a97*	12-A31	12-A33*	12-A47*
	12-A80	12-A82*	12-C00*	12-C42	12-C62	12-C64*	12-C79*	12-D15	12-D16*	12-D28*	12-D73	12-D75*	12-D87*	12-E21
	12-E23*	12-E36*	12-E67	12-E69*	12-E83*	12-F14	12-F16*	12-F30*	12-G05*	26-58	32-99	46-11		
RMR	4-578#	12-=06	12-=08	29-51	29-83	29-87	30-232	30-250	30-253					
RMSN	4-697#													
RMSNI	6-0#	46-17												
RMSNO	6-0#													
RMWC	4-764#	12-11*	12-12	12-:03	12-:70	12-:36	12-<02	12-<60						

RMWCI 6-0# 23-10 23-23 23-44 23-115 46-14

M 10

SEQ 0335

7

RMWCO	6-0#	12-997*	12-:64*	12-:30*	12-:96*	12-<54*	23-24	23-45						
RQA	4-661#	12-177	28-104	31-82										
RQB	4-662#	12-177	28-104	31-82										
RTC	4-506#	12-874												
SA1	4-543#													
SA16	4-539#													
SA2	4-542#													
SA4	4-541#													
SAB	4-540#													
SADMSK	4-547#	23-68												
SAVREG	14-16	35-10#												
SC	4-715#	12-128	21-92	21-109	21-125	28-32								
SCO	4-596#													
SC1	4-595#													
SC2	4-594#													
SC3	4-593#													
SC4	4-592#													
SCHSTS	12-D05	12-D61	12-E09	12-E55	12-F02	12-G29	32-40#							
SCOPE	4-483#	12-2	12-34	12-63	12-91	12-196	12-238	12-263	12-314	12-332	12-434	12-459	12-476	12-500
	12-522	12-547	12-569	12-599	12-617	12-648	12-684	12-719	12-751	12-816	12-870	12-901	12-945	12-974
	12-:40	12-:07	12-:73	12-<39	12-<81	12-=13	12-=46	12->13	12->41	12->69	12-?03	12-?37	12-?72	12-@06
	12-@40	12-@89	12-A40	12-A88	12-B22	12-B59	12-B95	12-C72	12-D23	12-D83	12-E28	12-E76	12-F22	12-F60
	12-F99	12-G43	13-9											
SCTMSK	4-598#													
SEARCH	4-512#	12-C03	12-C96	12-D52	12-D90	12-E32	12-E80	12-F31	12-F65	12-G20	22-104	23-5		
SECERR	12-235	12-329	12-456	12-473	12-497	12-519	12-544	12-566	12-596	12-614	12-645	12-681	12-748	12-810
	12-898	12-942	12->38	12->66	12->95	12-?30	12-?64	12-?98	12-@32	12-@76	12-A25	12-A74	12-B19	12-B56
	12-B92	12-D09	12-D65	12-E15	12-E61	12-F08	12-F57	12-F96	12-G33	12-G68	22-24#			
SEEK	4-501#	12->20	12->48	12->76	12-?11	12-?44	12-?79	12-@13	12-@45	12-@93	12-A44	12-A94	12-B29	12-B76
	12-C80	12-D32	12-G50											
SEKSTS	12->34	12->62	12->91	12-?26	12-?60	12-?94	12-@28	12-@71	12-A20	12-A70	12-B88	12-C95	12-D49	12-G64
	26-19#													
SHUT	34-13#	35-8	35-8											
SIZCLK	11-21	19-3#												
SKI	4-674#	12-185	12-307	12-364	12-367	12-404	12-407	12-408	15-175	26-48	26-51	26-70	26-100	26-102
	26-106	26-117	26-179	28-117	30-92	30-116	30-119	30-218	32-89	32-92	32-111	32-125	32-130	32-132
	32-137	32-223	33-74	33-89	33-108	33-111								
SNGPRT	4-644#	12-69	12-74	12-79	12-84									
STACK	4-483#	9-19	12-2	12-34	12-63	12-91	12-196	12-238	12-263	12-314	12-332	12-434	12-459	12-476
	12-500	12-522	12-547	12-569	12-599	12-617	12-648	12-684	12-719	12-751	12-816	12-870	12-901	12-945
	12-974	12-:40	12-:07	12-:73	12-<39	12-<81	12-=13	12-=46	12->13	12->41	12->69	12-?03	12-?37	12-?72
	12-@06	12-@40	12-@89	12-A40	12-A88	12-B22	12-B59	12-B95	12-C72	12-D23	12-D83	12-E28	12-E76	12-F22
	12-F60	12-F99	12-G43											
STANDA	9-63	10-3#												
START	4-776	9-14#	11-34	12-29	34-16									
STCDRV	12-495	12-612	12-643	12-679	12-746	12-808	12-896	12-940	12-B52	12-E11	12-E57	12-F04	12-F92	33-27#
STKLMT	4-483#													
STOP	34-3#	35-6												
STSD1	42-1	42-2	42-8	42-10	42-11	42-12	46-13#							
STSD2	42-1	42-2	42-8	42-10	42-11	42-12	46-14#							
STSD3	46-16#													
STSD4	42-1	42-2	42-8	42-10	42-11	42-12	46-17#							
STSF	43-1	43-1	43-1	43-2	43-2	43-2	43-8	43-8	43-8	43-10	43-10	43-10	43-11	43-11
	43-11	43-12	43-12	43-12	47-7#									
STSM1	41-1	41-2	41-8	41-9	41-11	41-12	41-13	45-21#						
STSM2	41-1	41-2	41-8	41-9	41-11	41-12	41-13	45-23#						

STSH3 45-25#

B 11

SEQ 0337

|

TST16	12-522#	35-6												
TST17	12-547#	35-6												
TST2	12-34#	35-6												
TST20	12-569#	35-6												
TST21	12-599#	35-6												
TST22	12-617#	35-6												
TST23	12-648#	35-6												
TST24	12-684#	35-6												
TST25	12-719#	35-6												
TST26	12-751#	35-6												
TST27	12-816#	35-6												
TST3	12-63#	35-6												
TST30	12-870#	35-6												
TST31	12-901#	35-6												
TST32	12-945#	35-6												
TST33	12-974#	35-6												
TST34	12-:40#	35-6												
TST35	12-:07#	35-6												
TST36	12-:73#	35-6												
TST37	12-<39#	35-6												
TST4	12-91#	35-6												
TST40	12-<81#	35-6												
TST41	12-=13#	35-6												
TST42	12-=46#	35-6												
TST43	12->13#	35-6												
TST44	12->41#	35-6												
TST45	12->69#	35-6												
TST46	12-?03#	35-6												
TST47	12-?37#	35-6												
TST5	12-196#	35-6												
TST50	12-?72#	35-6												
TST51	12-@06#	35-6												
TST52	12-@40#	35-6												
TST53	12-@89#	35-6												
TST54	12-A40#	35-6												
TST55	12-A88#	35-6												
TST56	12-B22#	35-6												
TST57	12-B59#	35-6												
TST6	12-238#	35-6												
TST60	12-B95#	35-6												
TST61	12-C72#	35-6												
TST62	12-D23#	35-6												
TST63	12-D83#	35-6												
TST64	12-E28#	35-6												
TST65	12-E76#	35-6												
TST66	12-F22#	35-6												
TST67	12-F60#	35-6												
TST7	12-263#	35-6												
TST70	12-F99#	35-6												
TST71	12-G43#	35-6												
TSTNMB	14-24*	14-25*	14-27	14-132#										
TSTPRP	12-461	12-478	12-502	12-524	12-549	12-571	12-601	12-619	12-650	12-686	12-721	12-753	12-818	12-872
	12-903	12-947	12-976	12-:42	12-:09	12-:75	12-<41	12-<83	12-=15	12-=48	12->16	12->44	12->73	12-?06
	12-?40	12-?75	12-@09	12-@43	12-@91	12-A42	12-B24	12-B62	12-C74	12-D25	12-D85	12-E30	12-E78	12-F24
	12-F62	12-G01	12-G46	15-38#										
TSTQUE	6-0#	11-4	11-5*	11-42	11-49	12-2	12-34	12-63	12-91	12-196	12-238	12-263	12-314	12-332

12-434 12-459 12-476 12-500 12-522 12-547 12-569^{F 11} 12-599 12-617 12-648 12-684 12-719 12-751 12-816
SEQ 0341

SSCMRE	4-783#													
SSCMTM	4-783#	5-0	5-0	5-0	5-0	5-0								
SSESCA	4-483#													
SSNEWT	4-483#	12-2	12-34	12-63	12-91	12-196	12-238	12-263	12-314	12-332	12-434	12-459	12-476	12-500
	12-522	12-547	12-569	12-599	12-617	12-648	12-684	12-719	12-751	12-816	12-870	12-901	12-945	12-974
	12-:40	12-:07	12-:73	12-<39	12-<81	12-=13	12-=46	12->13	12->41	12->69	12-?03	12-?37	12-?72	12-a06
	12-a40	12-a89	12-A40	12-A88	12-B22	12-B59	12-B95	12-C72	12-D23	12-D83	12-E28	12-E76	12-F22	12-F60
	12-F99	12-G43												
SSSET	35-10	35-10	35-10	35-10	35-10	35-10	35-10	35-10	35-10	35-10	35-10	35-10	35-10	35-10#
SSSETM	9-19	9-19#												
SSSKIP	4-483#													
.SACT1	4-475#	4-778												
.SAPT8	4-475#	5-0	5-0#											
.SAPTH	4-475#	4-781												
.SAPTY	4-475#	35-12												
.SCATC	4-471#	4-776												
.SCMTA	4-472#	4-783												
.SEOP	4-472#	13-20												
.SERRO	4-472#	35-7												
.SERRT	4-472#													
.SPOWE	4-474#	35-11												
.SRDDE	4-473#													
.SRDOC	4-473#	35-9												
.SREAD	4-473#	35-8												
.SSAVE	4-474#	35-1												
.SSCOP	4-472#	35-6												
.SSIZE	4-474#													
.STRAP	4-474#	35-10												
.STYPB	4-473#	35-2												
.STYPD	4-473#	35-3												
.STYPE	4-472#	35-5												
.STYPO	4-473#	35-4												
.EQUAT	4-471#	4-483												
.HEADE	4-471#	4-479												
.SETUP	4-471#	4-774												
.SWRHI	4-471#	4-480												
.SWRLO	4-471#	4-480#	4-481											
CALCLR	4-186#	12-36	12-65	12-93	12-198	12-226	12-240	12-265	12-316	12-334	12-436	12-981	12-:47	12-:14
	12-:80	12-=58	12-a57	12-A06	12-A56	12-A90	12-B97	12-D99	12-E45	12-E92				
CALSUB	4-200#	12-100	12-118	12-124	12-205	12-222	12-229	12-231	12-235	12-248	12-269	12-323	12-326	12-329
	12-339	12-379	12-425	12-446	12-449	12-452	12-456	12-461	12-467	12-469	12-471	12-473	12-478	12-485
	12-491	12-493	12-495	12-497	12-502	12-512	12-514	12-519	12-524	12-534	12-538	12-540	12-542	12-544
	12-549	12-556	12-558	12-560	12-562	12-564	12-566	12-571	12-584	12-589	12-592	12-594	12-596	12-601
	12-606	12-608	12-610	12-612	12-614	12-619	12-625	12-627	12-641	12-643	12-645	12-650	12-655	12-663
	12-667	12-669	12-677	12-679	12-681	12-686	12-691	12-704	12-706	12-708	12-716	12-721	12-726	12-738
	12-746	12-748	12-753	12-777	12-792	12-794	12-802	12-804	12-806	12-808	12-810	12-818	12-863	12-872
	12-878	12-880	12-894	12-896	12-898	12-903	12-915	12-919	12-921	12-938	12-940	12-942	12-947	12-952
	12-954	12-965	12-976	12-:09	12-:12	12-:14	12-:42	12-:76	12-:79	12-:81	12-:09	12-:42	12-:45	12-:47
	12-:75	12-<08	12-<11	12-<13	12-<41	12-<44	12-<47	12-<66	12-<69	12-<71	12-<73	12-<83	12-<86	12-<94
	12-=03	12-=15	12-=19	12-=31	12-=33	12-=48	12-=62	12-=64	12-=66	12->16	12->26	12->30	12->32	12->34
	12->36	12->38	12->44	12->54	12->58	12->60	12->62	12->64	12->66	12->73	12->82	12->87	12->89	12->91
	12->93	12->95	12-?06	12-?18	12-?22	12-?24	12-?26	12-?28	12-?30	12-?40	12-?51	12-?56	12-?58	12-?60
	12-?62	12-?64	12-?75	12-?86	12-?90	12-?92	12-?94	12-?96	12-?98	12-a09	12-a20	12-a24	12-a26	12-a28
	12-a30	12-a32	12-a43	12-a59	12-a61	12-a63	12-a67	12-a69	12-a71	12-a74	12-a76	12-a91	12-A08	12-A10
	12-A12	12-A16	12-A18	12-A20	12-A23	12-A25	12-A42	12-A58	12-A60	12-A62	12-A66	12-A68	12-A70	12-A72
	12-A74	12-B05	12-B07	12-B09	12-B17	12-B19	12-B24	12-B37	12-B39	12-B48	12-B50	12-B52	12-B54	12-B56

	12-B62	12-B72	12-B79	12-B82	12-B84	12-B86	12-B88	12-B90	12-B92	12-C12	12-C15	12-C17	12-C74	12-C88
	12-C91	12-C93	12-C95	12-C97	12-D01	12-D03	12-D05	12-D07	12-D09	12-D25	12-D41	12-D45	12-D47	12-D49
	12-D53	12-D57	12-D59	12-D61	12-D63	12-D65	12-D85	12-E01	12-E05	12-E07	12-E09	12-E11	12-E13	12-E15
	12-E30	12-E47	12-E51	12-E53	12-E55	12-E57	12-E59	12-E61	12-E78	12-E94	12-E98	12-F00	12-F02	12-F04
	12-F06	12-F08	12-F24	12-F40	12-F44	12-F46	12-F55	12-F57	12-F62	12-F75	12-F77	12-F88	12-F90	12-F92
	12-F94	12-F96	12-G01	12-G13	12-G21	12-G25	12-G27	12-G29	12-G31	12-G33	12-G46	12-G56	12-G60	12-G62
	12-G64	12-G66	12-G68											
COMMEN	4-483#													
ENDCOM	4-483#													
ERROR	4-12#	11-23	12-26	12-36	12-49	12-53	12-59	12-65	12-67	12-86	12-93	12-100	12-118	12-124
	12-132	12-137	12-146	12-152	12-162	12-169	12-174	12-181	12-187	12-193	12-198	12-205	12-222	12-226
	12-229	12-231	12-235	12-240	12-242	12-248	12-254	12-259	12-265	12-269	12-277	12-284	12-290	12-297
	12-305	12-311	12-316	12-318	12-323	12-326	12-329	12-334	12-336	12-339	12-344	12-351	12-357	12-363
	12-369	12-375	12-377	12-379	12-384	12-391	12-397	12-403	12-409	12-415	12-421	12-423	12-425	12-431
	12-436	12-438	12-440	12-442	12-446	12-449	12-452	12-456	12-461	12-463	12-467	12-469	12-471	12-473
	12-478	12-480	12-482	12-485	12-488	12-491	12-493	12-495	12-497	12-502	12-504	12-506	12-508	12-512
	12-514	12-517	12-519	12-524	12-534	12-538	12-540	12-542	12-544	12-549	12-551	12-553	12-560	12-562
	12-564	12-566	12-571	12-584	12-589	12-592	12-594	12-596	12-601	12-603	12-606	12-608	12-610	12-612
	12-614	12-619	12-621	12-625	12-627	12-633	12-639	12-641	12-643	12-645	12-650	12-652	12-655	12-661
	12-663	12-665	12-667	12-669	12-675	12-677	12-679	12-681	12-686	12-688	12-691	12-697	12-704	12-706
	12-708	12-714	12-716	12-721	12-723	12-726	12-732	12-734	12-736	12-738	12-744	12-746	12-748	12-753
	12-777	12-779	12-792	12-794	12-795	12-802	12-804	12-806	12-808	12-810	12-818	12-840	12-849	12-863
	12-864	12-872	12-874	12-878	12-880	12-886	12-892	12-894	12-896	12-898	12-903	12-915	12-919	12-921
	12-926	12-931	12-936	12-938	12-940	12-942	12-947	12-949	12-952	12-954	12-960	12-963	12-965	12-971
	12-976	12-981	12-986	12-988	12-992	12-:09	12-:12	12-:14	12-:21	12-:32	12-:42	12-:47	12-:52	12-:54
	12-:76	12-:79	12-:81	12-:88	12-:99	12-:09	12-:14	12-:19	12-:21	12-:25	12-:42	12-:45	12-:47	12-:54
	12-:65	12-:75	12-:80	12-:85	12-:87	12-:91	12-<08	12-<11	12-<13	12-<20	12-<31	12-<41	12-<47	12-<49
	12-<66	12-<71	12-<73	12-<83	12-<88	12-<90	12-<92	12-<94	12-<97	12-<99	12-=01	12-=03	12-=10	12-=15
	12-=19	12-=31	12-=33	12-=40	12-=48	12-=58	12-=62	12-=64	12-=66	12-=73	12->16	12->26	12->30	12->32
	12->34	12->36	12->38	12->44	12->54	12->58	12->60	12->62	12->64	12->66	12->73	12->82	12->87	12->89
	12->91	12->93	12->95	12-?06	12-?18	12-?22	12-?24	12-?26	12-?28	12-?30	12-?40	12-?51	12-?56	12-?58
	12-?60	12-?62	12-?64	12-?75	12-?86	12-?90	12-?92	12-?94	12-?96	12-?98	12-a09	12-a20	12-a24	12-a26
	12-a28	12-a30	12-a32	12-a43	12-a57	12-a59	12-a61	12-a63	12-a67	12-a69	12-a71	12-a74	12-a76	12-a91
	12-A06	12-A08	12-A10	12-A12	12-A16	12-A18	12-A20	12-A23	12-A25	12-A42	12-A56	12-A58	12-A60	12-A62
	12-A66	12-A68	12-A70	12-A72	12-A74	12-A90	12-A92	12-B05	12-B07	12-B09	12-B15	12-B17	12-B19	12-B24
	12-B37	12-B39	12-B45	12-B48	12-B50	12-B52	12-B54	12-B56	12-B62	12-B72	12-B79	12-B84	12-B86	12-B88
	12-B90	12-B92	12-B97	12-C12	12-C15	12-C17	12-C35	12-C57	12-C74	12-C88	12-C91	12-C93	12-C95	12-C97
	12-D01	12-D03	12-D05	12-D07	12-D09	12-D25	12-D41	12-D45	12-D47	12-D49	12-D53	12-D57	12-D59	12-D61
	12-D63	12-D65	12-D85	12-D99	12-E01	12-E05	12-E07	12-E09	12-E11	12-E13	12-E15	12-E30	12-E45	12-E47
	12-E51	12-E53	12-E55	12-E57	12-E59	12-E61	12-E78	12-E92	12-E94	12-E98	12-F00	12-F02	12-F04	12-F06
	12-F08	12-F24	12-F26	12-F40	12-F44	12-F46	12-F53	12-F55	12-F57	12-F62	12-F75	12-F77	12-F84	12-F88
	12-F90	12-F92	12-F94	12-F96	12-G01	12-G13	12-G21	12-G25	12-G27	12-G29	12-G31	12-G33	12-G46	12-G56
	12-G60	12-G62	12-G64	12-G66	12-G68									
ESCAPE	4-483#													
GETPRI	4-483#													
GETREG	4-163#	12-67												
GETSWR	4-483#	9-25	9-25#	11-27										
MSG	4-8#	12-G36#	12-G43	12-G71#										
MULT	4-483#													
NEWTST	4-483#	12-2	12-34	12-63	12-91	12-196	12-238	12-263	12-314	12-332	12-434	12-459	12-476	12-500
	12-522	12-547	12-569	12-599	12-617	12-648	12-684	12-719	12-751	12-816	12-870	12-901	12-945	12-974
	12-:40	12-:07	12-:73	12-<39	12-<81	12-=13	12-=46	12->13	12->41	12->69	12-?03	12-?37	12-?72	12-a06
	12-a40	12-a89	12-A40	12-A88	12-B22	12-B59	12-B95	12-C72	12-D23	12-D83	12-E28	12-E76	12-F22	12-F60
	12-F99	12-G43												
NWTST	4-44#	12-2	12-34	12-63	12-91	12-196	12-238	12-263	12-314	12-332	12-434	12-459	12-476	12-500
	12-522	12-547	12-569	12-599	12-617	12-648	12-684	12-719	12-751	12-816	12-870	12-901	12-945	12-974

12-:40 12-;07 12-;73 12-<39 12-<81 12-=13 12-=46^{J 11} 12->13 12->41 12->69 12-?03 12-?37 12-?72 12-@06
SEQ 0345

	12-240	12-289	12-A40	12-A88	12-B22	12-B59	12-B95	12-C72	12-D23	12-D83	12-E28	12-E76	12-F22	12-F60
POP	12-F99	12-G43												
	4-483#	12-19	12-20	12-24	12-25	12-45	12-46	12-57	12-58	12->10	12->11	12-C68	12-C69	16-25
	16-26	16-27	17-70	18-59	19-18	20-34	21-147	21-151	22-131	25-51	27-25	28-60	31-58	31-59
PUSH	35-1	35-3	35-9	35-11	35-11	35-12	35-12							
	4-483#	12-5	12-6	12-38	12-39	12-769	12-770	12-771	12-772	12-828	12-829	12-830	12-835	12-836
	12-837	12-=52	12-=53	16-10	16-11	16-12	17-35	18-29	19-3	20-8	21-143	22-128	25-20	27-12
PUTREG	28-54	31-51	31-52	35-1	35-3	35-9	35-11	35-11	35-12	35-12	35-12			
	4-441#	12-242	12-318	12-336	12-377	12-423	12-438	12-440	12-442	12-463	12-480	12-482	12-504	12-506
	12-508	12-551	12-553	12-603	12-621	12-652	12-665	12-688	12-723	12-734	12-736	12-779	12-840	12-849
	12-874	12-949	12-963	12-986	12-988	12-992	12-:52	12-:54	12-:19	12-:21	12-:25	12-:85	12-:87	12-:91
	12-<49	12-<88	12-<90	12-<92	12-<97	12-<99	12-=01	12-A92	12-F26					
REPORT	4-483#													
RGBFMC	4-77#	6-0	6-0											
SETPRI	4-483#	9-23	11-41	34-6	34-10	35-8								
SETTRA	35-10	35-10	35-10	35-10	35-10	35-10	35-10	35-10	35-10	35-10	35-10	35-10	35-10	35-10#
SETUP	4-483#	9-19												
SKIP	4-483#													
SLASH	4-483#													
STARS	4-483#	4-778	4-781	4-781	4-781	5-0	5-0	5-0	12-2	12-2	12-34	12-34	12-63	12-63
	12-91	12-91	12-196	12-196	12-238	12-238	12-263	12-263	12-314	12-314	12-332	12-332	12-434	12-434
	12-459	12-459	12-476	12-476	12-500	12-500	12-522	12-522	12-547	12-547	12-569	12-569	12-599	12-599
	12-617	12-617	12-648	12-648	12-684	12-684	12-719	12-719	12-751	12-751	12-816	12-816	12-870	12-870
	12-901	12-901	12-945	12-945	12-974	12-974	12-:40	12-:40	12-:07	12-:07	12-:73	12-:73	12-<39	12-<39
	12-<81	12-<81	12-=13	12-=13	12-=46	12-=46	12->13	12->13	12->41	12->41	12->69	12->69	12-?03	12-?03
	12-?37	12-?37	12-?72	12-?72	12-a06	12-a06	12-a40	12-a40	12-a89	12-a89	12-A40	12-A40	12-A88	12-A88
	12-B22	12-B22	12-B59	12-B59	12-B95	12-B95	12-C72	12-C72	12-D23	12-D23	12-D83	12-D83	12-E28	12-E28
	12-E76	12-E76	12-F22	12-F22	12-F60	12-F60	12-F99	12-F99	12-G43	12-G43	13-20	14-2	15-57	15-70
	15-84	15-107	15-138	15-160	15-193	21-2	22-26	22-34	22-119	23-1	23-3	28-2	35-1	35-2
	35-3	35-4	35-5	35-6	35-7	35-8	35-8	35-8	35-8	35-8	35-9	35-10	35-11	35-11
	35-12													
SWRSU	4-483#	9-19	9-19#											
TAGS	4-111#	5-0												
TRMTRP	35-10#													
TYPBIN	4-483#													
TYPDEC	4-483#	13-20	13-20											
TYPNAM	4-471#	4-483#	9-25											
TYPNUM	4-483#													
TYPOCS	4-483#	9-99	14-23	14-27	14-32	14-34								
TYPOCT	4-483#	10-56	35-8											
TYPTXT	4-483#	9-6	9-39	9-50	9-56	9-57	11-24	13-20	13-20	34-15				
XPER	4-20#	7-4	7-7	7-10	7-13	7-16	7-19	7-22	7-25	7-28	7-31	7-34	7-37	7-40
	7-43	7-46	7-49	7-52	7-55	7-58	7-61	7-64	7-67	7-70	7-73	7-76	7-79	7-82
	7-85	7-88	7-91	7-94	7-97	7-100	7-103	7-106	7-109	7-112	7-115	7-118	7-121	7-124
	7-127	7-131	7-134	7-137	7-140	7-143	7-147	7-151	7-155	7-158	7-161	7-164	7-167	7-170
	7-173	7-176	7-179	7-183	7-186	7-189	7-192	7-195	7-198	7-201	7-204	7-207	7-210	7-213
	7-216	7-219	7-222	7-225	7-228	7-231	7-234	7-237	7-240	7-243	7-246	7-249	7-252	7-255
	7-258	7-261	7-264	7-267	7-270	7-273	7-276	7-279	7-282	7-285	7-288	7-291	7-294	7-297
	7-300	7-303	7-306	7-309	7-312	7-315	7-318	7-321	7-324	7-327	7-330	7-333	7-336	7-339
	7-342	7-345	7-348	7-352	7-355	7-358	7-361	7-364	7-367	7-370	7-373	7-376	7-379	7-382
	7-385	7-388	7-391	7-394	7-397	7-400	7-403	7-406	7-409	7-412	7-415	7-418	7-421	7-424
	7-427	7-430	7-433	7-436	7-439	7-442	7-445	7-448	7-451	7-454	7-457	7-460	7-463	7-466
	7-469	7-472	7-475	7-478	7-481	7-484	7-487	7-490	7-493	7-496	7-499	7-502	7-505	7-508
	7-511	7-514	7-517	7-520	7-523	7-526	7-529	7-532	7-535	7-538	7-541	7-544	7-547	7-550
	7-553	7-557	7-560	7-564	7-567	7-570	7-574	7-578	7-582	7-587	7-591	7-595	7-598	7-601
	7-604	7-607	7-610	7-613	7-616	7-619	7-622	7-625	7-628	7-631	7-634	7-637	7-640	7-643

7-646 7-649 7-652 7-655 7-658 7-661 7-664^{L 11} 7-667 7-670 7-673 7-676 7-679 7-682 7-685
SEQ 0347

7-688 7-691 7-694 7-697 7-700 7-703 7-706 7-709 7-712 7-715 7-718 7-721 7-724