

RM02/03/05

RM05/3/2 FCTNL TST 1
CZRMMBO

AH-F922B-MC
FICHE 1 OF 2

AUG 1981
COPYRIGHT © 80-81
MADE IN USA



The main body of the document is a large, dense grid of data. Each cell in the grid contains a small table or set of data points. The text is extremely faint and difficult to read, but the overall structure is a regular grid of approximately 15 columns and 25 rows of data blocks. Each data block appears to have a header section followed by several rows of numerical or alphanumeric values.

RM02/03/05

RM05/3/2 FCTNL TST 1
CZRMMBO

AH-F922B-MC
FICHE 2 OF 2

AUG 1981
COPYRIGHT © 80-81
MADE IN USA



.REM \

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

IDENTIFICATION

PRODUCT CODE: AC-F921B-MC
PRODUCT NAME: CZRMMB0 RM05/3/2 FUNCTIONAL TEST, PT 1
PRODUCT DATE: APRIL 1981
MAINTAINER: CX DIAGNOSTIC GROUP
AUTHOR: MIKE LEAVITT

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

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

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

COPYRIGHT (C) 1980,1981 DIGITAL EQUIPMENT CORPORATION

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

CONTENTS

- 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
 - 7. EXECUTION TIME
- 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
- 20K 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 TEST, 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 MAY BE STARTED AT LOCATION 200 OR 204. STARTING AT 200 WILL BE THE NORMAL STARTING ADDRESS. STARTING AT 204 WILL ENABLE THE RH/RM BASE ADDRESS TO BE CHANGED. IF RUNNING IN A STAND-ALONE ENVIRONMENT, THE PROGRAM USES CONSOLE DIALOGUE TO ALLOW THE OPERATOR TO CONTROL TEST CONDITIONS.

3.4 HALTING

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

THE PROGRAM CAN BE HALTED BY TYPING CONTROL C FROM THE CONSOLE OR BY PRESSING THE HALT SWITCH ON THE PROCESSOR FRONT PANEL.

3.5 RESTARTING

THE PROGRAM CAN BE RESTARTED AT ADDRESS 200 OR 204. (SEE SECTION 3.3)

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 (L) N ?". IF THE OPERATOR RESPONDS WITH A 'Y', THE PROGRAM WILL TYPE A BRIEF HELP MESSAGE WHICH WILL LIST SWITCH OPTIONS, ETC. ANY OTHER RESPONSE TO THE QUESTION IS CONSIDERED A 'N' AND NO HELP TEXT IS TYPED. THIS QUESTION IS ONLY ASKED ON THE INITIAL PROGRAM START AND NOT ON SUBSEQUENT START-UP'S.

ON THE PROGRAM INITIAL START AND WHEN RESTARTING AT LOCATION 204, THE OPERATOR MAY CHANGE THE RH/RM BASE ADDRESSES WITH THE FOLLOWING DIALOGUE.

EXAMPLE 1

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

EXAMPLE 2

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

ON THE INITIAL START, THE NEXT 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 TYPE ANY STRING OF DRIVE NUMBER(S) TO BE TESTED AND TERMINATE THE INPUT WITH A 'CARRIAGE RETURN'. NO COMMAS OR ANY OTHER SEPARATORS ARE NEEDED WHEN ENTERING THE DRIVE NUMBERS AS A STRING. THE PROGRAM ENTERS THE COMMA SEPARATOR AUTOMATICALLY AFTER TYPING EACH NUMBER. ON ALL SUBSEQUENT STARTS, ONLY THE 'DRIVE(S):' PROMPT IS TYPED.

172 THE DIAGNOSTIC THEN INITIALIZES AND REPORTS THE STATUS OF THE
173 DRIVES WHICH WERE PREVIOUSLY SPECIFIED FOR TESTING. THE FOLLOWING
174 IS AN EXAMPLE PRINTOUT:
175

176 'UNIT STATUS:
177 0 ONLINE RM03
178 1 LOAD DEVICE
179 2 OFFLINE RM05
180 3 NOT PRESENT
181 4 NOT PRESENT
182 5 NOT AN RM05/3/2
183 6 NOT PRESENT
184 7 NOT PRESENT'
185

186 THE ABOVE UNIT STATUS SHOWS THAT DRIVE 0 WILL BE TESTED, WHILE DRIVES
187 1 - 7 WILL NOT BE TESTED.
188

189 THE DIAGNOSTIC THEN TYPES THE FOLLOWING MESSAGE, BASED ON THE STATUS
190 OF THE DRIVE:
191

192 'DRIVE(S) TO BE TESTED, 0'
193

194 IF NO DRIVES ARE AVAILABLE FOR TESTING THE FOLLOWING MESSAGE WILL BE
195 TYPED TO THE OPERATOR:
196

197 'DRIVE(S) TO BE TESTED, NONE'
198

199 THE PROGRAM WILL THEN, EITHER START TESTING THE DRIVES AVAILABLE FOR
200 TESTING OR RETURN TO THE BEGINNING OF THE PROGRAM AND WAIT.
201

202 ONCE THE DRIVES START TESTING, THE FOLLOWING MESSAGE WILL OCCUR
203 AS EACH DRIVE BEGINS TO BE TESTED:
204

205 'DRIVE 0'
206

207 AFTER ALL THE DRIVES ARE COMPLETELY TESTED, THE END OF PASS
208 MESSAGE WILL BE TYPED (SEE SECTION 4.3) AND THE PROGRAM WILL START
209 TESTING ALL THE DRIVES AGAIN. THIS WILL CONTINUE UNTIL THE PROGRAM
210 IS HALTED BY THE OPERATOR.
211

212 NOTE: THE LETTER LOCATED WITHIN THE BRACKETS () INDICATES THE TYPE
213 OF RESPONSE REQUIRED BY THE USER, D=DECIMAL, O=OCTAL AND
214 L=LETTER.
215

216 4.3 PROGRESS REPORTS 217

218 AN END OF PASS REPORT OCCURS EACH TIME THE PROGRAM IS EXECUTED
219 FOR ALL DEVICES IN THE TEST QUE. THE END OF PASS REPORT IS AS FOLLOWS.
220

221 'END OF PASS 1'
222

223 THE FOLLOWING MESSAGE WILL ALSO OCCUR IF THERE WERE ERRORS SINCE
224 THE LAST END OF PASS REPORT.
225

226 'TOTAL ERRORS SINCE LAST REPORT 0'
227
228

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

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 (DRIVE) BEING TESTED, THE DRIVE TYPE, 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.

THE FOLLOWING PRINTOUT SHOWS A TYPICAL ERROR MESSAGE FOR THIS PROGRAM:

```

DRV# 0 - RM03, TEST# 14, ERR# 326, PC=016654
MASSBUS DATA BUS PARITY ERROR 'MDPE' (RMCS2, BIT 8) DETECTED
DURING WRITE COMMAND
EXPECTED   RECEVD
040300     040700
RMCS1      RMCS2   RMDS   RMER1   RMER2   RMAS
144252     040700  010700  000000  000000  00CJ00
RMWC       RMBA    RMDA   RMOF   RMDC   RMEC1  RMEC2
177403     104604  000002  010000  000000  004066  000000
RMMR1      RMMR2   RMDT   RMSN
000010     011777  024026  177777

```

4.7 EXECUTION TIME

PASS 1 OF THE PROGRAM TAKES ABOUT 20 SECONDS. PASS 2 AND SUBSEQUENT PASSES TAKE 1 MINUTES 40 SECONDS.

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.

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

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.

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

TEST 1 CONTROLLER ACCESS TEST

PURPOSE:

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

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 THE END OF PASS HANDLER.

TEST 2 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.

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

TEST 3 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.

TEST 4 ERROR CLEAR TEST

PURPOSE:

TO VERIFY THAT ALL APPLICABLE RH 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.

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

TEST 5 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;

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

TEST 6 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.

TEST 7 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.

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

TEST 10 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.

TEST 11 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.

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

TEST 12 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.

TEST 13 IVC RECALIBRATE TEST

PURPOSE:

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 RMD5 TO RESET. THE PROGRAM THEN EXECUTES A RECALIBRATE COMMAND AND VERIFIES THAT 'IVC' STATUS SETS AND THAT 'PIP' REMAINS INACTIVE.

TEST 14 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.

TEST 15 RECALIBRATE AT OFFSET

PURPOSE:

TO VERIFY THAT OFFSET MODE DOES NOT AFFECT RECALIBRATE.

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

PROCEDURE:

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

TEST 16 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:

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;

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

TEST 17 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.

TEST 20 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.

TEST 21 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.

TEST 22 WRITE ATA TEST

PURPOSE:

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

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.

TEST 23 ERROR/ATA TEST

PURPOSE:

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

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.

TEST 24 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.

TEST 25 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.

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:

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.

TEST 26 RETURN TO CENTERLINE TEST

PURPOSE:

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

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.

TEST 27 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;

TEST 30 RMDC CLEAR OFFSET TEST

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

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.

TEST 31 ILLEGAL FUNCTION TEST

PURPOSE:

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

TEST 32 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.

TEST 33 INVALID ADDRESS ERROR TEST

PURPOSE:

TO VERIFY IAE ERROR DETECTION.

PROCEDURE:

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

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

TEST 34 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.

TEST 35 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.

TEST 36 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.

TEST 37 PARITY ERROR TEST

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

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.

TEST 40 ILLEGAL REGISTER TEST

PURPOSE:

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

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 (41 - 67)

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.

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

TEST 41 SEEK TO FIRST CYLINDER TEST

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

TEST 42 SEEK TO LAST CYLINDER TEST

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

TEST 43 SEEK PRIME CYLINDERS TEST

THIS TEST SEEKS FORWARD TO EACH PRIME CYLINDER ADDRESS, I.E., CYLINDERS 1, 2, 4, 8, 16, 32, 64, 128, 256, AND 512.

TEST 44 SEEK ZERO DIFFERENCE TEST

THIS TEST EXECUTES SUCCESSIVE SEEKS TO CYLINDER 0.

TEST 45 SEEK MAXIMUM DIFFERENCE FORWARD

THIS TEST DOES A RECALIBRATE COMMAND, REGARDLESS OF THE CONDITION OF 'PIP' OR 'SKI', AND THEN EXECUTES A SEEK TO THE LAST CYLINDER.

TEST 46 SEEK ADJACENT FORWARD TEST

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

TEST 47 SEEK ADJACENT REVERSE TEST

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

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

TEST 50 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. THE TEST IS REPEATED FOR 16 BIT FORMAT.

TEST 51 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.

TEST 52 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.

TEST 53 IVC SEEK TEST

PURPOSE:

TO VERIFY THAT INVALID COMMAND STATUS, BIT 12 OF RMR2 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.

TEST 54 ABORT SEEK TEST

PURPOSE:

TO VERIFY THAT THE RM05/3/2 INHIBITS A SEEK WHEN AN ABORT

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

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.

TEST 55 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.

TEST 56 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.

TEST 57 SEARCH ON CYLINDER

PURPOSE:

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

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

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.

TEST 60 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 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.

TEST 61 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.

TEST 62 SEARCH INVALID TRACK

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

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.

TEST 63 SEARCH INVALID CYLINDER

PURPOSE:

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

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.

TEST 64 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.

TEST 65 ABORT SEARCH TEST

PURPOSE:

TO VERIFY THAT THE RM05/3/2 INHIBITS A SEARCH WHEN AN ABORT

1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350

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.

TEST 66 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.

TEST 67 HEAD ALIGNMENT SEEK

PURPOSE:

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.

PROCEDURE:

THE TEST EXECUTES A SEEK TO CYLINDER 245., TO POSITION THE HEADS OVER THE HEAD ALIGNMENT CYLINDER.

1
470
471

```

;*LAST REVISION 04-APR-81
.TITLE CZRMBO RM05/3/2 FCTNL TST 1
;*COPYRIGHT (C) 1981
;*DIGITAL EQUIPMENT CORPORATION
;*COLORADO SPGS., CO. 80919
;*
;*PROGRAM BY MIKE LEAVITT
;*
;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
;*PACKAGE (MAINDEC-11-DZQAC-C5), 18-MAR-81

```

472

```

.SBTTL OPERATIONAL SWITCH SETTINGS
;*
;*      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

```

473

```

.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

```

474
475

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

```

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

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

```

000000 NOP = 000000 ;NOP COMMAND
000002 ILF02 = 000002 ;ILLEGAL COMMAND
000004 SEEK = 000004 ;SEEK COMMAND
000006 RECAL = 000006 ;RECALIBRATE COMMAND
000010 DRVCLR = 000010 ;DRIVE CLEAR COMMAND
000012 RELEASE = 000012 ;RELEASE COMMAND
000014 OFFSET = 000014 ;OFFSET COMMAND
000016 RTC = 000016 ;RETURN TO CENTERLINE COMMAND
000020 RIP = 000020 ;READ IN PRESET COMMAND
000022 PAKACK = 000022 ;PACK ACKNOWLEDGE COMMAND
000022 PACACK = PAKACK
000024 ILF24 = 000024 ;ILLEGAL COMMAND
000026 ILF26 = 000026 ;ILLEGAL COMMAND
    
```

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	000030	SEARCH = 000030	:SEARCH COMMAND
507	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
508	000050	WCD = 000050	:WRITE CHECK DATA COMMAND
509	000052	WCH = 000052	:WRITE CHECK HEADER AND DATA
510	000054	ILF54 = 000054	:ILLEGAL COMMAND
511	000056	ILF56 = 000056	:ILLEGAL COMMAND
512	000060	WD = 000060	:WRITE DATA COMMAND
513	000062	WH = 000062	:WRITE HEADER AND DATA COMMAND
514	000064	ILF64 = 000064	:ILLEGAL COMMAND
515	000066	ILF66 = 000066	:ILLEGAL COMMAND
516	000070	RD = 000070	:READ DATA COMMAND
517	000072	RH = 000072	:READ HEADER AND DATA COMMAND
518	000074	ILF74 = 000074	:ILLEGAL COMMAND
519	000076	ILF76 = 000076	:ILLEGAL COMMAND
520			
521		:*RMDA DISK ADDRESS REGISTER	
522			
523		:TRACK ADDRESS DEFINITIONS	
524	010000	TA16 = BIT12	:TRACK ADDRESS 16.
525	004000	TA8 = BIT11	:TRACK ADDRESS 8.
526	002000	TA4 = BIT10	:TRACK ADDRESS 4.
527	001000	TA2 = BIT09	:TRACK ADDRESS 2.
528	000400	TA1 = BIT08	:TRACK ADDRESS 1.
529			
530		:SECTOR ADDRESS DEFINITIONS	
531	000020	SA16 = BIT04	:SECTOR ADDRESS 16.
532	000010	SA8 = BIT03	:SECTOR ADDRESS 8.
533	000004	SA4 = BIT02	:SECTOR ADDRESS 4.
534	000002	SA2 = BIT01	:SECTOR ADDRESS 2.
535	000001	SA1 = BIT00	:SECTOR ADDRESS 1.
536			
537		:TRACK & SECTOR MASKS	
538	177400	TADMSK = 177400	:TRACK ADDRESS MASK
539	000377	SADMSK = 000377	:SECTOR ADDRESS MASK
540			
541		:*RMDS DRIVE STATUS REGISTER	
542			
543	100000	ATA = BIT15	:ATTENTION ACTIVE
544	040000	ERR = BIT14	:COMPOSITE ERROR
545	020000	PIP = BIT13	:POSITIONING IN PROGRESS
546	010000	MOL = BIT12	:MEDIUM ON LINE
547	004000	WRL = BIT11	:WRITE LOCK
548	002000	LBT = BIT10	:LAST BLOCK TRANSFERRED
549	001000	PGM = BIT09	:PROGRAMMABLE
550	000400	DPR = BIT08	:DRIVE PRESENT
551	000200	DRY = BIT07	:DRIVE READY
552	000100	VV = BIT06	:VOLUME VALID
553	000001	OM = BIT00	:OFFSET MODE ACTIVE
554			
555		:*RMER1 ERROR REGISTER #1	


```

556
557      100000      DCK      = BIT15      ;DATA CHECK ERROR
558      040000      UNS      = BIT14      ;DRIVE UNSAFE
559      020000      OPI      = BIT13      ;OPERATION INCOMPLETE
560      010000      DTE      = BIT12      ;DRIVE TIMING ERROR
561      004000      WLE      = BIT11      ;WRITE LOCK ERROR
562      002000      IAE      = BIT10      ;INVALID ADDRESS ERROR
563      001000      AOE      = BIT09      ;ADDRESS OVERFLOW ERROR
564      000400      HCRC     = BIT08      ;HEADER CRC ERROR
565      000200      HCE      = BIT07      ;HEADER COMPARE ERROR
566      000100      ECH      = BIT06      ;ECC 'HARD' ERROR
567      000040      WCF      = BIT05      ;WRITE CLOCK FAILURE
568      000020      FER      = BIT04      ;FORMAT ERROR
569      000010      PAR      = BIT03      ;PARITY ERROR
570      000004      RMR      = BIT02      ;REGISTER MODIFICATION REFUSED
571      000002      ILR      = BIT01      ;ILLEGAL REGISTER
572      000001      ILF      = BIT00      ;ILLEGAL FUNCTION
573
574      115760      NDTMSK   = DCK!DTE!WLE!AOE!HCRC!HCE!ECH!WCF!FER
575      ;'NDTMSK' IS USED TO MASK ERROR REGISTER 1 DURING NON - DATA
576      ;COMMANDS, I.E., HOUSEKEEPING AND POSITIONING COMMANDS
577
578      ;*RMAS ATTENTION SUMMARY REGISTER
579
580      000377      ATNMSK   = 377      ;MASK FOR ATTENTION BITS
581
582      ;*RMLA LOOK AHEAD REGISTER
583
584      002000      SC4      = BIT10      ;SECTOR COUNT = 16
585      001000      SC3      = BIT09      ;SECTOR COUNT = 8
586      000400      SC2      = BIT08      ;SECTOR COUNT = 4
587      000200      SC1      = BIT07      ;SECTOR COUNT = 2
588      000100      SC0      = BIT06      ;SECTOR COUNT = 1
589
590      003700      SCTMSK   = 003700    ;SECTOR COUNT MASK
591
592      ;*RMMR1 MAINTENANCE REGISTER #1
593
594      ;WRITE ONLY BITS
595      100000      DBCK     = BIT15      ;DEBUG CLOCK
596      040000      DBEN     = BIT14      ;DEBUG CLOCK ENABLE
597      020000      DEBL     = BIT13      ;DIAGNOSTIC END OF BLOCK
598      010000      DTO      = BIT12      ;DIAGNOSTIC TIMEOUT
599      004000      MCLK     = BIT11      ;MAINTENANCE CLOCK
600      002000      MRD      = BIT10      ;READ DATA
601      001000      MUR      = BIT09      ;UNIT READY
602      000400      MOC      = BIT08      ;ON CYLINDER
603      000200      MSER     = BIT07      ;SEEK ERROR
604      000100      MDF      = BIT06      ;DRIVE FAULT
605      000040      MS       = BIT05      ;SECTOR PULSE
606      000010      MWP      = BIT03      ;WRITE PROTECT
607      000004      MI       = BIT02      ;INDEX PULSE
608      000002      MSC      = BIT01      ;SECTOR COMPARE
609      000001      DMD      = BIT00      ;DIAGNOSTIC MODE
610
611      ;READ ONLY BITS
612      100000      OCC      = BIT15      ;OCCUPIED

```

613	040000	RG	= BIT14	:RUN AND GO
614	020000	EFL	= BIT13	:END OF BLOCK
615	010000	REX	= BIT12	:EXCEPTION
616	004000	ESRC	= BIT11	:ENABLE SEARCH
617	002000	PLFS	= BIT10	:LOOKING FOR SYNC
618	001000	ECRC	= BIT09	:ENABLE CRC OUT
619	000400	PDA	= BIT08	:DATA AREA
620	000200	PHA	= BIT07	:HEADER AREA
621	000100	CONT	= BIT06	:CONTINUE
622	000040	WC	= BIT05	:WORD CLOCK
623	000020	EECC	= BIT04	:ENABLE ECC OUT
624	000010	MWD	= BIT03	:WRITE DATA BIT
625	000004	LS	= BIT02	:LAST SECTOR
626	000002	LST	= BIT01	:LAST SECTOR AND TRACK
627	000001	DMD	= BIT00	:DIAGNOSTIC MODE
628				
629		:*RMDT DRIVE TYPE REGISTER		
630				
631	100000	NSA	= BIT15	:NOT SECTOR ADDRESSED = 0
632	040000	TAP	= BIT14	:TAPE DRIVE = 0
633	020000	MOH	= BIT13	:MOVING HEAD = 1
634	004000	DRQ	= BIT11	:DRIVE REQUEST REQUIRED
635				
636	020024	SNGPRT	= 020024	:SINGLE PORT DRIVE TYPE (RM02)
637	024024	DULPRT	= 024024	:DUAL PORT DRIVE TYPE (RM02)
638				
639		:*RMOF OFFSET REGISTER		
640				
641	010000	FMT16	= BIT12	:16 BIT WORD FORMAT
642	004000	ECI	= BIT11	:ECC INHIBIT
643	002000	HCI	= BIT10	:HEADER COMPARE INHIBIT
644	000200	OFD	= BIT07	:OFFSET FORWARD
645				
646		:*RMDC DESIRED CYLINDER ADDRESS REGISTER		
647				
648	001777	CYLMSK	= 001777	:MASK FOR CYLINDER ADDRESS
649				
650		:*RMMR2 MAINTENANCE REGISTER #2		
651				
652		:READ ONLY BITS		
653	100000	RQA	= BIT15	:PORT A REQUEST
654	040000	RQB	= BIT14	:PORT B REQUEST
655	020000	TAG	= BIT13	:TAG CONTROL
656	010000	TST	= BIT12	:COMMAND SEQUENCE TEST BIT
657	004000	CC	= BIT11	:CONTROL OR CYLINDER TAG
658	002000	CH	= BIT10	:CONTROL OR HEAD TAG
661	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

```

663          ;*RMER2 ERROR REGISTER 2
664
665          100000      BSE      = BIT15      ;BAD SECTOR ERROR
666          040000      SKI      = BIT14      ;SEEK INCOMPLETE
667          020000      OPE      = BIT13      ;OPERATOR PLUG ERROR
668          010000      IVC      = BIT12      ;INVALID COMMAND ERROR
669          004000      LSC      = BIT11      ;LOSS OF SYSTEM CLOCK
670          002000      LBC      = BIT10      ;LOSS OF BIT CLOCK
671          000200      DVC      = BIT07      ;DEVICE CHECK
672          000010      DPE      = BIT03      ;DATA PARITY ERROR
673
674          .SBTTL      PROGRAM MNEMONICS
675
676          100000      MSE      = BIT15      ;MANUFACTURING DETECTED SECTOR ERROR
677          040000      USE      = BIT14      ;USER DETECTED SECTOR ERROR
678
679          .SBTTL      RM REGISTER INDEX VALUES
680
681          000000      RMCS1    = 00          ;CONTROL STATUS REGISTER #1
682          000006      RMDA     = 06          ;DISK ADDRESS REGISTER
683          000012      RMDS     = 12          ;DRIVE STATUS REGISTER
684          000014      RMER1    = 14          ;ERROR REGISTER #1
685          000016      RMAS     = 16          ;ATTENTION SUMMARY REGISTER
686          000020      RMLA     = 20          ;LOOK AHEAD REGISTER
687          000024      RMMR1    = 24          ;MAINTENANCE REGISTER
688          000026      RMDT     = 26          ;DRIVE TYPE REGISTER
689          000030      RMSN     = 30          ;SERIAL NUMBER REGISTER
690          000032      RMOF     = 32          ;OFFSET REGISTER
691          000034      RMDC     = 34          ;DESIRED CYLINDER REGISTER
692          000036      RMHR     = 36          ;HOLDING REGISTER
693          000040      RMMR2    = 40          ;MAINTENANCE REGISTER #2
694          000042      RMER2    = 42          ;ERROR REGISTER #2
695          000044      RMEC1    = 44          ;ECC POSITION REGISTER
696          000046      RMEC2    = 46          ;ECC PATTERN REGISTER
697          000050      ILRG50   = 50          ;ILLEGAL REGISTER 50
698          000052      ILRG52   = 52          ;ILLEGAL REGISTER 52
699          000054      ILRG54   = 54          ;ILLEGAL REGISTER 54
700          000056      ILRG56   = 56          ;ILLEGAL REGISTER 56
701          000060      ILRG60   = 60          ;ILLEGAL REGISTER 60
702          000062      ILRG62   = 62          ;ILLEGAL REGISTER 62
703          000064      ILRG64   = 64          ;ILLEGAL REGISTER 64
704          000066      ILRG66   = 66          ;ILLEGAL REGISTER 66
705          000070      ILRG70   = 70          ;ILLEGAL REGISTER 70
706          000072      ILRG72   = 72          ;ILLEGAL REGISTER 72
707          000074      ILRG74   = 74          ;ILLEGAL REGISTER 74
708          000076      ILRG76   = 76          ;ILLEGAL REGISTER 76
709
710          000077      IDXMSK   = 77          ;MASK FOR REGISTER INDEX NUMBER
711
712          .SBTTL      RH CONTROLLER REGISTER BIT DEFINITIONS
713
714          ;*RMCS1 CONTROL STATUS REGISTER #1
715
716          100000      SC        = BIT15      ;SPECIAL CONDITION-READ ONLY
717          040000      TRE        = BIT14      ;TRANSFER ERROR
718          020000      MCPE       = BIT13      ;MASSBUS CONTROL BUS PARITY ERROR-READ ONLY
719          002000      PSEL       = BIT10      ;PORT B SELECT
  
```

711	001000	A17	= BIT09	:ADDRESS EXTENSION
712	000400	A16	= BIT08	:ADDRESS EXTENSION
713	000200	RDY	= BIT07	:READY-READ ONLY
714	000100	IE	= BIT06	:INTERRUPT ENABLE
715				
716		;*RMCS2 RH CONTROL STATUS REGISTER #2		
717				
718	100000	DLT	= BIT15	:DATA LATE-READ ONLY
719	040000	WCE	= BIT14	:WRITE CHECK ERROR-READ ONLY
720	020000	UPE	= BIT13	:UNIBUS PARITY ERROR
721	010000	NED	= BIT12	:NONEXISTANT DRIVE-READ ONLY
722	004000	NEM	= BIT11	:NONEXISTANT MEMORY-READ ONLY
723	002000	PGE	= BIT10	:PROGRAM ERROR-READ ONLY
724	001000	MXF	= BIT09	:MISSED TRANSFER
725	000400	MDPE	= BIT08	:MASSBUS DATA BUS PARITY ERROR-READ ONLY
726	000200	OR	= BIT07	:OUTPUT READY-READ ONLY
727	000100	IR	= BIT06	:INPUT READY-READ ONLY
728	000040	CLR	= BIT05	:CONTROLLER CLEAR
729	000020	PAT	= BIT04	:PARITY TEST
730	000010	BAI	= BIT03	:UNIBUS ADDRESS INCREMENT INHIBIT
733	000004	U2	= BIT02	:UNIT SELECT
	000002	U1	= BIT01	:UNIT SELECT
	000001	U0	= BIT00	:UNIT SELECT
734				
735		:UNIT SELECT MASK		
736				
737	000007	UNTMSK	= 7	:UNIT SELECT MASK
738				
739		;*RMCS3 RH70 CONTROL STATUS REGISTER #3		
740				
741	100000	APE	= BIT15	:ADDRESS PARITY ERROR
742	040000	DPEHI	= BIT14	:DATA PARITY ERROR HIGH WORD
743	020000	DPELO	= BIT13	:DATA PARITY ERROR LOW WORD
744	010000	WCEHI	= BIT12	:WRITE CHECK ERROR HIGH WORD
745	004000	WCELO	= BIT11	:WRITE CHECK ERROR LOW WORD
746	002000	DBL	= BIT10	:DOUBLE WORD TRANSFER
747	000100	IE	= BIT06	:INTERRUPT ENABLE
748	000010	IPCK3	= BIT03	:INVERT PARITY CHECK
749	000004	IPCK2	= BIT02	:INVERT PARITY CHECK
750	000002	IPCK1	= BIT01	:INVERT PARITY CHECK
751	000001	IPCK0	= BIT00	:INVERT PARITY CHECK
752				
753		.SBTTL RH CONTROLLER REGISTER INDEX VALUES		
754				
755	000000	RMCS1	= 00	:CONTROL, STATUS REGISTER #1
756	000002	RMWC	= 02	:WORD COUNT REGISTER
757	000004	RMBA	= 04	:BUS ADDRESS REGISTER
758	000010	RMCS2	= 10	:CONTROL, STATUS REGISTER #2
759	000022	RMDB	= 22	:DATA BUFFER
760	000050	RMBAE	= 50	:BUS ADDRESS EXTENSION
761	000052	RMCS3	= 52	:CONTROL, STATUS REGISTER #3
762				
763	176700	ABASE	= 176700	:UNIBUS ADDRESS
764	120254	AVECT1	= 120254	:UNIBUS VECTOR ADDRESS AND PRIORITY
765				

1

.SBTTL TRAP CATCHER

000000

. = 0

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

000174 000174
 000174 000000
 000176 000000

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

.SBTTL STARTING ADDRESS(ES)

2 000200 000137 005422
 3 000204 000137 005412
 4
 5

JMP @#START ;:JUMP TO STARTING ADDRESS OF PROGRAM
 JMP @#START1 ;:CHANGE RH/RM BUS ADDRESS

.SBTTL ACT11 HOOKS

 ;:HOOKS REQUIRED BY ACT11

000046 000210
 000046 000046
 000052 041630
 000052 000052
 000052 000000
 000052 000210

\$SVPC= . ;:SAVE PC
 . = 46
 \$ENDAD ;:1)SET LOC.46 TO ADDRESS OF \$ENDAD IN .\$EOP
 . = 52
 .WORD 0 ;:2)SET LOC.52 TO ZERO
 . = \$SVPC ;:RESTORE PC

6 001100
 7
 8

. = 1100
 .SBTTL APT PARAMETER BLOCK

 ;:SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT

000024 001100
 000024 000024
 000044 000200
 000044 000044
 000044 001100
 001100

.\$X= . ;:SAVE CURRENT LOCATION
 . = 24 ;:SET POWER FAIL TO POINT TO START OF PROGRAM
 200 ;:FOR APT START UP
 . = 44 ;:POINT TO APT INDIRECT ADDRESS PNTR.
 \$APTHDR ;:POINT TO APT HEADER BLOCK
 . = . \$X ;:RESET LOCATION COUNTER

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

001100
 001100 000000
 001102 001222
 001104 000024
 001106 000024
 001110 000024
 001112 000042
 9 001114

\$APTHD:
 \$HIBTS: .WORD 0 ;:TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
 \$MBADR: .WORD \$MAIL ;:ADDRESS OF APT MAILBOX (BITS 0-15)
 \$TSTM: .WORD 20. ;:RUN TIM OF LONGEST TEST
 \$PASTM: .WORD 20. ;:RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
 \$UNITM: .WORD 20. ;:ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDED UNIT
 .WORD ;:LENGTH MAILBOX-ETABLE (WORDS)
 TAGADR=.

0

.SBTTL COMMON TAGS

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

001114 001114
 001114 000000
 001116 000
 001117 000
 001120 000000
 001122 000000
 001124 000000
 001126 000000
 001130 000
 001131 001
 001132 000000
 001134 000000
 001136 000000
 001140 000000
 001142 000000
 001144 000000
 001146 000000
 001150 000
 001151 000
 001152 000000
 001154 177570
 001156 177570
 001160 177560
 001162 177562
 001164 177564
 001166 177566
 001170 000
 001171 002
 001172 012
 001173 000
 001174 000000
 001176 000000
 001200 000000
 001202 000000
 001204 000000
 001206 000000
 001210 000000
 001212 207 377 377
 001216 077
 001217 015
 001220 012 000

.=TAGADR
 \$CMTAG: .WORD 0
 \$STNM: .BYTE 0
 \$ERFLG: .BYTE 0
 \$ICNT: .WORD 0
 \$LPADR: .WORD 0
 \$LPERR: .WORD 0
 \$ERTTL: .WORD 0
 \$ITEMB: .BYTE 0
 \$ERMAX: .BYTE 1
 \$ERRPC: .WORD 0
 \$GDADR: .WORD 0
 \$BDADR: .WORD 0
 \$GDDAT: .WORD 0
 \$BDDAT: .WORD 0
 .WORD 0
 \$AUTOB: .BYTE 0
 \$INTAG: .BYTE 0
 .WORD 0
 SWR: .WORD DSWR
 DISPLAY: .WORD DDISP
 \$TKS: 177560
 \$TKB: 177562
 \$TPS: 177564
 \$TPB: 177566
 \$NULL: .BYTE 0
 \$FILLS: .BYTE 2
 \$FILLC: .BYTE 12
 \$TPFLG: .BYTE 0
 \$TMP0: .WORD 0
 \$TMP1: .WORD 0
 \$TMP2: .WORD 0
 \$TMP3: .WORD 0
 \$TMP4: .WORD 0
 \$TIMES: 0
 \$ESCAPE: 0
 \$BELL: .ASCIZ <207><377><377>
 \$QUES: .ASCII /?/
 \$CRLF: .ASCII <15>
 \$LF: .ASCIZ <12>

:::START OF COMMON TAGS
 :::CONTAINS THE TEST NUMBER
 :::CONTAINS ERROR FLAG
 :::CONTAINS SUBTEST ITERATION COUNT
 :::CONTAINS SCOPE LOOP ADDRESS
 :::CONTAINS SCOPE RETURN FOR ERRORS
 :::CONTAINS TOTAL ERRORS DETECTED
 :::CONTAINS ITEM CONTROL BYTE
 :::CONTAINS MAX. ERRORS PER TEST
 :::CONTAINS PC OF LAST ERROR INSTRUCTION
 :::CONTAINS ADDRESS OF 'GOOD' DATA
 :::CONTAINS ADDRESS OF 'BAD' DATA
 :::CONTAINS 'GOOD' DATA
 :::CONTAINS 'BAD' DATA
 :::RESERVED--NOT TO BE USED
 :::AUTOMATIC MODE INDICATOR
 :::INTERRUPT MODE INDICATOR
 :::ADDRESS OF SWITCH REGISTER
 :::ADDRESS OF DISPLAY REGISTER
 :::TTY KBD STATUS
 :::TTY KBD BUFFER
 :::TTY PRINTER STATUS REG. ADDRESS
 :::TTY PRINTER BUFFER REG. ADDRESS
 :::CONTAINS NULL CHARACTER FOR FILLS
 :::CONTAINS # OF FILLER CHARACTERS REQUIRED
 :::INSERT FILL CHARS. AFTER A 'LINE FEED'
 :::'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
 :::USER DEFINED
 :::USER DEFINED
 :::USER DEFINED
 :::USER DEFINED
 :::USER DEFINED
 :::USER DEFINED
 :::MAX. NUMBER OF ITERATIONS
 :::ESCAPE ON ERROR ADDRESS
 :::CODE FOR BELL
 :::QUESTION MARK
 :::CARRIAGE RETURN
 :::LINE FEED

.SBTTL APT MAILBOX-ETABLE

001222
 001222 000000
 001224 000000
 001226 000000

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

.SBTTL USER DEFINED TAGS

001326 000000
 001330 000000

CHGADR: .WORD 0 ;CHANGE RH/RM BUS ADDRESS = -1, NO CHANGE = 0
 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.

001332 000
 001333 000

LSTRK: .BYTE 0 ;LO BYTE = 0
 .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
 RMASI: .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
 RMASO: .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	RMSNO: .WORD	0	:SERIAL NUMBER REGISTER
001442	000000	RMOFO: .WORD	0	:OFFSET REGISTER
001444	000000	RMDCO: .WORD	0	:DESIRED CYLINDER REGISTER
001446	000000	RMHRO: .WORD	0	:HOLDING REGISTER
001450	000000	RMMR20: .WORD	0	:MAINTENANCE REGISTER #2
001452	000000	RMER20: .WORD	0	:ERROR REGISTER #2
001454	000000	RMEC10: .WORD	0	:ECC POSITION REGISTER
001456	000000	RMEC20: .WORD	0	:ECC PATTERN REGISTER
001460	000000	RMBAE0: .WORD	0	:BUS ADDRESS EXTENSION REGISTER
001462	000000	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	TSTQUE: .WORD	0	:CONTAINS DEVICE POINTER
001466		.BLKW	8.	:TEST QUE FOR DEVICES UNDER TEST
001506	000000	.WORD	0	:TABLE TERMINATOR GOES HERE WHEN :ALL 8. DEVICES ARE UNDER TEST.

001510	000000	CLKADR: .WORD	0	:UNIBUS ADDRESS OF KW11 CLOCK
001512	000000	CLKVCT: .WORD	0	:VECTOR ADDRESS OF KW11 CLOCK

;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.
 001514 GETINX: .BLKB 23. ;GET INDEX TABLE

;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.
 001543 PUTINX: .BLKB 23. ;PUT INDEX TABLE

;PUT TAGS HERE

.SBTTL ERROR POINTER TABLE

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

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

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

001572
 001572 067246
 001574 073332
 001576 073456
 001600 073546
 001602 067252
 001604 073332
 001606 073456
 001610 073546
 001612 067260
 001614 073332
 001616 073456
 001620 073546
 001622 067266
 001624 073332
 001626 073456
 001630 073546
 001632 067224
 001634 073332
 001636 073456
 001640 073546

\$ERRTB:
 ;ERROR 1 WRONG UNIT SELECTED
 EMT1
 EHT1
 EDT1
 EFT1
 ;ERROR 2 DEVICE WENT UNAVAILABLE
 EMT2
 EHT1
 EDT1
 EFT1
 ;ERROR 3 DEVICE WENT NONEXISTENT
 EMT3
 EHT1
 EDT1
 EFT1
 ;ERROR 4 CONTROLLER NOT READY
 EMT4
 EHT1
 EDT1
 EFT1
 ;ERROR 5 DRIVE NOT READY AND GO NOT RESET
 EMT5
 EHT1
 EDT1
 EFT1
 ;ERROR 6 UNEXPECTED VALUE FOR "ATA" STATUS

18	001642	067302	EMT6	
	001644	073332	EHT1	
	001646	073456	EDT1	
	001650	073546	EFT1	
19				
20			:ERROR 7	BUS TIMEOUT TRYING TO READ OR WRITE REGISTER
21				
	001652	067310	EMT7	
	001654	000000	0	
	001656	000000	0	
	001660	000000	0	
22				
23			:ERROR 10	DRIVE NOT READY BUT GO IS RESET
24				
	001662	067316	EMT10	
	001664	073332	EHT1	
	001666	073456	EDT1	
	001670	073546	EFT1	
25				
26			:ERROR 11	GO NOT RESET BUT DRIVE IS READY
27				
	001672	067322	EMT11	
	001674	073332	EHT1	
	001676	073456	EDT1	
	001700	073546	EFT1	
28				
29			:ERROR 12	INCORRECT FUNCTION CODE
30				
	001702	067326	EMT12	
	001704	073332	EHT1	
	001706	073456	EDT1	
	001710	073546	EFT1	
31				
32			:ERROR 13	PARITY ERROR READING REMOTE REGISTERS
33				
	001712	067334	EMT13	
	001714	073332	EHT1	
	001716	073456	EDT1	
	001720	073546	EFT1	
34				
35			:ERROR 14	TRANSFER ERROR IS INCORRECT
36				
	001722	067346	EMT14	
	001724	073332	EHT1	
	001726	073456	EDT1	
	001730	073546	EFT1	
37				
38			:ERROR 15	INCORRECT WORD COUNT
39				

ERROR POINTER TABLE				
	001732 067354		EMT15	
	001734 073332		EHT1	
	001736 073456		EDT1	
	001740 073546		EFT1	
40				
41		:ERROR 16		INCORRECT BUS ADDRESS
42				
	001742 067362		EMT16	
	001744 073332		EHT1	
	001746 073456		EDT1	
	001750 073546		EFT1	
43				
44		:ERROR 17		INCORRECT LBT STATUS
45				
	001752 067372		EMT17	
	001754 073332		EHT1	
	001756 073456		EDT1	
	001760 073546		EFT1	
46				
47		:ERROR 20		INCORRECT AOE
48				
	001762 067402		EMT20	
	001764 073332		EHT1	
	001766 073456		EDT1	
	001770 073546		EFT1	
49				
50		:ERROR 21		INCORRECT DISK ADDRESS
51				
	001772 067412		EMT21	
	001774 073332		EHT1	
	001776 073456		EDT1	
	002000 073546		EFT1	
52				
53		:ERROR 22		INCORRECT CYLINDER ADDRESS
54				
	002002 067422		EMT22	
	002004 073332		EHT1	
	002006 073456		EDT1	
	002010 073546		EFT1	
55				
56		:ERROR 23		INCORRECT WLE STATUS
57				
	002012 067432		EMT23	
	002014 073332		EHT1	
	002016 073456		EDT1	
	002020 073546		EFT1	
58				
59		:ERROR 24		INCORRECT UPE STATUS
60				
	002022 067442		EMT24	

	002024	073332	EHT1	
	002026	073456	EDT1	
	002030	073546	EFT1	
61				
62				:ERROR 25 INCORRECT WCF STATUS
63				
	002032	067452	EMT25	
	002034	073332	EHT1	
	002036	073456	EDT1	
	002040	073546	EFT1	
64				
65				:ERROR 26 INCORRECT WCE STATUS
66				
	002042	067462	EMT26	
	002044	073332	EHT1	
	002046	073456	EDT1	
	002050	073546	EFT1	
67				
68				:ERROR 27 INCORRECT MDPE STATUS
69				
	002052	067472	EMT27	
	002054	073332	EHT1	
	002056	073456	EDT1	
	002060	073546	EFT1	
70				
71				:ERROR 30 INCORRECT DCK STATUS
72				
	002062	067502	EMT30	
	002064	073332	EHT1	
	002066	073456	EDT1	
	002070	073546	EFT1	
73				
74				:ERROR 31 INCORRECT ECH STATUS
75				
	002072	067512	EMT31	
	002074	073332	EHT1	
	002076	073456	EDT1	
	002100	073546	EFT1	
76				
77				:ERROR 32 DLT SHOULD NOT BE SET
78				
	002102	067522	EMT32	
	002104	073332	EHT1	
	002106	073456	EDT1	
	002110	073546	EFT1	
79				
80				:ERROR 33 MXF SHOULD NOT BE SET
81				
	002112	067532	EMT33	
	002114	073332	EHT1	

	002116	073456	EDT1	
	002120	073546	EFT1	
82				
83				:ERROR 34 DTE SHOULD NOT BE SET
84				
	002122	067542	EMT34	
	002124	073332	EHT1	
	002126	073456	EDT1	
	002130	073546	EFT1	
85				
86				:ERROR 35 INCORRECT HCRC STATUS
87				
	002132	067552	EMT35	
	002134	073332	EHT1	
	002136	073456	EDT1	
	002140	073546	EFT1	
88				
89				:ERROR 36 INCORRECT HCE STATUS
90				
	002142	067562	EMT36	
	002144	073332	EHT1	
	002146	073456	EDT1	
	002150	073546	EFT1	
91				
92				:ERROR 37 INCORRECT FER STATUS
93				
	002152	067572	EMT37	
	002154	073332	EHT1	
	002156	073456	EDT1	
	002160	073546	EFT1	
94				
95				:ERROR 40 DPE SHOULD NOT BE SET (NOT A DATA COMMAND)
96				
	002162	067602	EMT40	
	002164	073332	EHT1	
	002166	073456	EDT1	
	002170	073546	EFT1	
97				
98				:ERROR 41 LOST "MOL" DURING PACK ACKNOWLEDGE
99				
	002172	067610	EMT41	
	002174	073332	EHT1	
	002176	073456	EDT1	
	002200	073546	EFT1	
100				
101				:ERROR 42 UNSAFE ERROR DURING PACK ACKNOWLEDGE
102				
	002202	067620	EMT42	
	002204	073332	EHT1	
	002206	073456	EDT1	

Line	Address	Module	Error Code	Description
	002210	073546	EFT1	
103				
104			:ERROR 43	"OPI" ERROR DURING PACK ACKNOWLEDGE
105				
	002212	067632	EMT43	
	002214	073332	EHT1	
	002216	073456	EDT1	
	002220	073546	EFT1	
106				
107			:ERROR 44	"RMR" ERROR DURING PACK ACKNOWLEDGE
108				
	002222	067642	EMT44	
	002224	073332	EHT1	
	002226	073456	EDT1	
	002230	073546	EFT1	
109				
110			:ERROR 45	"ILR" ERROR DURING PACK ACKNOWLEDGE
111				
	002232	067652	EMT45	
	002234	073332	EHT1	
	002236	073456	EDT1	
	002240	073546	EFT1	
112				
113			:ERROR 46	"ILF" ERROR DURING PACK ACKNOWLEDGE
114				
	002242	067662	EMT46	
	002244	073332	EHT1	
	002246	073456	EDT1	
	002250	073546	EFT1	
115				
116			:ERROR 47	COMPOSITE ERROR STATUS IS INCORRECT
117				
	002252	067672	EMT47	
	002254	073332	EHT1	
	002256	073456	EDT1	
	002260	073546	EFT1	
118				
119			:ERROR 50	PARITY ERROR WRITING REMOTE REGISTERS
120				
	002262	067700	EMT50	
	002264	073332	EHT1	
	002266	073456	EDT1	
	002270	073546	EFT1	
121				
122			:ERROR 51	INCORRECT IAE STATUS DURING SEEK COMMAND
123				
	002272	067710	EMT51	
	002274	073332	EHT1	
	002276	073456	EDT1	
	002300	073546	EFT1	

124				
125				
126		:ERROR 52	OPI ERROR DURING SEEK - MEDIUM IS NOT ON LINE	
	002302	067722	EMT52	
	002304	073332	EHT1	
	002306	073456	EDT1	
	002310	073546	EFT1	
127				
128		:ERROR 53	OPI ERROR DURING SEEK - MEDIUM IS ON LINE. ASSUME	
129		:	ON CYLINDER LATCH DIDN'T RESET	
130				
	002312	067740	EMT53	
	002314	073332	EHT1	
	002316	073456	EDT1	
	002320	073546	EFT1	
131				
132		:ERROR 54	SEEK INCOMPLETE ERROR DURING SEEK COMMAND	
133				
	002322	067756	EMT54	
	002324	073332	EHT1	
	002326	073456	EDT1	
	002330	073546	EFT1	
134				
135		:ERROR 55	DEVICE CHECK DURING SEEK COMMAND	
136				
	002332	067766	EMT55	
	002334	073332	EHT1	
	002336	073456	EDT1	
	002340	073546	EFT1	
137				
138		:ERROR 56	PIP IS STILL SET AFTER SEEK - SKI IS RESET	
139				
	002342	070000	EMT56	
	002344	073332	EHT1	
	002346	073456	EDT1	
	002350	073546	EFT1	
140				
141		:ERROR 57	ATA DID NOT SET DURING SEEK COMMAND	
142				
	002352	070016	EMT57	
	002354	073332	EHT1	
	002356	073456	EDT1	
	002360	073546	EFT1	
143				
144		:ERROR 60	IVC ERROR DURING SEEK COMMAND - LOST	
145		:	VOLUME VALID	
146				
	002362	070026	EMT60	
	002364	073332	EHT1	
	002366	073456	EDT1	

Line	Code	Module	Label	Description
	002370	073546	EFT1	
147				
148			:ERROR 61	ERRONEOUS IVC ERROR DURING SEEK COMMAND -
149			:	VOLUME VALID IS STILL SET
150				
	002372	070044	EMT61	
	002374	073332	EHT1	
	002376	073456	EDT1	
	002400	073546	EFT1	
151				
152			:ERROR 62	MOL IS ZERO, BUT OPI WAS NOT
153			:	REPORTED DURING SEEK COMMAND
154				
	002402	070064	EMT62	
	002404	073332	EHT1	
	002406	073456	EDT1	
	002410	073546	EFT1	
155				
156			:ERROR 63	UNUSED
157				
	002412	000000	0	
	002414	000000	0	
	002416	000000	0	
	002420	000000	0	
158				
159			:ERROR 64	DRIVE DID NOT DETECT "IVC" ERROR DURING SEEK
160				
	002422	070102	EMT64	
	002424	073332	EHT1	
	002426	073456	EDT1	
	002430	073546	EFT1	
161				
162			:ERROR 65	DRIVE EXECUTED A SEEK WITH ERROR SET
163				
	002432	070122	EMT65	
	002434	073332	EHT1	
	002436	073456	EDT1	
	002440	073546	EFT1	
164				
165			:ERROR 66	UNEXPECTED ERROR SET IN RMER1
166				
	002442	070142	EMT66	
	002444	073332	EHT1	
	002446	073456	EDT1	
	002450	073546	EFT1	
167				
168			:ERROR 67	UNEXPECTED ERROR SET IN RMER2
169				
	002452	070154	EMT67	
	002454	073332	EHT1	

	002456 073456		EDT1	
	002460 073546		EFT1	
170				
171		;ERROR 70		ERRONEOUS "IAE" ERROR DURING RECALIBRATE
172				
	002462 070166		EMT70	
	002464 073332		EHT1	
	002466 073456		EDT1	
	002470 073546		EFT1	
173				
174		;ERROR 71		"ILF" ERROR DURING RECALIBRATE
175				
	002472 070176		EMT71	
	002474 073332		EHT1	
	002476 073456		EDT1	
	002500 073546		EFT1	
176				
177		;ERROR 72		"OPI" ERROR DURING RECALIBRATE DUE TO 'MOL' = 0
178				
	002502 070206		EMT72	
	002504 073332		EHT1	
	002506 073456		EDT1	
	002510 073546		EFT1	
179				
180		;ERROR 73		"OPI" ERROR DURING RECALIBRATE BECAUSE ON
181		:		CYLINDER DIDNT DROP
182				
	002512 070224		EMT73	
	002514 073332		EHT1	
	002516 073456		EDT1	
	002520 073546		EFT1	
183				
184		;ERROR 74		"IVC" ERROR DURING RECALIBRATE - 'VV' = 0
185				
	002522 070242		EMT74	
	002524 073332		EHT1	
	002526 073456		EDT1	
	002530 073546		EFT1	
186				
187		;ERROR 75		ERRONEOUS "IVC" ERROR DURING RECALIBRATE - 'VV' = 1
188				
	002532 070252		EMT75	
	002534 073332		EHT1	
	002536 073456		EDT1	
	002540 073546		EFT1	
189				
190		;ERROR 76		"SKI" ERROR DURING RECALIBRATE
191				
	002542 070272		EMT76	
	002544 073332		EHT1	

Line	Address	Code	Message
	002546 073456	EDT1	
	002550 073546	EFT1	
192			
193			
194			:ERROR 77 'DVC' OCCURRED DURING RECALIBRATE
	002552 070302	EMT77	
	002554 073332	EHT1	
	002556 073456	EDT1	
	002560 073546	EFT1	
195			
196			
197			:ERROR 100 LOST 'MOL' DURING RECALIBRATE - 'OPI' = 0
	002562 070314	EMT100	
	002564 073332	EHT1	
	002566 073456	EDT1	
	002570 073546	EFT1	
198			
199			
200			:ERROR 101 LOST 'VV' DURING RECALIBRATE - 'IVC' = 0
	002572 070332	EMT101	
	002574 073332	EHT1	
	002576 073456	EDT1	
	002600 073546	EFT1	
201			
202			
203			:ERROR 102 'ATA' DID NOT SET DURING RECALIBRATE
	002602 070350	EMT102	
	002604 073332	EHT1	
	002606 073456	EDT1	
	002610 073546	EFT1	
204			
205			
206			:ERROR 103 'DM' DID NOT RESET DURING RECALIBRATE
	002612 070360	EMT103	
	002614 073332	EHT1	
	002616 073456	EDT1	
	002620 073546	EFT1	
207			
208			
209			:ERROR 104 'PIP' IS STIL SET AFTER RECALIBRATE
	002622 070372	EMT104	
	002624 073332	EHT1	
	002626 073456	EDT1	
	002630 073546	EFT1	
210			
211			
212			:ERROR 105 UNEXPECTED 'ILR' ERROR DURING RECALIBRATE
	002632 070410	EMT105	
	002634 073332	EHT1	
	002636 073456	EDT1	

ERROR POINTER TABLE

	002640	073546	EFT1	
213				
214			:ERROR 106	UNEXPECTED 'RMR' ERROR DURING RECALIBRATE
215				
	002642	070420	EMT106	
	002644	073332	EHT1	
	002646	073456	EDT1	
	002650	073546	EFT1	
216				
217			:ERROR 107	'UNS' ERROR DURING RECALIBRATE - AC POWER IS LOW
218				
	002652	070430	EMT107	
	002654	073332	EHT1	
	002656	073456	EDT1	
	002660	073546	EFT1	
219				
220			:ERROR 110	CANNOT ACCESS MASSBUS CONTROLLER VIA UNIBUS
221				
	002662	070450	EMT110	
	002664	073354	EHT110	
	002666	073474	EDT110	
	002670	073564	EFT110	
222				
223			:ERROR 111	NONEXISTENT DEVICE
224				
	002672	070462	EMT111	
	002674	073360	EHT111	
	002676	073476	EDT111	
	002700	073566	EFT111	
225				
226			:ERROR 112	DEVICE NOT AVAILABLE
227				
	002702	070470	EMT112	
	002704	073360	EHT111	
	002706	073476	EDT111	
	002710	073566	EFT111	
228				
229			:ERROR 113	BUS TIMEOUT-NED STATUS FAILURE
230				
	002712	070476	EMT113	
	002714	000000	0	
	002716	000000	0	
	002720	000000	0	
231				
232			:ERROR 114	DEVICE NOT AN RM05/3/2
233				
	002722	070512	EMT114	
	002724	073364	EHT114	
	002726	073500	EDT114	
	002730	073570	EFT114	

234				
235			:ERROR 115	RMCS1 NOT INITIALIZED BY UNIBUS
236	002732	070520		EMT115
	002734	073332		EHT1
	002736	073456		EDT1
	002740	073546		EFT1
237				
238			:ERROR 116	RMBA NOT INITIALIZED BY UNIBUS
239	002742	070530		EMT116
	002744	073332		EHT1
	002746	073456		EDT1
	002750	073546		EFT1
240				
241			:ERROR 117	RMCS2 NOT INITIALIZED BY UNIBUS
242	002752	070540		EMT117
	002754	073332		EHT1
	002756	073456		EDT1
	002760	073546		EFT1
243				
244			:ERROR 120	RMER1 NOT INITIALIZED BY UNIBUS
245	002762	070550		EMT120
	002764	073332		EHT1
	002766	073456		EDT1
	002770	073546		EFT1
246				
247			:ERROR 121	RMAS NOT INITIALIZED BY UNIBUS
248	002772	070560		EMT121
	002774	073332		EHT1
	002776	073456		EDT1
	003000	073546		EFT1
249				
250			:ERROR 122	RMMR1 NOT INITIALIZED BY UNIBUS
251	003002	070570		EMT122
	003004	073332		EHT1
	003006	073456		EDT1
	003010	073546		EFT1
252				
253			:ERROR 123	RMDS NOT INITIALIZED BY UNIBUS
254	003012	070600		EMT123
	003014	073332		EHT1
	003016	073456		EDT1
	003020	073546		EFT1

255				
256			;ERROR 124	RMEC2 NOT INITIALIZED BY UNIBUS
257	003022	070610	EMT124	
	003024	073332	EHT1	
	003026	073456	EDT1	
	003030	073546	EFT1	
258				
259			;ERROR 125	RMIR2 NOT INITIALIZED BY UNIBUS
260	003032	070620	EMT125	
	003034	073332	EHT1	
	003036	073456	EDT1	
	003040	073546	EFT1	
261				
262			;ERROR 126	RMCS1 NOT CLEARED BY CONTROLLER CLEAR
263	003042	070630	EMT126	
	003044	073332	EHT1	
	003046	073456	EDT1	
	003050	073546	EFT1	
264				
265			;ERROR 127	RMBA NOT CLEARED BY CONTROLLER CLEAR
266	003052	070642	EMT127	
	003054	073332	EHT1	
	003056	073456	EDT1	
	003060	073546	EFT1	
267				
268			;ERROR 130	RMCS2 NOT CLEARED BY CONTROLLER CLEAR
269	003062	070654	EMT130	
	003064	073332	EHT1	
	003066	073456	EDT1	
	003070	073546	EFT1	
270				
271			;ERROR 131	RMER1 NOT CLEARED BY CONTROLLER CLEAR
272	003072	070666	EMT131	
	003074	073332	EHT1	
	003076	073456	EDT1	
	003100	073546	EFT1	
273				
274			;ERROR 132	RMA5 NOT CLEARED BY CONTROLLER CLEAR
275	003102	070700	EMT132	
	003104	073332	EHT1	
	003106	073456	EDT1	
	003110	073546	EFT1	

276

ERROR POINTER TABLE

277			:ERROR 133	RMMR1 NOT CLEARED BY CONTROLLER CLEAR
278	003112	070712	EMT133	
	003114	073332	EHT1	
	003116	073456	EDT1	
	003120	073546	EFT1	
279				
280			:ERROR 134	RMDS NOT CLEARED BY CONTROLLER CLEAR
281	003122	070724	EMT134	
	003124	073332	EHT1	
	003126	073456	EDT1	
	003130	073546	EFT1	
282				
283			:ERROR 135	RMEC2 NOT CLEARED BY CONTROLLER CLEAR
284	003132	070736	EMT135	
	003134	073332	EHT1	
	003136	073456	EDT1	
	003140	073546	EFT1	
285				
286			:ERROR 136	RMMR2 NOT CLEARED BY CONTROLLER CLEAR
287	003142	070750	EMT136	
	003144	073332	EHT1	
	003146	073456	EDT1	
	003150	073546	EFT1	
288				
289			:ERROR 137	RMCS1 NOT CLEARED BY ERROR CLEAR
290	003152	070762	EMT137	
	003154	073332	EHT1	
	003156	073456	EDT1	
	003160	073546	EFT1	
291				
292			:ERROR 140	RMCS2 NOT CLEARED BY ERROR CLEAR
293	003162	070772	EMT140	
	003164	073332	EHT1	
	003166	073456	EDT1	
	003170	073546	EFT1	
294				
295			:ERROR 141	RMCS1 NOT CLEARED BY DRIVE CLEAR
296	003172	071002	EMT141	
	003174	073332	EHT1	
	003176	073456	EDT1	
	003200	073546	EFT1	
297				
298			:ERROR 142	RMDS NOT CLEARED BY DRIVE CLEAR

ERROR POINTER TABLE

299	003202 071012	EMT142	
	003204 073332	EHT1	
	003206 073456	EDT1	
	003210 073546	EFT1	
300			
301		:ERROR 143	RMER1 NOT CLEARED BY DRIVE CLEAR
302	003212 071022	EMT143	
	003214 073332	EHT1	
	003216 073456	EDT1	
	003220 073546	EFT1	
303			
304		:ERROR 144	RMAS NOT CLEARED BY DRIVE CLEAR
305	003222 071032	EMT144	
	003224 073332	EHT1	
	003226 073456	EDT1	
	003230 073546	EFT1	
306			
307		:ERROR 145	RMMR1 NOT CLEARED BY DRIVE CLEAR
308	003232 071042	EMT145	
	003234 073332	EHT1	
	003236 073456	EDT1	
	003240 073546	EFT1	
309			
310		:ERROR 146	RMMR2 NOT CLEARED BY DRIVE CLEAR
311	003242 071052	EMT146	
	003244 073332	EHT1	
	003246 073456	EDT1	
	003250 073546	EFT1	
312			
313		:ERROR 147	RMER2 NOT CLEARED BY DRIVE CLEAR
314	003252 071062	EMT147	
	003254 073332	EHT1	
	003256 073456	EDT1	
	003260 073546	EFT1	
315			
316		:ERROR 150	RMEC2 NOT CLEARED BY DRIVE CLEAR
317	003262 071072	EMT150	
	003264 073332	EHT1	
	003266 073456	EDT1	
	003270 073546	EFT1	
318			
319		:ERROR 151	MEDIUM NOT ON LINE
320			

Line	Code	Address	Label	Description
	003272	071102	EMT151	
	003274	073332	EHT1	
	003276	073456	EDT1	
	003300	073546	EFT1	
321				
322			:ERROR 152	DRIVE FAULT
323				
	003302	071114	EMT152	
	003304	073332	EHT1	
	003306	073456	EDT1	
	003310	073546	EFT1	
324				
325			:ERROR 153	UNSAFE SHOULD BE SET BECAUSE DVC IS SET
326				
	003312	071126	EMT153	
	003314	073332	EHT1	
	003316	073456	EDT1	
	003320	073546	EFT1	
327				
328			:ERROR 154	UNSAFE SHOULD NOT BE SET, AC IS LOW
329				
	003322	071144	EMT154	
	003324	073332	EHT1	
	003326	073456	EDT1	
	003330	073546	EFT1	
330				
331			:ERROR 155	VOLUME VALID NOT SET BY PACK ACK
332				
	003332	071162	EMT155	
	003334	073332	EHT1	
	003336	073456	EDT1	
	003340	073546	EFT1	
333				
334			:ERROR 156	OFFSET MODE NOT SET BY OFFSET COMMAND
335				
	003342	071174	EMT156	
	003344	073332	EHT1	
	003346	073456	EDT1	
	003350	073546	EFT1	
336				
337			:ERROR 157	OFFSET MODE NOT RESET BY RTC COMMAND
338				
	003352	071206	EMT157	
	003354	073332	EHT1	
	003356	073456	EDT1	
	003360	073546	EFT1	
339				
340			:ERROR 160	RMOF NOT RESET BY RIP COMMAND
341				
	003362	071220	EMT160	

003364 073332
 003366 073456
 003370 073546

EHT1
 EDT1
 EFT1

342
 343
 344

:ERROR 161 RMDA NOT RESET BY RIP COMMAND

003372 071230
 003374 073332
 003376 073456
 003400 073546

EMT161
 EHT1
 EDT1
 EFT1

345
 346
 347

:ERROR 162 RMDC NOT RESET BY RIP COMMAND

003402 071242
 003404 073332
 003406 073456
 003410 073546

EMT162
 EHT1
 EDT1
 EFT1

348
 349
 350
 351

:ERROR 163 DATA WAS ECC CORRECTED BUT DOES NOT COMPARE WITH WRITE BUFFER

003412 073126
 003414 073414
 003416 073512
 003420 073602

EMT336
 EHT336
 EDT336
 EFT336

352
 353
 354

:ERROR 164 OPI SHOULD NOT BE SET

003422 071264
 003424 073332
 003426 073456
 003430 073546

EMT164
 EHT1
 EDT1
 EFT1

355
 356
 357

:ERROR 165 IVC SHOULD NOT BE SET

003432 071272
 003434 073332
 003436 073456
 003440 073546

EMT165
 EHT1
 EDT1
 EFT1

358
 359
 360

:ERROR 166 IAE SHOULD NOT BE SET

003442 071300
 003444 073332
 003446 073456
 003450 073546

EMT166
 EHT1
 EDT1
 EFT1

361
 362
 363

:ERROR 167 NEM SHOULD NOT BE SET

003452 071306

EMT167

ERROR POINTER TABLE

	003454	073332	EHT1	
	003456	073456	EDT1	
	003460	073546	EFT1	
364				
365			:ERROR	170 INCORRECT "MOL" STATUS DURING DIAGNOSTIC MODE
366				
	003462	071314	EMT170	
	003464	073332	EHT1	
	003466	073456	EDT1	
	003470	073546	EFT1	
367				
368			:ERROR	171 "ATA" NOT SET DURING RETURN TO CENTERLINE
369				
	003472	071326	EMT171	
	003474	073332	EHT1	
	003476	073456	EDT1	
	003500	073546	EFT1	
370				
371			:ERROR	172 "ATA" NOT SET BY-OFFSET COMMAND
372				
	003502	071336	EMT172	
	003504	073332	EHT1	
	003506	073456	EDT1	
	003510	073546	EFT1	
373				
374			:ERROR	173 RMER2 NOT INITIALIZED BY UNIBUS INIT
375				
	003512	071346	EMT173	
	003514	073332	EHT1	
	003516	073456	EDT1	
	003520	073546	EFT1	
376				
377			:ERROR	174 RMER2 NOT INITIALIZED BY CONTROLLER CLEAR
378				
	003522	071356	EMT174	
	003524	073332	EHT1	
	003526	073456	EDT1	
	003530	073546	EFT1	
379				
380			:ERROR	175 SELECTED DEVICE IS IN WRITE PROTECT
381				
	003532	071370	EMT175	
	003534	073332	EHT1	
	003536	073456	EDT1	
	003540	073546	EFT1	
382				
383			:ERROR	176 CANNOT SET DIAGNOSTIC MODE
384				
	003542	071376	EMT176	
	003544	073332	EHT1	

ERROR POINTER TABLE				
	003546 073456		EDT1	
	003550 073546		EFT1	
385				
386		:ERROR 177		--RESERVED FOR POWER MONITOR BIT FAILURE--
387				
	003552 000000		0	
	003554 000000		0	
	003556 000000		0	
	003560 000000		0	
388				
389		:ERROR 200		INCORRECT 'PIP' STATUS DURING DIAGNOSTIC MODE
390				
	003562 071406		EMT200	
	003564 073332		EHT1	
	003566 073456		EDT1	
	003570 073546		EFT1	
391				
392		:ERROR 201		INCORRECT 'WRL' STATUS DURING DIAGNOSTIC MODE
393				
	003572 071420		EMT201	
	003574 073332		EHT1	
	003576 073456		EDT1	
	003600 073546		EFT1	
394				
395		:ERROR 202		INCORRECT 'SKI' STATUS DURING DIAGNOSTIC MODE
396				
	003602 071432		EMT202	
	003604 073332		EHT1	
	003606 073456		EDT1	
	003610 073546		EFT1	
397				
398		:ERROR 203		INCORRECT 'DVC' STATUS DURING DIAGNOSTIC MODE
399				
	003612 071444		EMT203	
	003614 073332		EHT1	
	003616 073456		EDT1	
	003620 073546		EFT1	
400				
401		:ERROR 204		'VV' WAS NOT RESET BY MAINTENANCE UNIT READY
402				
	003622 071456		EMT204	
	003624 073332		EHT1	
	003626 073456		EDT1	
	003630 073546		EFT1	
403				
404		:ERROR 205		SELECTED DEVICE HAS A PERSISTENT 'SKI' ERROR
405				
	003632 071474		EMT205	
	003634 073332		EHT1	
	003636 073456		EDT1	

	003640	073546	EFT1	
406				
407				
408				:ERROR 206 "LBC" DID NOT SET DURING DIAGNOSTIC MODE
	003642	071504	EMT206	
	003644	073332	EHT1	
	003646	073456	EDT1	
	003650	073546	EFT1	
409				
410				:ERROR 207 UNEXPECTED LOSS OF 'MOL' - MEDIUM IS OFF LINE
411				
	003652	071514	EMT207	
	003654	073332	EHT1	
	003656	073456	EDT1	
	003660	073546	EFT1	
412				
413				:ERROR 210 UNEXPECTED LOSS OF VOLUME VALID - 'VV' = 0
414				
	003662	071526	EMT210	
	003664	073332	EHT1	
	003666	073456	EDT1	
	003670	073546	EFT1	
415				
416				:ERROR 211 UNEXPECTED MECHANICAL MOTION - 'PIP' = 1
417				
	003672	071534	EMT211	
	003674	073332	EHT1	
	003676	073456	EDT1	
	003700	073546	EFT1	
418				
419				:ERROR 212 UNEXPECTED DEVICE FAULT - 'DVC' = 1
420				
	003702	071550	EMT212	
	003704	073332	EHT1	
	003706	073456	EDT1	
	003710	073546	EFT1	
421				
422				:ERROR 213 UNEXPECTED SEEK INCOMPLETE ERROR - 'SKI' = 1
423				
	003712	071564	EMT213	
	003714	073332	EHT1	
	003716	073456	EDT1	
	003720	073546	EFT1	
424				
425				:ERROR 214 DRIVE EXECUTED A RECALIBRATE WITH ERROR SET
426				
	003722	071574	EMT214	
	003724	073344	EHT2	
	003726	073466	EDT2	
	003730	073556	EFT2	

427				
428				
429			:ERROR 215	DRIVE DID NOT DETECT "IVC" ERROR DURING RECALIBRATE
	003732	071614	EMT215	
	003734	073344	EHT2	
	003736	073466	EDT2	
	003740	073556	EFT2	
430				
431				
432			:ERROR 216	INCORRECT "IVC" STATUS
	003742	071626	EMT216	
	003744	073332	EHT1	
	003746	073456	EDT1	
	003750	073546	EFT1	
433				
434				
435			:ERROR 217	INCORRECT "IAE" STATUS
	003752	071636	EMT217	
	003754	073332	EHT1	
	003756	073456	EDT1	
	003760	073546	EFT1	
436				
437				
438			:ERROR 220	INCORRECT "WLE" STATUS
	003762	071646	EMT220	
	003764	073332	EHT1	
	003766	073456	EDT1	
	003770	073546	EFT1	
439				
440				
441			:ERROR 221	INCORRECT "OPI" STATUS
	003772	071656	EMT221	
	003774	073332	EHT1	
	003776	073456	EDT1	
	004000	073546	EFT1	
442				
443				
444			:ERROR 222	RM DID NOT DETECT RMR ERROR
	004002	071666	EMT222	
	004004	073332	EHT1	
	004006	073456	EDT1	
	004010	073546	EFT1	
445				
446				
447			:ERROR 223	RM DID NOT DETECT PARITY ERROR ON MASSBUS CONTROL BUS
	004012	071676	EMT223	
	004014	073370	EHT223	
	004016	073502	EDT223	
	004020	073572	EFT223	

448				
449			:ERROR	224 UNUSED
450	004022	000000		0
	004024	000000		0
	004026	000000		0
	004030	000000		0
451				
452			:ERROR	225 UNUSED
453	004032	000000		0
	004034	000000		0
	004036	000000		0
	004040	000000		0
454				
455			:ERROR	226 UNUSED
456	004042	000000		0
	004044	000000		0
	004046	000000		0
	004050	000000		0
457				
458			:ERROR	227 UNUSED
459	004052	000000		0
	004054	000000		0
	004056	000000		0
	004060	000000		0
460				
461			:ERROR	230 UNUSED
462	004062	000000		0
	004064	000000		0
	004066	000000		0
	004070	000000		0
463				
464			:ERROR	231 UNUSED
465	004072	000000		0
	004074	000000		0
	004076	000000		0
	004100	000000		0
466				
467			:ERROR	232 UNUSED
468	004102	000000		0
	004104	000000		0
	004106	000000		0
	004110	000000		0

469

470			:ERROR	233	UNUSED
471	004112	000000		0	
	004114	000000		0	
	004116	000000		0	
	004120	000000		0	
472					
473			:ERROR	234	UNUSED
474	004122	000000		0	
	004124	000000		0	
	004126	000000		0	
	004130	000000		0	
475					
476			:ERROR	235	UNUSED
477	004132	000000		0	
	004134	000000		0	
	004136	000000		0	
	004140	000000		0	
478					
479			:ERROR	236	UNUSED
480	004142	000000		0	
	004144	000000		0	
	004146	000000		0	
	004150	000000		0	
481					
482			:ERROR	237	UNUSED
483	004152	000000		0	
	004154	000000		0	
	004156	000000		0	
	004160	000000		0	
484					
485			:ERROR	240	UNUSED
486	004162	000000		0	
	004164	000000		0	
	004166	000000		0	
	004170	000000		0	
487					
488			:ERROR	241	UNUSED
489	004172	000000		0	
	004174	000000		0	
	004176	000000		0	
	004200	000000		0	
490					
491			:ERROR	242	UNUSED

492	004202	000000	0	
	004204	000000	0	
	004206	000000	0	
	004210	000000	0	
493				
494			:ERROR	243 UNUSED
495	004212	000000	0	
	004214	000000	0	
	004216	000000	0	
	004220	000000	0	
496				
497			:ERROR	244 UNUSED
498	004222	000000	0	
	004224	000000	0	
	004226	000000	0	
	004230	000000	0	
499				
500			:ERROR	245 UNUSED
501	004232	000000	0	
	004234	000000	0	
	004236	000000	0	
	004240	000000	0	
502				
503			:ERROR	246 "ATA" NOT RESET BY GO WHEN "ERR" = 0
504	004242	071752	EMT246	
	004244	073332	EHT1	
	004246	073456	EDT1	
	004250	073546	EET1	
505				
506			:ERROR	247 "ATA" NOT RESET BY WRITING RMAS
507	004252	071762	EMT247	
	004254	073332	EHT1	
	004256	073456	EDT1	
	004260	073546	EFT1	
508				
509			:ERROR	250 "ATA" WAS RESET BY GO WHEN "ERR" = 1
510	004262	071774	EMT250	
	004264	073332	EHT1	
	004266	073456	EDT1	
	004270	073546	EFT1	
511				
512			:ERROR	251 PROGRAM INTERRUPT WAS NOT GENERATED
513				

	004272	072010	EMT251
	004274	073344	EHT2
	004276	073466	EDT2
	004300	073556	EFT2
514			
515			:ERROR 252 PROGRAM INTERRUPT SHOULD NOT HAVE BEEN GENERATED
516			
	004302	072016	EMT252 -
	004304	073344	EHT2
	004306	073466	EDT2
	004310	073556	EFT2
517			
518			:ERROR 253 OFFSET MODE WAS NOT RESET BY WRITING RMDC
519			
	004312	072024	EMT253
	004314	073332	EHT1
	004316	073456	EDT1
	004320	073546	EFT1
520			
521			:ERROR 254 INCORRECT "ILF" STATUS
522			
	004322	072042	EMT254
	004324	073332	EHT1
	004326	073456	EDT1
	004330	073546	EFT1
523			
524			:ERROR 255 INCORRECT "ATA" STATUS
525			
	004332	072052	EMT255
	004334	073332	EHT1
	004336	073456	EDT1
	004340	073546	EFT1
526			
527			:ERROR 256 INCORRECT "ILR" STATUS
528			
	004342	072062	EMT256
	004344	073402	EHT256
	004346	073502	EDT223
	004350	073572	EFT223
529			
530			:ERROR 257 INVALID IAE STATUS DURING SEARCH COMMAND
531			
	004352	072072	EMT257
	004354	073332	EHT1
	004356	073456	EDT1
	004360	073546	EFT1
532			
533			:ERROR 260 "IVC" WAS NOT DETECTED DURING SEARCH COMMAND
534			
	004362	072104	EMT260

	004364	073332		EHT1	
	004366	073456		EDT1	
	004370	073546		EFT1	
535					
536			;ERROR	261	DRIVE EXECUTED SEARCH WITH ERROR SET
537					
	004372	072116		EMT261	
	004374	073332		EHT1	
	004376	073456		EDT1	
	004400	073546		EFT1	
538					
539			;ERROR	262	"LBC" ERROR NOT SET DURING DIAGNOSTIC MODE
540					
	004402	072136		EMT262	
	004404	073332		EHT1	
	004406	073456		EDT1	
	004410	073546		EFT1	
541					
542			;ERROR	263	"SKI" ERROR DURING SEARCH COMMAND
543					
	004412	072146		EMT263	
	004414	073332		EHT1	
	004416	073456		EDT1	
	004420	073546		EFT1	
544					
545			;ERROR	264	"IVC" ERROR DURING SEARCH - LOST VOLUME VALID
546					
	004422	072156		EMT264	
	004424	073332		EHT1	
	004426	073456		EDT1	
	004430	073546		EFT1	
547					
548			;ERROR	265	ERRONEOUS IVC ERROR DURING SEARCH - VOLUME IS VALID
549					
	004432	072176		EMT265	
	004434	073332		EHT1	
	004436	073456		EDT1	
	004440	073546		EFT1	
550					
551			;ERROR	266	DEVICE FAULT (DVC) DURING SEARCH
552					
	004442	072216		EMT266	
	004444	073332		EHT1	
	004446	073456		EDT1	
	004450	073546		EFT1	
553					
554			;ERROR	267	SKI SHOULD HAVE BEEN SET BECAUSE CYLINDER
555			:		ADDRESS IS TOO LARGE
556					
	004452	072230		EMT267	

ERROR POINTER TABLE

	004454	073332		EHT1
	004456	073456		EDT1
	004460	073546		EFT1
557				
558			:ERROR 270	OPI ERROR DURING SEARCH BECAUSE MOL = 0
559				
	004462	072246		EMT270
	004464	073332		EHT1
	004466	073456		EDT1
	004470	073546		EFT1
560				
561			:ERROR 271	OPI ERROR DURING SEARCH BECAUSE ON CYLINDER
562			:	DIDN'T DROP
563				
	004472	072262		EMT271
	004474	073332		EHT1
	004476	073456		EDT1
	004500	073546		EFT1
564				
565			:ERROR 272	LOST MOL DURING SEARCH, OPI IS NOT SET
566				
	004502	072300		EMT272
	004504	073332		EHT1
	004506	073456		EDT1
	004510	073546		EFT1
567				
568			:ERROR 273	PIP STIL SET AFTER SEARCH
569				
	004512	072316		EMT273
	004514	073332		EHT1
	004516	073456		EDT1
	004520	073546		EFT1
570				
571			:ERROR 274	PARITY ERROR OCCURRED WHILE WRITING REMOTE
572			:	REGISTERS BUT MXF DID NOT SET
573				
	004522	072334		EMT274
	004524	073332		EHT1
	004526	073456		EDT1
	004530	073546		EFT1
574				
575			:ERROR 275	MXF ERROR - COMPOSITE ERROR OCCURRED BEFORE DATA
576			:	COMMAND STARTED
577				
	004532	072352		EMT275
	004534	073332		EHT1
	004536	073456		EDT1
	004540	073546		EFT1
578				
579			:ERROR 276	"OPI" ERROR DURING DATA TRANSFER BECAUSE "MOL" WAS

Line	Address	Offset	Instruction	Comment
580				ZERO
581	004542	072364	EMT276	
	004544	073332	EHT1	
	004546	073456	EDT1	
	004550	073546	EFT1	
582				
583			:ERROR 277	"OPI" ERROR DURING DATA TRANSFER BECAUSE 1) ON
584			:	CYLINDER DIDN'T DROP OR 2) SEARCH TIMED OUT OR
585			:	3) RUN TIMED OUT
586	004552	072400	EMT277	
	004554	073332	EHT1	
	004556	073456	EDT1	
	004560	073546	EFT1	
587				
588			:ERROR 300	"IVC" ERROR DURING DATA TRANSFER BECAUSE VOLUME
589			:	WAS NOT VALID
590	004562	072416	EMT300	
	004564	073332	EHT1	
	004566	073456	EDT1	
	004570	073546	EFT1	
591				
592			:ERROR 301	ERRONEOUS "IVC" ERROR DURING DATA TRANSFER - VOLUME
593			:	IS VALID
594	004572	072436	EMT301	
	004574	073332	EHT1	
	004576	073456	EDT1	
	004600	073546	EFT1	
595				
596			:ERROR 302	"ILR" ERROR DURING DATA TRANSFER
597	004602	072460	EMT302	
	004604	073332	EHT1	
	004606	073456	EDT1	
	004610	073546	EFT1	
598				
599			:ERROR 303	"ILF" ERROR DURING DATA TRANSFER
600	004612	072472	EMT303	
	004614	073332	EHT1	
	004616	073456	EDT1	
	004620	073546	EFT1	
601				
602			:ERROR 304	"RMR" ERROR DURING DATA TRANSFER
603	004622	072504	EMT304	
	004624	073332	EHT1	
	004626	073456	EDT1	

Line	Address	Code	Message
	004630	073546	EFT1
604			
605			
606			:ERROR 305 INCORRECT "IAE" STATUS DURING DATA TRANSFER
	004632	072516	EMT305
	004634	073332	EHT1
	004636	073456	EDT1
	004640	073546	EFT1
607			
608			
609			:ERROR 306 "SKI" ERROR DURING DATA TRANSFER
	004642	072530	EMT306
	004644	073332	EHT1
	004646	073456	EDT1
	004650	073546	EFT1
610			
611			
612			:ERROR 307 DRIVE DID NOT DETECT SKI ERROR DUE TO CYLINDER
	004652	072540	EMT307
	004654	073332	EHT1
	004656	073456	EDT1
	004660	073546	EFT1
613			
614			
615			:ERROR 310 DEVICE FAULT DURING DATA TRANSFER
	004662	072560	EMT310
	004664	073332	EHT1
	004666	073456	EDT1
	004670	073546	EFT1
616			
617			
618			:ERROR 311 LOSS OF BIT CLOCK DURING DATA TRANSFER
	004672	072572	EMT311
	004674	073332	EHT1
	004676	073456	EDT1
	004700	073546	EFT1
619			
620			
621			:ERROR 312 LOSS OF SYSTEM CLOCK DURING DATA TRANSFER
	004702	072604	EMT312
	004704	073332	EHT1
	004706	073456	EDT1
	004710	073546	EFT1
622			
623			
624			:ERROR 313 UNSAFE ERROR DURING DATA TRANSFER (DVC = 0)
	004712	072616	EMT313
	004714	073332	EHT1
	004716	073456	EDT1
	004720	073546	EFT1

625				
626			:ERROR 314	DRIVE TIMING ERROR DURING DATA TRANSFER
627	004722	072636		EMT314
	004724	073332		EHT1
	004726	073456		EDT1
	004730	073546		EFT1
628				
629			:ERROR 315	WRITE LOCK ERROR
630	004732	072650		EMT315
	004734	073332		EHT1
	004736	073456		EDT1
	004740	073546		EFT1
631				
632			:ERROR 316	ERRONEOUS WRITE LOCK ERROR
633	004742	072662		EMT316
	004744	073332		EHT1
	004746	073456		EDT1
	004750	073546		EFT1
634				
635			:ERROR 317	HEADER CRC ERROR DURING DATA TRANSFER
636	004752	072674		EMT317
	004754	073332		EHT1
	004756	073456		EDT1
	004760	073546		EFT1
637				
638			:ERROR 320	FORMAT ERROR DURING DATA TRANSFER
639	004762	072704		EMT320
	004764	073332		EHT1
	004766	073456		EDT1
	004770	073546		EFT1
640				
641			:ERROR 321	HEADER COMPARE ERROR DURING DATA TRANSFER
642	004772	072714		EMT321
	004774	073332		EHT1
	004776	073456		EDT1
	005000	073546		EFT1
643				
644			:ERROR 322	HEADER ERRORS SHOULD NOT BE SET
645	005002	072724		EMT322
	005004	073332		EHT1
	005006	073456		EDT1
	005010	073546		EFT1

Line No.	Address	Code	Module	Description
646				
647				
648	005012	072732	EMT323	:ERROR 323 DATA CHECK ERROR DURING DATA TRANSFER
	005014	073332	EHT1	
	005016	073456	EDT1	
	005020	073546	EFT1	
649				
650				
651	005022	072742	EMT324	:ERROR 324 CORRECTABLE DATA CHECK ERROR DURING DATA TRANSFER
	005024	073332	EHT1	
	005026	073456	EDT1	
	005030	073546	EFT1	
652				
653				
654	005032	072754	EMT325	:ERROR 325 UNCORRECTABLE DATA CHECK ERROR DURING DATA TRANSFER
	005034	073332	EHT1	
	005036	073456	EDT1	
	005040	073546	EFT1	
655				
656				
657	005042	072766	EMT326	:ERROR 326 DATA PARITY ERROR DURING READ COMMAND
	005044	073332	EHT1	
	005046	073456	EDT1	
	005050	073546	EFT1	
658				
659				
660	005052	073004	EMT327	:ERROR 327 OFFSET MODE NOT RESET BY WRITE COMMAND
	005054	073332	EHT1	
	005056	073456	EDT1	
	005060	073546	EFT1	
661				
662				
663	005062	073016	EMT330	:ERROR 330 DATA PARITY ERROR DURING WRITE COMMAND
	005064	073332	EHT1	
	005066	073456	EDT1	
	005070	073546	EFT1	
664				
665				
666	005072	073026	EMT331	:ERROR 331 WRITE CLOCK FAILURE DURING WRITE COMMAND
	005074	073332	EHT1	
	005076	073456	EDT1	
	005100	073546	EFT1	
667				

668			:ERROR 332	DATA LATE ERROR DURING DATA TRANSFER
669	005102	073040		EMT332
	005104	073332		EHT1
	005106	073456		EDT1
	005110	073546		EFT1
670				
671			:ERROR 333	PIP STIL SET AFTER DATA TRANSFER - SKI = 0
672	005112	073052		EMT333
	005114	073332		EHT1
	005116	073456		EDT1
	005120	073546		EFT1
673				
674			:ERROR 334	LOST MOL DURING DATA TRANSFER - OPI = 0
675	005122	073070		EMT334
	005124	073332		EHT1
	005126	073456		EDT1
	005130	073546		EFT1
676				
677			:ERROR 335	LOST VOLUME VALID DURING DATA TRANSFER - IVC = 0
678	005132	073106		EMT335
	005134	073332		EHT1
	005136	073456		EDT1
	005140	073546		EFT1
679				
680			:ERROR 336	DATA READ DOES NOT COMPARE WITH DATA WRITTEN
681	005142	073126		EMT336
	005144	073414		EHT336
	005146	073512		EDT336
	005150	073602		EFT336
682				
683			:ERROR 337	WRITE CHECK ERROR NOT DETECTED
684	005152	073136		EMT337
	005154	073426		EHT337
	005156	073522		EDT337
	005160	073612		EFT337
685				
686			:ERROR 340	WRITE CHECK ERROR AT UNEXPECTED ADDRESS
687	005162	073146		EMT340
	005164	073414		EHT336
	005166	073512		EDT336
	005170	073602		EFT336
688				
689			:ERROR 341	INCORRECT DATA DURING WRITE CHECK ERROR

690	005172 073160	EMT341	
	005174 073414	EHT336	
	005176 073512	EDT336	
	005200 073602	EFT336	
691			
692		:ERROR 342	'IVC' ERROR NOT DETECTED DURING DATA TRANSFER
693	005202 073166	EMT342	
	005204 073332	EHT1	
	005206 073456	EDT1	
	005210 073546	EFT1	
694			
695		:ERROR 343	'FER' NOT DETECTED DURING DATA TRANSFER
696	005212 073200	EMT343	
	005214 073332	EHT1	
	005216 073456	EDT1	
	005220 073546	EFT1	
697			
698		:ERROR 344	'HCE' NOT DETECTED DURING DATA TRANSFER
699	005222 073212	EMT344	
	005224 073440	EHT344	
	005226 073532	EDT344	
	005230 073622	EFT344	
700			
701		:ERROR 345	'BSE' NOT DETECTED DURING DATA TRANSFER
702	005232 073224	EMT345	
	005234 073332	EHT1	
	005236 073456	EDT1	
	005240 073546	EFT1	
703			
704		:ERROR 346	HEADER ERROR WAS DETECTED W/ HCI SET
705	005242 073234	EMT346	
	005244 073332	EHT1	
	005246 073456	EDT1	
	005250 073546	EFT1	
706			
707		:ERROR 347	DATA TRANSFER NOT ABORTED W/ COMP ERROR SET
708	005252 073250	EMT347	
	005254 073332	EHT1	
	005256 073456	EDT1	
	005260 073546	EFT1	
709			
710		:ERROR 350	LOST VOLUME VALID DURING SEARCH - 'IVC' = 0
711			

ERROR POINTER TABLE

	005262	073262		EMT350
	005264	073332		EHT1
	005266	073456		EDT1
	005270	073546		EFT1
712				
713			:ERROR 351	'ATA' DID NOT SET DURING SEARCH
714				
	005272	073300		EMT351
	005274	073332		EHT1
	005276	073456		EDT1
	005300	073546		EFT1
715				
716			:ERROR 352	PROGRAM TIMEOUT WHILE TESTING RMLA
717				
	005302	073310		EMT352
	005304	000000		0
	005306	000000		0
	005310	000000		0
718				
719			:ERROR 353	LOOK AHEAD TEST FAILS
720				
	005312	073314		EMT353
	005314	073452		EHT353
	005316	073544		EDT353
	005320	073632		EFT353
721				
722			:ERROR 354	BSE SHOULD NOT BE SET
723				
	005322	073324		EMT354
	005324	073332		EHT1
	005326	073456		EDT1
	005330	073546		EFT1
724				
725			:PUT ERROR TABLE HERE	

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

```
.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
2
3
4 005332 011600
5 005334 005740
6 005336 022626
6 005340 104401 005346
   005344 000417
   005404
7 005404 010046
8 005406 104402
9 005410 000240
10
11
12
13
14 005412 012737 177777 001326
15 005420 000402
16
17 005422 005037 001326
18 005426 000240
19 005430 005227 000000
20 005434 001375
21 005436 000005
22
23

;THIS ROUTINE HANDLES UNEXPECTED TIMEOUTS
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 R0,-(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
START1: MOV #-1,CHGADR ;CHANGE RH/RM BUS ADDRESS
        BR START2
START: CLR CHGADR ;NO CHANGE IN ADDRESS
START2: 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
    
```

```

005640 012716 005646          64$:  MOV    #65$, (SP)          ;;SET UP FOR TRAP RETURN
005644 000002
005646 012737 000176 001154 65$:  MOV    #SWREG,SWR          ;;POINT TO SOFTWARE SWR
005654 012737 000174 001156 66$:  MOV    #DISPREG,DISPLAY
005662 012637 000004          66$:  MOV    (SP)+,@#ERRVEC      ;;RESTORE ERROR VECTOR

005666 005037 001230          CLR    $PASS              ;;CLEAR PASS COUNT
005672 132737 000200 001243 BITB   #APTSIZE,$ENVM      ;;TEST USER SIZE UNDER APT
005700 001403          BEQ    67$                ;;YES,USE NON-APT SWITCH
005702 012737 001244 001154 67$:  MOV    #$$SWREG,SWR      ;;NO,USE APT SWITCH REGISTER
005710

24 005710 012737 005332 000004 ;:SETUP 'TIMEOUT' TRAP VECTOR FOR UNEXPECTED BUS TIMEOUTS
25 005710 012737 005332 000004 MOV    #BADTMO,ERRVEC    ;:SETUP FOR UNEXPECTED TIMEOUT
26 005716 012737 000300 000006 MOV    #PR6,ERRVEC+2    ;:LEVEL 6
27
28 .SBTTL  TYPL PROGRAM NAME
   ;:TYPE THE NAME OF THE PROGRAM IF FIRST PASS
005724 005227 177777          ILL    #-1                ;;FIRST TIME?
005730 001035          BNE    68$                ;;BRANCH IF NO
005732 022737 041630 000042 68$:  CMP    #SENDAD,@#42      ;;ACT-11?
005740 001431          BEQ    68$                ;;BRANCH IF YES
005742 104401 005750          TYPE   ,69$              ;;TYPE ASCIZ STRING
005746 000426          BR     68$                ;;GET OVER THE ASCIZ
   ;:69$: .ASCIZ <CRLF>@CZRMBO - RM05/3/2 FUNCTIONAL TEST, PT 1@<CRLF>
   ;:68$:
   .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
006024 005737 000042          TST    @#42                ;;ARE WE RUNNING UNDER XXDP/ACT?
006030 001012          BNE    70$                ;;BRANCH IF YES
006032 123727 001242 000001 70$:  CMPB   $ENV,#1            ;;ARE WE RUNNING UNDER APT?
006040 001406          BEQ    70$                ;;BRANCH IF YES
006042 023727 001154 000176 71$:  CMP    SWR,#SWREG        ;;SOFTWARE SWITCH REG SELECTED?
006050 001005          BNE    71$                ;;BRANCH IF NO
006052 104407          GTSWR                    ;;GET SOFT-SWR SETTINGS
006054 000403          BR     71$
006056 112737 000001 001150 70$:  MOVB   #1,$AUTOB         ;;SET AUTO-MODE INDICATOR
006064 71$:

29
30 ;:THE FOLLOWING FINDS OUT THE PROGRAM CONTROL MODE:
31 ;:PAPER TAPE (MANUAL), ACT11, XXDP CHAIN OR DUMP
32
33 006064 005037 001330          CLR    XXDP                ;:CLEAR 'XXDP' LOAD DEVICE STORAGE
34 006070 122737 000016 000041 72$:  CMPB   #16,@#41          ;:LOADED FROM AN RM05/3/2 ?
35 006076 001160          BNE    3$                ;:BRANCH IF NOT
36 006100 013737 000040 001330 73$:  MOV    @#40,XXDP         ;:GET DEVICE INDICATOR AND NUMBER
37 006106 122737 000007 001330 74$:  CMPB   #7,XXDP           ;:IS IT A VALID NUMBER ?
38 006114 103002          BHIS   1$                ;:YES
39 006116 105037 001330          CLRB   XXDP              ;:NO, DEFAULT TO DRIVE 0
40 006122 005737 000042          TST    @#42              ;:CHAIN MODE OR ACT11 AUTO ACCEPT ?
41 006126 001425          BEQ    2$                ;:BR IF NEITHER
42 006130 104401 006136          TYPE   ,73$              ;:TYPE ASCIZ STRING
006134 000412          BR     72$              ;:GET OVER THE ASCIZ
   ;:73$: .ASCIZ <CRLF>/NOT TESTING DRIVE /
   ;:72$:
43 006162 005046          CLR    -(SP)              ;:CLEAR WORD ON STACK
44 006164 113716 001330 75$:  MOVB   XXDP,(SP)         ;:GET DRIVE ADDRESS
45 006170 104403          TYPOS                    ;:TYPE THE ADDRESS
46 006172 001          .BYTE  1                ;:ONLY 1 CHARACTER

```

```

47 006173      000
48 006174 104401 001217
49 006200 000517
50
51 006202 005227 177777
52 006206 001114
53 006210 104401 006216
    006214 000410
    006236
54 006236 005046
55 006240 113716 001330
56 006244 104403
57 006246      001
58 006247      000
59 006250 104401 006256
    006254 000431
    006340
60 006340 104401 006346
    006344 000435
    006440
64
65 006440 005737 000042
66 006444 001561
67 006446 012737 000377 001300
68
69
70 006454
71 006454 132737 000200 001243
72 006462 001146
73
74 006464 005001
75 006466 013700 001276
76 006472 104401 066720
77
78 006476 136137 067236 001300
79 006504 001531
80 006506 104401 001217
81 006512 010146
    006514 104403
    006516      002
    006517      000
82 006520 104401 067130
83
84 006524 012760 000040 000010
85 006532 010160 000010
86 006536 005760 000012
87 006542 032760 010000 000010
88 006550 001051
89 006552 032760 004000 000000
90 006560 001450
91 006562 016002 000026
92 006566 012737 066736 006766
93 006574 022702 020025
94 006600 001430

    .BYTE 0 ;SUPPRESS LEADING ZEROS
    TYPE , $CRLF ;CR-LF
    BR 3$ ;GET NUMBER OF DRIVES

2$: INC #-1 ;FIRST TIME THRU HERE ?
    BNE 3$ ;NO
    TYPE , 75$ ;:TYPE ASCIZ STRING
    BR 74$ ;:GET OVER THE ASCIZ
    ;:75$: .ASCIZ <CRLF>/TO TEST DRIVE /
    ;:74$:

    CLR -(SP) ;CLEAR WORD ON STACK
    MOVB XXDP,(SP) ;GET DRIVE ADDRESS
    TYPOS ;TYPE DRIVE ADDRESS
    .BYTE 1 ;ONLY 1 CHARACTER
    .BYTE 0 ;SUPPRESS LEADING ZEROS
    TYPE , 77$ ;:TYPE ASCIZ STRING
    BR 76$ ;:GET OVER THE ASCIZ
    ;:77$: .ASCIZ /, HALT PROGRAM, REMOVE RRDP PACK AND REPLACE IT/<CRLF>
    ;:76$:

    TYPE , 78$ ;:TYPE ASCIZ STRING
    BR 3$ ;:GET OVER THE ASCIZ
    ;:78$: .ASCIZ /WITH A WORK PACK, CLEAR LOCATION 40 AND RESTART PROGRAM./<CRLF>
    ;:3$:

    ;CHECK FOR AUTO MODE OR STANDLONE MODE
    TST @#42 ;RUNNING IN AUTO MODE ?
    BEQ STANDALONE ;BR IF NO
    MOV #377,$DEV ;SET DEVICE MAP FOR ALL DRIVES

    ;PROGRAM IS RUNNING IN AUTO MODE - SEE IF SIZING IS ALLOWED
    XSIZ:
    BITB #BIT7,$ENV ;SIZING ALLOWED ?
    BNE 12$ ;NO

    CLR R1 ;START FROM DRIVE 0
    MOV $BASE,R0 ;LOAD THE BASE ADDRESS
    TYPE ,SYSTAT ;TYPE 'UNIT STATUS:'

1$: BITB ATNTBL(R1),$DEV ;IS DEVICE PRESENT IN MAP ?
    BEQ 11$ ;BR IF NO
    TYPE , $CRLF ;CR-LF
    MOV R1,-(SP) ;SAVE R1 FOR TYPEOUT
    TYPOS ;GO TYPE--OCTAL ASCII
    .BYTE 2 ;TYPE 2 DIGIT(S)
    .BYTE 0 ;SUPPRESS LEADING ZEROS
    TYPE ,BLNKS4 ;TYPE 4 BLANKS

    MOV #CLR,RMCS2(R0) ;CLEAR MASS BUS
    MOV R1,RMCS2(R0) ;LOAD THE DRIVE ADDRESS
    TST RMD5(R0) ;ACCESS DRIVE REGISTER
    BIT #NED,RMCS2(R0) ;IS DRIVE PRESENT ?
    BNE 3$ ;BR IF NO
    BIT #DVA,RMCS1(R0) ;IS DRIVE AVAILABLE ?
    BEQ 4$ ;BR IF NO
    MOV RMDT(R0),R2 ;SAVE DRIVE TYPE REGISTER IN R2
    MOV #RM02,10$ ;ASSUME RM02 DEVICE
    CMP #20025,R2 ;SINGLE PORT RM02 ?
    BEQ 2$ ;BR IF YES
    
```

95	006602	022702	024025			CMP	#24025,R2	:DUAL PORT RM02 ?
96	006606	001425				BEQ	2\$:BR IF YES
97	006610	012737	066743	006766		MOV	#\$RM03,10\$:ASSUME RM03 DEVICE
98	006616	022702	020024			CMP	#20024,R2	:SINGLE PORT RM03 ?
99	006622	001417				BEQ	2\$:BR IF YES
100	006624	022702	024024			CMP	#24024,R2	:DUAL PORT RM03 ?
101	006630	001414				BEQ	2\$:BR IF YES
102	006632	012737	066750	006766		MOV	#\$RM05,10\$:ASSUME RM05 DEVICE
103	006640	022702	020027			CMP	#20027,R2	:SINGLE PORT RM05 ?
104	006644	001406				BEQ	2\$:BR IF YES
105	006646	022702	024027			CMP	#24027,R2	:DUAL PORT RM05 ?
106	006652	001403				BEQ	2\$:BR IF YES
107	006654	104401	066755			TYPE	,NOTRM	:DRIVE NOT AN RM05/3/2
108	006660	000412				BR	5\$:CHECK NEXT DRIVE
109	006662	032760	010000	000012	2\$:	BIT	#MOL,RMDS(R0)	:IS MEDIUM ON LINE ?
110	006670	001412				BEQ	6\$:BR IF NO
111	006672	000417				BR	7\$	
112								
113	006674	104401	067013		3\$:	TYPE	,NOTPRS	:DRIVE NOT PRESENT
114	006700	000402				BR	5\$:CHECK NEXT DRIVE
115								
116	006702	104401	067030		4\$:	TYPE	,NOTAVL	:DRIVE NOT AVAILABLE
117	006706	146137	067236	001300	5\$:	BICB	ATNTBL(R1),SDEV	:CLEAR DEVICE FROM BIT MAP
118	006714	000425				BR	11\$:CHECK NEXT DRIVE
119								
120	006716	104401	067047		6\$:	TYPE	,UNTOFF	:DRIVE OFFLINE
121	006722	146137	067236	001300		BICB	ATNTBL(R1),SDEV	:CLEAR DEVICE FROM BIT MAP
122	006730	000413				BR	9\$:PRINT DRIVE TYPE
123								
124	006732	005737	001330		7\$:	TST	XXDP	:LOADED FROM RM05/3/2 ?
125	006736	001406				BEQ	8\$:NO
126	006740	123701	001330			CMPB	XXDP,R1	:IS THIS THE DRIVE ?
127	006744	001360				BNE	5\$:BR IF NO
128	006746	104401	066776			TYPE	,LODEV	:DRIVE IS LOAD DEVICE
129	006752	000755				BR	5\$	
130								
131	006754	104401	067060		8\$:	TYPE	,UNTON	:DRIVE ONLINE
132	006760	104401	067132		9\$:	TYPE	,BLNKS2	:TYPE 2 BLANKS
133	006764	104401				TYPE		:PRINT DRIVE TYPE
134	006766	000000			10\$:	.WORD	0	:MESSAGE ADDRESS HERE
135								
136	006770	005201			11\$:	INC	R1	:INCREMENT THE DRIVE ADDRESS
137	006772	020127	000007			CMP	R1,#7	:ALL DRIVES ARE CHECKED ?
138	006776	003637				BLE	1\$:BRANCH IF NOT
139								
140	007000	104401	001217		12\$:	TYPE	,SCRLF	:CR-LF
141	007004	000137	007466			JMP	CMNSTART	:JUMP TO COMMON START


```

1      .SBTTL  STANDALONE INPUT ROUTINES
2
3      STANDALONE:
4      007010      004737  064210      JSR      PC,$TKINT      :INITIALIZE CONSOLE
5
6      007014      005227  177777      INC      #-1           :FIRST TIME THRU HERE ?
7      007020      001023              BNE      2$           :BR IF NO
8
9      :SEE IF OPERATOR WANTS HELP TEXT
10
11     007022      104401  066344      TYPE    ,MSHELP      :WANT HELP ?
12     007026      104411              RDCHR                   :GET RESPONSE
13     007030      012637  001176      MOV     (SP)+,$TMP1   :SAVE AND ECHO RESPONSE
14     007034      123727  001176      CMPB   $TMP1,#'Y     :WAS IT A YES RESPONSE ?
15     007042      001005              BNE      1$           :NO
16     007044      104401  001176      TYPE    , $TMP1      :TYPE 'Y'
17     007050      104401  103274      TYPE    ,HELP ;      :YES - TYPE HELP TEXT
18     007054      000414              BR       3$
19     007056      104401  067124      1$:    TYPE    ,N      :TYPE 'N'
20     007062      104401  001217      TYPE    , $CRLF      :CR-LF
21     007066      000407              BR       3$
22
23     :SEE IF USER WANTS TO CHANGE UNIBUS ADDRESS
24     007070      2$:
25     007070      005737  001326      TST    CHGADR        :CHANGE RH/RM BUS ADDRESS ?
26     007074      001457              BEQ     7$           :BR IF NO
27     007076      005737  001326      CLR    CHGADR        :NO CHANGE NEXT TIME
28     007102      104401  001217      TYPE    , $CRLF      :CR-LF
29
30     :DIALOGUE TO CHANGE THE UNIBUS ADDRESS, VECTOR ADDRESS AND INTERRUPT PRIORITY
31     3$:
32     007106      104401  066375      TYPE    ,CNSLO1      :TYPE CURRENT BUS ADDRESS
33     007112      013746  001276      MOV     $BASE,-(SP)  :SAVE $BASE FOR TYPEOUT
34     007116      104402              TYPOC                   :GO TYPE--OCTAL ASCII(ALL DIGITS)
35     007120      104401  067132      TYPE    ,BLNKS2     :TYPE 2 BLANKS
36     007124      104413              RDOCT                   :GET NEW BUS ADDRESS
37     007126      012637  001176      MOV     (SP)+,$TMP1   :CARRIAGE RETURN ?
38     007132      001412              BEQ     5$           :YES-SKIP TO NEXT ENTRY
39     007134      022737  160000  001176      CMP     #160000,$TMP1 :BASE ADDRESS IN I/O PAGE ?
40     007142      101403              BLOS   4$           :YES
41     007144      104401  066405      TYPE    ,CNSLO2     :TYPE WARNING MESSAGE
42     007150      000756              BR       3$         :TRY AGAIN
43     007152      013737  001176  001276  4$:    MOV     $TMP1,$BASE  :STORE NEW BUS ADDRESS
44
45     5$:    TYPE    ,CNSLO3
46     007160      104401  066447      CLR     -(SP)
47     007164      005046              MOV     $VECT1,(SP)  :GET CURRENT VECTOR ADDRESS
48     007166      113716  001272      TYPOC                   :TYPE 2 BLANKS
49     007172      104402              RDOCT                   :GET NEW VECTOR ADDRESS
50     007174      104401  067132      TYPE    ,BLNKS2     :TYPE 2 BLANKS
51     007200      104413              RDOCT                   :GET NEW VECTOR ADDRESS
52     007202      012637  001176      MOV     (SP)+,$TMP1   :CARRIAGE RETURN?
53     007206      001412              BEQ     7$           :YES-SKIP TO NEXT ENTRY
54     007210      022737  001000  001176      CMP     #1000,$TMP1  :VECTOR ADDRESS < 1000 ?
55     007216      101003              BHI    6$           :YES!!
56     007220      104401  066456      TYPE    ,CNSLO4     :TYPE WARNING MESSAGE
57     007224      000755              BR       5$         :RETRY
58     007226      113737  001176  001272  6$:    MOV     $TMP1,$VECT1 :STORE NEW VECTOR ADDRESS

```

```

57
58 ;DIALOGUE TO INPUT DEVICE NUMBERS
59 007234 005227 177777 7$: INC #-1 ;FIRST TIME THRU ?
60 007240 001002 BNE 8$ ;BR IF NO
61 007242 104401 066512 TYPE ,CNSLO7 ;TYPE INPUT INSTRUCTIONS
62 007246 104401 001217 8$: TYPE ,$CRLF ;CR-LF
63 007252 005037 001300 9$: CLR $DEVM ;CLEAR DEVICE MAP
64 007256 104401 066676 TYPE ,MSDRVS ;TYPE 'DRIVE(S): '
65 007262 104411 RDCHR
66 007264 012637 001176 MOV (SP)+,$TMP1 ;GET RESPONSE
67 007270 023727 001176 000101 CMP $TMP1,#'A ;IS INPUT 'A' ?
68 007276 001007 BNE 10$ ;NO
69 007300 104401 066330 TYPE ,ALL ;YES, TYPE 'ALL' AND GO
70 007304 012737 000377 001300 MOV #377,$DEVM ;SET DEVICE MAP FOR ALL DRIVES
71 007312 000137 006454 JMP XSIZ ;AUTO SIZE.
72
73 007316 023727 001176 000015 10$: CMP $TMP1,#CR ;CARRIAGE RETURN ?
74 007324 001436 BEQ 12$ ;YES
75 007326 104401 001176 TYPE , $TMP1 ;ECHO RESPONSE
76 007332 023727 001176 000060 CMP $TMP1,#'0 ;NUMBER < 0 ?
77 007340 002430 BLT 12$ ;YES
78 007342 023727 001176 000067 CMP $TMP1,#'7 ;NUMBER > 7 ?
79 007350 003427 BLE 13$ ;NO
80 007352 000423 BR 12$ ;ILLEGAL INPUT
81
82 007354 104411 11$: RDCHR
83 007356 012637 001176 MOV (SP)+,$TMP1 ;GET RESPONSE
84 007362 023727 001176 000015 CMP $TMP1,#CR ;CARRIAGE RETURN ?
85 007370 001432 BEQ 14$ ;YES
86 007372 104401 066341 TYPE ,COMMA ;TYPE ','
87 007376 104401 001176 TYPE , $TMP1 ;ECHO RESPONSE
88 007402 023727 001176 000060 CMP $TMP1,#'0 ;NUMBER < 0 ?
89 007410 002404 BLT 12$ ;YES
90 007412 023727 001176 000067 CMP $TMP1,#'7 ;NUMBER > 7 ?
91 007420 003403 BLE 13$ ;NO
92 007422 104401 066654 12$: TYPE ,CNSLO8 ;TYPE CR-LF '' ?ILLEGAL INPUT''
93 007426 000711 BR 9$ ;RETRY
94
95 007430 013701 001176 13$: MOV $TMP1,R1 ;R1 = DRIVE NUMBER
96 007434 042701 177770 BIC #'C7,R1
97 007440 156137 067236 001300 BISB ATNTBL(R1),$DEVM ;SET DEVICE IN MAP
98 007446 122737 000377 001300 CMPB #377,$DEVM ;DONE ?
99 007454 101337 BHI 11$ ;NO
100 007456 104401 001217 14$: TYPE , $CRLF ;CR-LF
101 007462 000137 006454 JMP XSIZ ;GO SIZE DEVICES

```

```

1      ;ASSEMBLE TEST QUE FROM DEVICE MAP
2 007466 CMNSTART:
3 007466 104401 067070      TYPE      ,DRIVES      ;TYPE 'DRIVE(S) TO BE TESTED'
4 007472 013700 001300      MOV      $DEVM,R0      ;R0 = DEVICE MAP
5 007476 001004              BNE      1$             ;BR IF DRIVES TO TEST
6 007500 104401 066341      TYPE      ,COMMA      ;TYPE ','
7 007504 104401 067117      TYPE      ,NONE      ;TYPE 'NONE'
8 007510 012701 001466      1$:      MOV      #TSTQUE+2,R1 ;R1 = ADDRESS OF FIRST ENTRY IN QUE
9 007514 010137 001464      MOV      R1,TSTQUE    ;INITIALIZE ENTRY POINTER
10 007520 012702 000001      MOV      #1,R2        ;R2 = DEVICE POINTER
11 007524 005003              CLR      R3            ;R3 = DEVICE NUMBER
12 007526 030200              2$:      BIT      R2,R0    ;IS THIS DEVICE IN MAP ?
13 007530 001413              BEQ      3$           ;NO !!
14 007532 104401 066341      TYPE      ,COMMA      ;TYPE ','
15 007536 010311              MOV      R3,(R1)      ;YES - ENTER DEVICE NUMBER IN QUE
16 007540 010346              MOV      R3,-(SP)     ;SAVE R3 FOR TYPEOUT
    007542 104403              TYPOS    ;GO TYPE--OCTAL ASCII
    007544 001              .BYTE   1            ;TYPE 1 DIGIT(S)
    007545 000              .BYTE   0            ;SUPPRESS LEADING ZEROS
17 007546 116361 067236 000001 MOVVB   ATNTBL(R3),1(R1) ;ENTER ATTENTION BIT IN QUE
18 007554 062701 000002      ADD     #2,R1        ;ADVANCE ENTRY POINTER
19 007560 006302              3$:      ASL     R2          ;ADVANCE DEVICE POINTER
20 007562 105702              TSTB    R2           ;DONE ALL DEVICES ?
21 007564 001402              BEQ     4$           ;YES
22 007566 005203              INC     R3           ;ADVANCE DEVICE NUMBER
23 007570 000756              BR      2$          ;ENTER NEXT DEVICE
24 007572 005011              4$:      CLR     (R1)    ;TERMINATE TEST QUE
25 007574 104401 001217      TYPE    ,SCRLF      ;TYPE CR-LF
26
27      ;SIZE FOR CLOCK
28 007600 004737 043400      JSR     PC,SIZCLK    ;SEE IF CLOCK PRESENT
29 007604 000425              BR      6$           ;YES - CLOCK IS PRESENT
30 007606 104401 007614      TYPE    ,65$        ;TYPE ASCII STRING
    007612 000413              BR      64$         ;GET OVER THE ASCII
    007642              65$:      .ASCIIZ <CRLF>/NO 'L' OR 'P' CLOCK/
31 007642 005737 000042      64$:      TST     @#42        ;ANY MONITOR PRESENT ?
32 007646 001002              BNE     5$           ;BR IF YES
33 007650 000137 005422      JMP     START        ;JUMP TO START
34 007654 000137 041620      5$:      JMP     $GET42     ;RETURN CONTROL TO MONITOR
35 007660
36
37 007660 000240              READY:  NOP           ;READY TO START TEST
38 007662 105737 001300      TSTB   $DEVM        ;ANY DRIVES IN MAP ?
39 007666 001007              BNE     2$           ;BR IF YES
40 007670 005737 000042      TST    @#42        ;ANY MONITOR PRESENT ?
41 007674 001002              BNE     1$           ;BR IF YES
42 007676 000137 005422      JMP     START        ;JUMP TO START
43 007702 000137 041620      1$:      JMP     $GET42     ;RETURN CONTROL TO MONITOR
44
45 007706 105037 001116      2$:      CLRB   $TSTNM     ;RESET TEST NUMBER
46 007712 005037 001206      CLR    $TIMES      ;INITIALIZE NUMBER OF ITERATIONS
47 007716 004737 064210      JSR    PC,$TKINT   ;INITIALIZE TTY
48 007722 012746 000240      MOV    #PR5,-(SP)  ;PUT NEW PS ON STACK
    007726 012746 007734      MOV    #64$,-(SP) ;PUT NEW PC ON STACK
    007732 000002              RTI              ;POP NEW PC AND PS
    007734              64$:

```

```
49 007734 117737 171524 001234      MOV      @TSTQUE,$UNIT      ;LOAD DRIVE NUMBER
50
51                                     ;CLEAR MASSBUS CONTROLLER, SELECT DRIVE AND DETERMINE THE LAST TRACK
52                                     ;OF THE--DIFFERENT DRIVE TYPES
53
54 007742 012737 002000 001332      MOV      #TA4,LSTRK        ;ASSUME LAST TRACK FOR RM02/3 = 4.
55 007750 013700 001276                MOV      $BASE,R0          ;R0 = UNIBUS ADDRESS
56 007754 012760 000040 000010      MOV      #CLR,RMCS2(R0)    ;CLEAR MASSBUS
57 007762 117760 171476 000010      MOV      @TSTQUE,RMCS2(R0) ;SELECT DEVICE UNDER TEST
58 007770 016002 000026                MOV      RMDT(R0),R2       ;GET RMDT AND
59 007774 042702 177770                BIC      #177770,R2        ;SAVE DRIVE TYPE BITS
60 010000 022702 000007                CMP      #7,R2             ;IS IT AN RM05 ?
61 010004 001003                BNE      3$                ;NO, MUST BE AN RM02 OR RM03
62 010006 012737 011000 001332      MOV      #TA16!TA2,LSTRK  ;YES--SET LAST TRACK = 18.
63
64                                     ;TYPE DRIVE NUMBER TO BE TESTED($UNIT)
65
66 010014 104401 001217      3$:      TYPE      , $CRLF        ;CR-LF
67 010020 104401 066712      TYPE      ,MSGDRV         ;TYPE 'DRIVE'
68 010024 013746 001234      MOV      $UNIT,-(SP)      ;SAVE $UNIT FOR TYPEOUT
                                ;TYPE DRIVE NUMBER
                                ;GO TYPE--OCTAL ASCII
                                ;TYPE 2 DIGIT(S)
                                ;SUPPRESS LEADING ZEROS
                                ;THESE TWO LOOPS ARE ADDED TO
                                ;WAIT FOR TTY
        010030 104403      TYPOS
        010032      .BYTE 2
        010033      .BYTE 0
69 010034 005004      CLR      R4
70 010036 005304      DEC     R4
71 010040 001376      BNE     -2
72 010042 005304      DEC     R4
73 010044 001376      BNE     -2
```

```

1
010046
010046 000004
010050 000240
010052 012706 001100
010056 013700 001276
010062 013701 001464
010066 012737 000001 001226
2
3 010074 005001
4 010076 013746 000004
010102 013746 000006
5 010106 012737 010210 000004
6 010114 012737 000300 000006
7
8 010122 110160 000001
9 010126 010160 000002
10 010132 016002 000002
11 010136 010160 000004
12 010142 016002 000004
13 010146 016046 000010
14 010152 010160 000010
15 010156 016002 000010
16 010162 012660 000010
17 010166 010160 000022
18 010172 016002 000022
19 010176 012637 000006
010202 012637 000004
20 010206 000417
21
22 010210 022626
23 010212 012637 000006
010216 012637 000004
24 010222 104110
25 010224 005737 000042
26 010230 001002
27 010232 000137 005412
28 010236 005037 001300
29 010242 000137 041432
30 010246
31
32
  
```

```

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

```

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

CLR R1
MOV ERRVEC,-(SP) ;;PUSH ERRVEC ON STACK
MOV ERRVEC+2,-(SP) ;;PUSH ERRVEC+2 ON STACK
MOV #1$,ERRVEC
MOV #PR6,ERRVEC+2

MOV#B R1,RMCS1+1(R0) ;MOVE HI BYTE TO RMCS1
MOV R1,RMWC(R0) ;MOVE WORD COUNT REGISTER
MOV RMWC(R0),R2
MOV R1,RMBA(R0) ;MOVE BUS ADDRESS REGISTER
MOV RMBA(R0),R2
MOV RMCS2(R0),-(SP) ;;PUSH RMCS2(R0) ON STACK
MOV R1,RMCS2(R0) ;MOVE CONTROL STATUS REGISTER
MOV RMCS2(R0),R2
MOV (SP)+,RMCS2(R0) ;;POP STACK INTO RMCS2(R0)
MOV R1,RMDB(R0) ;MOVE DATA BUFFER
MOV RMDB(R0),R2
MOV (SP)+,ERRVEC+2 ;;POP STACK INTO ERRVEC+2
MOV (SP)+,ERRVEC ;;POP STACK INTO ERRVEC
BR 3$ ;NO BUS TIMEOUT OCCURRED

1$: CMP (SP)+,(SP)+ ;ADJUST STACK
MOV (SP)+,ERRVEC+2 ;;POP STACK INTO ERRVEC+2
MOV (SP)+,ERRVEC ;;POP STACK INTO ERRVEC
EMT 110
TST @#42 ;STAND ALONE MODE ?
BNE 2$ ;NO!!
JMP START1 ;YES-GO GET $BASE
2$: CLR $DEVN ;FUDGE NO DRIVES IN MAP
JMP $EOP ;RETURN TO $EOP

3$:
  
```

```

*****
*TEST 2 UNIBUS INITIALIZE TEST
*****
TST2:
  
```

```

010246
010246 000004
010250 000240
010252 012706 001100
010256 013700 001276
010262 013701 001464
010266 012737 000001 001206
010274 012737 000002 001226
33
34 010302 004737 052264
010306 000404
  
```

```

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 #2,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

JSR PC,CNTCLR ;GO ISSUE CONTROLLER CLEAR
BR 1$ ;GO TO 1$ IF NO ERROR
  
```

```

T2 UNIBUS INITIALIZE TEST
010310 000240 NOP ;RETURN HERE IF ERROR
010312 104000 EMT ;ERROR NUMBER DEFINED BY SUBROUTINE
010314 000137 011116 JMP 15$ ;GO TO 15$ IF ERROR
35 010320 1$: MOV #65,R2 ;SET OR AND RESET IR
36 010320 012702 000101 MOV #0,RMDB0 ;WRITE ZEROS IN DATA SILO
37 010324 012737 000000 001432 MOVB #RMDB,PUTINX ;SETUP REGISTER INDEX
38 010332 112737 000022 001543 MOVB #200,PUTINX+1
39 010340 112737 000200 001544
40 010346 2$: JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
41 010346 004737 043160 BR 3$ ;GO TO 3$ IF NO ERROR
010352 000404 NOP ;RETURN HERE IF ERROR
010354 000240 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
010356 104000 JMP 15$ ;GO TO 15$ IF ERROR
42 010364 005302 3$: DEC R2 ;DECREMENT COUNT
43 010366 001367 BNE 2$ ;WRITE SILO AGAIN
44 010370 012737 177777 001414 MOV #-1,RMBA0 ;RMBA = ALL 1'S
45 010376 012737 177777 001424 MOV #-1,RMER10 ;RMER1 = ALL 1'S
46 010404 012737 177777 001452 MOV #-1,RMER20 ;RMER2 = ALL 1'S
47 010412 012737 040001 001434 MOV #DMD!DBEN,RMMR10 ;SET DIAGNOSTIC MODE
48 010420 012737 003577 001410 MOV #003577,RMCS10
49 010426 012737 021037 001420 MOV #021037,RMCS20
50
51 010434 012702 001543 MOV #PUTINX,R2 ;R2 = ADDRESS OF INDEX TABLE
52 010440 112722 000004 MOVB #RMBA,(R2)+
53 010444 112722 000014 MOVB #RMER1,(R2)+
54 010450 112722 000042 MOVB #RMER2,(R2)+
55 010454 112722 000024 MOVB #RMMR1,(R2)+
56 010460 112722 000000 MOVB #RMCS1,(R2)+
57 010464 112722 000010 MOVB #RMCS2,(R2)+
58 010470 112722 000200 MOVB #200,(R2)+
59 010474 004737 043160 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
010500 000404 BR 4$ ;GO TO 4$ IF NO ERROR
010502 000240 NOP ;RETURN HERE IF ERROR
010504 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
60 010512 000005 011116 JMP 15$ ;GO TO 15$ IF ERROR
61 010514 004737 064210 4$: RESET ;UNIBUS INITIALIZE
62 010520 111160 000010 JSR PC,$TKINT ;INITIALIZE CONSOLE
63 010524 004737 042624 MOVB (R1),RMCS2(R0) ;SELECT DEVICE
64 JSR PC,GETSTS ;GO SET UP FOR STATUS FETCH
65 010530 004737 042710 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
010534 000403 BR 5$ ;GO TO 5$ IF NO ERROR
010536 000240 NOP ;RETURN HERE IF ERROR
010540 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
66 010542 000565 BR 15$ ;SKIP REMAINDER OF TEST
67
68 010544 013737 001334 001142 5$: MOV RMCS1I,$BDDAT ;VERIFY RMCS1
69 010552 042737 100000 001142 BIC #SC,$BDDAT ;IGNORE SPECIAL CONDITION
70 010560 012737 004200 001140 MOV #DVA!RDY,$GDDAT ;EXPECT DVA & RDY
71 010566 023737 001140 001142 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED, RECEIVED
72 010574 001401 BEQ 6$ ;BRANCH IF EQUAL
73 010576 104115 EMT 115
74
75 010600 005037 001140 6$: CLR $GDDAT ;VERIFY RMBA IS ZERO
76 010604 013737 001340 001142 MOV RMBAI,$BDDAT
77 010612 001401 BEQ 7$ ;BRANCH IF ZERO

```

```

78 010614 104116 EMT 116
79
80 010616 013737 001344 001142 7$: MOV RMCS2I,$BDDAT ;VERIFY RMCS2
81 010624 005046 CLR -(SP) ;EXPECT IR & UNIT NUMBER
82 010626 111116 MOVB (R1),(SP)
83 010630 052716 000100 BIS #IR,(SP)
84 010634 012637 001140 MOV (SP)+,$GDDAT
85 010640 023737 001140 001142 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED & RECEIVED
86 010646 001401 BEQ 8$ ;BRANCH IF EQUAL
87 010650 104117 EMT 117
88
89 010652 005037 001140 8$: CLR $GDDAT ;VERIFY RMER1
90 010656 013737 001350 001142 MOV RMER1I,$BDDAT
91 010664 042737 040000 001142 BIC #UNS,$BDDAT ;IGNORE UNSAFE
92 010672 001401 BEQ 9$ ;BRANCH IF ZERO
93 010674 104120 EMT 120
94
95 010676 013737 001352 001142 9$: MOV RMASI,$BDDAT ;VERIFY RMAS
96 010704 005002 CLR R2 ;CLEAR ALL BUT THIS
97 010706 116102 000001 MOVB 1(R1),R2 ;DRIVES ATTENTION BIT
98 010712 000302 SWAB R2
99 010714 005102 COM R2
100 010716 000240 NOP
101 010720 040237 001142 BIC R2,$BDDAT
102 010724 001401 BEQ 10$ ;BRANCH IF ITS 0
103 010726 104121 EMT 121
104
105 010730 013737 001360 001142 10$: MOV RMMR1I,$BDDAT ;VERIFY RMMR
106 010736 042737 000046 001142 BIC #WC!LS!LST,$BDDAT ;IGNORE WORD CLOCK, SCT, TRK
107 010744 012737 000010 001140 MOV #MWD,$GDDAT ;EXPECT WRITE DATA BIT
108 010752 023737 001140 001142 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED AND RECEIVED
109 010760 001401 BEQ 11$ ;BRANCH IF 0
110 010762 104122 EMT 122
111
112 010764 005037 001140 11$: CLR $GDDAT ;EXPECT ZEROS
113 010770 013737 001402 001142 MOV RMEC2I,$BDDAT ;VERIFY RMEC2 = 0
114 010776 001401 BEQ 12$
115 011000 104124 EMT 124
116
117 011002 013737 001374 001142 12$: MOV RMMR2I,$BDDAT ;VERIFY RMMR2
118 011010 042737 140000 001142 BIC #RQA!RQB,$BDDAT
119 011016 012737 011777 001140 MOV #TST!1777,$GDDAT ;EXPECT TEST,TAG BIT ON
120 011024 023737 001140 001142 CMP $GDDAT,$BDDAT
121 011032 001401 BEQ 13$
122 011034 104125 EMT 125
123
124 011036 005037 001140 13$: CLR $GDDAT ;EXPECT ALL ZEROS
125 011042 013737 001376 001142 MOV RMER2I,$BDDAT ;VERIFY RMER2
126 011050 042737 040200 001142 BIC #SKI!DVC,$BDDAT ;IGNORE DEVICE ERRORS
127 011056 001401 BEQ 14$ ;BRANCH IF OTHER BITS 0
128 011060 104173 EMT 173
129 011062 013737 001346 001142 14$: MOV RMDSI,$BDDAT ;CHECK DRIVE STATUS REGISTER
130 011070 042737 177177 001142 BIC #^C<DPR!DRY>,$BDDAT
131 011076 012737 000600 001140 MOV #DPR!DRY,$GDDAT ;EXPECTED STATUS
132 011104 023737 001142 001140 CMP $BDDAT,$GDDAT ;COMPARE EXPECTED & RECEIVED STATUS
133 011112 001401 BEQ 15$ ;DRIVE STATUS IS OK
134 011114 104123 EMT 123

```

135 011116
136
137

15\$:

```

:*****
:*TEST 3          CONTROLLER CLEAR TEST
:*****
TST3:

```

```

011116 000004
011120 000240
011122 012706 001100
011126 013700 001276
011132 013701 001464
011136 012737 000003 001226
138
139 011144 004737 052264
011150 000404
011152 000240
011154 104000
011156 000137 011454
140 011162
141 011162 012702 000101
142 011166 012737 000000 001432
143 011174 112737 000022 001543
144 011202 112737 000200 001544
145 011210
146 011210 004737 043160
011214 000404
011216 000240
011220 104000
011222 000137 011454
147 011226 005302
148 011230 001367
149 011232 012737 177777 001414
150 011240 012737 177777 001424
151 011246 012737 177777 001452
152 011254 012737 040001 001434
153 011262 012737 003577 001410
154 011270 012737 021037 001420
155 011276 012702 001543
156 011302 112722 000004
157 011306 112722 000014
158 011312 112722 000042
159 011316 112722 000024
160 011322 112722 000000
161 011326 112722 000010
162 011332 112722 000200
163 011336 004737 043160
011342 000403
011344 000240
011346 104000
164 011350 000430
165
166 011352
167 011352 004737 052264
011356 000404
011360 000240
011362 104000
011364 000137 011454

```

```

SCOPE ;SCOPE CALL
NOP ;START OF TEST
MOV #STACK,SP ;INITIALIZE STACK POINTER
MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
MOV #3,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

JSR PC,CNTCLR ;GO ISSUE CONTROLLER CLEAR
BR 1$ ;GO TO 1$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
EMT ;ERROR NUMBER DEFINED BY SUBROUTINE
JMP 8$ ;GO TO 8$ IF ERROR

1$: MOV #65.,R2
MOV #0,RMDBO ;WRITE ZEROS IN DATA SILO
MOVB #RMDB,PUTINX ;SETUP REGISTER INDEX TABLE
MOVB #200,PUTINX+1

2$: JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
BR 3$ ;GO TO 3$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
EMT ;ERROR # DEFINED BY PUT SUBROUTINE
JMP 8$ ;GO TO 8$ IF ERROR

3$: DEC R2 ;DECREMENT COUNT
BNE 2$ ;AND WRITE SILO AGAIN IF NOT DONE
MOV #-1,RMBA0
MOV #-1,RMER10
MOV #-1,RMER20
MOV #DMD!DBEN,RMMR10
MOV #003577,RMCS10
MOV #021037,RMCS20
MOV #PUTINX,R2 ;R2 = ADDRESS OF INDEX TABLE
MOVB #RMBA,(R2)+
MOVB #RMER1,(R2)+
MOVB #RMER2,(R2)+
MOVB #RMMR1,(R2)+
MOVB #RMCS1,(R2)+
MOVB #RMCS2,(R2)+
MOVB #200,(R2)+
JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
BR 4$ ;GO TO 4$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
EMT ;ERROR # DEFINED BY PUT SUBROUTINE
BR 7$

4$: JSR PC,CNTCLR ;GO ISSUE CONTROLLER CLEAR
BR 5$ ;GO TO 5$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
EMT ;ERROR NUMBER DEFINED BY SUBROUTINE
JMP 8$ ;GO TO 8$ IF ERROR

```



```
168 011370 004737 042624 5$: JSR PC,GETSTS ;SETUP TO READ ALL REGISTERS
169
170 011374 004737 042710 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
011400 000404 BR 6$ ;GO TO 6$ IF NO ERROR
011402 000240 NOP ;RETURN HERE IF ERROR
011404 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
011406 000137 011454 JMP 8$ ;GO TO 8$ IF ERROR
171 011412
172 011412 004737 052402 6$: JSR PC,CLRSTS ;GO VERIFY CONTROLLER CLEAR OPERATION
011416 000405 BR 7$ ;GO TO 7$ IF NO ERROR
011420 000240 NOP ;RETURN HERE IF ERROR
011422 104000 EMT ;ERROR # DEFINED BY CLRSTS SUBROUTINE
011424 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
011426 000137 011454 JMP 8$ ;GO TO 8$ IF ERROR
173 011432
174 011432 012737 000000 001410 7$: MOV #NOP, RMCS10 ;CHANGE FUNCTION CODE FOR ERROR CHECK
175
176 011440 004737 044540 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
011444 000403 BR 8$ ;GO TO 8$ IF NO ERROR
011446 000240 NOP ;RETURN HERE IF ERROR
011450 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
011452 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
177 011454 8$:
178
179
```

```
:::*****
:*TEST 4 ERROR CLEAR TEST
:::*****
TST4:
```

```
011454
011454 000004 SCOPE ;SCOPE CALL
011456 000240 NOP ;START OF TEST
011460 012706 001100 MOV #STACK, SP ;INITIALIZE STACK POINTER
011464 013700 001276 MOV $BASE, R0 ;R0 = UNIBUS ADDRESS
011470 013701 001464 MOV TSTQUE, R1 ;(R1) = DEVICE BEING TESTED
011474 012737 000004 001226 MOV #4, $TESTN ;SET TEST NUMBER IN APT MAIL BOX
180
181 011502 004737 052264 JSR PC,CNTCLR ;GO ISSUE CONTROLLER CLEAR
011506 000404 BR 10$ ;GO TO 10$ IF NO ERROR
011510 000240 NOP ;RETURN HERE IF ERROR
011512 104000 EMT ;ERROR NUMBER DEFINED BY SUBROUTINE
011514 000137 011660 JMP 50$ ;GO TO 50$ IF ERROR
182 011520
183 011520 112737 000000 001543 10$: MOVB #RMCS1, PUTINX ;SETUP PUT INDEX TABLE
011526 112737 000200 001544 MOVB #200, PUTINX+1 ;SET TERMINATOR BYTE
011534 012737 040000 001410 MOV #TRE, RMCS10 ;SET RMCS1 OUTPUT BUFFER = TRE
011542 004737 043160 JSR PC,PUT ;GO WRITE RMCS1 VIA PUT SUBROUTINE
011546 000403 BR 20$ ;GO TO 20$ IF NO ERROR
011550 000240 NOP ;RETURN HERE IF ERROR
011552 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
184 011554 000441 BR 50$ ;SKIP REST OF TEST
185 011556 112737 000000 001514 20$: MOVB #RMCS1, GETINX
186 011564 112737 000010 001515 MOVB #RMCS2, GETINX+1
187 011572 112737 000200 001516 MOVB #200, GETINX+2
188
189 011600 004737 042710 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
011604 000403 BR 30$ ;GO TO 30$ IF NO ERROR
011606 000240 NOP ;RETURN HERE IF ERROR
011610 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
```

```

190 011612 000422          BR      50$          ;SKIP REST OF TEST
191 011614 013737 001334 001142 30$:  MOV    RMCS11,$BDDAT ;CHECK TRE & MCPE
192 011622 005037 001140          CLR    $GDDAT      ;EXPECT 0'S
193 011626 042737 117777 001142          BIC    #^C<TRE!MCPE>,$BDDAT
194 011634 001401          BEQ    40$          ;BRANCH IF TRE & MCPE = 0
195 011636 104137          EMT    137
196
197 011640 013737 001344 001142 40$:  MOV    RMCS21,$BDDAT ;CHECK RMCS2
198 011646 042737 104377 001142          BIC    #^C<WCE!UPE!NED!PGE!MXF!MDPE>,$BDDAT
199 011654 001401          BEQ    50$
200 011656 104140          EMT    140

```

201
202 011660
203
204

50\$:

```

:*****
:*TEST 5      DRIVE STATUS TEST
:*****
TST5:

```

```

011660
011660 000004          SCOPE          ;SCOPE CALL
011662 000240          NOP           ;START OF TEST
011664 012706 001100          MOV    #STACK,SP ;INITIALIZE STACK POINTER
011670 013700 001276          MOV    $BASE,R0  ;R0 = UNIBUS ADDRESS
011674 013701 001464          MOV    TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
011700 012737 000005 001226          MOV    #5,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
205
206 011706 004737 052264          JSR    PC,CNTCLR ;GO ISSUE CONTROLLER CLEAR
011712 000404          BR      1$      ;GO TO 1$ IF NO ERROR
011714 000240          NOP           ;RETURN HERE IF ERROR
011716 104000          EMT           ;ERROR NUMBER DEFINED BY SUBROUTINE
011720 000137 012174          JMP    8$      ;GO TO 8$ IF ERROR
207 011724          1$:
208 011724 004737 042624          JSR    PC,GETSTS
209
210 011730 004737 042710          JSR    PC,GET   ;GO READ REGISTER(S) WITH GET SUBROUTINE
011734 000403          BR      2$      ;GO TO 2$ IF NO ERROR
011736 000240          NOP           ;RETURN HERE IF ERROR
011740 104000          EMT           ;ERROR # DEFINED BY GET SUBROUTINE
211 011742 000514          BR      8$      ;SKIP REST OF TEST
212 011744 032737 010000 001346 2$:  BIT    #MOL,RMDSI ;MEDIUM ON LINE??
213 011752 001016          BNE    3$      ;YES!!
214 011754 013737 001346 001140          MOV    RMDSI,$GDDAT ;EXPECTED STATUS
215 011762 042737 162000 001140          BIC    #ATA!ERR!PIP!LBT,$GDDAT
216 011770 052737 010000 001140          BIS    #MOL,$GDDAT
217 011776 013737 001346 001142          MOV    RMDSI,$BDDAT ;RECEIVED STATUS
218 012004 104151          EMT    151     ;MOL STATUS INCORRECT
219 012006 000440          BR      5$
220
221 012010 032737 000200 001376 3$:  BIT    #DVC,RMER2I ;IS THERE A DEVICE CHECK??
222 012016 001422          BEQ    4$      ;BR IF NO
223 012020 013737 001376 001142          MOV    RMER2I,$BDDAT ;RECEIVED STATUS
224 012026 005037 001140          CLR    $GDDAT  ;EXPECTED STATUS
225 012032 104152          EMT    152     ;DRIVE FAULT
226
227 012034 032737 040000 001350          BIT    #UNS,RMER1I ;IS UNS SET??
228 012042 001022          BNE    5$      ;YES!!
229 012044 013737 001350 001142          MOV    RMER1I,$BDDAT ;RECEIVED STATUS
230 012052 012737 040000 001140          MOV    #UNS,$GDDAT ;EXPECTED STATUS

```

```
231 012060 104153 EMT 153 ;'DVC' IS SET BUT 'UNS' IS NOT
232 012062 000412 BR 5$
233
234 012064 032737 040000 001350 4$: BIT #UNS,RMER1I ;IS UNS SET??
235 012072 001410 BEQ 6$ ;NO!!
236 012074 013737 001350 001142 MOV RMER1I,$BDDAT ;RECEIVED STATUS
237 012102 005037 001140 CLR $GDDAT ;EXPECTED STATUS
238 012106 104154 EMT 154 ;'DVC' IS NOT SET, BUT 'UNS' IS
239 012110 000137 041374 5$: JMP $EOSP ;SKIP REST OF TEST
240
241 012114 032737 004000 001346 6$: BIT #WRL,RMDSI ;IS WRITE PROTECT ON??
242 012122 001412 BEQ 7$ ;NO!!
243 012124 013737 001346 001140 MOV RMDSI,$GDDAT ;EXPECTED STATUS
244 012132 042737 166076 001140 BIC #^C<MOL!PGM!DPR!DRY!VV!OM>,$GDDAT ;RECEIVED STATUS
245 012140 013737 001346 001142 MOV RMDSI,$BDDAT ;SELECTED DEVICE IN WRITE PROTECT
246 012146 104175 EMT 175
247
248 012150 032737 040000 001376 7$: BIT #SKI,RMER2I ;IS SKI SET??
249 012156 001406 BEQ 8$ ;NO!!
250 012160 013737 001376 001142 MOV RMER2I,$BDDAT ;RECEIVED STATUS
251 012166 005037 001140 CLR $GDDAT ;EXPECTED STATUS
252 012172 104205 EMT 205 ;PERSISTENT SEEK INCOMPLETE ERROR
253 012174 8$:
254
255
```

```
::*****
:*TEST 6 PRIMARY/SECONDARY ERROR TEST
::*****
```

```
TST6:
012174 SCOPE ;SCOPE CALL
012174 000004 NOP ;START OF TEST
012176 000240 MOV #STACK,SP ;INITIALIZE STACK POINTER
012200 012706 001100 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
012204 013700 001276 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
012210 013701 001464 MOV #6,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
012214 012737 000006 001226
256
257 012222 004737 052264 JSR PC,CNTCLR ;GO ISSUE CONTROLLER CLEAR
012226 000404 BR 1$ ;GO TO 1$ IF NO ERROR
012230 000240 NOP ;RETURN HERE IF ERROR
012232 104000 EMT ;ERROR NUMBER DEFINED BY SUBROUTINE
012234 000137 012354 JMP 5$ ;GO TO 5$ IF ERROR
258 012240 1$:
259 012240 112737 000000 001543 MOV #RMCS1,PUTINX ;SETUP PUT INDEX TABLE
012246 112737 000200 001544 MOV #200,PUTINX+1 ;SET TERMINATOR BYTE
012254 012737 000023 001410 MOV #PAKACK!GO,RMCS10 ;SET RMCS1 OUTPUT BUFFER = PAKACK!GO
012262 004737 043160 JSR PC,PUT ;GO WRITE RMCS1 VIA PUT SUBROUTINE
012266 000403 BR 2$ ;GO TO 2$ IF NO ERROR
012270 000240 NOP ;RETURN HERE IF ERROR
012272 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
260 012274 000427 BR 5$ ;SKIP REST OF TEST IF ERROR
261 012276 004737 042624 2$: JSR PC,GETSTS ;SETUP FOR STATUS FETCH
262 012302 004737 043522 JSR PC,TIMOUT ;WAIT FOR COMPLETION OF NOP
263
264 012306 004737 042710 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
012312 000403 BR 3$ ;GO TO 3$ IF NO ERROR
012314 000240 NOP ;RETURN HERE IF ERROR
012316 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
265 012320 000415 BR 5$ ;SKIP REST OF TEST IF ERROR
```

```

266 012322 3$:
267 012322 004737 043706 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
    012326 000404 BR 4$ ;GO TO 4$ IF NO ERROR
    012330 000240 NOP ;RETURN HERE IF ERROR
    012332 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
    012334 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
268 012336 000406 BR 5$ ;SKIP REST OF TEST IF ERROR
269 012340
270 012340 004737 044540 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
    012344 000403 BR 5$ ;GO TO 5$ IF NO ERROR
    012346 000240 NOP ;RETURN HERE IF ERROR
    012350 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
    012352 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
271 012354
272
273
    
```

```

*****
:*TEST 7 DIAGNOSTIC MODE TEST
*****
TST7:
    
```

```

012354
012354 000004 SCOPE ;SCOPE CALL
012356 000240 NOP ;START OF TEST
012360 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
012364 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
012370 013701 001464 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
012374 012737 000007 001226 MOV #7,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
274
275 012402 004737 052264 JSR PC,CNTCLR ;GO ISSUE CONTROLLER CLEAR
    012406 000404 BR 1$ ;GO TO 1$ IF NO ERROR
    012410 000240 NOP ;RETURN HERE IF ERROR
    012412 104000 EMT ;ERROR NUMBER DEFINED BY SUBROUTINE
    012414 000137 013426 JMP 21$ ;GO TO 21$ IF ERROR
276 012420
277 012420 112737 000024 001543 1$: MOVB #RMMR1,FUTINX ;SETUP PUT INDEX TABLE
    012426 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
    012434 012737 000001 001434 MOV #DMD,RMMR10 ;SET RMMR1 OUTPUT BUFFER = DMD
    012442 004737 043160 JSR PC,PUT ;GO WRITE RMMR1 VIA PUT SUBROUTINE
    012446 000404 BR 2$ ;GO TO 2$ IF NO ERROR
    012450 000240 NOP ;RETURN HERE IF ERROR
    012452 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
    012454 000137 013426 JMP 21$ ;GO TO 21$ IF ERROR
278 012460 004737 042624 2$: JSR PC,GETSTS ;SETUP FOR STATUS FETCH
279
280 012464 004737 042710 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
    012470 000404 BR 3$ ;GO TO 3$ IF NO ERROR
    012472 000240 NOP ;RETURN HERE IF ERROR
    012474 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
    012476 000137 013426 JMP 21$ ;GO TO 21$ IF ERROR
281 012502 032737 000001 001360 3$: BIT #DMD,RMMR1I ;IS DIAGNOSTIC MODE SET??
282 012510 001011 BNE 4$ ;YES!!
283 012512 012737 000001 001140 MOV #DMD,$GDDAT ;EXPECTED STATUS
284 012520 013737 001360 001142 MOV RMMR1I,$BDDAT ;RECEIVED STATUS
285 012526 104176 EMT 176
286 012530 000137 013426 JMP 21$ ;SKIP REST OF TEST IF DMD = 0
287 012534 032737 010000 001346 4$: BIT #MOL,RMDSI ;IS 'MOL' = 0 ??
288 012542 001411 BEQ 5$ ;YES!!
289 012544 013737 001346 001142 MOV RMDSI,$BDDAT ;SETUP BAD DATA FOR TYPEOUT
290 012552 042737 167777 001142 BIC #^CMOL,$BDDAT
    
```

291	012560	005037	001140		CLR	\$GDDAT	:SETUP GOOD DATA FOR TYPEOUT
292	012564	104170			EMT	170	
293	012566	032737	020000	001346	5\$:	BIT	#PIP,RMDSI :IS PIP SET??
294	012574	001012			BNE	6\$:YES!!
295	012576	013737	001346	001142	MOV	RMDSI,\$BDDAT	:SETUP BAD DATA FOR TYPEOUT
296	012604	042737	157777	001142	BIC	#^CPIP,\$BDDAT	
297	012612	012737	020000	001140	MOV	#PIP,\$GDDAT	:EXPECTED PIP SET
298	012620	104200			EMT	200	
299	012622	032737	004000	001346	6\$:	BIT	#WRL,RMDSI :IS WRITE LOCK OFF??
300	012630	001411			BEQ	7\$:YES!!
301	012632	013737	001346	001142	MOV	RMDSI,\$BDDAT	:SETUP BAD DATA FOR TYPEOUT
302	012640	042737	173777	001142	BIC	#^CWRL,\$BDDAT	
303	012646	005037	001140		CLR	\$GDDAT	:EXPECTED WRL = 0
304	012652	104201			EMT	201	
305	012654	032737	040000	001376	7\$:	BIT	#SKI,RMER2I :IS SKI = 0
306	012662	001411			BEQ	8\$:YES!!
307	012664	013737	001376	001142	MOV	RMER2I,\$BDDAT	:SETUP BAD DATA FOR TYPEOUT
308	012672	042737	137777	001142	BIC	#^CSKI,\$BDDAT	
309	012700	005037	001140		CLR	\$GDDAT	:SKI SHOULD BE 0
310	012704	104202			EMT	202	
311	012706	032737	000200	001376	8\$:	BIT	#DVC,RMER2I :IS DEVICE CHECK = 0??
312	012714	001411			BEQ	9\$:YES!!
313	012716	013737	001376	001142	MOV	RMER2I,\$BDDAT	:SETUP BAD DATA FOR TYPEOUT
314	012724	042737	177577	001142	BIC	#^CDVC,\$BDDAT	
315	012732	005037	001140		CLR	\$GDDAT	:DVC SHOULD BE 0
316	012736	104203			EMT	203	
317	012740				9\$:		
318	012740	112737	000024	001543	MOVB	#RMMR1,PUTINX	:SETUP PUT INDEX TABLE
	012746	112737	000200	001544	MOVB	#200,PUTINX+1	:SET TERMINATOR BYTE
	012754	012737	001711	001434	MOV	#DMD!MUR!MOC!MSER!MDF!MWP,RMMR10	:SET RMMR1 OUTPUT BUFFER = DMD!MUR!M
	012762	004737	043160		JSR	PC,PUT	:GO WRITE RMMR1 VIA PUT SUBROUTINE
	012766	000404			BR	10\$:GO TO 10\$ IF NO ERROR
	012770	000240			NOP		:RETURN HERE IF ERROR
	012772	104000			EMT		:ERROR DEFINED BY PUT SUBROUTINE
	012774	000137	013426		JMP	21\$:GO TO 21\$ IF ERROR
319	013000				10\$:		
320	013000	004737	042710		JSR	PC,GET	:GO READ REGISTER(S) WITH GET SUBROUTINE
	013004	000404			BR	11\$:GO TO 11\$ IF NO ERROR
	013006	000240			NOP		:RETURN HERE IF ERROR
	013010	104000			EMT		:ERROR # DEFINED BY GET SUBROUTINE
	013012	000137	013426		JMP	21\$:GO TO 21\$ IF ERROR
321	013016	032737	000001	001360	11\$:	BIT	#DMD,RMMR1I :IS DIAGNOSTIC MODE SET??
322	013024	001011			BNE	12\$:YES!!
323	013026	012737	000001	001140	MOV	#DMD,\$GDDAT	:EXPECTED STATUS
324	013034	013737	001360	001142	MOV	RMMR1I,\$BDDAT	:RECEIVED STATUS
325	013042	104176			EMT	176	
326	013044	000137	013426		JMP	21\$:SKIP REST OF TEST IF DMD = 0
327	013050	032737	010000	001346	12\$:	BIT	#MOL,RMDSI :IS MOL = 1??
328	013056	001012			BNE	13\$:YES!!
329	013060	013737	001346	001142	MOV	RMDSI,\$BDDAT	:SETUP BAD DATA FOR TYPEOUT
330	013066	042737	167777	001142	BIC	#^CMOL,\$BDDAT	
331	013074	012737	010000	001140	MOV	#MOL,\$GDDAT	:EXPECTED MOL = 1
332	013102	104170			EMT	170	
333	013104	032737	020000	001346	13\$:	BIT	#PIP,RMDSI :IS PIP = 0 ?
334	013112	001411			BEQ	14\$:YES!!
335	013114	013737	001346	001142	MOV	RMDSI,\$BDDAT	:SETUP BAD DATA FOR TYPEOUT
336	013122	042737	157777	001142	BIC	#^CPIP,\$BDDAT	

```

337 013130 005037 001140 CLR $GDDAT ;EXPECTED PIP STATUS
338 013134 104200 EMT 200
339 013136 032737 004000 001346 14$: BIT #WRL,RMDSI ;IS WRL SET??
340 013144 001012 BNE 15$ ;YES!!
341 013146 013737 001346 001142 MOV RMDSI,$BDDAT ;SETUP BAD DATA FOR TYPEOUT
342 013154 042737 173777 001142 BIC #^CWRL,$BDDAT
343 013162 012737 004000 001140 MOV #WRL,$GDDAT ;EXPECTED GOOD STATUS
344 013170 104201 EMT 201
345 013172 032737 040000 001376 15$: BIT #SKI,RMER2I ;IS SKI SET??
346 013200 001012 BNE 16$ ;YES!!
347 013202 013737 001376 001142 MOV RMER2I,$BDDAT ;BAD DATA FOR TYPEOUT
348 013210 042737 137777 001142 BIC #^CSKI,$BDDAT
349 013216 012737 040000 001140 MOV #SKI,$GDDAT ;EXPECTED SKI ON
350 013224 104202 EMT 202
351 013226 032737 000200 001376 16$: BIT #DVC,RMER2I ;IS DVC SET ??
352 013234 001012 BNE 17$ ;YES!!
353 013236 013737 001376 001142 MOV RMER2I,$BDDAT ;BAD DATA FOR TYPEOUT
354 013244 042737 177577 001142 BIC #^CDVC,$BDDAT
355 013252 012737 000200 001140 MOV #DVC,$GDDAT ;EXPECTED DVC ON
356 013260 104203 EMT 203
357 013262 032737 000100 001346 17$: BIT #VV,RMDSI ;MUR SHOULD HAVE RESET VOLUME VALID
358 013270 001411 BEQ 18$ ;BRANCH IF IT DID
359 013272 013737 001346 001142 MOV RMDSI,$BDDAT ;BAD DATA FOR TYPEOUT
360 013300 042737 177677 001142 BIC #^CVV,$BDDAT
361 013306 005037 001140 CLR $GDDAT ;GOOD DATA FOR TYPEOUT
362 013312 104204 EMT 204
363 013314 18$:
364 013314 112737 000000 001543 MOVB #RMCS1,PUTINX ;SETUP PUT INDEX TABLE
013322 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
013330 012737 000011 001410 MOV #DRVCLR!GO,RMCS10 ;SET RMCS1 OUTPUT BUFFER = DRVCLR!GO
013336 004737 043160 JSR PC,PUT ;GO WRITE RMCS1 VIA PUT SUBROUTINE
013342 000404 BR 19$ ;GO TO 19$ IF NO ERROR
013344 000240 NOP ;RETURN HERE IF ERROR
013346 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
013350 000137 013426 JMP 21$ ;GO TO 21$ IF ERROR
365 013354 19$:
366 013354 004737 042710 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
013360 000404 BR 20$ ;GO TO 20$ IF NO ERROR
013362 000240 NOP ;RETURN HERE IF ERROR
013364 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
013366 000137 013426 JMP 21$ ;GO TO 21$ IF ERROR
367 013372 032737 002000 001376 20$: BIT #LBC,RMER2I ;IS LBC SET ??
368 013400 001012 BNE 21$ ;YES!!
369 013402 013737 001376 001142 MOV RMER2I,$BDDAT ;BAD DATA FOR TYPEOUT
370 013410 012737 002000 001140 MOV #LBC,$GDDAT ;GOOD DATA FOR TYPEOUT
371 013416 042737 002000 001142 BIC #LBC,$BDDAT
372 013424 104262 EMT 262
373 013426 21$:
374
375

```

```

:*****
:*TEST 10 PACK ACKNOWLEDGE TEST
:*****
TST10:

```

```

013426 000004 SCOPE ;SCOPE CALL
013430 000240 NOP ;START OF TEST
013432 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
013436 013700 001276 MOV $BASE,RO ;RO = UNIBUS ADDRESS

```

	013442	013701	001464		MOV	TSTQUE,R1	;(R1) = DEVICE BEING TESTED
	013446	012737	000010	001226	MOV	#10,\$TESTN	;;SET TEST NUMBER IN APT MAIL BOX
376							
377	013454	004737	052264		JSR	PC,CNTCLR	;GO ISSUE CONTROLLER CLEAR
	013460	000404			BR	1\$;GO TO 1\$ IF NO ERROR
	013462	000240			NOP		;RETURN HERE IF ERROR
	013464	104000			EMT		;ERROR NUMBER DEFINED BY SUBROUTINE
	013466	000137	013726		JMP	8\$;GO TO 8\$ IF ERROR
378	013472						
379	013472	112737	000024	001543	MOV	#RMMR1,PUTINX	;SETUP PUT INDEX TABLE
	013500	112737	000200	001544	MOV	#200,PUTINX+1	;SET TERMINATOR BYTE
	013506	012737	000001	001434	MOV	#DMD,RMMR10	;SET RMMR1 OUTPUT BUFFER = DMD
	013514	004737	043160		JSR	PC,PUT	;GO WRITE RMMR1 VIA PUT SUBROUTINE
	013520	000404			BR	2\$;GO TO 2\$ IF NO ERROR
	013522	000240			NOP		;RETURN HERE IF ERROR
	013524	104000			EMT		;ERROR DEFINED BY PUT SUBROUTINE
	013526	000137	013726		JMP	8\$;GO TO 8\$ IF ERROR
380	013532						
381	013532	112737	000024	001543	MOV	#RMMR1,PUTINX	;SETUP PUT INDEX TABLE
	013540	112737	000200	001544	MOV	#200,PUTINX+1	;SET TERMINATOR BYTE
	013546	012737	000000	001434	MOV	#0,RMMR10	;SET RMMR1 OUTPUT BUFFER = 0
	013554	004737	043160		JSR	PC,PUT	;GO WRITE RMMR1 VIA PUT SUBROUTINE
	013560	000404			BR	3\$;GO TO 3\$ IF NO ERROR
	013562	000240			NOP		;RETURN HERE IF ERROR
	013564	104000			EMT		;ERROR DEFINED BY PUT SUBROUTINE
	013566	000137	013726		JMP	8\$;GO TO 8\$ IF ERROR
382	013572						
383	013572	112737	000000	001543	MOV	#RMCS1,PUTINX	;SETUP PUT INDEX TABLE
	013600	112737	000200	001544	MOV	#200,PUTINX+1	;SET TERMINATOR BYTE
	013606	012737	000023	001410	MOV	#PAKACK!GO,RMCS10	;SET RMCS1 OUTPUT BUFFER = PAKACK!GO
	013614	004737	043160		JSR	PC,PUT	;GO WRITE RMCS1 VIA PUT SUBROUTINE
	013620	000404			BR	4\$;GO TO 4\$ IF NO ERROR
	013622	000240			NOP		;RETURN HERE IF ERROR
	013624	104000			EMT		;ERROR DEFINED BY PUT SUBROUTINE
	013626	000137	013726		JMP	8\$;GO TO 8\$ IF ERROR
384	013632	004737	042624		JSR	PC,GETSTS	
385	013636	004737	043522		JSR	PC,TIMOUT	;WAIT FOR COMPLETION
386							
387	013642	004737	042710		JSR	PC,GET	;GO READ REGISTER(S) WITH GET SUBROUTINE
	013646	000403			BR	5\$;GO TO 5\$ IF NO ERROR
	013650	000240			NOP		;RETURN HERE IF ERROR
	013652	104000			EMT		;ERROR # DEFINED BY GET SUBROUTINE
388	013654	000424			BR	8\$;SKIP REMAINDER OF TEST
389	013656						
390	013656	004737	043706		JSR	PC,PRIERR	;GO CHECK FOR PRIMARY ERRORS
	013662	000404			BR	6\$;GO TO 6\$ IF NO ERROR
	013664	000240			NOP		;RETURN HERE IF ERROR
	013666	104000			EMT		;ERROR # DEFINED BY PRIERR SUBROUTINE
	013670	004736			JSR	PC,@(SP)+	;GO BACK FOR MORE ERROR CHECKS
391	013672	000415			BR	8\$;SKIP TO NEXT TEST
392	013674						
393	013674	004737	053262		JSR	PC,ACKSTS	;GO VERIFY PACK ACKNOWLEDGE
	013700	000404			BR	7\$;GO TO 7\$ IF NO ERROR
	013702	000240			NOP		;RETURN HERE IF ERROR
	013704	104000			EMT		;ERROR # DEFINED BY ACKSTS SUBROUTINE
	013706	004736			JSR	PC,@(SP)+	;GO BACK FOR MORE ERROR CHECKS
394	013710	000406			BR	8\$;SKIP TO NEXT TEST

395
 396 013712
 397 013712 004737 044540
 013716 000403
 013720 000240
 013722 104000
 013724 004736
 398 013726
 399
 400

```

7$:      JSR      PC,SECERR      ;GO CHECK FOR SECONDARY ERRORS
        BR       8$             ;GO TO 8$ IF NO ERROR
        NOP      ;RETURN HERE IF ERROR
        EMT     ;ERROR # DEFINED BY SECERR SUBROUTINE
        JSR     PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
    
```

```

*****
:*TEST 11      RECALIBRATE TEST
*****
TST11:
    
```

013726
 013726 000004
 013730 000240
 013732 012706 001100
 013736 013700 001276
 013742 013701 001464
 013746 012737 000001 001206
 013754 012737 000011 001226

```

SCOPE      ;SCOPE CALL
NOP        ;START OF TEST
MOV        #STACK,SP ;INITIALIZE STACK POINTER
MOV        $BASE,R0  ;R0 = UNIBUS ADDRESS
MOV        TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
MOV        #1,$TIMES ;DO 1 ITERATION
MOV        #11,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
    
```

401
 402 013762 004737 041650
 013766 050020

```

JSR        PC,TSTPRP      ;PREPARE DEVICE FOR TEST
WORD      050020          ;TASK DESCRIPTOR AS FOLLOWS:
                                ;CLEAR CONTROLLER & SELECT DEVICE
                                ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                ;VERIFY PACK ACKNOWLEDGE
    
```

013770 000404
 013772 000240
 013774 104000
 013776 000137 014144

```

BR         1$             ;GO TO 1$ IF NO ERROR
NOP        ;RETURN HERE IF ERROR
EMT       ;ERROR # DEFINED BY TSTPRP SUBROUTINE
JMP       6$             ;GO TO 6$ IF ERROR
    
```

403 014002
 404 014002 112737 000000 001543
 014010 112737 000200 001544
 014016 012737 000007 001410
 014024 004737 043160
 014030 000404
 014032 000240
 014034 104000

```

1$:      MOV      #RMCS1,PUTINX ;SETUP PUT INDEX TABLE
        MOV      #200,PUTINX+1 ;SET TERMINATOR BYTE
        MOV      #RECAL!GO,RMCS10 ;SET RMCS1 OUTPUT BUFFER = RECAL!GO
        JSR     PC,PUT          ;GO WRITE RMCS1 VIA PUT SUBROUTINE
        BR      2$             ;GO TO 2$ IF NO ERROR
        NOP      ;RETURN HERE IF ERROR
        EMT     ;ERROR DEFINED BY PUT SUBROUTINE
        JMP     6$             ;GO TO 6$ IF ERROR
    
```

014036 000137 014144
 405 014042 004737 042624
 406 014046 004737 043522
 407

```

2$:      JSR     PC,GETSTS      ;GO SETUP FOR STATUS FETCH
        JSR     PC,TIMOUT      ;WAIT FOR COMPLETION
    
```

408 014052 004737 042710
 014056 000404
 014060 000240
 014062 104000
 014064 000137 014144

```

JSR        PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
BR         3$             ;GO TO 3$ IF NO ERROR
NOP        ;RETURN HERE IF ERROR
EMT       ;ERROR # DEFINED BY GET SUBROUTINE
JMP       6$             ;GO TO 6$ IF ERROR
    
```

409 014070
 410 014070 004737 043706
 014074 000405
 014076 000240
 014100 104000
 014102 004736
 014104 000137 014144

```

3$:      JSR     PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
        BR      4$             ;GO TO 4$ IF NO ERROR
        NOP      ;RETURN HERE IF ERROR
        EMT     ;ERROR # DEFINED BY PRIERR SUBROUTINE
        JSR     PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
        JMP     6$             ;GO TO 6$ IF ERROR
    
```

411 014110
 412 014110 004737 054056
 014114 000405

```

4$:      JSR     PC,RCLSTS      ;GO VERIFY RECALIBRATE OPERATION
        BR      5$             ;GO TO 5$ IF NO ERROR
    
```



```

T11 RECALIBRATE TEST
014116 000240 NOP ;RETURN HERE IF ERROR
014120 104000 EMT ;ERROR # DEFINED BY RCLSTS SUBROUTINE
014122 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
014124 000137 014144 JMP 6$ ;GO TO 6$ IF ERROR
413 014130 5$:
414 014130 004737 044540 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
014134 000403 BR 6$ ;GO TO 6$ IF NO ERROR
014136 000240 NOP ;RETURN HERE IF ERROR
014140 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
014142 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
415 014144 6$:
416
417
*****
:TEST 12 ABORT RECALIBRATE TEST
*****
TST12:
014144 000004 SCOPE ;SCOPE CALL
014146 000240 NOP ;START OF TEST
014150 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
014154 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
014160 013701 001464 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
014164 012737 000001 001206 MOV #1,$TIMES ;DO 1 ITERATION
014172 012737 000012 001226 MOV #12,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
418
419 014200 004737 041650 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
014204 054130 .WORD 054130 ;TASK DESCRIPTOR AS FOLLOWS:
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF "SKI" OR "PIP" IS SET
;VERIFY RECALIBRATION
;GO TO 1$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 9$ IF ERROR
014206 000404 BR 1$
014210 000240 NOP
014212 104000 EMT
014214 000137 014452 JMP 9$
420 014220 1$:
421 014220 112737 000014 001543 MOV #RMER1,PUTINX ;SETUP PUT INDEX TABLE
014226 112737 000200 001544 MOV #200,PUTINX+1 ;SET TERMINATOR BYTE
014234 012737 040000 001424 MOV #UNS,RMER10 ;SET RMER1 OUTPUT BUFFER = UNS
014242 004737 043160 JSR PC,PUT ;GO WRITE RMER1 VIA PUT SUBROUTINE
014246 000404 BR 2$ ;GO TO 2$ IF NO ERROR
014250 000240 NOP ;RETURN HERE IF ERROR
014252 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
014254 000137 014452 JMP 9$ ;GO TO 9$ IF ERROR
422 014260 2$:
423 014260 112737 000000 001543 MOV #RMCS1,PUTINX ;SETUP PUT INDEX TABLE
014266 112737 000200 001544 MOV #200,PUTINX+1 ;SET TERMINATOR BYTE
014274 012737 000007 001410 MOV #RECAL!GO,RMCS10 ;SET RMCS1 OUTPUT BUFFER = RECAL!GO
014302 004737 043160 JSR PC,PUT ;GO WRITE RMCS1 VIA PUT SUBROUTINE
014306 000404 BR 3$ ;GO TO 3$ IF NO ERROR
014310 000240 NOP ;RETURN HERE IF ERROR
014312 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
014314 000137 014452 JMP 9$ ;GO TO 9$ IF ERROR
424 014320 004737 042624 3$: JSR PC,GETSTS ;SETUP FOR STATUS FETCH
425
426 014324 004737 042710 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE

```

```

014330 000404 BR 4$ :GO TO 4$ IF NO ERROR
014332 000240 NOP :RETURN HERE IF ERROR
014334 104000 EMT :ERROR # DEFINED BY GET SUBROUTINE
427 014336 000137 014452 JMP 9$ :GO TO 9$ IF ERROR
428 014342 032737 020000 001346 4$: BIT #PIP,RMDSI :DID THE DRIVE RECALIBRATE??
429 014350 001401 BEQ 5$ :NO!!
430 014352 104214 EMT 214
431 014354 004737 043522 5$: JSR PC,TIMOUT :WAIT FOR COMPLETION
432 014360 004737 042710 JSR PC,GET :GO READ REGISTER(S) WITH GET SUBROUTINE
014364 000404 BR 6$ :GO TO 6$ IF NO ERROR
014366 000240 NOP :RETURN HERE IF ERROR
014370 104000 EMT :ERROR # DEFINED BY GET SUBROUTINE
433 014372 000137 014452 JMP 9$ :GO TO 9$ IF ERROR
434 014376 004737 043706 6$: JSR PC,PRIERR :GO CHECK FOR PRIMARY ERRORS
014402 000405 BR 7$ :GO TO 7$ IF NO ERROR
014404 000240 NOP :RETURN HERE IF ERROR
014406 104000 EMT :ERROR # DEFINED BY PRIERR SUBROUTINE
435 014410 004736 JSR PC,@(SP)+ :GO BACK FOR MORE ERROR CHECKS
014412 000137 014452 JMP 9$ :GO TO 9$ IF ERROR
436 014416 004737 057766 7$: JSR PC,STCDRVSTS :GO CHECK FOR CHANGES IN DRIVE STATUS
014422 000405 BR 8$ :GO TO 8$ IF NO ERROR
014424 000240 NOP :RETURN HERE IF ERROR
014426 104000 EMT :ERROR # DEFINED BY STCDRVSTS SUBROUTINE
437 014430 004736 JSR PC,@(SP)+ :GO BACK FOR MORE ERROR CHECKS
014432 000137 014452 JMP 9$ :GO TO 9$ IF ERROR
438 014436 004737 044540 8$: JSR PC,SECERR :GO CHECK FOR SECONDARY ERRORS
014442 000403 BR 9$ :GO TO 9$ IF NO ERROR
014444 000240 NOP :RETURN HERE IF ERROR
014446 104000 EMT :ERROR # DEFINED BY SECERR SUBROUTINE
439 014450 004736 JSR PC,@(SP)+ :GO BACK FOR MORE ERROR CHECKS
440 014452
441

```

```

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

```

```

014452 000004 SCOPE :SCOPE CALL
014454 000240 NOP :START OF TEST
014456 012706 001100 MOV #STACK,SP :INITIALIZE STACK POINTER
014462 013700 001276 MOV $BASE,R0 :R0 = UNIBUS ADDRESS
014466 013701 001464 MOV TSTQUE,R1 : (R1) = DEVICE BEING TESTED
014472 012737 000001 001206 MOV #1,$TIMES :DO 1 ITERATION
442 014500 012737 000013 001226 MOV #13,$TESTN :SET TEST NUMBER IN APT MAIL BOX
443 014506 004737 041650 JSR PC,TSTPRP :PREPARE DEVICE FOR TEST
014512 054130 .WORD 054130 :TASK DESCRIPTOR AS FOLLOWS:
: CLEAR CONTROLLER & SELECT DEVICE
: VERIFY CONTROLLER CLEAR OPERATION
: PACK ACKNOWLEDGE IF VOLUME NOT VALID
: VERIFY PACK ACKNOWLEDGE
: RECALIBRATE IF "SKI" OR "PIP" IS SET
: VERIFY RECALIBRATION
014514 000404 BR 1$ :GO TO 1$ IF NO ERROR

```

```

014516 000240      NOP      ;RETURN HERE IF ERROR
014520 104000      EMT      ;ERROR # DEFINED BY TSTPRP SUBROUTINE
014522 000137 014762 JMP      8$      ;GO TO 8$ IF ERROR
444 014526      1$:
445 014526 112737 000024 001543 MOVB    #RMMR1,PUTINX ;SETUP PUT INDEX TABLE
014534 112737 000200 001544 MOVB    #200,PUTINX+1 ;SET TERMINATOR BYTE
014542 012737 000001 001434 MOV     #DMD,RMMR1O   ;SET RMMR1 OUTPUT BUFFER = DMD
014550 004737 043160 JSR     PC,PUT        ;GO WRITE RMMR1 VIA PUT SUBROUTINE
014554 000404      BR      2$      ;GO TO 2$ IF NO ERROR
014556 000240      NOP      ;RETURN HERE IF ERROR
014560 104000      EMT      ;ERROR DEFINED BY PUT SUBROUTINE
014562 000137 014762 JMP      8$      ;GO TO 8$ IF ERROR
446 014566      2$:
447 014566 112737 000024 001543 MOVB    #RMMR1,PUTINX ;SETUP PUT INDEX TABLE
014574 112737 000200 001544 MOVB    #200,PUTINX+1 ;SET TERMINATOR BYTE
014602 012737 000000 001434 MOV     #0,RMMR1O     ;SET RMMR1 OUTPUT BUFFER = 0
014610 004737 043160 JSR     PC,PUT        ;GO WRITE RMMR1 VIA PUT SUBROUTINE
014614 000404      BR      3$      ;GO TO 3$ IF NO ERROR
014616 000240      NOP      ;RETURN HERE IF ERROR
014620 104000      EMT      ;ERROR DEFINED BY PUT SUBROUTINE
014622 000137 014762 JMP      8$      ;GO TO 8$ IF ERROR
448 014626      3$:
449 014626 112737 000000 001543 MOVB    #RMCS1,PUTINX ;SETUP PUT INDEX TABLE
014634 112737 000200 001544 MOVB    #200,PUTINX+1 ;SET TERMINATOR BYTE
014642 012737 000007 001410 MOV     #RECAL!GO,RMCS1O ;SET RMCS1 OUTPUT BUFFER = RECAL!GO
014650 004737 043160 JSR     PC,PUT        ;GO WRITE RMCS1 VIA PUT SUBROUTINE
014654 000404      BR      4$      ;GO TO 4$ IF NO ERROR
014656 000240      NOP      ;RETURN HERE IF ERROR
014660 104000      EMT      ;ERROR DEFINED BY PUT SUBROUTINE
014662 000137 014762 JMP      8$      ;GO TO 8$ IF ERROR
450 014666 004737 042624 4$: JSR     PC,GETSTS    ;SETUP FOR STATUS FETCH
451 014672 004737 043522 JSR     PC,TIMOUT    ;WAIT FOR RECAL TO COMPLETE
452
453 014676 004737 042710 JSR     PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
014702 000404      BR      5$      ;GO TO 5$ IF NO ERROR
014704 000240      NOP      ;RETURN HERE IF ERROR
014706 104000      EMT      ;ERROR # DEFINED BY GET SUBROUTINE
014710 000137 014762 JMP      8$      ;GO TO 8$ IF ERROR
454 014714      5$:
455 014714 004737 043706 JSR     PC,PRIERR    ;GO CHECK FOR PRIMARY ERRORS
014720 000405      BR      6$      ;GO TO 6$ IF NO ERROR
014722 000240      NOP      ;RETURN HERE IF ERROR
014724 104000      EMT      ;ERROR # DEFINED BY PRIERR SUBROUTINE
014726 004736      JSR     PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
014730 000137 014762 JMP      8$      ;GO TO 8$ IF ERROR
456 014734 032737 010000 001376 6$: BIT     #IVC,RMER2I  ;IVC SHOULD BE SET
457 014742 001001      BNE     7$      ;IT IS - GO CHECK SECONDARY ERRORS
458 014744 104215      EMT      215
459 014746      7$:
460 014746 004737 044540 JSR     PC,SECERR    ;GO CHECK FOR SECONDARY ERRORS
014752 000403      BR      8$      ;GO TO 8$ IF NO ERROR
014754 000240      NOP      ;RETURN HERE IF ERROR
014756 104000      EMT      ;ERROR # DEFINED BY SECERR SUBROUTINE
014760 004736      JSR     PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
461 014762      8$:
462
463

```

TEST 14 IAE RECALIBRATE TEST

TST14:

014762					SCOPE		:SCOPE CALL
014762	000004				NOP		:START OF TEST
014764	000240				MOV	#STACK,SP	:INITIALIZE STACK POINTER
014766	012706	001100			MOV	\$BASE,R0	:R0 = UNIBUS ADDRESS
014772	013700	001276			MOV	TSTQUE,R1	:(R1) = DEVICE BEING TESTED
014776	013701	001464			MOV	#1,\$TIMES	;;DO 1 ITERATION
015002	012737	000001	001206		MOV	#14,\$TESTN	;;SET TEST NUMBER IN APT MAIL BOX
015010	012737	000014	001226				
464							
465	015016	004737	041650		JSR	PC,TSTPRP	:PREPARE DEVICE FOR TEST
	015022	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 1\$ IF NO ERROR
							:RETURN HERE IF ERROR
							:ERROR # DEFINED BY TSTPRP SUBROUTINE
							:GO TO 6\$ IF ERROR
	015024	000404			BR	1\$	
	015026	000240			NOP		
	015030	104000			EMT		
	015032	000137	015224		JMP	6\$	
466	015036			1\$:			
467	015036	012737	177777	001444	MOV	#-1,RMDCO	:RMDC WILL BE ALL ONES
468	015044	012737	177777	001416	MOV	#-1,RMDAO	:RMDA WILL BE ALL ONES
469	015052	012737	000007	001410	MOV	#RECAL!GO,RMCS10	
470	015060	012702	001543		MOV	#PUTINX,R2	:R2 POINTS TO PUT INDEX TABLE
471	015064	112722	000034		MOVB	#RMDC,(R2)+	:SETUP PUT INDEX TABLE
472	015070	112722	000006		MOVB	#RMDA,(R2)+	
473	015074	112722	000000		MOVB	#RMCS1,(R2)+	
474	015100	112722	000200		MOVB	#200,(R2)+	
475	015104	004737	043160		JSR	PC,PUT	:GO WRITE REGISTER(S) WITH PUT SUBROUTINE
	015110	000404			BR	2\$:GO TO 2\$ IF NO ERROR
	015112	000240			NOP		:RETURN HERE IF ERROR
	015114	104000			EMT		:ERROR # DEFINED BY PUT SUBROUTINE
	015116	000137	015224		JMP	6\$:GO TO 6\$ IF ERROR
476	015122	004737	042624	2\$:	JSR	PC,GETSTS	:SETUP FOR STATUS FETCH
477	015126	004737	043522		JSR	PC,TIMOUT	:WAIT FOR GO TO RESET
478							
479	015132	004737	042710		JSR	PC,GET	:GO READ REGISTER(S) WITH GET SUBROUTINE
	015136	000404			BR	3\$:GO TO 3\$ IF NO ERROR
	015140	000240			NOP		:RETURN HERE IF ERROR
	015142	104000			EMT		:ERROR # DEFINED BY GET SUBROUTINE
	015144	000137	015224		JMP	6\$:GO TO 6\$ IF ERROR
480	015150			3\$:			
481	015150	004737	043706		JSR	PC,PRIERR	:GO CHECK FOR PRIMARY ERRORS
	015154	000405			BR	4\$:GO TO 4\$ IF NO ERROR
	015156	000240			NOP		:RETURN HERE IF ERROR
	015160	104000			EMT		:ERROR # DEFINED BY PRIERR SUBROUTINE
	015162	004736			JSR	PC,@(SP)+	:GO BACK FOR MORE ERROR CHECKS
	015164	000137	015224		JMP	6\$:GO TO 6\$ IF ERROR
482	015170			4\$:			
483	015170	004737	054056		JSR	PC,RCLSTS	:GO VERIFY RECALIBRATE OPERATION
	015174	000405			BR	5\$:GO TO 5\$ IF NO ERROR
	015176	000240			NOP		:RETURN HERE IF ERROR
	015200	104000			EMT		:ERROR # DEFINED BY RCLSTS SUBROUTINE

```

015202 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
015204 000137 015224 JMP 6$ ;GO TO 6$ IF ERROR
484 015210 5$:
485 015210 004737 044540 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
015214 000403 BR 6$ ;GO TO 6$ IF NO ERROR
015216 000240 NOP ;RETURN HERE IF ERROR
015220 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
015222 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
486 015224 6$:
487
488
*****
;*TEST 15 RECALIBRATE AT OFFSET
*****
TST15:
015224 000004 SCOPE ;SCOPE CALL
015226 000240 NOP ;START OF TEST
015230 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
015234 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
015240 013701 001464 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
015244 012737 000001 001206 MOV #1,$TIMES ;DO 1 ITERATION
015252 012737 000015 001226 MOV #15,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
489
490 015260 004737 041650 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
015264 050020 .WORD 050020 ;TASK DESCRIPTOR AS FOLLOWS:
;CLEAR CONTROLLER & SELECT DEVICE
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
015266 000404 BR 1$ ;GO TO 1$ IF NO ERROR
015270 000240 NOP ;RETURN HERE IF ERROR
015272 104000 EMT ;ERROR # DEFINED BY TSTPRP SUBROUTINE
015274 000137 015502 JMP 7$ ;GO TO 7$ IF ERROR
491 015300 1$:
492 015300 112737 000000 001543 MOVB #RMCS1,PUTINX ;SETUP PUT INDEX TABLE
015306 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
015314 012737 000015 001410 MOV #OFFSET!GO,RMCS10 ;SET RMCS1 OUTPUT BUFFER = OFFSET!GO
015322 004737 043160 JSR PC,PUT ;GO WRITE RMCS1 VIA PUT SUBROUTINE
015326 000404 BR 2$ ;GO TO 2$ IF NO ERROR
015330 000240 NOP ;RETURN HERE IF ERROR
015332 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
015334 000137 015502 JMP 7$ ;GO TO 7$ IF ERROR
493 015340 2$:
494 015340 112737 000000 001543 MOVB #RMCS1,PUTINX ;SETUP PUT INDEX TABLE
015346 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
015354 012737 000007 001410 MOV #RECAL!GO,RMCS10 ;SET RMCS1 OUTPUT BUFFER = RECAL!GO
015362 004737 043160 JSR PC,PUT ;GO WRITE RMCS1 VIA PUT SUBROUTINE
015366 000404 BR 3$ ;GO TO 3$ IF NO ERROR
015370 000240 NOP ;RETURN HERE IF ERROR
015372 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
015374 000137 015502 JMP 7$ ;GO TO 7$ IF ERROR
495 015400 3$:
496
497 ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
015400 004737 042624 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
498
499 ;WAIT FOR COMMAND TO COMPLETE
015404 004737 043522 JSR PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
500

```

```

T15 RECALIBRATE AT OFFSET
501 015410 004737 042710 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
    015414 000404 BR 4$ ;GO TO 4$ IF NO ERROR
    015416 000240 NOP ;RETURN HERE IF ERROR
    015420 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
    015422 000137 015502 JMP 7$ ;GO TO 7$ IF ERROR
502 015426 4$:
503 015426 004737 043706 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
    015432 000405 BR 5$ ;GO TO 5$ IF NO ERROR
    015434 000240 NOP ;RETURN HERE IF ERROR
    015436 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
    015440 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
    015442 000137 015502 JMP 7$ ;GO TO 7$ IF ERROR
504 015446 5$:
505 015446 004737 054056 JSR PC,RCLSTS ;GO VERIFY RECALIBRATE OPERATION
    015452 000405 BR 6$ ;GO TO 6$ IF NO ERROR
    015454 000240 NOP ;RETURN HERE IF ERROR
    015456 104000 EMT ;ERROR # DEFINED BY RCLSTS SUBROUTINE
    015460 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
    015462 000137 015502 JMP 7$ ;GO TO 7$ IF ERROR
506 015466 6$:
507 015466 004737 044540 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
    015472 000403 BR 7$ ;GO TO 7$ IF NO ERROR
    015474 000240 NOP ;RETURN HERE IF ERROR
    015476 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
    015500 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
508 015502 7$:
509
510
*****
*TEST 16 DRIVE CLEAR TEST
*****
TST16:
015502
015502 000004 SCOPE ;SCOPE CALL
015504 000240 NOP ;START OF TEST
015506 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
015512 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
015516 013701 001464 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
015522 012737 000016 001226 MOV #16,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
511
512 015530 004737 041650 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
    015534 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
    015536 000404 BR 1$ ;GO TO 1$ IF NO ERROR
    015540 000240 NOP ;RETURN HERE IF ERROR
    015542 104000 EMT ;ERROR # DEFINED BY TSTPRP SUBROUTINE
    015544 000137 015752 JMP 6$ ;GO TO 6$ IF ERROR
513 015550 1$:
514 015550 012737 000000 001416 MOV #0,RMDAO
515 015556 012737 000011 001410 MOV #DRVCLR!GO,RMCS10
516 015564 012737 177777 001424 MOV #-1,RMER10
517 015572 012737 177777 001452 MOV #-1,RMER20
518 015600 042737 004000 001452 BIC #LSC,RMER20 ;DELETE FOR PASS 2 ETCH
519 015606 012702 001543 MOV #PUTINX,R2 ;R2 POINTS TO INDEX TABLE

```



```

016006 000404 BR 1$ ;RECALIBRATE IF "SKI" OR "PIP" IS SET
016010 000240 NOP ;VERIFY RECALIBRATION
016012 104000 EMT ;GO TO 1$ IF NO ERROR
016014 000137 016202 JMP 7$ ;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 7$ IF ERROR
543 016020 1$:
544 016020 112737 000000 001543 MOVB #RMCS1,PUTINX ;SETUP PUT INDEX TABLE
016026 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
016034 012737 000001 001410 MOV #NOP!GO,RMCS10 ;SET RMCS1 OUTPUT BUFFER = NOP!GO
016042 004737 043160 JSR PC,PUT ;GO WRITE RMCS1 VIA PUT SUBROUTINE
016046 000404 BR 2$ ;GO TO 2$ IF NO ERROR
016050 000240 NOP ;RETURN HERE IF ERROR
016052 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
016054 000137 016202 JMP 7$ ;GO TO 7$ IF ERROR
545 016060 004737 042624 2$: JSR PC,GETSTS ;SETUP FOR STATUS FETCH
546
547 016064 004737 042710 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
016070 000404 BR 3$ ;GO TO 3$ IF NO ERROR
016072 000240 NOP ;RETURN HERE IF ERROR
016074 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
016076 000137 016202 JMP 7$ ;GO TO 7$ IF ERROR
548 016102 3$:
549 016102 004737 043706 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
016106 000405 BR 4$ ;GO TO 4$ IF NO ERROR
016110 000240 NOP ;RETURN HERE IF ERROR
016112 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
016114 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
016116 000137 016202 JMP 7$ ;GO TO 7$ IF ERROR
550 016122 4$:
551 016122 004737 050400 JSR PC,CMPERRSTS ;CHECK ANY ERRORS NOT MASKED
016126 115760 .WORD NDTMSK ;MASK FOR RMER1
016130 000010 .WORD DPE ;MASK FOR RMER2
016132 000405 BR 5$ ;GO TO 5$ IF NO ERROR
016134 000240 NOP ;RETURN HERE IF ERROR
016136 104000 EMT ;ERROR # DEFINED BY CMPERRSTS SUBROUTINE
016140 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
016142 000137 016202 JMP 7$ ;GO TO 7$ IF ERROR
552 016146 5$:
553 016146 004737 057766 JSR PC,STCDRVSTS ;GO CHECK FOR CHANGES IN DRIVE STATUS
016152 000405 BR 6$ ;GO TO 6$ IF NO ERROR
016154 000240 NOP ;RETURN HERE IF ERROR
016156 104000 EMT ;ERROR # DEFINED BY STCDRVSTS SUBROUTINE
016160 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
016162 000137 016202 JMP 7$ ;GO TO 7$ IF ERROR
554 016166 6$:
555 016166 004737 044540 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
016172 000403 BR 7$ ;GO TO 7$ IF NO ERROR
016174 000240 NOP ;RETURN HERE IF ERROR
016176 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
016200 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
556 016202 7$:
557
558

```

016202

```

*****
;*TEST 20 OFFSET TEST
*****
TST20:

```


T20	016202	000004				SCOPE		:SCOPE CALL
	016204	000240				NOP		:START OF TEST
	016206	012706	001100			MOV	#STACK,SP	:INITIALIZE STACK POINTER
	016212	013700	001276			MOV	\$BASE,R0	:R0 = UNIBUS ADDRESS
	016216	013701	001464			MOV	TSTQUE,R1	:(R1) = DEVICE BEING TESTED
	016222	012737	000020	001226		MOV	#20,\$TESTN	::SET TEST NUMBER IN APT MAIL BOX
559								
560	016230	004737	041650			JSR	PC,TSTPRP	:PREPARE DEVICE FOR TEST
	016234	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 1\$ IF NO ERROR
								:RETURN HERE IF ERROR
								:ERROR # DEFINED BY TSTPRP SUBROUTINE
								:GO TO 9\$ IF ERROR
	016236	000404				BR	1\$	
	016240	000240				NOP		
	016242	104000				EMT		
	016244	000137	016526			JMP	9\$	
561	016250				1\$:			
562	016250	112737	000000	001543		MOVB	#RMCS1,PUTINX	:SETUP PUT INDEX TABLE
	016256	112737	000200	001544		MOVB	#200,PUTINX+1	:SET TERMINATOR BYTE
	016264	012737	000015	001410		MOV	#OFFSET!GO,RMCS10	:SET RMCS1 OUTPUT BUFFER = OFFSET!GO
	016272	004737	043160			JSR	PC,PUT	:GO WRITE RMCS1 VIA PUT SUBROUTINE
	016276	000404				BR	2\$:GO TO 2\$ IF NO ERROR
	016300	000240				NOP		:RETURN HERE IF ERROR
	016302	104000				EMT		:ERROR DEFINED BY PUT SUBROUTINE
	016304	000137	016526			JMP	9\$:GO TO 9\$ IF ERROR
563	016310	004737	042624		2\$:	JSR	PC,GETSTS	:SETUP FOR STATUS FETCH
564	016314	004737	043522			JSR	PC,TIMOUT	:WAIT FOR GO TO RESET
565								
566	016320	004737	042710			JSR	PC,GET	:GO READ REGISTER(S) WITH GET SUBROUTINE
	016324	000404				BR	3\$:GO TO 3\$ IF NO ERROR
	016326	000240				NOP		:RETURN HERE IF ERROR
	016330	104000				EMT		:ERROR # DEFINED BY GET SUBROUTINE
	016332	000137	016526			JMP	9\$:GO TO 9\$ IF ERROR
567	016336				3\$:			
568	016336	004737	043706			JSR	PC,PRIERR	:GO CHECK FOR PRIMARY ERRORS
	016342	000405				BR	4\$:GO TO 4\$ IF NO ERROR
	016344	000240				NOP		:RETURN HERE IF ERROR
	016346	104000				EMT		:ERROR # DEFINED BY PRIERR SUBROUTINE
	016350	004736				JSR	PC,@(SP)+	:GO BACK FOR MORE ERROR CHECKS
	016352	000137	016526			JMP	9\$:GO TO 9\$ IF ERROR
569	016356	032737	000001	001346	4\$:	BIT	#OM,RMDSI	:OFFSET MODE ON??
570	016364	001012				BNE	5\$:YES!!
571	016366	013737	001346	001142		MOV	RMDSI,\$BDDAT	:BAD DATA FOR TYPEOUT
572	016374	042737	177776	001142		BIC	#^COM,\$BDDAT	
573	016402	012737	000001	001140		MOV	#OM,\$GDDAT	:GOOD DATA FOR TYPEOUT
574	016410	104156				EMT	156	
575	016412	032737	100000	001346	5\$:	BIT	#ATA,RMDSI	:WAS ATTENTION SET ??
576	016420	001012				BNE	6\$:YES!!
577	016422	013737	001346	001142		MOV	RMDSI,\$BDDAT	:BAD DATA FOR TYPEOUT
578	016430	042737	077777	001142		BIC	#^CATA,\$BDDAT	
579	016436	012737	100000	001140		MOV	#ATA,\$GDDAT	:GOOD DATA FOR TYPEOUT
580	016444	104172				EMT	172	
581	016446				6\$:			
582	016446	004737	050400			JSR	PC,MPERRSTS	:CHECK ANY ERRORS NOT MASKED

```

016452 115760 .WORD NDTMSK :MASK FOR RMER1
016454 000010 .WORD DPE :MASK FOR RMER2
016456 000405 BR 7$ :GO TO 7$ IF NO ERROR
016460 000240 NCP :RETURN HERE IF ERROR
016462 104000 EMT :ERROR # DEFINED BY CMPERRSTS SUBROUTINE
016464 004736 JSR PC,@(SP)+ :GO BACK FOR MORE ERROR CHECKS
016466 000137 016526 JMP 9$ :GO TO 9$ IF ERROR
583 016472 7$:
584 016472 004737 057766 JSR PC,STCDRVSTS :GO CHECK FOR CHANGES IN DRIVE STATUS
016476 000405 BR 8$ :GO TO 8$ IF NO ERROR
016500 000240 NOP :RETURN HERE IF ERROR
016502 104000 EMT :ERROR # DEFINED BY STCDRVSTS SUBROUTINE
016504 004736 JSR PC,@(SP)+ :GO BACK FOR MORE ERROR CHECKS
016506 000137 016526 JMP 9$ :GO TO 9$ IF ERROR
585 016512 8$:
586 016512 004737 044540 JSR PC,SECERR :GO CHECK FOR SECONDARY ERRORS
016516 000403 BR 9$ :GO TO 9$ IF NO ERROR
016520 000240 NOP :RETURN HERE IF ERROR
016522 104000 EMT :ERROR # DEFINED BY SECERR SUBROUTINE
016524 004736 JSR PC,@(SP)+ :GO BACK FOR MORE ERROR CHECKS
587 016526 9$:
588
589

```

```

:*****
:*TEST 21 GO/ATA TEST
:*****
TST21:

```

```

016526 000004 SCOPE :SCOPE CALL
016530 000240 NOP :START OF TEST
016532 012706 001100 MOV #STACK,SP :INITIALIZE STACK POINTER
016536 013700 001276 MOV $BASE,R0 :R0 = UNIBUS ADDRESS
016542 013701 001464 MOV TSTQUE,R1 :(R1) = DEVICE BEING TESTED
016546 012737 000021 001226 MOV #21,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
590
591 016554 004737 041650 JSR PC,TSTPRP :PREPARE DEVICE FOR TEST
016560 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
016562 000404 BR 1$ :GO TO 1$ IF NO ERROR
016564 000240 NOP :RETURN HERE IF ERROR
016566 104000 EMT :ERROR # DEFINED BY TSTPRP SUBROUTINE
016570 000137 017146 JMP 12$ :GO TO 12$ IF ERROR
592 016574 1$:
593 016574 112737 000000 001543 MOVB #RMCS1,PUTINX :SETUP PUT INDEX TABLE
016602 112737 000200 001544 MOVB #200,PUTINX+1 :SET TERMINATOR BYTE
016610 012737 000015 001410 MOV #OFFSET!GO,RMCS10 :SET RMCS1 OUTPUT BUFFER = OFFSET!GO
016616 004737 043160 JSR PC,PUT :GO WRITE RMCS1 VIA PUT SUBROUTINE
016622 000404 BR 2$ :GO TO 2$ IF NO ERROR
016624 000240 NOP :RETURN HERE IF ERROR
016626 104000 EMT :ERROR DEFINED BY PUT SUBROUTINE
016630 000137 017146 JMP 12$ :GO TO 12$ IF ERROR
594 016634 004737 042624 2$: JSR PC,GETSTS :SETUP TO READ ALL REGISTERS
595
596 016640 004737 042710 JSR PC,GET :GO READ REGISTER(S) WITH GET SUBROUTINE

```

121	016644	000404				BR	3\$:GO TO 3\$ IF NO ERROR
	016646	000240				NOP			:RETURN HERE IF ERROR
	016650	104000				EMT			:ERROR # DEFINED BY GET SUBROUTINE
	016652	000137	017146			JMP	12\$:GO TO 12\$ IF ERROR
597	016656	032737	100000	001346	3\$:	BIT	#ATA,RMDSI		:IS ATTENTION SET??
598	016664	001012				BNE	4\$:YES!!
599	016666	012737	100000	001140		MOV	#ATA,\$GDDAT		:GOOD DATA FOR TYPEOUT
600	016674	013737	001346	001142		MOV	RMDSI,\$BDDAT		:BAD DATA FOR TYPEOUT
601	016702	042737	077777	001142		BIC	#^CATA,\$BDDAT		
602	016710	104172				EMT	172		
603	016712				4\$:				
604	016712	004737	050400			JSR	PC,CMPERRSTS		:CHECK ANY ERRORS NOT MASKED
	016716	000000				.WORD	0		:MASK FOR RMER1
	016720	000000				.WORD	0		:MASK FOR RMER2
	016722	000405				BR	5\$:GO TO 5\$ IF NO ERROR
	016724	000240				NOP			:RETURN HERE IF ERROR
	016726	104000				EMT			:ERROR # DEFINED BY CMPERRSTS SUBROUTINE
	016730	004736				JSR	PC,@(SP)+		:GO BACK FOR MORE ERROR CHECKS
	016732	000137	017146			JMP	12\$:GO TO 12\$ IF ERROR
605	016736				5\$:				
606	016736	112737	000000	001543		MOVB	#RMCS1,PUTINX		:SETUP PUT INDEX TABLE
	016744	112737	000200	001544		MOVB	#200,PUTINX+1		:SET TERMINATOR BYTE
	016752	012737	000001	001410		MOV	#NOP!GO,RMCS10		:SET RMCS1 OUTPUT BUFFER = NOP!GO
	016760	004737	043160			JSR	PC,PUT		:GO WRITE RMCS1 VIA PUT SUBROUTINE
	016764	000404				BR	6\$:GO TO 6\$ IF NO ERROR
	016766	000240				NOP			:RETURN HERE IF ERROR
	016770	104000				EMT			:ERROR DEFINED BY PUT SUBROUTINE
	016772	000137	017146			JMP	12\$:GO TO 12\$ IF ERROR
607	016776				6\$:				
608	016776	004737	042710			JSR	PC,GET		:GO READ REGISTER(S) WITH GET SUBROUTINE
	017002	000404				BR	7\$:GO TO 7\$ IF NO ERROR
	017004	000240				NOP			:RETURN HERE IF ERROR
	017006	104000				EMT			:ERROR # DEFINED BY GET SUBROUTINE
	017010	000137	017146			JMP	12\$:GO TO 12\$ IF ERROR
609	017014				7\$:				
610	017014	004737	043706			JSR	PC,PRIERR		:GO CHECK FOR PRIMARY ERRORS
	017020	000405				BR	8\$:GO TO 8\$ IF NO ERROR
	017022	000240				NOP			:RETURN HERE IF ERROR
	017024	104000				EMT			:ERROR # DEFINED BY PRIERR SUBROUTINE
	017026	004736				JSR	PC,@(SP)+		:GO BACK FOR MORE ERROR CHECKS
	017030	000137	017146			JMP	12\$:GO TO 12\$ IF ERROR
611	017034	032737	100000	001346	8\$:	BIT	#ATA,RMDSI		:IS ATTENTION RESET??
612	017042	001411				BEQ	9\$:YES
613	017044	013737	001346	001142		MOV	RMDSI,\$BDDAT		:BAD DATA FOR TYPEOUT
614	017052	042737	077777	001142		BIC	#^CATA,\$BDDAT		
615	017060	005037	001140			CLR	\$GDDAT		:GOOD DATA FOR TYPEOUT
616	017064	104246				EMT	246		
617	017066				9\$:				
618	017066	004737	050400			JSR	PC,CMPERRSTS		:CHECK ANY ERRORS NOT MASKED
	017072	115760				.WORD	NDTMSK		:MASK FOR RMER1
	017074	000010				.WORD	DPE		:MASK FOR RMER2
	017076	000405				BR	10\$:GO TO 10\$ IF NO ERROR
	017100	000240				NOP			:RETURN HERE IF ERROR
	017102	104000				EMT			:ERROR # DEFINED BY CMPERRSTS SUBROUTINE
	017104	004736				JSR	PC,@(SP)+		:GO BACK FOR MORE ERROR CHECKS
	017106	000137	017146			JMP	12\$:GO TO 12\$ IF ERROR
619	017112				10\$:				

```

620 017112 004737 057766      JSR    PC,STCDRVSTS      ;GO CHECK FOR CHANGES IN DRIVE STATUS
      017116 000405          BR     11$              ;GO TO 11$ IF NO ERROR
      017120 000240          NOP                    ;RETURN HERE IF ERROR
      017122 104000          EMT                    ;ERROR # DEFINED BY STCDRVSTS SUBROUTINE
      017124 004736          JSR    PC,@(SP)+        ;GO BACK FOR MORE ERROR CHECKS
      017126 000137 017146    JMP     12$              ;GO TO 12$ IF ERROR
621 017132                    11$:
622 017132 004737 044540      JSR    PC,SECERR         ;GO CHECK FOR SECONDARY ERRORS
      017136 000403          BR     12$              ;GO TO 12$ IF NO ERROR
      017140 000240          NOP                    ;RETURN HERE IF ERROR
      017142 104000          EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      017144 004736          JSR    PC,@(SP)+        ;GO BACK FOR MORE ERROR CHECKS
623 017146                    12$:
624
625
    
```

 *TEST 22 WRITE ATA TEST

 TST22:

```

017146
017146 000004      SCOPE                ;SCOPE CALL
017150 000240      NOP                    ;START OF TEST
017152 012706 001100  MOV    #STACK,SP        ;INITIALIZE STACK POINTER
017156 013700 001276  MOV    $BASE,R0         ;R0 = UNIBUS ADDRESS
017162 013701 001464  MOV    TSTQUE,R1       ;(R1) = DEVICE BEING TESTED
017166 012737 000022 001226  MOV    #22,$TESTN      ;:SET TEST NUMBER IN APT MAIL BOX
626
627 017174 004737 041650      JSR    PC,TSTPRP        ;PREPARE DEVICE FOR TEST
      017200 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
017202 000404      BR     1$              ;GO TO 1$ IF NO ERROR
017204 000240      NOP                    ;RETURN HERE IF ERROR
017206 104000      EMT                    ;ERROR # DEFINED BY TSTPRP SUBROUTINE
017210 000137 017512    JMP     9$              ;GO TO 9$ IF ERROR
628 017214                    1$:
629 017214 112737 000000 001543  MOVB   #RMCS1,PUTINX    ;SETUP PUT INDEX TABLE
      017222 112737 000200 001544  MOVB   #200,PUTINX+1    ;SET TERMINATOR BYTE
      017230 012737 000015 001410  MOV    #OFFSET!GO,RMCS10 ;SET RMCS1 OUTPUT BUFFER = OFFSET!GO
      017236 004737 043160      JSR    PC,PUT           ;GO WRITE RMCS1 VIA PUT SUBROUTINE
      017242 000404          BR     2$              ;GO TO 2$ IF NO ERROR
      017244 000240          NOP                    ;RETURN HERE IF ERROR
      017246 104000          EMT                    ;ERROR DEFINED BY PUT SUBROUTINE
      017250 000137 017512    JMP     9$              ;GO TO 9$ IF ERROR
630 017254 004737 042624      2$: JSR    PC,GETSTS        ;SETUP TO READ ALL REGISTERS
631
632 017260 004737 042710      JSR    PC,GET           ;GO READ REGISTER(S) WITH GET SUBROUTINE
      017264 000404          BR     3$              ;GO TO 3$ IF NO ERROR
      017266 000240          NOP                    ;RETURN HERE IF ERROR
      017270 104000          EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      017272 000137 017512    JMP     9$              ;GO TO 9$ IF ERROR
633 017276 032737 100000 001346 3$: BIT    #ATA,RMDSI      ;IS ATTENTION SET??
634 017304 001012          BNE   4$              ;YES!!
635 017306 013737 001346 001142  MOV    RMDSI,$BDDAT    ;BAD DATA FOR TYPEOUT
636 017314 042737 077777 001142  BIC    #^CATA,$BDDAT
    
```



```

;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
;VERIFY RECALIBRATION
;GO TO 1$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 10$ IF ERROR

017546 000404 BR 1$
017550 000240 NOP
017552 104000 EMT
017554 000137 020104 JMP 10$
663 017560 112737 000000 001543 1$: MOVB #RMCS1,PUTINX ;SETUP PUT INDEX TABLE
664 017560 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
017566 112737 000015 001410 MOV #OFFSET!GO,RMCS10 ;SET RMCS1 OUTPUT BUFFER = OFFSET!GO
017574 012737 043160 JSR PC,PUT ;GO WRITE RMCS1 VIA PUT SUBROUTINE
017602 004737 043160 BR 2$ ;GO TO 2$ IF NO ERROR
017606 000404 NOP ;RETURN HERE IF ERROR
017610 000240 EMT ;ERROR DEFINED BY PUT SUBROUTINE
017612 104000 JMP 10$ ;GO TO 10$ IF ERROR
017614 000137 020104 JSR PC,GETSTS ;SETUP TO READ ALL REGISTERS
665 017620 004737 042624 2$: JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
666 017624 004737 042710 BR 3$ ;GO TO 3$ IF NO ERROR
017630 000404 NOP ;RETURN HERE IF ERROR
017632 000240 EMT ;ERROR # DEFINED BY GET SUBROUTINE
017634 104000 JMP 10$ ;GO TO 10$ IF ERROR
668 017642 032737 100000 001346 3$: BIT #ATA,RMDSI ;IS ATA SET??
669 017650 001012 BNE 4$ ;YES!!
670 017652 013737 001346 001142 MOV RMDSI,$BDDAT ;BAD DATA FOR TYPEOUT
671 017660 042737 077777 001142 BIC #^CATA,$BDDAT
672 017666 012737 100000 001140 MOV #ATA,$GDDAT ;GOOD DATA FOR TYPEOUT
673 017674 104172 EMT 172
674 017676 4$:
675 017676 112737 000014 001543 MOVB #RMER1,PUTINX ;SETUP PUT INDEX TABLE
017704 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
017712 012737 040000 001424 MOV #UNS,RMER10 ;SET RMER1 OUTPUT BUFFER = UNS
017720 004737 043160 JSR PC,PUT ;GO WRITE RMER1 VIA PUT SUBROUTINE
017724 000404 BR 5$ ;GO TO 5$ IF NO ERROR
017726 000240 NOP ;RETURN HERE IF ERROR
017730 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
017732 000137 020104 JMP 10$ ;GO TO 10$ IF ERROR
676 017736 5$:
677 017736 112737 000000 001543 MOVB #RMCS1,PUTINX ;SETUP PUT INDEX TABLE
017744 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
017752 012737 000001 001410 MOV #NOP!GO,RMCS10 ;SET RMCS1 OUTPUT BUFFER = NOP!GO
017760 004737 043160 JSR PC,PUT ;GO WRITE RMCS1 VIA PUT SUBROUTINE
017764 000404 BR 6$ ;GO TO 6$ IF NO ERROR
017766 000240 NOP ;RETURN HERE IF ERROR
017770 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
017772 000137 020104 JMP 10$ ;GO TO 10$ IF ERROR
678 017776 6$:
679 017776 004737 042710 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
020002 000404 BR 7$ ;GO TO 7$ IF NO ERROR
020004 000240 NOP ;RETURN HERE IF ERROR
020006 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
020010 000137 020104 JMP 10$ ;GO TO 10$ IF ERROR
680 020014 032737 100000 001346 7$: BIT #ATA,RMDSI ;IS ATA STILL SET??
681 020022 001012 BNE 8$ ;YES!!
682 020024 012737 100000 001140 MOV #ATA,$GDDAT ;GOOD DATA FOR TYPEOUT

```

```

683 020032 013737 001346 001142      MOV      RMDSI,$BDDAT      ;BAD DATA FOR TYPEOUT
684 020040 042737 077777 001142      BIC      #^CATA,$BDDAT
685 020046 104250                      EMT      250
686 020050                      8$:
687 020050 004737 057766      JSR      PC,STCDRVSTS     ;GO CHECK FOR CHANGES IN DRIVE STATUS
        020054 000405                      BR       9$              ;GO TO 9$ IF NO ERROR
        020056 000240                      NOP
        020060 104000                      EMT
        020062 004736                      JSR      PC,@(SP)+       ;ERROR # DEFINED BY STCDRVSTS SUBROUTINE
        020064 000137 020104      JMP      10$             ;GO BACK FOR MORE ERROR CHECKS
688 020070                      9$:
689 020070 004737 044540      JSR      PC,SECERR        ;GO CHECK FOR SECONDARY ERRORS
        020074 000403                      BR       10$            ;GO TO 10$ IF NO ERROR
        020076 000240                      NOP
        020100 104000                      EMT
        020102 004736                      JSR      PC,@(SP)+       ;ERROR # DEFINED BY SECERR SUBROUTINE
690 020104                      10$:
691
692

```

```

:*****
:*TEST 24      PROGRAM INTERRUPT TEST
:*****
TST24:

```

```

020104
020104 000004      SCOPE      ;SCOPE CALL
020106 000240      NOP        ;START OF TEST
020110 012706 001100  MOV      #STACK,SP     ;INITIALIZE STACK POINTER
020114 013700 001276  MOV      $BASE,R0      ;R0 = UNIBUS ADDRESS
020120 013701 001464  MOV      TSTQUE,R1     ;(R1) = DEVICE BEING TESTED
020124 012737 000024 001226  MOV      #24,$TESTN    ;:SET TEST NUMBER IN APT MAIL BOX
693
694 020132 004737 041650  JSR      PC,TSTPRP     ;PREPARE DEVICE FOR TEST
        020136 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
020140 000404      BR       1$            ;GO TO 1$ IF NO ERROR
020142 000240      NOP
020144 104000      EMT
020146 000137 020574      JMP      16$          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
695 020152                      1$:
696 020152 113702 001272  MOVB     $VECT1,R2     ;R2 = INTERRUPT ADDRESS
697 020156 042702 177400  BIC      #^C<377>,R2   ;CLEAR SIGN EXTENSION
698 020162 011204      MOV      (R2),R4      ;SAVE HANDLER ADDRESS
699 020164 016205 000002  MOV      2(R2),R5     ;SAVE HANDLER PRIORITY
700 020170 012712 020434  MOV      #9$,(R2)     ;WRITE HANDLER ADDRESS AND
701 020174 012762 000300 000002  MOV      #PR6,2(R2)   ;PRIORITY FOR THIS TEST
702 020202 113703 001273  MOVB     $VECT1+1,R3   ;R3 = INTERRUPT LEVEL
703 020206 042703 177400  BIC      #^C<377>,R3   ;CLEAR SIGN EXTENSION
704 020212 000303      SWAB     R3
705 020214 005303      DEC     R3            ;DECREMENT INTERRUPT LEVEL
706 020216 100001      BPL     2$
707 020220 005003      CLR     R3
708 020222 042703 177437  BIC      #^CPR7,R3
709 020226 000240      NOP
710 020230 012746 000300  MOV      #PR6,-(SP)   ;:PUSH #PR6 ON STACK

```



```

744 020456 000137 020574          JMP      16$          ;GO TO 16$ IF ERROR
745 020462 004737 043706          11$:    JSR      PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
      020466 000405          BR       12$          ;GO TO 12$ IF NO ERROR
      020470 000240          NOP      ;RETURN HERE IF ERROR
      020472 104000          EMT     ;ERROR # DEFINED BY PRIERR SUBROUTINE
      020474 004736          JSR     PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      020476 000137 020574          JMP     16$          ;GO TO 16$ IF ERROR
746 020502 004737 050400          12$:    JSR     PC,CMPERRSTS ;CHECK ANY ERRORS NOT MASKED
747 020506 115760          .WORD   NDTMSK      ;MASK FOR RMER1
      020510 000010          .WORD   DPE         ;MASK FOR RMER2
      020512 000405          BR       13$          ;GO TO 13$ IF NO ERROR
      020514 000240          NOP     ;RETURN HERE IF ERROR
      020516 104000          EMT     ;ERROR # DEFINED BY CMPERRSTS SUBROUTINE
      020520 004736          JSR     PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      020522 000137 020574          JMP     16$          ;GO TO 16$ IF ERROR
748 020526 004737 057766          13$:    JSR     PC,STCDRVSTS ;GO CHECK FOR CHANGES IN DRIVE STATUS
749 020532 000405          BR       14$          ;GO TO 14$ IF NO ERROR
      020534 000240          NOP     ;RETURN HERE IF ERROR
      020536 104000          EMT     ;ERROR # DEFINED BY STCDRVSTS SUBROUTINE
      020540 004736          JSR     PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      020542 000137 020574          JMP     16$          ;GO TO 16$ IF ERROR
750 020546 004737 044540          14$:    JSR     PC,SECERR    ;GO CHECK FOR SECONDARY ERRORS
751 020552 000405          BR       15$          ;GO TO 15$ IF NO ERROR
      020554 000240          NOP     ;RETURN HERE IF ERROR
      020556 104000          EMT     ;ERROR # DEFINED BY SECERR SUBROUTINE
      020560 004736          JSR     PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      020562 000137 020574          JMP     16$          ;GO TO 16$ IF ERROR
752 020566 010412          MOV     R4,(R2)
753 020566 010562 000002          MOV     R5,2(R2)
754 020570
755 020574
756
757

```

```

*****
*TEST 25      INHIBIT INTERRUPT TEST
*****
TST25:

```

```

020574          SCOPE          ;SCOPE CALL
020574 000004          NOP          ;START OF TEST
020576 000240          MOV     #STACK,SP ;INITIALIZE STACK POINTER
020600 012706 001100          MOV     $BASE,R0  ;R0 = UNIBUS ADDRESS
020604 013700 001276          MOV     TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
020610 013701 001464          MOV     #25,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
020614 012737 000025 001226
758 020622 004737 041650          JSR     PC,TSTPRP  ;PREPARE DEVICE FOR TEST
759 020626 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
020630 000404          BR       1$        ;GO TO 1$ IF NO ERROR
020632 000240          NOP          ;RETURN HERE IF ERROR

```

```

020634 104000
020636 000137 021142
760 020642 113702 001272
761 020642 042702 177400
762 020646 011204
763 020652 016205 000002
764 020654 012712 021102
765 020660 012762 000300 000002
766 020664 113703 001273
767 020672 042703 177400
768 020676 012746 000300
769 020702 012746 021134
770 020706 010346
771 020712 100001
772 020714 100001
773 020716 005016
774 020720 042716 177437
775 020722 012746 021070
776 020726 010346
777 020732 012746 021016
778 020734 004737 042624
779 020740
780
781 020744 112737 000000 001543
020752 112737 000200 001544
020760 012737 000115 001410
020766 004737 043160
020772 000404
020774 000240
020776 104000
021000 000137 021142
782 021004
783 021004 004737 043522
784 021010 012703 000205
785 021014 000002
786
787 021016 005303
788 021020 100376
789
790 021022 112737 000000 001543
021030 112737 000200 001544
021036 012737 000000 001410
021044 004737 043160
021050 000404
021052 000240
021054 104000
021056 000137 021142
791 021062 012703 000205
792 021066 000002
793
794 021070 012712 021104
795 021074 005303
796 021076 100376
797 021100 000002
798
799 021102 022626
800 021104 022626

```

```

EMT
JMP 13$ :ERROR # DEFINED BY TSTPRP SUBROUTINE
:GO TO 13$ IF ERROR

1$:
MOV B $VECT1,R2 :R2 = INTERRUPT ADDRESS
BIC #^C<377>,R2 :CLEAR SIGN EXTENSION
MOV (R2),R4 :SAVE HANDLER ADDRESS
MOV 2(R2),R5 :SAVE HANDLER PRIORITY
MOV #8$, (R2) :WRITE HANDLER ADDRESS AND
MOV #PR6,2(R2) :PRIORITY FOR THIS TEST
MOV B $VECT1+1,R3 :R3 = INTERRUPT LEVEL
BIC #^C<377>,R3 :CLEAR SIGN EXTENSION
MOV #PR6,-(SP) :PUSH #PR6 ON STACK
MOV #12$,-(SP) :PUSH #12$ ON STACK
MOV R3,-(SP) :PUSH R3 ON STACK
DEC (SP)
BPL 2$
CLR (SP)
2$:
BIC #^CPR7,(SP)
MOV #6$,-(SP) :PUSH #6$ ON STACK
MOV R3,-(SP) :PUSH R3 ON STACK
MOV #4$,-(SP) :PUSH #4$ ON STACK
JSR PC,GETSTS :SETUP TO READ ALL REGISTERS

MOV B #RMCS1,PUTINX :SETUP PUT INDEX TABLE
MOV B #200,PUTINX+1 :SET TERMINATOR BYTE
MOV #OFFSET!GO!IE,RMCS10 :SET RMCS1 OUTPUT BUFFER = OFFSET!GO!IE
JSR PC,PUT :GO WRITE RMCS1 VIA PUT SUBROUTINE
BR 3$ :GO TO 3$ IF NO ERROR
NOP :RETURN HERE IF ERROR
EMT :ERROR DEFINED BY PUT SUBROUTINE
JMP 13$ :GO TO 13$ IF ERROR

3$:
JSR PC,TIMOUT :WAIT FOR GO = 0
MOV #205,R3 :R3 = GROSS TIMER
RTI

4$:
DEC R3 :DELAY
BPL 4$

MOV B #RMCS1,PUTINX :SETUP PUT INDEX TABLE
MOV B #200,PUTINX+1 :SET TERMINATOR BYTE
MOV #NOP,RMCS10 :SET RMCS1 OUTPUT BUFFER = NOP
JSR PC,PUT :GO WRITE RMCS1 VIA PUT SUBROUTINE
BR 5$ :GO TO 5$ IF NO ERROR
NOP :RETURN HERE IF ERROR
EMT :ERROR DEFINED BY PUT SUBROUTINE
JMP 13$ :GO TO 13$ IF ERROR

5$:
MOV #205,R3
RTI

6$:
MOV #9$, (R2)
7$:
DEC R3
BPL 7$
RTI :NO ERROR

8$:
CMP (SP)+,(SP)+ :ADJUST STACK
9$:
CMP (SP)+,(SP)+ :ADJUST STACK

```

```

T25 INHIBIT INTERRUPT TEST
801 021106 012716 021114      MOV      #10$, (SP)
802 021112 000002              RTI
803 021114                    10$:
804 021114 004737 042710      JSR      PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
      021120 000404          BR      11$          ;GO TO 11$ IF NO ERROR
      021122 000240          NOP
      021124 104000          EMT
      021126 000137 021142    JMP      13$          ;ERROR # DEFINED BY GET SUBROUTINE
805 021132                    11$:
      021132 104252          EMT      252
806
807 021134 010412            12$:  MOV      R4,(R2)
808 021136 010562 000002    MOV      R5,2(R2)
809 021142
810
811
      ;*****
      ;*TEST 26      RETURN TO CENTERLINE TEST
      ;*****
      TST26:
      021142
      C21142 000004          SCOPE
      021144 000240          NOP
      021146 012706 001100    MOV      #STACK,SP   ;SCOPE CALL
      021152 013700 001276    MOV      $BASE,R0    ;START OF TEST
      021156 013701 001464    MOV      TSTQUE,R1   ;INITIALIZE STACK POINTER
      021162 012737 000026 001226 MOV      #26,$TESTN  ;R0 = UNIBUS ADDRESS
      ;(R1) = DEVICE BEING TESTED
      ;:SET TEST NUMBER IN APT MAIL BOX
812
813 021170 004737 041650      JSR      PC,TSTPRP   ;PREPARE DEVICE FOR TEST
      021174 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
      021176 000404          BR      1$
      021200 000240          NOP
      021202 104000          EMT
      021204 000137 021464    JMP      9$
      ;GO TO 1$ IF NO ERROR
      ;RETURN HERE IF ERROR
      ;ERROR # DEFINED BY TSTPRP SUBROUTINE
      ;GO TO 9$ IF ERROR
814 021210                    1$:
815 021210 112737 000000 001543 MOVB     #RMCS1,PUTINX ;SETUP PUT INDEX TABLE
      021216 112737 000200 001544 MOVB     #200,PUTINX+1 ;SET TERMINATOR BYTE
      021224 012737 000017 001410 MOV      #RTC!GO,RMCS10 ;SET RMCS1 OUTPUT BUFFER = RTC!GO
      021232 004737 043160      JSR      PC,PUT
      021236 000404          BR      2$
      021240 000240          NOP
      021242 104000          EMT
      021244 000137 021464    JMP      9$
      ;GO WRITE RMCS1 VIA PUT SUBROUTINE
      ;GO TO 2$ IF NO ERROR
      ;RETURN HERE IF ERROR
      ;ERROR DEFINED BY PUT SUBROUTINE
      ;GO TO 9$ IF ERROR
816 021250 004737 042624      2$:  JSR      PC,GETSTS
817 021254 004737 043522      JSR      PC,TIMOUT
      ;SETUP FOR STATUS FETCH
      ;WAIT FOR RECAL TO COMPLETE
818
819 021260 004737 042710      JSR      PC,GET
      021264 000404          BR      3$
      021266 000240          NOP
      021270 104000          EMT
      021272 000137 021464    JMP      9$
      ;GO READ REGISTER(S) WITH GET SUBROUTINE
      ;GO TO 3$ IF NO ERROR
      ;RETURN HERE IF ERROR
      ;ERROR # DEFINED BY GET SUBROUTINE
      ;GO TO 9$ IF ERROR
820 021276                    3$:
821 021276 004737 043706      JSR      PC,PRIERR   ;GO CHECK FOR PRIMARY ERRORS

```

```

021302 000405 BR 4$ :GO TO 4$ IF NO ERROR
021304 000240 NOP :RETURN HERE IF ERROR
021306 104000 EMT :ERROR # DEFINED BY PRIERR SUBROUTINE
021310 004736 JSR PC,@(SP)+ :GO BACK FOR MORE ERROR CHECKS
021312 000137 021464 JMP 9$ :GO TO 9$ IF ERROR
822 021316 032737 000001 001346 4$: BIT #OM,RMDSI :OFFSET MODE OFF??
823 021324 001411 BEQ 5$ :YES!!
824 021326 005037 001140 CLR $GDDAT :GOOD DATA FOR TYPEOUT
825 021332 013737 001346 001142 MOV RMDSI,$BDDAT :BAD DATA FOR TYPEOUT
826 021340 042737 177776 001142 BIC #^COM,$BDDAT
827 021346 104157 EMT 157
828 021350 032737 100000 001346 5$: BIT #ATA,RMDSI :WAS ATTENTION SET ??
829 021356 001012 BNE 6$ :YES!!
830 021360 013737 001346 001142 MOV RMDSI,$BDDAT :BAD DATA FOR TYPEOUT
831 021366 042737 077777 001142 BIC #^CATA,$BDDAT
832 021374 012737 100000 001140 MOV #ATA,$GDDAT :GOOD DATA FOR TYPEOUT
833 021402 104171 EMT 171
834 021404 6$:
835 021404 004737 050400 JSR PC,CMPEERRSTS :CHECK ANY ERRORS NOT MASKED
021410 115760 .WORD NDTMSK :MASK FOR RMER1
021412 000010 .WORD DPE :MASK FOR RMER2
021414 000405 BR 7$ :GO TO 7$ IF NO ERROR
021416 000240 NOP :RETURN HERE IF ERROR
021420 104000 EMT :ERROR # DEFINED BY CMPEERRSTS SUBROUTINE
021422 004736 JSR PC,@(SP)+ :GO BACK FOR MORE ERROR CHECKS
021424 000137 021464 JMP 9$ :GO TO 9$ IF ERROR
836 021430 7$:
837 021430 004737 057766 JSR PC,STCDRVSTS :GO CHECK FOR CHANGES IN DRIVE STATUS
021434 000405 BR 8$ :GO TO 8$ IF NO ERROR
021436 000240 NOP :RETURN HERE IF ERROR
021440 104000 EMT :ERROR # DEFINED BY STCDRVSTS SUBROUTINE
021442 004736 JSR PC,@(SP)+ :GO BACK FOR MORE ERROR CHECKS
021444 000137 021464 JMP 9$ :GO TO 9$ IF ERROR
838 021450 8$:
839 021450 004737 044540 JSR PC,SECERR :GO CHECK FOR SECONDARY ERRORS
021454 000403 BR 9$ :GO TO 9$ IF NO ERROR
021456 000240 NOP :RETURN HERE IF ERROR
021460 104000 EMT :ERROR # DEFINED BY SECERR SUBROUTINE
021462 004736 JSR PC,@(SP)+ :GO BACK FOR MORE ERROR CHECKS
840 021464 9$:
841
842

```

```

*****
:*TEST 27 READ IN PRESET TEST
*****
TST27:

```

```

021464
021464 000004 SCOPE :SCOPE CALL
021466 000240 NOP :START OF TEST
021470 012706 001100 MOV #STACK,SP :INITIALIZE STACK POINTER
021474 013700 001276 MOV $BASE,R0 :R0 = UNIBUS ADDRESS
021500 013701 001464 MOV TSTQUE,R1 : (R1) = DEVICE BEING TESTED
021504 012737 000027 001226 MOV #27,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
843
844 021512 004737 041650 JSR PC,TSTPRP :PREPARE DEVICE FOR TEST
021516 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 1$ IF NO ERROR
                                ;RETURN HERE IF ERROR
                                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                ;GO TO 10$ IF ERROR

021520 000404 BR 1$
021522 000240 NOP
021524 104000 EMT
021526 000137 022046 JMP 10$
845 021532 1$:
846 021532 012702 001543 MOV #PUTINX,R2 ;R2 POINTS TO INDEX TABLE
847 021536 112722 000006 MOVB #RMDA,(R2)+
848 021542 112722 000034 MOVB #RMDC,(R2)+
849 021546 112722 000032 MOVB #RMOF,(R2)+
850 021552 112722 000000 MOVB #RMCS1,(R2)+
851 021556 112722 000200 MOVB #200,(R2)+
852 021562 012737 177777 001416 MOV #-1,RMDAO
853 021570 012737 177777 001444 MOV #-1,RMDCO
854 021576 012737 016200 001442 MOV #FMT16!ECI!HCI!OFD,RMOFO
855 021604 012737 000021 001410 MOV #RIP!GO,RMCS10
856 021612 004737 043160 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
021616 000404 BR 2$ ;GO TO 2$ IF NO ERROR
021620 000240 NOP ;RETURN HERE IF ERROR
021622 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
021624 000137 022046 JMP 10$ ;GO TO 10$ IF ERROR
857 021630 004737 042624 2$: JSR PC,GETSTS ;SETUP FOR STATUS FETCH
858 021634 004737 043522 JSR PC,TIMOUT ;WAIT FOR RIP TO COMPLETE
859
860 021640 004737 042710 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
021644 000404 BR 3$ ;GO TO 3$ IF NO ERROR
021646 000240 NOP ;RETURN HERE IF ERROR
021650 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
021652 000137 022046 JMP 10$ ;GO TO 10$ IF ERROR
861 021656 3$:
862 021656 004737 043706 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
021662 000405 BR 4$ ;GO TO 4$ IF NO ERROR
021664 000240 NOP ;RETURN HERE IF ERROR
021666 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
021670 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
021672 000137 022046 JMP 10$ ;GO TO 10$ IF ERROR
863 021676 013737 001366 001142 4$: MOV RMOFI,$BDDAT ;IS RMOF RESET??
864 021704 042737 161577 001142 BIC #^C<FMT16!ECI!HCI!OFD>,$BDDAT
865 021712 001403 BEQ 5$ ;YES!!
866 021714 005037 001140 CLR $GDDAT ;GOOD DATA FOR TYPEOUT
867 021720 104160 EMT 160
868 021722 005737 001342 5$: TST RMDAI ;IS RMDA RESET??
869 021726 001406 BEQ 6$ ;YES!!
870 021730 013737 001342 001142 MOV RMDAI,$BDDAT ;BAD DATA FOR TYPEOUT
871 021736 005037 001140 CLR $GDDAT ;GOOD DATA FOR TYPEOUT
872 021742 104161 EMT 161
873 021744 005737 001370 6$: TST RMDCI ;IS RMDC RESET??
874 021750 001406 BEQ 7$ ;YES!!
875 021752 013737 001370 001142 MOV RMDCI,$BDDAT ;BAD DATA FOR TYPEOUT
876 021760 005037 001140 CLR $GDDAT ;GOOD DATA FOR TYPEOUT
877 021764 104162 EMT 162
878 021766 7$:
879 021766 004737 050400 JSR PC,CMPEPERRSTS ;CHECK ANY ERRORS NOT MASKED
021772 115760 .WORD ND1MSK ;MASK FOR RMER1
021774 000010 .WORD DPE ;MASK FOR RMER2

```

```

021776 000405 BR 8$ ;GO TO 8$ IF NO ERROR
022000 000240 NOP ;RETURN HERE IF ERROR
022002 104000 EMT ;ERROR # DEFINED BY CMPERRSTS SUBROUTINE
022004 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
022006 000137 022046 JMP 10$ ;GO TO 10$ IF ERROR
880 022012 8$: JSR PC,STCDRVSTS ;GO CHECK FOR CHANGES IN DRIVE STATUS
881 022012 004737 057766 BR 9$ ;GO TO 9$ IF NO ERROR
022016 000405 NOP ;RETURN HERE IF ERROR
022020 000240 EMT ;ERROR # DEFINED BY STCDRVSTS SUBROUTINE
022022 104000 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
022024 004736 022046 JMP 10$ ;GO TO 10$ IF ERROR
882 022032 9$: JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
883 022032 004737 044540 BR 10$ ;GO TO 10$ IF NO ERROR
022036 000403 NOP ;RETURN HERE IF ERROR
022040 000240 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
022042 104000 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
022044 004736 10$:
884 022046
885
886

```

 *TEST 30 RMDC CLEAR OFFSET TEST

 TST30:

```

022046 000004 SCOPE ;SCOPE CALL
022050 000240 NOP ;START OF TEST
022052 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
022056 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
022062 013701 001464 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
022066 012737 000030 001226 MOV #30,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
887
888 022074 004737 041650 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
022100 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
022102 000404 BR 1$ ;GO TO 1$ IF NO ERROR
022104 000240 NOP ;RETURN HERE IF ERROR
022106 104000 EMT ;ERROR # DEFINED BY TSTPRP SUBROUTINE
022110 000137 022364 JMP 8$ ;GO TO 8$ IF ERROR
889 022114 1$: MOV #RMCS1,PUTINX ;SETUP PUT INDEX TABLE
890 022114 112737 000000 001543 MOV #200,PUTINX+1 ;SET TERMINATOR BYTE
022122 112737 000200 001544 MOV #OFFSET!GO,RMCS10 ;SET RMCS1 OUTPUT BUFFER = OFFSET!GO
022130 012737 000015 001410 JSR PC,PUT ;GO WRITE RMCS1 VIA PUT SUBROUTINE
022136 004737 043160 BR 2$ ;GO TO 2$ IF NO ERROR
022142 000404 NOP ;RETURN HERE IF ERROR
022144 000240 EMT ;ERROR DEFINED BY PUT SUBROUTINE
022146 104000 JMP 8$ ;GO TO 8$ IF ERROR
891 022154 004737 042624 2$: JSR PC,GETSTS ;SETUP FOR STATUS
892
893 022160 004737 042710 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
022164 000404 BR 3$ ;GO TO 3$ IF NO ERROR
022166 000240 NOP ;RETURN HERE IF ERROR

```

```

022170 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
022172 000137 022364 JMP 8$ ;GO TO 8$ IF ERROR
894 022176 3$: JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
895 022176 004737 043706 BR 4$ ;GO TO 4$ IF NO ERROR
022202 000405 NOP ;RETURN HERE IF ERROR
022204 000240 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
022206 104000 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
022210 004736 JMP 8$ ;GO TO 8$ IF ERROR
896 022212 000137 022364 4$: BIT #OM,RMSI ;OFFSET ON??
897 022224 001013 BNE 5$ ;YES
898 022226 013737 001346 001142 MOV RMSI,$BDDAT ;BAD DATA FOR TYPEOUT
899 022234 042737 177776 001142 BIC #^COM,$BDDAT
900 022242 012737 000001 001140 MOV #OM,$GDDAT ;GOOD DATA FOR TYPEOUT
901 022250 104156 EMT 156
902 022252 000444 BR 8$ ;SKIP REST OF TEST
903 022254 5$:
904 022254 112737 000034 001543 MOVB #RMDC,PUTINX ;SETUP PUT INDEX TABLE
022262 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
022270 012737 000000 001444 MOV #0,RMDCO ;SET RMDC OUTPUT BUFFER = 0
022276 004737 043160 JSR PC,PUT ;GO WRITE RMDC VIA PUT SUBROUTINE
022302 000404 BR 6$ ;GO TO 6$ IF NO ERROR
022304 000240 NOP ;RETURN HERE IF ERROR
022306 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
022310 000137 022364 JMP 8$ ;GO TO 8$ IF ERROR
905 022314 6$:
906 022314 004737 042710 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
022320 000404 BR 7$ ;GO TO 7$ IF NO ERROR
022322 000240 NOP ;RETURN HERE IF ERROR
022324 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
022326 000137 022364 JMP 8$ ;GO TO 8$ IF ERROR
907 022332 032737 000001 001346 7$: BIT #OM,RMSI ;DID OFFSET MODE RESET??
908 022340 001411 BEQ 8$
909 022342 013737 001346 001142 MOV RMSI,$BDDAT ;BAD DATA FOR TYPEOUT
910 022350 042737 177776 001142 BIC #^COM,$BDDAT
911 022356 005037 001140 CLR $GDDAT ;GOOD DATA FOR TYPEOUT
912 022362 104253 EMT 253
913 022364 8$:
914
915

```

```

*****
:*TEST 31 ILLEGAL FUNCTION TEST
*****
TST31:

```

```

022364
022364 000004 SCOPE ;SCOPE CALL
022366 000240 NOP ;START OF TEST
022370 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
022374 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
022400 013701 001464 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
022404 012737 000031 001226 MOV #31,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
916
917 022412 004737 041650 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
022416 040000 .WORD 040000 ;TASK DESCRIPTOR AS FOLLOWS:
;CLEAR CONTROLLER & SELECT DEVICE
022420 000404 BR 1$ ;GO TO 1$ IF NO ERROR
022422 000240 NOP ;RETURN HERE IF ERROR
022424 104000 EMT ;ERROR # DEFINED BY TSTPRP SUBROUTINE
022426 000137 023150 JMP 13$ ;GO TO 13$ IF ERROR

```

```

918 022432
919 022432 004737 042624
920 022436 012702 000000
921 022442
922 022442 004737 052264
    022446 000404
    022450 000240
    022452 104000
    022454 000137 023150
923 022460
924
925
926
927 022460 112737 000024 001543
    022466 112737 000200 001544
    022474 012737 000001 001434
    022502 004737 043160
    022506 000404
    022510 000240
    022512 104000
    022514 000137 023150
928 022520
929 022520 112737 000024 001543
    022526 112737 000200 001544
    022534 012737 001401 001434
    022542 004737 043160
    022546 000404
    022550 000240
    022552 104000
    022554 000137 023150
930 022560
931
932
933 022560 112737 000000 001543
    022566 112737 000200 001544
    022574 012737 000023 001410
    022602 004737 043160
    022606 000404
    022610 000240
    022612 104000
    022614 000137 023150
934 022620
935 022620 012737 000001 001410
936 022626 050237 001410
937 022632 012737 103274 001414
938 022640 012737 177777 001412
939 022646 005037 001416
940 022652 005037 001444
941 022656 012737 010000 001442
942 022664 012703 001543
943 022670 112723 000004
944 022674 112723 000002
945 022700 112723 000032
946 022704 112723 000006
947 022710 112723 000034
948 022714 112723 000000
949 022720 112713 000200

```

1\$: JSR PC,GETSTS ;SETUP FOR STATUS
MOV #NOP,R2

2\$: JSR PC,CNTCLR ;GO ISSUE CONTROLLER CLEAR
BR 3\$;GO TO 3\$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
EMT ;ERROR NUMBER DEFINED BY SUBROUTINE
JMP 13\$;GO TO 13\$ IF ERROR

3\$: ;SET DIAGNOSTIC MODE AND DIAGNOSTIC UNIT READY, DIAGNOSTIC ON
;CYLINDER

4\$: 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
BR 4\$;GO TO 4\$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
EMT ;ERROR DEFINED BY PUT SUBROUTINE
JMP 13\$;GO TO 13\$ IF ERROR

5\$: MOV #RMMR1,PUTINX ;SETUP PUT INDEX TABLE
MOV #200,PUTINX+1 ;SET TERMINATOR BYTE
MOV #DMD!MUR!MOC,RMMR10 ;SET RMMR1 OUTPUT BUFFER = DMD!MUR!MOC
JSR PC,PUT ;GO WRITE RMMR1 VIA PUT SUBROUTINE
BR 5\$;GO TO 5\$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
EMT ;ERROR DEFINED BY PUT SUBROUTINE
JMP 13\$;GO TO 13\$ IF ERROR

6\$: ;SET VOLUME VALID BEFORE ISSUING FUNCTION CODE BEING TESTED

6\$: MOV #RMCS1,PUTINX ;SETUP PUT INDEX TABLE
MOV #200,PUTINX+1 ;SET TERMINATOR BYTE
MOV #PAKACK!GO,RMCS10 ;SET RMCS1 OUTPUT BUFFER = PAKACK!GO
JSR PC,PUT ;GO WRITE RMCS1 VIA PUT SUBROUTINE
BR 6\$;GO TO 6\$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
EMT ;ERROR DEFINED BY PUT SUBROUTINE
JMP 13\$;GO TO 13\$ IF ERROR

6\$: MOV #GO,RMCS10
BIS R2,RMCS10 ;WRITE FUNCTION CODE IN BUFFER
MOV #BUFONE,RMBA0 ;DUMMY BUS ADDRESS
MOV #-1,RMWCO ;DUMMY WORD COUNT
CLR RMDA0 ;CLEAR DISK ADDRESS
CLR RMDCO ;CLEAR CYLINDER ADDRESS
MOV #FMT16,RMOFO ;16 BIT FORMAT
MOV #PUTINX,R3 ;WRITE REGISTER INDEX TABLE
MOV #RMBA,(R3)+
MOV #RMWC,(R3)+
MOV #RMOF,(R3)+
MOV #RMDA,(R3)+
MOV #RMDC,(R3)+
MOV #RMCS1,(R3)+
MOV #200,(R3)


```

950 022724 004737 043160 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
    022730 000404 BR 7$ ;GO TO 7$ IF NO ERROR
    022732 000240 NOP ;RETURN HERE IF ERROR
    022734 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
    022736 000137 023150 JMP 13$ ;GO TO 13$ IF ERROR
951 022742 004737 043522 7$: JSR PC,TIMOUT ;WAIT FOR GO TO RESET
952
953 022746 004737 042710 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
    022752 000404 BR 8$ ;GO TO 8$ IF NO ERROR
    022754 000240 NOP ;RETURN HERE IF ERROR
    022756 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
    022760 000137 023150 JMP 13$ ;GO TO 13$ IF ERROR
954 022764 8$:
955 022764 004737 043706 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
    022770 000405 BR 9$ ;GO TO 9$ IF NO ERROR
    022772 000240 NOP ;RETURN HERE IF ERROR
    022774 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
    022776 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
    023000 000137 023150 JMP 13$ ;GO TO 13$ IF ERROR
956 023004 016237 067136 001140 9$: MOV FNCDTB(R2),SGDDAT ;GET ENTRY FROM FUNCTION CODE TABLE
957 023012 042737 177776 001140 BIC #^CILF,SGDDAT
958 023020 013737 001350 001142 MOV RMERR1,$BDDAT ;BAD DATA FOR TYPEOUT
959 023026 042737 177776 001142 BIC #^CILF,$BDDAT
960 023034 023737 001140 001142 CMP $GDDAT,$BDDAT ;IS ILF STATUS CORRECT?
961 023042 001402 BEQ 10$ ;YES!!
962 023044 104254 EMT 254
963 023046 000440 BR 13$
964 023050 016237 067136 001140 10$: MOV FNCDTB(R2),SGDDAT
965 023056 032737 040000 001346 BIT #ERR,RMDSI ;WAS AN ERROR DETECTED??
966 023064 001403 BEQ 11$ ;NO!!
967 023066 052737 100000 001140 BIS #ATA,SGDDAT ;YES - ATA SHOULD BE ON
968 023074 042737 077777 001140 11$: BIC #^CATA,SGDDAT
969 023102 013737 001346 001142 MOV RMDSI,$BDDAT ;GET DRIVE'S ATTENTION
970 023110 042737 077777 001142 BIC #^CATA,$BDDAT
971 023116 023737 001140 001142 CMP $GDDAT,$BDDAT ;IS ATA STATUS OK??
972 023124 001402 BEQ 12$ ;YES!!
973 023126 104255 EMT 255
974 023130 000407 BR 13$
975 023132 062702 000002 12$: ADD #2,R2 ;GO TO NEXT FUNCTION CODE
976 023136 022702 000076 CMP #ILF76,R2 ;DONE??
977 023142 103402 BLO 13$ ;YES !!
978 023144 000137 022442 JMP 2$ ;TEST NEXT FUNCTION
979 023150 13$:
980
981

```

```

*****
:TEST 32 INVALID COMMAND TEST
*****
TST32:

```

```

023150
023150 000004 SCOPE ;SCOPE CALL
023152 000240 NOP ;START OF TEST
023154 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
023160 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
023164 013701 001464 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
023170 012737 000032 001226 MOV #32,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
982
983 023176 004737 041650 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
    023202 040000 .WORD 040000 ;TASK DESCRIPTOR AS FOLLOWS:

```

```

023204 000404 BR 1$ ;CLEAR CONTROLLER & SELECT DEVICE
023206 000240 NOP ;GO TO 1$ IF NO ERROR
023210 104000 EMT ;RETURN HERE IF ERROR
023212 000137 023674 JMP 13$ ;ERROR # DEFINED BY TSTPRP SUBROUTINE
984 023216 1$: ;GO TO 13$ IF ERROR
985 023216 004737 042624 JSR PC,GETSTS ;SETUP FOR STATUS
986 023222 012702 000000 MOV #NOP,R2
987 023226 2$:
988 023226 004737 052264 JSR PC,CNTCLR ;GO ISSUE CONTROLLER CLEAR
023232 000404 BR 3$ ;GO TO 3$ IF NO ERROR
023234 000240 NOP ;RETURN HERE IF ERROR
023236 104000 EMT ;ERROR NUMBER DEFINED BY SUBROUTINE
023240 000137 023674 JMP 13$ ;GO TO 13$ IF ERROR
989 023244 3$:
990
991 ;SET DIAGNOSTIC MODE AND DIAGNOSTIC UNIT READY, DIAGNOSTIC ON
992 ;CYLINDER
993 023244 112737 000024 001543 MOVB #RMMR1,PUTINX ;SETUP PUT INDEX TABLE
023252 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
023260 012737 000001 001434 MOV #DMD,RMMR10 ;SET RMMR1 OUTPUT BUFFER = DMD
023266 004737 043160 JSR PC,PUT ;GO WRITE RMMR1 VIA PUT SUBROUTINE
023272 000404 BR 4$ ;GO TO 4$ IF NO ERROR
023274 000240 NOP ;RETURN HERE IF ERROR
023276 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
023300 000137 023674 JMP 13$ ;GO TO 13$ IF ERROR
994 023304 4$:
995 023304 112737 000024 001543 MOVB #RMMR1,PUTINX ;SETUP PUT INDEX TABLE
023312 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
023320 012737 001401 001434 MOV #DMD!MUR!MOC,RMMR10 ;SET RMMR1 OUTPUT BUFFER = DMD!MUR!MOC
023326 004737 043160 JSR PC,PUT ;GO WRITE RMMR1 VIA PUT SUBROUTINE
023332 000404 BR 5$ ;GO TO 5$ IF NO ERROR
023334 000240 NOP ;RETURN HERE IF ERROR
023336 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
023340 000137 023674 JMP 13$ ;GO TO 13$ IF ERROR
996 023344 5$:
997
998 ;VOLUME VALID IS LEFT RESET FOR THE TEST
999
1000 6$:
1001 023344
1002 023344 012737 000001 001410 MOV #GC,RMCS10
1003 023352 050237 001410 BIS R2,RMCS10 ;WRITE FUNCTION CODE IN BUFFER
1004 023356 012737 103274 001414 MOV #BUFONE,RMBA0 ;DUMMY BUS ADDRESS
1005 023364 012737 177777 001412 MOV #-1,RMWCO ;DUMMY WORD COUNT
1006 023372 005037 001416 CLR RMDAO ;CLEAR DISK ADDRESS
1007 023376 005037 001444 CLR RMDCO ;CLEAR CYLINDER ADDRESS
1008 023402 012737 010000 001442 MOV #FMT16,RMOFO ;16 BIT FORMAT
1009 023410 012703 001543 MOV #PUTINX,R3 ;WRITE REGISTER INDEX TABLE
1010 023414 112723 000004 MOVB #RMBA,(R3)+
1011 023420 112723 000002 MOVB #RMWC,(R3)+
1012 023424 112723 000032 MOVB #RMOF,(R3)+
1013 023430 112723 000006 MOVB #RMDA,(R3)+
1014 023434 112723 000034 MOVB #RMDC,(R3)+
1015 023440 112723 000000 MOVB #RMCS1,(R3)+
1016 023444 112713 000200 MOVB #200,(R3)
1017 023450 004737 043160 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
  
```

```

023454 000404 BR 7$ ;GO TO 7$ IF NO ERROR
023456 000240 NOP ;RETURN HERE IF ERROR
023460 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
1018 023462 000137 023674 JMP 13$ ;GO TO 13$ IF ERROR
1019 023466 004737 043522 7$: JSR PC,TIMOUT ;WAIT FOR GO TO RESET
1020 023472 004737 042710 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
023476 000404 BR 8$ ;GO TO 8$ IF NO ERROR
023500 000240 NOP ;RETURN HERE IF ERROR
023502 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
1021 023504 000137 023674 JMP 13$ ;GO TO 13$ IF ERROR
1022 023510 004737 043706 8$: JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
023514 000405 BR 9$ ;GO TO 9$ IF NO ERROR
023516 000240 NOP ;RETURN HERE IF ERROR
023520 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
023522 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
023524 000137 023674 JMP 13$ ;GO TO 13$ IF ERROR
1023 023530 016237 067136 001140 9$: MOV FNCDTB(R?),$GDDAT ;GET ENTRY FROM FUNCTION CODE TABLE
1024 023536 042737 167777 001140 BIC #^CIVC,$GDDAT
1025 023544 013737 001376 001142 MOV RMER2I,$BDDAT ;BAD DATA FOR TYPEOUT
1026 023552 042737 167777 001142 BIC #^CIVC,$BDDAT
1027 023560 023737 001140 001142 CMP $GDDAT,$BDDAT ;IS IVC STATUS CORRECT?
1028 023566 001402 BEQ 10$ ;YES!!
1029 023570 104216 EMT 216
1030 023572 000440 BR 13$
1031 023574 016237 067136 001140 10$: MOV FNCDTB(R2),$GDDAT
1032 023602 032737 040000 001346 BIT #ERR,RMDSI ;WAS AN ERROR DETECTED??
1033 023610 001403 BEQ 11$ ;NO!!
1034 023612 052737 100000 001140 BIS #ATA,$GDDAT ;YES - ATA SHOULD BE ON
1035 023620 042737 077777 001140 11$: BIC #^CATA,$GDDAT
1036 023626 013737 001346 001142 MOV RMDSI,$BDDAT ;GET DRIVE'S ATTENTION
1037 023634 042737 077777 001142 BIC #^CATA,$BDDAT
1038 023642 023737 001140 001142 CMP $GDDAT,$BDDAT ;IS ATA STATUS OK??
1039 023650 001402 BEQ 12$ ;YES!!
1040 023652 104255 EMT 255
1041 023654 000407 BR 13$
1042 023656 062702 000002 12$: ADD #2,R2 ;GO TO NEXT FUNCTION CODE
1043 023662 022702 000076 CMP #1LF76,R2 ;DONE??
1044 023666 103402 BLG 13$ ;YES !!
1045 023670 000137 023226 JMP 2$ ;TEST NEXT FUNCTION
1046 023674 13$:
1047
1048

```

 : TEST 33 INVALID ADDRESS ERROR TEST

 TST33:

```

023674 000004 SCOPE ;SCOPE CALL
023676 000240 NOP ;START OF TEST
023700 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
023704 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
023710 013701 001464 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
023714 012737 000033 001226 MOV #33,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
1049 1050 023722 004737 041650 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
023726 040000 .WORD 040000 ;TASK DESCRIPTOR AS FOLLOWS:
;CLEAR CONTROLLER & SELECT DEVICE

```

```

1051 023730 000404 BR 1$ :GO TO 1$ IF NO ERROR
      023732 000240 NOP :RETURN HERE IF ERROR
      023734 104000 EMT :ERROR # DEFINED BY TSTPRP SUBROUTINE
1051 023736 000137 024464 JMP 13$ :GO TO 13$ IF ERROR
1052 023742 004737 042624 1$: JSR PC,GETSTS :SETUP FOR STATUS
1053 023746 012702 000000 MOV #NOP,R2
1054 023752 2$: JSR PC,CNTCLR :GO ISSUE CONTROLLER CLEAR
1055 023756 000404 BR 3$ :GO TO 3$ IF NO ERROR
      023760 000240 NOP :RETURN HERE IF ERROR
      023762 104000 EMT :ERROR NUMBER DEFINED BY SUBROUTINE
      023764 000137 024464 JMP 13$ :GO TO 13$ IF ERROR
1056 023770 3$:
1057
1058 :SET DIAGNOSTIC MODE AND DIAGNOSTIC UNIT READY, DIAGNOSTIC ON
1059 :CYLINDER
1060 023770 112737 000024 001543 MOVB #RMMR1,PUTINX :SETUP PUT INDEX TABLE
      023776 112737 000200 001544 MOVB #200,PUTINX+1 :SET TERMINATOR BYTE
      024004 012737 000001 001434 MOV #DMD,RMMR10 :SET RMMR1 OUTPUT BUFFER = DMD
      024012 004737 043160 JSR PC,PUT :GO WRITE RMMR1 VIA PUT SUBROUTINE
      024016 000404 BR 4$ :GO TO 4$ IF NO ERROR
      024020 000240 NOP :RETURN HERE IF ERROR
      024022 104000 EMT :ERROR DEFINED BY PUT SUBROUTINE
      024024 000137 024464 JMP 13$ :GO TO 13$ IF ERROR
1061 024030 4$:
1062 024030 112737 000024 001543 MOVB #RMMR1,PUTINX :SETUP PUT INDEX TABLE
      024036 112737 000200 001544 MOVB #200,PUTINX+1 :SET TERMINATOR BYTE
      024044 012737 001401 001434 MOV #DMD!MUR!MOC,RMMR10 :SET RMMR1 OUTPUT BUFFER = DMD!MUR!MOC
      024052 004737 043160 JSR PC,PUT :GO WRITE RMMR1 VIA PUT SUBROUTINE
      024056 000404 BR 5$ :GO TO 5$ IF NO ERROR
      024060 000240 NOP :RETURN HERE IF ERROR
      024062 104000 EMT :ERROR DEFINED BY PUT SUBROUTINE
      024064 000137 024464 JMP 13$ :GO TO 13$ IF ERROR
1063 024070 5$:
1064
1065 :SET VOLUME VALID BEFORE ISSUING FUNCTION CODE BEING TESTED
1066 024070 112737 000000 001543 MOVB #RMCS1,PUTINX :SETUP PUT INDEX TABLE
      024076 112737 000200 001544 MOVB #200,PUTINX+1 :SET TERMINATOR BYTE
      024104 012737 000023 001410 MOV #PAKACK!GO,RMCS10 :SET RMCS1 OUTPUT BUFFER = PAKACK!GO
      024112 004737 043160 JSR PC,PUT :GO WRITE RMCS1 VIA PUT SUBROUTINE
      024116 000404 BR 6$ :GO TO 6$ IF NO ERROR
      024120 000240 NOP :RETURN HERE IF ERROR
      024122 104000 EMT :ERROR DEFINED BY PUT SUBROUTINE
      024124 000137 024464 JMP 13$ :GO TO 13$ IF ERROR
1067 024130 6$:
1068 024130 012737 000001 001410 MOV #GO,RMCS10
1069 024136 050237 001410 BIS R2,RMCS10 :WRITE FUNCTION CODE IN BUFFER
1070 024142 012737 103274 001414 MOV #BUFONE,RMBA0 :DUMMY BUS ADDRESS
1071 024150 012737 177777 001412 MOV #-1,RMWCO :DUMMY WORD COUNT
1072 024156 012737 177777 001416 MOV #-1,RMDAO :USE INVALID DISK ADDRESS
1073 024164 012737 177777 001444 MOV #-1,RMDCO :USE INVALID CYLINDER ADDRESS
1074 024172 012737 010000 001442 MOV #FMT16,RMOFO :16 BIT FORMAT
1075 024200 012703 001543 MOV #PUTINX,R3 :WRITE REGISTER INDEX TABLE
1076 024204 112723 000004 MOVB #RMBA,(R3)+
1077 024210 112723 000002 MOVB #RMWC,(R3)+
1078 024214 112723 000032 MOVB #RMOF,(R3)+
  
```



```

1115 024504 012737 000034 001226      MOV      #34,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
1116 024512 004737 041650      JSR      PC,TSTPRP      ;PREPARE DEVICE FOR TEST
      024516 040000      .WORD    040000        ;TASK DESCRIPTOR AS FOLLOWS:
      024520 000404      BR       1$            ;CLEAR CONTROLLER & SELECT DEVICE
      024522 000240      NOP      ;GO TO 1$ IF NO ERROR
      024524 104000      EMT     ;RETURN HERE IF ERROR
      024526 000137 025250      JMP     13$           ;ERROR # DEFINED BY TSTPRP SUBROUTINE
1117 024532 000137 025250      1$:      JMP     13$           ;GO TO 13$ IF ERROR
1118 024532 004737 042624      JSR      PC,GETSTS     ;SETUP FOR STATUS
1119 024536 012702 000000      MOV     #NOP,R2
1120 024542 000137 025250      2$:      JMP     13$           ;GO ISSUE CONTROLLER CLEAR
1121 024542 004737 052264      JSR      PC,CNTCLR     ;GO TO 3$ IF NO ERROR
      024546 000404      BR       3$            ;RETURN HERE IF ERROR
      024550 000240      NOP     ;ERROR NUMBER DEFINED BY SUBROUTINE
      024552 104000      EMT     ;GO TO 13$ IF ERROR
      024554 000137 025250      JMP     13$
1122 024560 000137 025250      3$:      JMP     13$
1123
1124      ;SET DIAGNOSTIC MODE AND DIAGNOSTIC UNIT READY, DIAGNOSTIC ON
1125      ;CYLINDER, AND DIAGNOSTIC WRITE LOCK
1126 024560 112737 000024 001543      MOV     #RMMR1,PUTINX ;SETUP PUT INDEX TABLE
      024566 112737 C00200 001544      MOV     #200,PUTINX+1 ;SET TERMINATOR BYTE
      024574 012737 000001 001434      MOV     #DMD,RMMR10   ;SET RMMR1 OUTPUT BUFFER = DMD
      024602 004737 043160      JSR      PC,PUT        ;GO WRITE RMMR1 VIA PUT SUBROUTINE
      024606 000404      BR       4$            ;GO TO 4$ IF NO ERROR
      024610 000240      NOP     ;RETURN HERE IF ERROR
      024612 104000      EMT     ;ERROR DEFINED BY PUT SUBROUTINE
      024614 000137 025250      JMP     13$           ;GO TO 13$ IF ERROR
1127 024620 000137 025250      4$:      JMP     13$
1128 024620 112737 000024 001543      MOV     #RMMR1,PUTINX ;SETUP PUT INDEX TABLE
      024626 112737 000200 001544      MOV     #200,PUTINX+1 ;SET TERMINATOR BYTE
      024634 012737 001411 001434      MOV     #DMD!MUR!MOC!MWP,RMMR10 ;SET RMMR1 OUTPUT BUFFER = DMD!MUR!MOC!MWP
      024642 004737 043160      JSR      PC,PUT        ;GO WRITE RMMR1 VIA PUT SUBROUTINE
      024646 000404      BR       5$            ;GO TO 5$ IF NO ERROR
      024650 000240      NOP     ;RETURN HERE IF ERROR
      024652 104000      EMT     ;ERROR DEFINED BY PUT SUBROUTINE
      024654 000137 025250      JMP     13$           ;GO TO 13$ IF ERROR
1129 024660 000137 025250      5$:      JMP     13$
1130
1131      ;SET VOLUME VALID BEFORE ISSUING FUNCTION CODE BEING TESTED
1132 024660 112737 000000 001543      MOV     #RMCS1,PUTINX ;SETUP PUT INDEX TABLE
      024666 112737 000200 001544      MOV     #200,PUTINX+1 ;SET TERMINATOR BYTE
      024674 012737 000023 001410      MOV     #PAKACK!GO,RMCS10 ;SET RMCS1 OUTPUT BUFFER = PAKACK!GO
      024702 004737 043160      JSR      PC,PUT        ;GO WRITE RMCS1 VIA PUT SUBROUTINE
      024706 000404      BR       6$            ;GO TO 6$ IF NO ERROR
      024710 000240      NOP     ;RETURN HERE IF ERROR
      024712 104000      EMT     ;ERROR DEFINED BY PUT SUBROUTINE
      024714 000137 025250      JMP     13$           ;GO TO 13$ IF ERROR
1133 024720 000137 025250      6$:      JMP     13$
1134 024720 012737 000001 001410      MOV     #GO,RMCS10    ;WRITE FUNCTION CODE IN BUFFER
1135 024726 050237 001410      BIS     R2,RMCS10
1136 024732 012737 103274 001414      MOV     #BUFONE,RMBA0 ;DUMMY BUS ADDRESS
1137 024740 012737 177777 001412      MOV     #-1,RMWCO    ;DUMMY WORD COUNT
1138 024746 005037 001416      CLR     RMDA0        ;CLEAR DISK ADDRESS
1139 024752 005037 001444      CLR     RMDCO        ;CLEAR CYLINDER ADDRESS
    
```

```

1140 024756 012737 010000 001442 MOV #FMT16,RMOFO ;16 BIT FORMAT
1141 024764 012703 001543 MOV #PUTINX,R3 ;WRITE REGISTER INDEX TABLE
1142 024770 112723 000004 MOVB #RMB,(R3)+
1143 024774 112723 000002 MOVB #RMWC,(R3)+
1144 025000 112723 000032 MOVB #RMOF,(R3)+
1145 025004 112723 000006 MOVB #RMDA,(R3)+
1146 025010 112723 000034 MOVB #RMDC,(R3)+
1147 025014 112723 000000 MOVB #RMCST,(R3)+
1148 025020 112713 000200 MOVB #200,(R3)
1149 025024 004737 043160 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
025030 000404 BR 7$ ;GO TO 7$ IF NO ERROR
025032 000240 NOP ;RETURN HERE IF ERROR
025034 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
025036 000137 025250 JMP 13$ ;GO TO 13$ IF ERROR
1150 025042 004737 043522 7$: JSR PC,TIMOUT ;WAIT FOR GO TO RESET
1151
1152 025046 004737 042710 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
025052 000404 BR 8$ ;GO TO 8$ IF NO ERROR
025054 000240 NOP ;RETURN HERE IF ERROR
025056 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
025060 000137 025250 JMP 13$ ;GO TO 13$ IF ERROR
1153 025064 8$:
1154 025064 004737 043706 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
025070 000405 BR 9$ ;GO TO 9$ IF NO ERROR
025072 000240 NOP ;RETURN HERE IF ERROR
025074 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
025076 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
025100 000137 025250 JMP 13$ ;GO TO 13$ IF ERROR
1155 025104 016237 067136 001140 9$: MOV FNCDTB(R2),SGDDAT ;GET ENTRY FROM FUNCTION CODE TABLE
1156 025112 042737 173777 001140 BIC #^CWLE,SGDDAT
1157 025120 013737 001350 001142 MOV RMERR1,$BDDAT ;BAD DATA FOR TYPEOUT
1158 025126 042737 173777 001142 BIC #^CWLE,$BDDAT
1159 025134 023737 001140 001142 CMP $GDDAT,$BDDAT ;IS WLE STATUS CORRECT?
1160 025142 001402 BEQ 10$ ;YES!!
1161 025144 104220 EMT 220
1162 025146 000440 BR 13$
1163 025150 016237 067136 001140 10$: MOV FNCDTB(R2),SGDDAT
1164 025156 032737 040000 001346 BIT #ERR,RMDSI ;WAS AN ERROR DETECTED??
1165 025164 001403 BEQ 11$ ;NO!!
1166 025166 052737 100000 001140 BIS #ATA,SGDDAT ;YES - ATA SHOULD BE ON
1167 025174 042737 077777 001140 11$: BIC #^CATA,SGDDAT
1168 025202 013737 001346 001142 MOV RMDSI,$BDDAT ;GET DRIVE'S ATTENTION
1169 025210 042737 077777 001142 BIC #^CATA,$BDDAT
1170 025216 023737 001140 001142 CMP $GDDAT,$BDDAT ;IS ATA STATUS OK??
1171 025224 001402 BEQ 12$ ;YES!!
1172 025226 104255 EMT 255
1173 025230 000407 BR 13$
1174 025232 062702 000002 12$: ADD #2,R2 ;GO TO NEXT FUNCTION CODE
1175 025236 022702 000076 CMP #1LF76,R2 ;DONE??
1176 025242 103402 BLO 13$ ;YES !!
1177 025244 000137 024542 JMP 2$ ;TEST NEXT FUNCTION
1178 025250 13$:
1179
1180

```

```

*****
: *TEST 35 ERROR ABORT TESTS
*****
TST35:

```

025250

	025250	000004			SCOPE		:SCOPE CALL
	025252	000240			NOP		:START OF TEST
	025254	012706	001100		MOV	#STACK,SP	:INITIALIZE STACK POINTER
	025260	013700	001276		MOV	\$BASE,R0	:R0 = UNIBUS ADDRESS
	025264	013701	001464		MOV	TSTQUE,R1	:(R1) = DEVICE BEING TESTED
	025270	012737	000035	001226	MOV	#35,\$TESTN	;;SET TEST NUMBER IN APT MAIL BOX
1181							
1182	025276	004737	041650		JSR	PC,TSTPRP	:PREPARE DEVICE FOR TEST
	025302	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 1\$ IF NO ERROR
							:RETURN HERE IF ERROR
							:ERROR # DEFINED BY TSTPRP SUBROUTINE
							:GO TO 9\$ IF ERROR
	025304	000404			BR	1\$	
	025306	000240			NOP		
	025310	104000			EMT		
	025312	000137	025602		JMP	9\$	
1183	025316					1\$:	
1184							
1185							:SETUP GET INDEX TABLE TO READ ALL REGISTERS
	025316	004737	042624		JSR	PC,GETSTS	:GO TO GETSTS SUBROUTINE
1186	025322	012702	000000		MOV	#NOP,R2	:R2 = FUNCTION CODE
1187	025326					2\$:	
1188	025326	004737	052264		JSR	PC,CNTCLR	:GO ISSUE CONTROLLER CLEAR
	025332	000404			BR	3\$:GO TO 3\$ IF NO ERROR
	025334	000240			NOP		:RETURN HERE IF ERROR
	025336	104000			EMT		:ERROR # DEFINED BY CNTCLR SUBROUTINE
	025340	000137	025602		JMP	9\$:GO TO 9\$ IF ERROR
1189	025344					3\$:	
1190	025344	112737	000014	001543	MOVB	#RMER1,PUTINX	:SETUP PUT INDEX TABLE
	025352	112737	000200	001544	MOVB	#200,PUTINX+1	:SET TERMINATOR BYTE
	025360	012737	040000	001424	MOV	#UNS,RMER10	:SET RMER1 OUTPUT BUFFER = UNS
	025366	004737	043160		JSR	PC,PUT	:GO WRITE RMER1 VIA PUT SUBROUTINE
	025372	000404			BR	4\$:GO TO 4\$ IF NO ERROR
	025374	000240			NOP		:RETURN HERE IF ERROR
	025376	104000			EMT		:ERROR DEFINED BY PUT SUBROUTINE
	025400	000137	025602		JMP	9\$:GO TO 9\$ IF ERROR
1191	025404					4\$:	
1192	025404	012737	000001	001410	MOV	#GO,RMCS10	
1193	025412	050237	001410		BIS	R2,RMCS10	:WRITE FUNCTION CODE IN BUFFER
1194	025416	012737	103274	001414	MOV	#BUFONE,RMBA0	:DUMMY BUS ADDRESS
1195	025424	012737	177777	001412	MOV	#-1,RMWCO	:DUMMY WORD COUNT
1196	025432	012737	000000	001416	MOV	#0,RMDAO	:CLEAR DISK ADDRESS
1197	025440	012737	000000	001444	MOV	#0,RMDCO	:CLEAR CYLINDER ADDRESS
1198	025446	012737	010000	001442	MOV	#FMT16,RMOFO	:16 BIT FORMAT
1199	025454	012703	001543		MOV	#PUTINX,R3	:WRITE REGISTER INDEX TABLE
1200	025460	112723	000004		MOVB	#RMBA,(R3)+	
1201	025464	112723	000002		MOVB	#RMWC,(R3)+	
1202	025470	112723	000032		MOVB	#RMOF,(R3)+	
1203	025474	112723	000006		MOVB	#RMDA,(R3)+	
1204	025500	112723	000034		MOVB	#RMDC,(R3)+	
1205	025504	112723	000000		MOVB	#RMCS1,(R3)+	
1206	025510	112713	000200		MOVB	#200,(R3)	
1207	025514	004737	043160		JSR	PC,PUT	:GO WRITE REGISTER(S) WITH PUT SUBROUTINE
	025520	000402			BR	5\$:GO TO 5\$ IF NO ERROR


```
1208 025522 000240      NOP      :RETURN HERE IF ERROR
      025524 104000      EMT      :ERROR # DEFINED BY PUT SUBROUTINE
1209 025526      5$:
1210      :WAIT FOR COMMAND TO COMPLETE
      025526 004737 043522 JSR      PC,TIMOUT :GO TO TIMOUT SUBROUTINE
1211      JSR      PC,GET :GO READ REGISTER(S) WITH GET SUBROUTINE
1212 025532 004737 042710 BR      6$ :GO TO 6$ IF NO ERROR
      025536 000404      NOP      :RETURN HERE IF ERROR
      025540 000240      EMT      :ERROR # DEFINED BY GET SUBROUTINE
      025542 104000      JMP      9$ :GO TO 9$ IF ERROR
1213 025544 000137 025602
1214 025550      6$:
      025550 004737 043706 JSR      PC,PRIERR :GO CHECK FOR PRIMARY ERRORS
      025554 000405      BR      7$ :GO TO 7$ IF NO ERROR
      025556 000240      NOP      :RETURN HERE IF ERROR
      025560 104000      EMT      :ERROR # DEFINED BY PRIERR SUBROUTINE
      025562 004736      JSR      PC,@(SP)+ :GO BACK FOR MORE ERROR CHECKS
      025564 000137 025602 JMP      9$ :GO TO 9$ IF ERROR
1215 025570      7$:
1216 025570      8$:
1217 025570 062702 000002 ADD      #2,R2 :ADVANCE FUNCTION CODE
1218 025574 022702 000076 CMP      #ILF76,R2 :DONE ALL COMMANDS
1219 025600 103252      BHS     2$ :NO !!
1220 025602      9$:
1221
1222
*****
:*TEST 36      RMR TEST
*****
TST36:
025602      SCOPE      :SCOPE CALL
025602 000004      NOP      :START OF TEST
025604 000240      MOV      #STACK,SP :INITIALIZE STACK POINTER
025606 012706 001100      MOV      $BASE,R0 :R0 = UNIBUS ADDRESS
025612 013700 001276      MOV      TSTQUE,R1 : (R1) = DEVICE BEING TESTED
025616 013701 001464      MOV      #36,$TESTN :SET TEST NUMBER IN APT MAIL BOX
025622 012737 000036 001226
1223
1224 025630 004737 041650 JSR      PC,TSTPRP :PREPARE DEVICE FOR TEST
      025634 040000      .WORD 040000 :TASK DESCRIPTOR AS FOLLOWS:
      :CLEAR CONTROLLER & SELECT DEVICE
      025636 000404      BR      1$ :GO TO 1$ IF NO ERROR
      025640 000240      NOP      :RETURN HERE IF ERROR
      025642 104000      EMT      :ERROR # DEFINED BY TSTPRP SUBROUTINE
      025644 000137 026242 JMP      11$ :GO TO 11$ IF ERROR
1225 025650      1$:
1226
1227      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      025650 004737 042624 JSR      PC,GETSTS :GO TO GETSTS SUBROUTINE
1228 025654      2$:
1229 025654 112737 000024 001543 MOVB     #RMMR1,PUTINX :SETUP PUT INDEX TABLE
      025662 112737 000200 001544 MOVB     #200,PUTINX+1 :SET TERMINATOR BYTE
      025670 012737 000001 001434 MOV      #DMD,RMMR10 :SET RMMR1 OUTPUT BUFFER = DMD
      025676 004737 043160 JSR      PC,PUT :GO WRITE RMMR1 VIA PUT SUBROUTINE
      025702 000404      BR      3$ :GO TO 3$ IF NO ERROR
      025704 000240      NOP      :RETURN HERE IF ERROR
      025706 104000      EMT      :ERROR DEFINED BY PUT SUBROUTINE
      025710 000137 026242 JMP      11$ :GO TO 11$ IF ERROR
```

```

1230 025714
1231 025714 112737 000024 001543 3$: MOVB #RMMR1,PUTINX ;SETUP PUT INDEX TABLE
      025722 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
      025730 012737 001401 001434 MOV #DMD!MUR!MOC,RMMR10 ;SET RMMR1 OUTPUT BUFFER = DMD!MUR!MOC
      025736 004737 043160 JSR PC,PUT ;GO WRITE RMMR1 VIA PUT SUBROUTINE
      025742 000404 BR 4$ ;GO TO 4$ IF NO ERROR
      025744 000240 NOP ;RETURN HERE IF ERROR
      025746 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
      025750 000137 026242 JMP 11$ ;GO TO 11$ IF ERROR
1232 025754
1233 025754 112737 000000 001543 4$: MOVB #RMCS1,PUTINX ;SETUP PUT INDEX TABLE
      025762 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
      025770 012737 000023 001410 MOV #PAKACK!GO,RMCS10 ;SET RMCS1 OUTPUT BUFFER = PAKACK!GO
      025776 004737 043160 JSR PC,PUT ;GO WRITE RMCS1 VIA PUT SUBROUTINE
      026002 000404 BR 5$ ;GO TO 5$ IF NO ERROR
      026004 000240 NOP ;RETURN HERE IF ERROR
      026006 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
      026010 000137 026242 JMP 11$ ;GO TO 11$ IF ERROR
1234 026014
1235 026014 004737 042710 5$: JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
      026020 000404 BR 6$ ;GO TO 6$ IF NO ERROR
      026022 000240 NOP ;RETURN HERE IF ERROR
      026024 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
      026026 000137 026242 JMP 11$ ;GO TO 11$ IF ERROR
1236 026032
1237 026032 000240 6$: NOP
1238 026034 112737 000024 001543 MOVB #RMMR1,PUTINX ;SETUP PUT INDEX TABLE
      026042 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
      026050 012737 041401 001434 MOV #DMD!MUR!MOC!DBEN,RMMR10 ;SET RMMR1 OUTPUT BUFFER = DMD!MUR!MOC!DBEN
      026056 004737 043160 JSR PC,PUT ;GO WRITE RMMR1 VIA PUT SUBROUTINE
      026062 000404 BR 7$ ;GO TO 7$ IF NO ERROR
      026064 000240 NOP ;RETURN HERE IF ERROR
      026066 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
      026070 000137 026242 JMP 11$ ;GO TO 11$ IF ERROR
1239 026074
1240 026074 112737 000000 001543 7$: MOVB #RMCS1,PUTINX ;SETUP PUT INDEX TABLE
      026102 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
      026110 012737 000001 001410 MOV #NOP!GO,RMCS10 ;SET RMCS1 OUTPUT BUFFER = NOP!GO
      026116 004737 043160 JSR PC,PUT ;GO WRITE RMCS1 VIA PUT SUBROUTINE
      026122 000404 BR 8$ ;GO TO 8$ IF NO ERROR
      026124 000240 NOP ;RETURN HERE IF ERROR
      026126 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
      026130 000137 026242 JMP 11$ ;GO TO 11$ IF ERROR
1241 026134
1242 026134 112737 000000 001543 8$: MOVB #RMCS1,PUTINX ;SETUP PUT INDEX TABLE
      026142 112737 000200 001544 MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
      026150 012737 000015 001410 MOV #OFFSET!GO,RMCS10 ;SET RMCS1 OUTPUT BUFFER = OFFSET!GO
      026156 004737 043160 JSR PC,PUT ;GO WRITE RMCS1 VIA PUT SUBROUTINE
      026162 000404 BR 9$ ;GO TO 9$ IF NO ERROR
      026164 000240 NOP ;RETURN HERE IF ERROR
      026166 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
      026170 000137 026242 JMP 11$ ;GO TO 11$ IF ERROR
1243 026174
1244 026174 004737 042710 9$: JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
      026200 000404 BR 10$ ;GO TO 10$ IF NO ERROR
      026202 000240 NOP ;RETURN HERE IF ERROR
      026204 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
  
```

```

1245 026206 000137 026242          JMP      11$          ;GO TO 11$ IF ERROR
1246 026212 000240          10$:      NOP
1247 026214 032737 000004 001350      BIT      #RMR,RMER11 ;IS RMR SET ??
1248 026222 001007          BNE     11$          ;YES !!
1249 026224 012737 000004 001140      MOV     #RMR,$GDDAT ;EXPECTED STATUS
1250 026232 013737 001350 001142      MOV     RMER11,$BDDAT ;RECEIVED STATUS
1251 026240 104222          EMT     222
1252 026242          11$:
1253
1254

*****
:TEST 37          PARITY ERROR TEST
*****
TST37:
026242          SCOPE          ;SCOPE CALL
026242 000004          NOP          ;START OF TEST
026244 000240          MOV     #STACK,SP ;INITIALIZE STACK POINTER
026246 012706 001100      MOV     $BASE,R0   ;R0 = UNIBUS ADDRESS
026252 013700 001276      MOV     TSTQUE,R1  ;(R1) = DEVICE BEING TESTED
026256 013701 001464      MOV     #37,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
1255 026262 012737 000037 001226      JSR     PC,TSTPRP  ;PREPARE DEVICE FOR TEST
1256 026270 004737 041650      .WORD  040000     ;TASK DESCRIPTOR AS FOLLOWS:
026274 040000          BR      1$          ;CLEAR CONTROLLER & SELECT DEVICE
026276 000404          NOP          ;GO TO 1$ IF NO ERROR
026300 000240          EMT          ;RETURN HERE IF ERROR
026302 104000          JMP     7$          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
1257 026310 000137 026462          1$:      ;GO TO .7$ IF ERROR
1258 026310 012702 000001      MOV     #1,R2      ;R2 = DATA PATTERN
1259 026314          2$:
1260 026314 004737 052264      JSR     PC,CNTCLR  ;GO ISSUE CONTROLLER CLEAR
026320 000404          BR      3$          ;GO TO 3$ IF NO ERROR
026322 000240          NOP          ;RETURN HERE IF ERROR
026324 104000          EMT          ;ERROR # DEFINED BY CNTCLR SUBROUTINE
026326 000137 026462          JMP     7$          ;GO TO 7$ IF ERROR
1261 026332          3$:
1262 026332 111103          MOVB   (R1),R3     ;SETUP RMCS2
1263 026334 042703 177770      BIC   #^CUNTMSK,R3
1264 026340 052703 000020      BIS   #PAT,R3
1265 026344 010337 001420      MOV   R3,RMCS20   ;OUTPUT VALUE TO RMCS2
1266 026350 010237 001416      MOV   R2,RMDAO    ;VALUE TO RMDA
1267 026354 012703 001543      MOV   #PUTINX,R3  ;WRITE REGISTER OUTPUT INDEX
1268 026360 112723 000010      MOVB  #RMCS2,(R3)+
1269 026364 112723 000006      MOVB  #RMDA,(R3)+
1270 026370 112723 000200      MOVB  #200,(R3)+
1271
1272 026374 004737 043160      JSR   PC,PUT      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
026400 000402          BR    4$          ;GO TO 4$ IF NO ERROR
026402 000240          NOP          ;RETURN HERE IF ERROR
026404 104000          EMT          ;ERROR # DEFINED BY PUT SUBROUTINE
1273 026406          4$:
1274 026406 004737 042710      JSR   PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
026412 000404          BR    5$          ;GO TO 5$ IF NO ERROR
026414 000240          NOP          ;RETURN HERE IF ERROR
026416 104000          EMT          ;ERROR # DEFINED BY GET SUBROUTINE
026420 000137 026462          JMP   7$          ;GO TO 7$ IF ERROR
    
```

```

1275 026424          5$:
1276 026424 032737 000010 001350      BIT    #PAR,RMER11    ;IS PARITY ERROR SET ??
1277 026432 001011          BNE    6$           ;YES !!
1278 026434 012737 000010 001140      MOV    #PAR,$GDDAT  ;EXPECTED STATUS
1279 026442 013737 001350 001142      MOV    RMER11,$BDDAT ;RECEIVED STATUS
1280 026450 010237 001174          MOV    R2,$TMP0     ;DATA PATTERN
1281 026454 104223          EMT    223
1282 026456          6$:
1283 026456 006302          ASL    R2           ;ADVANCE DATA PATTERN
1284 026460 001315          BNE    2$         ;BRANCH IF NOT DONE
1285 026462          7$:
1286
1287
*****
;TEST 40      ILLEGAL REGISTER TEST
*****
TST40:
026462          SCOPE          ;SCOPE CALL
026462 000004          NOP          ;START OF TEST
026464 000240          MOV    #STACK,SP  ;INITIALIZE STACK POINTER
026466 012706 001100      MOV    $BASE,R0   ;R0 = UNIBUS ADDRESS
026472 013700 001276      MOV    TSTQUE,R1  ;(R1) = DEVICE BEING TESTED
026476 013701 001464      MOV    #40,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
1288 026502 012737 000040 001226      JSR    PC,TSTPRP  ;PREPARE DEVICE FOR TEST
1289 026510 004737 041650      .WORD 040000     ;TASK DESCRIPTOR AS FOLLOWS:
026514 040000          ;CLEAR CONTROLLER & SELECT DEVICE
026516 000404          BR     1$        ;GO TO 1$ IF NO ERROR
026520 000240          NOP          ;RETURN HERE IF ERROR
026522 104000          EMT          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
026524 000137 027010      JMP    10$       ;GO TO 10$ IF ERROR
1290 026530          1$:
1291 026530 005005          CLR    R5         ;R5 = EXPECTED STATUS
1292 026532 004737 042624      JSR    PC,GETSTS  ;SETUP FOR STATUS
1293 026536 013746 000004      MOV    ERRVEC,-(SP) ;PUSH ERRVEC ON STACK
1294 026542 013746 000006      MOV    ERRVEC+2,-(SP) ;PUSH ERRVEC+2 ON STACK
1295 026546 012737 027006 000004      MOV    #9$,ERRVEC  ;SETUP FOR BUS TIMEOUT
1296 026554 012737 000300 000006      MOV    #PR6,ERRVEC+2
1297 026562 005002          CLR    R2         ;R2 = REGISTER INDEX
1298 026564          2$:
1299 026564 004737 052264      JSR    PC,CNTCLR  ;GO ISSUE CONTROLLER CLEAR
026570 000404          BR     3$        ;GO TO 3$ IF NO ERROR
026572 000240          NOP          ;RETURN HERE IF ERROR
026574 104000          EMT          ;ERROR NUMBER DEFINED BY SUBROUTINE
026576 000137 026722      JMP    7$        ;GO TO 7$ IF ERROR
1300 026602 010003          3$:      MOV    R0,R3     ;R3 = REGISTER ADDRESS
1301 026604 060203          ADD    R2,R3
1302 026606 005013          CLR    (R3)      ;CLEAR THE REGISTER
1303 026610 004737 050612      JSR    PC,DEVSEL  ;GO SELECT DEVICE
026614 000404          BR     4$        ;GO TO 4$ IF NO ERROR
026616 000240          NOP          ;RETURN HERE IF ERROR
026620 104000          EMT          ;ERROR # DEFINED BY DEVSEL SUBROUTINE
026622 000137 027010      JMP    10$       ;GO TO 10$ IF ERROR
1304 026626          4$:
1305 026626 004737 042710      JSR    PC,GET     ;GO READ REGISTER(S) WITH GET SUBROUTINE
026632 000404          BR     5$        ;GO TO 5$ IF NO ERROR
026634 000240          NOP          ;RETURN HERE IF ERROR
026636 104000          EMT          ;ERROR # DEFINED BY GET SUBROUTINE

```

```

1306 026640 000137 026722          5$: JMP      7$          :GO TO 7$ IF ERROR
1307 026644 004737 043706          JSR      PC,PRIERR   :GO CHECK FOR PRIMARY ERRORS
      026650 000405          BR       6$          :GO TO 6$ IF NO ERROR
      026652 000240          NOP      :RETURN HERE IF ERROR
      026654 104000          EMT     :ERROR # DEFINED BY PRIERR SUBROUTINE
      026656 004736          JSR      PC,@(SP)+  :GO BACK FOR MORE ERROR CHECKS
      026660 000137 026722          JMP      7$          :GO TO 7$ IF ERROR
1308 026664 010537 001140          6$: MOV      R5,$GDDAT  :GET DRIVE'S ILR STATUS
1309 026670 013737 001350 001142  MOV      RMER1I,$BDDAT
1310 026676 042737 177775 001142  BIC      #^CILR,$BDDAT
1311 026704 023737 001140 001142  CMP      $GDDAT,$BDDAT :IS ILR STATUS OK??
1312 026712 001403          BEQ      7$          :YES!!
1313 026714 010237 001174          MOV      R2,$TMP0   :SAVE R2 FOR ERROR DATA
1314 026720 104256          EMT     256
1336 026722 062702 000002          7$: ADD      #2,R2      :INCREMENT THE REGISTER ADDRESS
1337 026726 022702 000050          CMP      #50,R2     :TIME TO CHECK THE EXTEND ADDRESS REG ?
1338 026732 101314          BHI     2$          :BRANCH IF NOT
1339 026734 103420          BLO     8$          :BRANCH IF ALREADY SET UP
1340 026736 012705 000002          MOV      #ILR,R5    :EXCEPT ILR HAPPEND
1341 026742 012760 001400 000000  MOV      #A16!A17,RMCS1(R0) :SET EXTENDED ADDRESS BITS
1342 026750 012702 000054          MOV      #54,R2     :SET ADDRESS TO 54,IF 22 REGISTERS
1343 026754 016003 000050          MOV      50(R0),R3  :IS THIS 22 REG RH70 ?
1344 026760 042703 177774          BIC      #177774,R3 :LEFT ONLY BIT 0 AND BIT 1
1345 026764 022703 000003          CMP      #BIT0!BIT1,R3 :BIT 0 AND BIT 1 SET AT THE SAME TIME ?
1346 026770 001402          BEQ     8$          :BRANCH IF SO
1347 026772 012702 000050          MOV      #50,R2     :OTHERWISE,SET ADDRESS TO 50
1348 026776 022702 000074          8$: CMP      #74,R2   :ALL REGISTERS CHECKED ?
1349 027002 101402          BLOS   10$         :BRANCH IF SO
1350 027004 000667          BR      2$          :NEXT REGISTER
1351
1352 027006 022626          9$: CMP      (SP)+,(SP)+
1353 027010          10$:
1354 027010 012637 000006          MOV      (SP)+,ERRVEC+2  ;;POP STACK INTO ERRVEC+2
1355 027014 012637 000004          MOV      (SP)+,ERRVEC   ;;POP STACK INTO ERRVEC
1356

```

 :*TEST 41 SEEK FIRST CYLINDER

```

027020          TST41:
027020 000004          SCOPE          :SCOPE CALL
027022 000240          NOP           :START OF TEST
027024 012706 001100          MOV      #STACK,SP  :INITIALIZE STACK POINTER
027030 013700 001276          MOV      $BASE,R0   :R0 = UNIBUS ADDRESS
027034 013701 001464          MOV      TSTQUE,R1  :(R1) = DEVICE BEING TESTED
027040 012737 000001 001206          MOV      #1,$TIMES  ;;DO 1 ITERATION
027046 012737 000041 001226          MOV      #41,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
1357
1358 027054          1$:
1359 027054 004737 041650          JSR      PC,TSTPRP  :PREPARE DEVICE FOR TEST
      027060 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

```

	027062	000404			BR	2\$:GO TO 2\$ IF NO ERROR
	027064	000240			NOP			:RETURN HERE IF ERROR
	027066	104000			EMT			:ERROR # DEFINED BY TSTPRP SUBROUTINE
	027070	000137	027306		JMP	8\$:GO TO 8\$ IF ERROR
1360	027074						2\$:	
1361	027074	012737	000000	001444	MOV	#0, RMDCO		:CYLINDER = 0
1362	027102	012737	000000	001416	MOV	#0, RMDAO		:TRACK = 0, SECTOR = 0
1363	027110	012737	000005	001410	MOV	#SEEK!GO, RMCS10		:LOAD SEEK COMMAND IN BUFFER
1364	027116	012702	001543		MOV	#PUTINX, R2		:R2 POINTS TO REGISTER TABLE
1365	027122	112722	000034		MOVB	#RMDC, (R2)+		:WRITE REGISTER INDEX TABLE
1366	027126	112722	000006		MOVB	#RMDA, (R2)+		
1367	027132	112722	000000		MOVB	#RMCS1, (R2)+		
1368	027136	112722	000200		MOVB	#200, (R2)+		:WRITE TERMINATOR
1369	027142	004737	043160		JSR	PC, PUT		:GO WRITE REGISTER(S) WITH PUT SUBROUTINE
	027146	000404			BR	3\$:GO TO 3\$ IF NO ERROR
	027150	000240			NOP			:RETURN HERE IF ERROR
	027152	104000			EMT			:ERROR # DEFINED BY PUT SUBROUTINE
	027154	000137	027306		JMP	8\$:GO TO 8\$ IF ERROR
1370	027160	004737	042624		JSR	PC, GETSTS		:SETUP FOR STATUS FETCH
1371	027164	004737	043522		JSR	PC, TIMEOUT		:WAIT FOR SEEK TO COMPLETE
1372								
1373	027170	004737	042710		JSR	PC, GET		:GO READ REGISTER(S) WITH GET SUBROUTINE
	027174	000404			BR	4\$:GO TO 4\$ IF NO ERROR
	027176	000240			NOP			:RETURN HERE IF ERROR
	027200	104000			EMT			:ERROR # DEFINED BY GET SUBROUTINE
	027202	000137	027306		JMP	8\$:GO TO 8\$ IF ERROR
1374	027206						4\$:	
1375	027206	004737	043706		JSR	PC, PRIERR		:GO CHECK FOR PRIMARY ERRORS
	027212	000405			BR	5\$:GO TO 5\$ IF NO ERROR
	027214	000240			NOP			:RETURN HERE IF ERROR
	027216	104000			EMT			:ERROR # DEFINED BY PRIERR SUBROUTINE
	027220	004736			JSR	PC, @ (SP)+		:GO BACK FOR MORE ERROR CHECKS
	027222	000137	027306		JMP	8\$:GO TO 8\$ IF ERROR
1376	027226						5\$:	
1377	027226	004737	051024		JSR	PC, SEKSTS		:GO VERIFY RESULTS OF SEEK OPERATION
	027232	000405			BR	6\$:GO TO 6\$ IF NO ERROR
	027234	000240			NOP			:RETURN HERE IF ERROR
	027236	104000			EMT			:ERROR # DEFINED BY SEKSTS SUBROUTINE
	027240	004736			JSR	PC, @ (SP)+		:GO BACK FOR MORE ERROR CHECKS
	027242	000137	027306		JMP	8\$:GO TO 8\$ IF ERROR
1378	027246						6\$:	
1379	027246	004737	050400		JSR	PC, CMPERRSTS		:CHECK ANY ERRORS NOT MASKED
	027252	115760			.WORD	NDTMSK		:MASK FOR RMER1
	027254	000010			.WORD	DPE		:MASK FOR RMER2
	027256	000405			BR	7\$:GO TO 7\$ IF NO ERROR
	027260	000240			NOP			:RETURN HERE IF ERROR
	027262	104000			EMT			:ERROR # DEFINED BY CMPERRSTS SUBROUTINE
	027264	004736			JSR	PC, @ (SP)+		:GO BACK FOR MORE ERROR CHECKS
	027266	000137	027306		JMP	8\$:GO TO 8\$ IF ERROR
1380	027272						7\$:	
1381	027272	004737	044540		JSR	PC, SECERR		:GO CHECK FOR SECONDARY ERRORS
	027276	000403			BR	8\$:GO TO 8\$ IF NO ERROR
	027300	000240			NOP			:RETURN HERE IF ERROR
	027302	104000			EMT			:ERROR # DEFINED BY SECERR SUBROUTINE
	027304	004736			JSR	PC, @ (SP)+		:GO BACK FOR MORE ERROR CHECKS
1382	027306						8\$:	
1383								

1384

```
*****  
*TEST 42      SEEK LAST CYLINDER  
*****  
TST42:
```

027306	000004			SCOPE		:SCOPE CALL
027310	000240			NOP		:START OF TEST
027312	012706	001100		MOV	#STACK,SP	:INITIALIZE STACK POINTER
027316	013700	001276		MOV	\$BASE,R0	:R0 = UNIBUS ADDRESS
027322	013701	001464		MOV	TSTQUE,R1	:(R1) = DEVICE BEING TESTED
027326	012737	000001	001206	MOV	#1,\$TIMES	:DO 1 ITERATION
027334	012737	000042	001226	MOV	#42,\$TESTN	:SET TEST NUMBER IN APT MAIL BOX
1385						
1386	027342			1\$:		
1387	027342	004737	041650	JSR	PC,TSTPRP	:PREPARE DEVICE FOR TEST
	027346	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 8\$ IF ERROR
	027350	000404		BR	2\$	
	027352	000240		NOP		
	027354	104000		EMT		
	027356	000137	027574	JMP	8\$	
1388	027362			2\$:		
1389	027362	012737	001466	MOV	#822.,RMDCO	:LAST CYLINDER
1390	027370	012737	000000	MOV	#0,RMDAO	:TRACK = 0, SECTOR = 0
1391	027376	012737	000005	MOV	#SEEK!GO,RMCS10	:LOAD SEEK COMMAND IN BUFFER
1392	027404	012702	001543	MOV	#PUTINX,R2	:R2 POINTS TO REGISTER TABLE
1393	027410	112722	000034	MOV	#RMDC,(R2)+	:WRITE REGISTER INDEX TABLE
1394	027414	112722	000006	MOV	#RMDA,(R2)+	
1395	027420	112722	000000	MOV	#RMCS1,(R2)+	
1396	027424	112722	000200	MOV	#200,(R2)+	:WRITE TERMINATOR
1397	027430	004737	043160	JSR	PC,PUT	:GO WRITE REGISTER(S) WITH PUT SUBROUTINE
	027434	000404		BR	3\$:GO TO 3\$ IF NO ERROR
	027436	000240		NOP		:RETURN HERE IF ERROR
	027440	104000		EMT		:ERROR # DEFINED BY PUT SUBROUTINE
	027442	000137	027574	JMP	8\$:GO TO 8\$ IF ERROR
1398	027446	004737	042624	3\$:		
1399	027452	004737	043522	JSR	PC,GETSTS	:SETUP FOR STATUS FETCH
1400				JSR	PC,TIMOUT	:WAIT FOR SEEK TO COMPLETE
1401	027456	004737	042710	JSR	PC,GET	:GO READ REGISTER(S) WITH GET SUBROUTINE
	027462	000404		BR	4\$:GO TO 4\$ IF NO ERROR
	027464	000240		NOP		:RETURN HERE IF ERROR
	027466	104000		EMT		:ERROR # DEFINED BY GET SUBROUTINE
	027470	000137	027574	JMP	8\$:GO TO 8\$ IF ERROR
1402	027474			4\$:		
1403	027474	004737	043706	JSR	PC,PRIERR	:GO CHECK FOR PRIMARY ERRORS
	027500	000405		BR	5\$:GO TO 5\$ IF NO ERROR
	027502	000240		NOP		:RETURN HERE IF ERROR
	027504	104000		EMT		:ERROR # DEFINED BY PRIERR SUBROUTINE
	027506	004736		JSR	PC,@(SP)+	:GO BACK FOR MORE ERROR CHECKS
	027510	000137	027574	JMP	8\$:GO TO 8\$ IF ERROR
1404	027514			5\$:		
1405	027514	004737	051024	JSR	PC,SEKSTS	:GO VERIFY RESULTS OF SEEK OPERATION
	027520	000405		BR	6\$:GO TO 6\$ IF NO ERROR

```

027522 000240      NOP      ;RETURN HERE IF ERROR
027524 104000      EMT      ;ERROR # DEFINED BY SEKSTS SUBROUTINE
027526 004736      JSR      PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
027530 000137 027574 JMP      8$      ;GO TO 8$ IF ERROR
1406 027534      6$:      JSR      PC,CMPERRSTS ;CHECK ANY ERRORS NOT MASKED
1407 027534 004737 050400 .WORD   NDTMSK      ;MASK FOR RMER1
027540 115760      .WORD   DPE        ;MASK FOR RMER2
027542 000010      BR       7$      ;GO TO 7$ IF NO ERROR
027544 000405      NOP      ;RETURN HERE IF ERROR
027546 000240      EMT      ;ERROR # DEFINED BY CMPERRSTS SUBROUTINE
027550 104000      JSR      PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
027552 004736      JMP      8$      ;GO TO 8$ IF ERROR
1408 027554 000137 027574      7$:      JSR      PC,SECERR   ;GO CHECK FOR SECONDARY ERRORS
1409 027560 004737 044540 BR       8$      ;GO TO 8$ IF NO ERROR
027564 000403      NOP      ;RETURN HERE IF ERROR
027566 000240      EMT      ;ERROR # DEFINED BY SECERR SUBROUTINE
027570 104000      JSR      PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
027572 004736      8$:
1410 027574
1411
1412

```

```

*****
*TEST 43      SEEK PRIME CYLINDERS
*****
TST43:

```

```

027574 000004      SCOPE     ;SCOPE CALL
027576 000240      NOP      ;START OF TEST
027600 012706 001100 MOV      #STACK,SP ;INITIALIZE STACK POINTER
027604 013700 001276 MOV      $BASE,R0  ;R0 = UNIBUS ADDRESS
027610 013701 001464 MOV      TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
027614 012737 000002 001206 MOV      #2,$TIMES ;DO 2 ITERATIONS
027622 012737 000043 001226 MOV      #43,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
1413 027630 012737 000001 001444 1$:      MOV      #1,RMDCO   ;FIRST CYLINDER = 1
1414 027636
1415 027636
1416 027636 004737 041650 JSR      PC,TSTPRP ;PREPARE DEVICE FOR TEST
027642 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
027644 000404 BR       2$      ;GO TO 2$ IF NO ERROR
027646 000240 NOP      ;RETURN HERE IF ERROR
027650 104000 EMT      ;ERROR # DEFINED BY TSTPRP SUBROUTINE
027652 000137 030106 JMP      9$      ;GO TO 9$ IF ERROR
1417 027656      2$:      MOV      #0,RMDAO   ;TRACK = 0, SECTOR = 0
1418 027656 012737 000000 001416 MOV      #SEEK!GO,RMCS10 ;LOAD SEEK COMMAND IN BUFFER
1419 027664 012737 000005 001410 MOV      #PUTINX,R2 ;R2 POINTS TO REGISTER TABLE
1420 027672 012702 001543 MOV      #RMDC,(R2)+ ;WRITE REGISTER INDEX TABLE
1421 027676 112722 000034 MOV      #RMDA,(R2)+
1422 027702 112722 000006 MOV      #RMCS1,(R2)+
1423 027706 112722 000000 MOV      #200,(R2)+
1424 027712 112722 000200 ;WRITE TERMINATOR
1425 027716 004737 043160 JSR      PC,PUT    ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
027722 000404 BR       3$      ;GO TO 3$ IF NO ERROR

```



```

027724 000240      NOP
027726 104000      EMT
1426 027730 000137 030106      3$: JMP 9$ ;RETURN HERE IF ERROR
;ERROR # DEFINED BY PUT SUBROUTINE
;GO TO 9$ IF ERROR
1427 027734 004737 042624      JSR PC,GETSTS ;SETUP FOR STATUS FETCH
1428 027740 004737 043522      JSR PC,TIMOUT ;WAIT FOR SEEK TO COMPLETE
1429
1430 027744 004737 042710      JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
027750 000404      BR 4$ ;GO TO 4$ IF NO ERROR
027752 000240      NOP ;RETURN HERE IF ERROR
027754 104000      EMT ;ERROR # DEFINED BY GET SUBROUTINE
027756 000137 030106      JMP 9$ ;GO TO 9$ IF ERROR
1431 027762      4$:
1432 027762 004737 043706      JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
027766 000405      BR 5$ ;GO TO 5$ IF NO ERROR
027770 000240      NOP ;RETURN HERE IF ERROR
027772 104000      EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
027774 004736      JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
027776 000137 030106      JMP 9$ ;GO TO 9$ IF ERROR
1433 030002      5$:
1434 030002 004737 051024      JSR PC,SEKSTS ;GO VERIFY RESULTS OF SEEK OPERATION
030006 000405      BR 6$ ;GO TO 6$ IF NO ERROR
030010 000240      NOP ;RETURN HERE IF ERROR
030012 104000      EMT ;ERROR # DEFINED BY SEKSTS SUBROUTINE
030014 004736      JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
030016 000137 030106      JMP 9$ ;GO TO 9$ IF ERROR
1435 030022      6$:
1436 030022 004737 050400      JSR PC,CMPERRSTS ;CHECK ANY ERRORS NOT MASKED
030026 115760      .WORD NDTMSK ;MASK FOR RMER1
030030 000010      .WORD DPE ;MASK FOR RMER2
030032 000405      BR 7$ ;GO TO 7$ IF NO ERROR
030034 000240      NOP ;RETURN HERE IF ERROR
030036 104000      EMT ;ERROR # DEFINED BY CMPERRSTS SUBROUTINE
030040 004736      JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
030042 000137 030106      JMP 9$ ;GO TO 9$ IF ERROR
1437 030046      7$:
1438 030046 004737 044540      JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
030052 000405      BR 8$ ;GO TO 8$ IF NO ERROR
030054 000240      NOP ;RETURN HERE IF ERROR
030056 104000      EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
030060 004736      JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
030062 000137 030106      JMP 9$ ;GO TO 9$ IF ERROR
1439 030066      8$:
1440 030066 006337 001444      ASL RMDCO ;SHIFT TO NEXT PRIME CYLINDER
1441 030072 022737 001000 001444      CMP #512.,RMDCO ;IS THE TEST DONE??
1442 030100 103402      BLO 9$ ;YES!!
1443 030102 000137 027636      JMP 1$ ;GO DO NEXT CYLINDER
1444 030106      9$:
1445
1446

```

```

*****
;*TEST 44      SEEK ZERO DIFFERENCE
*****

```

```

TST44:
030106      SCOPE ;SCOPE CALL
030106 000004      NOP ;START OF TEST
030110 000240      MOV #STACK,SP ;INITIALIZE STACK POINTER
030112 012706 001100      MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
030116 013700 001276

```

```

030122 013701 001464      MOV    TSTQUE,R1      ;(R1) = DEVICE BEING TESTED
030126 012737 000044 001226  MOV    #44,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX

1447
1448 030134      1$:
1449 030134 004737 041650      JSR    PC,TSTFRP    ;PREPARE DEVICE FOR TEST
030140 054130      .WORD 054130      ;TASK DESCRIPTOR AS FOLLOWS:
                                           ;CLEAR CONTROLLER & SELECT DEVICE
                                           ;VERIFY CONTROLLER CLEAR OPERATION
                                           ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                           ;VERIFY PACK ACKNOWLEDGE
                                           ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
                                           ;VERIFY RECALIBRATION
                                           ;GO TO 2$ IF NO ERROR
                                           ;RETURN HERE IF ERROR
                                           ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                           ;GO TO 10$ IF ERROR

030142 000404      BR     2$
030144 000240      NOP
030146 104000      EMT
030150 000137 030406      JMP    10$

1450 030154      2$:
1451 030154 012703 000003      MOV    #3,R3        ;R3 = NUMBER OF SEEKS
1452 030160 012737 000000 001444      MOV    #0,RMDCO     ;CYLINDER = 0
1453 030166 012737 000000 001416      MOV    #0,RMDAO     ;TRACK = 0, SECTOR = 0
1454 030174 012737 000005 001410      MOV    #SEEK!GO,RMCS10 ;FUNCTION CODE FOR SEEK
1455 030202 012702 001543      MOV    #PUTINX,R2   ;R2 POINTS TO REGISTER INDEX
1456 030206 112722 000006      MOVB  #RMDA,(R2)+   ;WRITE REGISTER INDEX TABLE
1457 030212 112722 000034      MOVB  #RMDC,(R2)+
1458 030216 112722 000000      MOVB  #RMCS1,(R2)+
1459 030222 112722 000200      MOVB  #200,(R2)+   ;TERMINATE TABLE

1460 030226      3$:
1461 030226 004737 043160      JSR    PC,PUT       ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
030232 000404      BR     4$
030234 000240      NOP
030236 104000      EMT
030240 000137 030406      JMP    10$
1462 030244 004737 042624      4$: JSR    PC,GETSTS    ;SETUP FOR READING STATUS
1463 030250 004737 043522      JSR    PC,TIMOUT   ;WAIT FOR COMPLETION
1464
1465 030254 004737 042710      JSR    PC,GET       ;GO READ REGISTER(S) WITH GET SUBROUTINE
030260 000404      BR     5$
030262 000240      NOP
030264 104000      EMT
030266 000137 030406      JMP    10$
1466 030272      5$:
1467 030272 004737 043706      JSR    PC,PRIERR   ;GO CHECK FOR PRIMARY ERRORS
030276 000405      BR     6$
030300 000240      NOP
030302 104000      EMT
030304 004736      JSR    PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
030306 000137 030406      JMP    10$
1468 030312      6$:
1469 030312 004737 051024      JSR    PC,SEKSTS   ;GO VERIFY RESULTS OF SEEK OPERATION
030316 000405      BR     7$
030320 000240      NOP
030322 104000      EMT
030324 004736      JSR    PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
030326 000137 030406      JMP    10$
1470 030332      7$:
1471 030332 004737 050400      JSR    PC,CMPERRSTS ;CHECK ANY ERRORS NOT MASKED
030336 115760      .WORD NDTMSK      ;MASK FOR RMER1
  
```

```

030340 000010 .WORD DPE ;MASK FOR RMER2
030342 000405 BR 8$ ;GO TO 8$ IF NO ERROR
030344 000240 NOP ;RETURN HERE IF ERROR
030346 104000 EMT ;ERROR # DEFINED BY CMPERRSTS SUBROUTINE
030350 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
030352 00C137 030406 JMP 10$ ;GO TO 10$ IF ERROR
1472 030356 8$:
1473 030356 004737 044540 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
030362 000405 BR 9$ ;GO TO 9$ IF NO ERROR
030364 000240 NOP ;RETURN HERE IF ERROR
030366 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
030370 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
030372 000137 030406 JMP 10$ ;GO TO 10$ IF ERROR
1474 030376 9$:
1475 030376 005303 DEC R3 ;DONE ALL SEEKS??
1476 030400 001402 BEQ 10$ ;YES!!
1477 030402 000137 030226 JMP 3$ ;NO - GO DO NEXT SEEK
1478 030406 10$:
1479
1480

```

 *TEST 45 SEEK MAXIMUM DIFFERENCE FORWARD

 TST45:

```

030406 000004 SCOPE ;SCOPE CALL
030410 000240 NOP ;START OF TEST
030412 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
030416 013700 001276 MOV $BASE,R0 ;RC = UNIBUS ADDRESS
030422 013701 001464 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
030426 012737 000045 001226 MOV #45,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
1481
1482 030434 1$:
1483 030434 004737 041650 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
030440 052130 .WORD 052130 ;TASK DESCRIPTOR AS FOLLOWS:
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;VERIFY RECALIBRATION
030442 000404 BR 2$ ;GO TO 2$ IF NO ERROR
030444 000240 NOP ;RETURN HERE IF ERROR
030446 104000 EMT ;ERROR # DEFINED BY TSTPRP SUBROUTINE
030450 000137 030666 JMP 8$ ;GO TO 8$ IF ERROR
1484 030454 2$:
1485 030454 012737 001466 001444 MOV #822.,RMDCO ;LAST CYLINDER
1486 030462 012737 000000 001416 MOV #0,RMDAO ;TRACK = 0, SECTOR = 0
1487 030470 012737 000005 001410 MOV #SEEK!GO,RMCS10 ;LOAD SEEK COMMAND IN BUFFER
1488 030476 012702 001543 MOV #PUTINX,R2 ;R2 POINTS TO REGISTER TABLE
1489 030502 112722 000034 MOVB #RMDC,(R2)+ ;WRITE REGISTER INDEX TABLE
1490 030506 112722 000006 MOVB #RMDA,(R2)+
1491 030512 112722 000000 MOVB #RMCS1,(R2)+
1492 030516 112722 000200 MOVB #200,(R2)+
1493 030522 004737 043160 JSR PC,PUT ;WRITE TERMINATOR
030526 000404 BR 3$ ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
030530 000240 NOP ;GO TO 3$ IF NO ERROR
030532 104000 EMT ;RETURN HERE IF ERROR
030534 000137 030666 JMP 8$ ;ERROR # DEFINED BY PUT SUBROUTINE
1494 030540 004737 042624 3$: JSR PC,GETSTS ;GO TO 8$ IF ERROR
;SETUP FOR STATUS FETCH

```

```

1495 030544 004737 043522      JSR    PC,TIMOUT      ;WAIT FOR SEEK TO COMPLETE
1496
1497 030550 004737 042710      JSR    PC,GET        ;GO READ REGISTER(S) WITH GET SUBROUTINE
      030554 000404      BR     4$            ;GO TO 4$ IF NO ERROR
      030556 000240      NOP                    ;RETURN HERE IF ERROR
      030560 104000      EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      030562 000137 030666      JMP    8$            ;GO TO 8$ IF ERROR
1498 030566
1499 030566 004737 043706      4$: JSR    PC,PRIERR     ;GO CHECK FOR PRIMARY ERRORS
      030572 000405      BR     5$            ;GO TO 5$ IF NO ERROR
      030574 000240      NOP                    ;RETURN HERE IF ERROR
      030576 104000      EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      030600 004736      JSR    PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      030602 000137 030666      JMP    8$            ;GO TO 8$ IF ERROR
1500 030606
1501 030606 004737 051024      5$: JSR    PC,SEKSTS    ;GO VERIFY RESULTS OF SEEK OPERATION
      030612 000405      BR     6$            ;GO TO 6$ IF NO ERROR
      030614 000240      NOP                    ;RETURN HERE IF ERROR
      030616 104000      EMT                    ;ERROR # DEFINED BY SEKSTS SUBROUTINE
      030620 004736      JSR    PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      030622 000137 030666      JMP    8$            ;GO TO 8$ IF ERROR
1502 030626
1503 030626 004737 050400      6$: JSR    PC,CMPERRSTS ;CHECK ANY ERRORS NOT MASKED
      030632 115760      .WORD NDITMSK        ;MASK FOR RMER1
      030634 000010      .WORD DPE           ;MASK FOR RMER2
      030636 000405      BR     7$            ;GO TO 7$ IF NO ERROR
      030640 000240      NOP                    ;RETURN HERE IF ERROR
      030642 104000      EMT                    ;ERROR # DEFINED BY CMPERRSTS SUBROUTINE
      030644 004736      JSR    PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      030646 000137 030666      JMP    8$            ;GO TO 8$ IF ERROR
1504 030652
1505 030652 004737 044540      7$: JSR    PC,SECERR    ;GO CHECK FOR SECONDARY ERRORS
      030656 000403      BR     8$            ;GO TO 8$ IF NO ERROR
      030660 000240      NOP                    ;RETURN HERE IF ERROR
      030662 104000      EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      030664 004736      JSR    PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
1506 030666
1507
1508

```

```

*****
;*TEST 46      SEEK ADJACENT FORWARD
*****
TST46:

```

```

030666
030666 000004      SCOPE                ;SCOPE CALL
030670 000240      NOP                  ;START OF TEST
030672 012706 001100      MOV    #STACK,SP    ;INITIALIZE STACK POINTER
030676 013700 001276      MOV    $BASE,R0     ;R0 = UNIBUS ADDRESS
030702 013701 001464      MOV    TSTQUE,R1    ;(R1) = DEVICE BEING TESTED
030706 012737 000046 001226      MOV    #46,$TESTN   ;SET TEST NUMBER IN APT MAIL BOX
1509
1510 030714
1511 030714 004737 041650      1$: JSR    PC,TSTPRP    ;PREPARE DEVICE FOR TEST
      030720 054130      .WORD 054130        ;TASK DESCRIPTOR /S 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

```

```

030722 000404 BR 2$ ;VERIFY RECALIBRATION
030724 000240 NOP ;GO TO 2$ IF NO ERROR
030726 104000 EMT ;RETURN HERE IF ERROR
030730 000137 031170 JMP 10$ ;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 10$ IF ERROR
1512 030734 2$:
1513 030734 012737 000000 001444 MOV #0,RMDCO ;CYLINDER = 0
1514 030742 012737 000000 001416 MOV #0,RMDAO ;TRACK = 0, SECTOR = 0
1515 030750 012737 000005 001410 MOV #SEEK!GO,RMCS10 ;FUNCTION CODE FOR SEEK
1516 030756 012702 001543 MOV #PJTINX,R2 ;R2 POINTS TO REGISTER INDEX
1517 030762 112722 000034 MOVB #RMDC,(R2)+ ;WRITE REGISTER INDEX TABLE
1518 030766 112722 000006 MOVB #RMDA,(R2)+
1519 030772 112722 000000 MOVB #RMCS1,(R2)+
1520 030776 112722 000200 MOVB #200,(R2)+ ;TERMINATE REGISTER TABLE
1521 031002 3$:
1522 031002 004737 043160 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
031006 000404 BR 4$ ;GO TO 4$ IF NO ERROR
031010 000240 NOP ;RETURN HERE IF ERROR
031012 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
031014 000137 031170 JMP 10$ ;GO TO 10$ IF ERROR
1523 031020 004737 042624 4$: JSR PC,GETSTS ;SETUP FOR STATUS FETCH
1524 031024 004737 043522 JSR PC,TIMOUT ;WAIT FOR COMPLETION
1525
1526 031030 004737 042710 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
031034 000404 BR 5$ ;GO TO 5$ IF NO ERROR
031036 000240 NOP ;RETURN HERE IF ERROR
031040 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
031042 000137 031170 JMP 10$ ;GO TO 10$ IF ERROR
1527 031046 5$:
1528 031046 004737 043706 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
031052 000405 BR 6$ ;GO TO 6$ IF NO ERROR
031054 000240 NOP ;RETURN HERE IF ERROR
031056 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
031060 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
031062 000137 031170 JMP 10$ ;GO TO 10$ IF ERROR
1529 031066 6$:
1530 031066 004737 051024 JSR PC,SEKSTS ;GO VERIFY RESULTS OF SEEK OPERATION
031072 000405 BR 7$ ;GO TO 7$ IF NO ERROR
031074 000240 NOP ;RETURN HERE IF ERROR
031076 104000 EMT ;ERROR # DEFINED BY SEKSTS SUBROUTINE
031100 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
031102 000137 031170 JMP 10$ ;GO TO 10$ IF ERROR
1531 031106 7$:
1532 031106 004737 050400 JSR PC,CMPERRSTS ;CHECK ANY ERRORS NOT MASKED
031112 115760 .WORD ND1MSK ;MASK FOR RMER1
031114 000010 .WORD DPE ;MASK FOR RMER2
031116 000405 BR 8$ ;GO TO 8$ IF NO ERROR
031120 000240 NOP ;RETURN HERE IF ERROR
031122 104000 EMT ;ERROR # DEFINED BY CMPERRSTS SUBROUTINE
031124 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
031126 000137 031170 JMP 10$ ;GO TO 10$ IF ERROR
1533 031132 8$:
1534 031132 004737 044540 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
031136 000405 BR 9$ ;GO TO 9$ IF NO ERROR
031140 000240 NOP ;RETURN HERE IF ERROR
031142 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
031144 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS

```

```

1535 031146 000137 031170          9$:  JMP      10$          ;GO TO 10$ IF ERROR
1536 031152                                TST      RMDCO          ;FIRST TIME THROUGH??
1537 031152 005737 001444          BNE      10$          ;NO!!
1538 031160 005237 001444          INC      RMDCO          ;YES - SEEK TO ADJACENT CYLINDER FORWARD
1539 031164 000137 031002          JMP      3$
1540 031170                                10$:
1541
1542
  
```

```

;*****
;*TEST 47      SEEK ADJACENT REVERSE
;*****
TST47:
  
```

```

031170
031170 000004          SCOPE          ;SCOPE CALL
031172 000240          NOP           ;START OF TEST
031174 012706 001100  MOV      #STACK,SP ;INITIALIZE STACK POINTER
031200 013700 001276  MOV      $BASE,R0  ;R0 = UNIBUS ADDRESS
031204 013701 001464  MOV      TSTQUE,R1 ; (R1) = DEVICE BEING TESTED
031210 012737 000047 001226  MOV      #47,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
1543
1544 031216                                1$:
1545 031216 004737 041650  JSR      PC,TSTPRP  ;PREPARE DEVICE FOR TEST
031222 054130          .WORD     054130  ;TASK DESCRIPTOR AS FOLLOWS:
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
;VERIFY RECALIBRATION
031224 000404          BR       2$          ;GO TO 2$ IF NO ERROR
031226 000240          NOP           ;RETURN HERE IF ERROR
031230 104000          EMT          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
031232 000137 031474  JMP      10$          ;GO TO 10$ IF ERROR
1546 031236                                2$:
1547 031236 012737 000001 001444  MOV      #1,RMDCO   ;CYLINDER = 1
1548 031244 012737 000000 001416  MOV      #0,RMDAO   ;TRACK = 0, SECTOR = 0
1549 031252 012737 000005 001410  MOV      #SEEK!GO,RMCS10 ;FUNCTION CODE FOR SEEK
1550 031260 012702 001543          MOV      #PUTINX,R2 ;R2 POINTS TO REGISTER INDEX
1551 031264 112722 000006          MOVB    #RMDA,(R2)+ ;WRITE REGISTER INDEX TABLE
1552 031270 112722 000034          MOVB    #RMDC,(R2)+
1553 031274 112722 000000          MOVB    #RMCS1,(R2)+
1554 031300 112722 000200          MOVB    #200,(R2)+ ;TERMINATE TABLE
1555 031304                                3$:
1556 031304 004737 043160  JSR      PC,PUT     ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
031310 000404          BR       4$          ;GO TO 4$ IF NO ERROR
031312 000240          NOP           ;RETURN HERE IF ERROR
031314 104000          EMT          ;ERROR # DEFINED BY PUT SUBROUTINE
031316 000137 031474  JMP      10$          ;GO TO 10$ IF ERROR
1557 031322 004737 042624  4$:  JSR      PC,GETSTS  ;SETUP TO READ ALL REGISTERS
1558 031326 004737 043522  JSR      PC,TIMOUT  ;WAIT FOR SEEK TO COMPLETE
1559
1560 031332 004737 042710  JSR      PC,GET     ;GO READ REGISTER(S) WITH GET SUBROUTINE
031336 000404          BR       5$          ;GO TO 5$ IF NO ERROR
031340 000240          NOP           ;RETURN HERE IF ERROR
031342 104000          EMT          ;ERROR # DEFINED BY GET SUBROUTINE
031344 000137 031474  JMP      10$          ;GO TO 10$ IF ERROR
1561 031350                                5$:
1562 031350 004737 043706  JSR      PC,PRIERR  ;GO CHECK FOR PRIMARY ERRORS
  
```

```

031354 000405 BR 6$ ;GO TO 6$ IF NO ERROR
031356 000240 NOP ;RETURN HERE IF ERROR
031360 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
031362 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
031364 000137 031474 JMP 10$ ;GO TO 10$ IF ERROR
1563 031370 6$: JSR PC,SEKSTS ;GO VERIFY RESULTS OF SEEK OPERATION
1564 031370 004737 051024 BR 7$ ;GO TO 7$ IF NO ERROR
031374 000405 NOP ;RETURN HERE IF ERROR
031376 000240 EMT ;ERROR # DEFINED BY SEKSTS SUBROUTINE
031400 104000 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
031402 004736 JMP 10$ ;GO TO 10$ IF ERROR
031404 000137 031474
1565 031410 7$: JSR PC,CMPERRSTS ;CHECK ANY ERRORS NOT MASKED
1566 031410 004737 050400 /WORD NDTMSK ;MASK FOR RMER1
031414 115760 .WORD DPE ;MASK FOR RMER2
031416 000010 BR 8$ ;GO TO 8$ IF NO ERROR
031420 000405 NOP ;RETURN HERE IF ERROR
031422 000240 EMT ;ERROR # DEFINED BY CMPERRSTS SUBROUTINE
031424 104000 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
031426 004736 JMP 10$ ;GO TO 10$ IF ERROR
031430 000137 031474
1567 031434 8$: JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
1568 031434 004737 044540 BR 9$ ;GO TO 9$ IF NO ERROR
031440 000405 NOP ;RETURN HERE IF ERROR
031442 000240 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
031444 104000 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
031446 004736 JMP 10$ ;GO TO 10$ IF ERROR
031450 000137 031474
1569 031454 9$: CMP #1,RMDCO ;IS THIS THE FIRST SEEK??
1570 031454 022737 000001 001444 BNE 10$ ;NO!!
1571 031462 001004 DEC RMDCO ;YES - SEEK TO ADJACENT CYLINDER
1572 031464 005337 001444 JMP 3$ ;GO SEEK
1573 031470 000137 031304
1574 031474 10$:
1575
1576

```

```

*****
;*TEST 50 SEEK INVALID SECTOR
*****
TST50:

```

```

031474 000004 SCOPE ;SCOPE CALL
031474 000240 NOP ;START OF TEST
031500 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
031504 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
031510 013701 001464 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
031514 012737 000050 001226 MOV #50,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
1577
1578 031522 1$: JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
1579 031522 004737 041650 .WORD 054130 ;TASK DESCRIPTOR AS FOLLOWS:
031526 054130 ;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
;VERIFY RECALIBRATION
031530 000404 BR 2$ ;GO TO 2$ IF NO ERROR
031532 000240 NOP ;RETURN HERE IF ERROR

```



```

031770 000240      NOP      ;RETURN HERE IF ERROR
031772 104000      EMT      ;ERROR # DEFINED BY SEKSTS SUBROUTINE
031774 004736      JSR      PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
031776 000137 032112 JMP      14$      ;GO TO 14$ IF ERROR
1608 032002      10$:
1609 032002      11$:
1610 032002 004737 050400 JSR      PC, CMPERRSTS ;CHECK ANY ERRORS NOT MASKED
      032006 117760      .WORD      ;MASK FOR RMER1
      032010 000010      .WORD      ;MASK FOR RMER2
      032012 000405      BR        12$      ;GO TO 12$ IF NO ERROR
      032014 000240      NOP      ;RETURN HERE IF ERROR
      032016 104000      EMT      ;ERROR # DEFINED BY CMPERRSTS SUBROUTINE
      032020 004736      JSR      PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      032022 000137 032112 JMP      14$      ;GO TO 14$ IF ERROR
1611 032026      12$:
1612 032026 004737 044540 JSR      PC, SECERR   ;GO CHECK FOR SECONDARY ERRORS
      032032 000405      BR        13$      ;GO TO 13$ IF NO ERROR
      032034 000240      NOP      ;RETURN HERE IF ERROR
      032036 104000      EMT      ;ERROR # DEFINED BY SECERR SUBROUTINE
      032040 004736      JSR      PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      032042 000137 032112 JMP      14$      ;GO TO 14$ IF ERROR
1613 032046      13$:
1614 032046 005237 001416 INC      RMDAO      ;INCREMENT SECTOR ADDRESS
1615 032052 023727 001416 000100 CMP      RMDAO,#64. ;DONE YET ?
1616 032060 101662      BLOS     3$        ;NO - TEST NEXT SECTOR
1617
1618 032062 032737 010000 001442 BIT      #FMT16,RMOFO ;WAS TEST DONE FOR 16 BIT MODE YET ?
1619 032070 001010      BNE     14$      ;BR IF YES
1620 032072 012737 010000 001442 MOV      #FMT16,RMOFO ;SET 16 BIT FORMAT MODE
1621 032100 012737 000040 001416 MOV      #32.,RMDAO  ;SET INVALID SECTOR 32.
1622 032106 000137 031626      JMP     3$        ;NO - TEST NEXT SECTOR
1623 032112      14$:
1624
1625

```

```

*****
*TEST 51      SEEK INVALID TRACK
*****
TST51:

```

```

032112
032112 000004      SCOPE     ;SCOPE CALL
032114 000240      NOP      ;START OF TEST
032116 012706 001100 MOV      #STACK,SP  ;INITIALIZE STACK POINTER
032122 013700 001276 MOV      $BASE,R0   ;R0 = UNIBUS ADDRESS
032126 013701 001464 MOV      TSTQUE,R1  ;(R1) = DEVICE BEING TESTED
032132 012737 000051 001226 MOV      #51,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
1626
1627 032140 004737 041650 JSR      PC, TSTPRP  ;PREPARE DEVICE FOR TEST
      032144 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
      032146 000404      BR        1$        ;GO TO 1$ IF NO ERROR
      032150 000240      NOP      ;RETURN HERE IF ERROR
      032152 104000      EMT      ;ERROR # DEFINED BY TSTPRP SUBROUTINE
      032154 000137 032540 JMP      14$      ;GO TO 14$ IF ERROR
1628 032160      1$:

```

1629	032160	012737	000005	001410	MOV	#SEEK!GO,RMCS10	:SEEK COMMAND
1630	032166	012737	000000	001444	MOV	#0,RMDCO	:CYLINDER = 0
1631	032174	013737	001332	001416	MOV	LSTRK,RMDAO	:LAST TRACK, SECTOR = 0
1632	032202	105237	001417		INCB	RMDAO+1	:SETUP FIRST INVALID TRACK
1633	032206	012737	000000	001442	MOV	#0,RMOFO	:18 BIT FORMAT
1634	032214	012702	001543		MOV	#PUTINX,R2	:R2 POINTS TO REGISTER TABLE
1635	032220	112722	000032		MOVVB	#RMOF,(R2)+	:WRITE REGISTER INDEX
1636	032224	112722	000034		MOVVB	#RMDG,(R2)+	:TABLE FOR OUTPUT
1637	032230	112722	000006		MOVVB	#RMDA,(R2)+	
1638	032234	112722	000000		MOVVB	#RMCS1,(R2)+	
1639	032240	112722	000200		MOVVB	#200,(R2)+	:TERMINATE TABLE
1640	032244	004737	042624		JSR	PC,GETSTS	:SETUP INPUT TABLE FOR STATUS
1641	032250			2\$:			
1642	032250	004737	052264		JSR	PC,CNTCLR	:GO ISSUE CONTROLLER CLEAR
	032254	000404			BR	3\$:GO TO 3\$ IF NO ERROR
	032256	000240			NOP		:RETURN HERE IF ERROR
	032260	104000			EMT		:ERROR NUMBER DEFINED BY SUBROUTINE
	032262	000137	032540		JMP	14\$:GO TO 14\$ IF ERROR
1643	032266			3\$:			
1644	032266	004737	042710		JSR	PC,GET	:GO READ REGISTER(S) WITH GET SUBROUTINE
	032272	000404			BR	4\$:GO TO 4\$ IF NO ERROR
	032274	000240			NOP		:RETURN HERE IF ERROR
	032276	104000			EMT		:ERROR # DEFINED BY GET SUBROUTINE
	032300	000137	032540		JMP	14\$:GO TO 14\$ IF ERROR
1645	032304			4\$:			
1646	032304	004737	052402		JSR	PC,CLRSTS	:GO VERIFY CONTROLLER CLEAR OPERATION
	032310	000405			BR	5\$:GO TO 5\$ IF NO ERROR
	032312	000240			NOP		:RETURN HERE IF ERROR
	032314	104000			EMT		:ERROR # DEFINED BY CLRSTS SUBROUTINE
	032316	004736			JSR	PC,@(SP)+	:GO BACK FOR MORE ERROR CHECKS
	032320	000137	032540		JMP	14\$:GO TO 14\$ IF ERROR
1647	032324			5\$:			
1648	032324	004737	043160		JSR	PC,PUT	:GO WRITE REGISTER(S) WITH PUT SUBROUTINE
	032330	000404			BR	6\$:GO TO 6\$ IF NO ERROR
	032332	000240			NOP		:RETURN HERE IF ERROR
	032334	104000			EMT		:ERROR # DEFINED BY PUT SUBROUTINE
	032336	000137	032540		JMP	14\$:GO TO 14\$ IF ERROR
1649	032342			6\$:			
1650	032342	004737	043522		JSR	PC,TIMOUT	:WAIT FOR GO TO RESET
1651							
1652	032346	004737	042710		JSR	PC,GET	:GO READ REGISTER(S) WITH GET SUBROUTINE
	032352	000404			BR	7\$:GO TO 7\$ IF NO ERROR
	032354	000240			NOP		:RETURN HERE IF ERROR
	032356	104000			EMT		:ERROR # DEFINED BY GET SUBROUTINE
	032360	000137	032540		JMP	14\$:GO TO 14\$ IF ERROR
1653	032364			7\$:			
1654	032364	004737	043706		JSR	PC,PRIERR	:GO CHECK FOR PRIMARY ERRORS
	032370	000405			BR	8\$:GO TO 8\$ IF NO ERROR
	032372	000240			NOP		:RETURN HERE IF ERROR
	032374	104000			EMT		:ERROR # DEFINED BY PRIERR SUBROUTINE
	032376	004736			JSR	PC,@(SP)+	:GO BACK FOR MORE ERROR CHECKS
	032400	000137	032540		JMP	14\$:GO TO 14\$ IF ERROR
1655	032404			8\$:			
1656	032404	004737	051024		JSR	PC,SEKSTS	:GO VERIFY RESULTS OF SEEK OPERATION
	032410	000405			BR	9\$:GO TO 9\$ IF NO ERROR
	032412	000240			NOP		:RETURN HERE IF ERROR
	032414	104000			EMT		:ERROR # DEFINED BY SEKSTS SUBROUTINE

```

151 032416 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      032420 000137 032540 JMP 14$ ;GO TO 14$ IF ERROR
1657 032424 9$:
1658 032424 10$:
1659 032424 004737 050400 JSR PC,CMPERRSTS ;CHECK ANY ERRORS NOT MASKED
      032430 117760 .WORD IAE!NDTMSK ;MASK FOR RMER1
      032432 000010 .WORD DPE ;MASK FOR RMER2
      032434 000405 BR 11$ ;GO TO 11$ IF NO ERROR
      032436 000240 NOP ;RETURN HERE IF ERROR
      032440 104000 EMT ;ERROR # DEFINED BY CMPERRSTS SUBROUTINE
      032442 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      032444 000137 032540 JMP 14$ ;GO TO 14$ IF ERROR
1660 032450 11$:
1661 032450 004737 044540 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
      032454 000405 BR 12$ ;GO TO 12$ IF NO ERROR
      032456 000240 NOP ;RETURN HERE IF ERROR
      032460 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
      032462 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      032464 000137 032540 JMP 14$ ;GO TO 14$ IF ERROR
1662 032470 12$:
1663 032470 105237 001417 INCB RMDAO+1 ;INCREMENT TRACK ADDRESS
1664 032474 123727 001417 000200 CMPB RMDAO+1,#128. ;DONE??
1665 032502 101414 BLOS 13$ ;NO-DO NEXT TRACK
1666
1667 032504 032737 010000 001442 BIT #FMT16,RMOFO ;WAS TEST DONE FOR 16 BIT??
1668 032512 001012 BNE 14$ ;YES!!
1669 032514 012737 010000 001442 MOV #FMT16,RMOFO ;SET FORMAT 16
1670 032522 013737 001332 001416 MOV LSTRK,RMDAO ;LAST TRACK, SECTOR = 0
1671 032530 105237 001417 INCB RMDAO+1 ;SETUP FIRST INVALID TRACK ADDRESS
1672 032534 000137 032250 JMP 2$ ;DO TEST FOR 16 BIT FORMAT
1673
1674 032540 14$:
1675
1676

```

 :*TEST 52 SEEK INVALID CYLINDER

```

TST52:
032540 SCOPE ;SCOPE CALL
032540 000004 NOP ;START OF TEST
032542 000240 MOV #STACK,SP ;INITIALIZE STACK POINTER
032544 012706 001100 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
032550 013700 001276 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
032554 013701 001464 MOV #52,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
032560 012737 000052 001226
1677
1678 032566 004737 041650 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
      032572 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
      032574 000404 BR 1$ ;GO TO 1$ IF NO ERROR
      032576 000240 NOP ;RETURN HERE IF ERROR
      032600 104000 EMT ;ERROR # DEFINED BY TSTPRP SUBROUTINE
      032602 000137 033156 JMP 13$ ;GO TO 13$ IF ERROR
1679 032606 1$:

```

1680	032606	012737	000005	001410	MOV	#SEEK!GO,RMCS10	:SEEK FUNCTION CODE
1681	032614	012737	001467	001444	MOV	#823.,RMDCO	:START AT FIRST INVALID CYLINDER
1682	032622	012737	000000	001416	MOV	#0,RMDAO	:TRACK = 0, SECTOR = 0
1683	032630	012737	000000	001442	MOV	#0,RMOFO	:18 BIT FORMAT
1684	032636	012702	001543		MOV	#PUTINX,R2	:R2 POINTS TO INDEX TABLE
1685	032642	112722	000006		MOVB	#RMDA,(R2)+	:WRITE REGISTER TABLE
1686	032646	112722	000034		MOVB	#RMDC,(R2)+	
1687	032652	112722	000032		MOVB	#RMOF,(R2)+	
1688	032656	112722	000000		MOVB	#RMCS1,(R2)+	
1689	032662	112722	000200		MOVB	#200,(R2)+	:TERMINATE TABLE
1690	032666	004737	042624		JSR	PC,GETSTS	:SETUP FOR STATUS
1691	032672			2\$:			
1692	032672	004737	052264		JSR	PC,CNTCLR	:GO ISSUE CONTROLLER CLEAR
	032676	000404			BR	3\$:GO TO 3\$ IF NO ERROR
	032700	000240			NOP		:RETURN HERE IF ERROR
	032702	104000			EMT		:ERROR NUMBER DEFINED BY SUBROUTINE
	032704	000137	033156		JMP	13\$:GO TO 13\$ IF ERROR
1693	032710			3\$:			
1694	032710	004737	042710		JSR	PC,GET	:GO READ REGISTER(S) WITH GET SUBROUTINE
	032714	000404			BR	4\$:GO TO 4\$ IF NO ERROR
	032716	000240			NOP		:RETURN HERE IF ERROR
	032720	104000			EMT		:ERROR # DEFINED BY GET SUBROUTINE
	032722	000137	033156		JMP	13\$:GO TO 13\$ IF ERROR
1695	032726			4\$:			
1696	032726	004737	052402		JSR	PC,CLRSTS	:GO VERIFY CONTROLLER CLEAR OPERATION
	032732	000405			BR	5\$:GO TO 5\$ IF NO ERROR
	032734	000240			NOP		:RETURN HERE IF ERROR
	032736	104000			EMT		:ERROR # DEFINED BY CLRSTS SUBROUTINE
	032740	004736			JSR	PC,@(SP)+	:GO BACK FOR MORE ERROR CHECKS
	032742	000137	033156		JMP	13\$:GO TO 13\$ IF ERROR
1697	032746			5\$:			
1698	032746	004737	043160		JSR	PC,PUT	:GO WRITE REGISTER(S) WITH PUT SUBROUTINE
	032752	000404			BR	6\$:GO TO 6\$ IF NO ERROR
	032754	000240			NOP		:RETURN HERE IF ERROR
	032756	104000			EMT		:ERROR # DEFINED BY PUT SUBROUTINE
	032760	000137	033156		JMP	13\$:GO TO 13\$ IF ERROR
1699	032764			6\$:			
1700	032764	004737	043522		JSR	PC,TIMOUT	:WAIT FOR GO TO RESET
1701							
1702	032770	004737	042710		JSR	PC,GET	:GO READ REGISTER(S) WITH GET SUBROUTINE
	032774	000404			BR	7\$:GO TO 7\$ IF NO ERROR
	032776	000240			NOP		:RETURN HERE IF ERROR
	033000	104000			EMT		:ERROR # DEFINED BY GET SUBROUTINE
	033002	000137	033156		JMP	13\$:GO TO 13\$ IF ERROR
1703	033006			7\$:			
1704	033006	004737	043706		JSR	PC,PRIERR	:GO CHECK FOR PRIMARY ERRORS
	033012	000405			BR	8\$:GO TO 8\$ IF NO ERROR
	033014	000240			NOP		:RETURN HERE IF ERROR
	033016	104000			EMT		:ERROR # DEFINED BY PRIERR SUBROUTINE
	033020	004736			JSR	PC,@(SP)+	:GO BACK FOR MORE ERROR CHECKS
	033022	000137	033156		JMP	13\$:GO TO 13\$ IF ERROR
1705	033026			8\$:			
1706	033026	004737	051024		JSR	PC,SEKSTS	:GO VERIFY RESULTS OF SEEK OPERATION
	033032	000405			BR	9\$:GO TO 9\$ IF NO ERROR
	033034	000240			NOP		:RETURN HERE IF ERROR
	033036	104000			EMT		:ERROR # DEFINED BY SEKSTS SUBROUTINE
	033040	004736			JSR	PC,@(SP)+	:GO BACK FOR MORE ERROR CHECKS

```

1707 033042 000137 033156          9$: JMP 13$ ;GO TO 13$ IF ERROR
1708 033046 004737 050400          JSR PC,CMPERRSTS ;CHECK ANY ERRORS NOT MASKED
      033052 117760                .WORD IAE!NDTMSK ;MASK FOR RMER1
      033054 000010                .WORD DPE ;MASK FOR RMER2
      033056 000405                BR 10$ ;GO TO 10$ IF NO ERROR
      033060 000240                NOP ;RETURN HERE IF ERROR
      033062 104000                EMT ;ERROR # DEFINED BY CMPERRSTS SUBROUTINE
      033064 004736                JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      033066 000137 033156          JMP 13$ ;GO TO 13$ IF ERROR
1709 033072 004737 044540          10$: JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
1710 033076 000405                BR 11$ ;GO TO 11$ IF NO ERROR
      033100 000240                NOP ;RETURN HERE IF ERROR
      033102 104000                EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
      033104 004736                JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      033106 000137 033156          JMP 13$ ;GO TO 13$ IF ERROR
1711 033112 005237 001444          11$: INC RMDCO ;INCREMENT CYLINDER ADDRESS
1712 033112 023727 001444 002000  CMP RMDCO,#1024. ;DONE??
1713 033116 002412                BLT 12$ ;NO-DO NEXT CYLINDER
1714 033124 032737 010000 001442  BIT #FMT16,RMOFO ;16 BIT FORMAT TESTED??
1715 033126 001010                BNE 13$ ;YES !!
1716 033134 012737 010000 001442  MOV #FMT16,RMOFO ;SETUP FOR 16 BIT TEST
1717 033136 001467 001444          MOV #823.,RMDCO ;START AT FIRST INVALID CYLINDER
1718 033144 032672                JMP 2$
1719 033152 000137 032672          12$:
1720 033156 000137 032672          13$:
1721 033156
1722 033156
1723 033156
1724 033156
  
```

```

*****
*TEST 53 IVC SEEK TEST
*****
TST53:
  
```

```

033156 000004                SCOPE ;SCOPE CALL
033160 000240                NOP ;START OF TEST
033162 012706 001100          MOV #STACK,SP ;INITIALIZE STACK POINTER
033166 013700 001276          MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
033172 013701 001464          MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
033176 012737 000053 001226  MOV #53,$TSTN ;SET TEST NUMBER IN APT MAIL BOX
1725 033204 004737 052264          JSR PC,CNTCLR ;GO ISSUE CONTROLLER CLEAR
1726 033210 000404                BR 1$ ;GO TO 1$ IF NO ERROR
      033212 000240                NOP ;RETURN HERE IF ERROR
      033214 104000                EMT ;ERROR NUMBER DEFINED BY SUBROUTINE
      033216 000137 033516          JMP 8$ ;GO TO 8$ IF ERROR
1727 033222 112737 000024 001543  1$: MOVB #RMMR1,PUTINX ;SETUP PUT INDEX TABLE
1728 033230 112737 000200 001544  MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
      033236 012737 000001 001434  MOV #DMD,RMMR10 ;SET RMMR1 OUTPUT BUFFER = DMD
      033244 004737 043160          JSR PC,PUT ;GO WRITE RMMR1 VIA PUT SUBROUTINE
      033250 000404                BR 2$ ;GO TO 2$ IF NO ERROR
      033252 000240                NOP ;RETURN HERE IF ERROR
      033254 104000                EMT ;ERROR DEFINED BY PUT SUBROUTINE
      033256 000137 033516          JMP 8$ ;GO TO 8$ IF ERROR
1729 033262 012737 000005 001410  2$: MOV #SEEK!GO,RMCS10
1730 033262
  
```

```

1731 033270 012737 000000 001434 MOV #0,RMMR10
1732 033276 012737 000000 001444 MOV #0,RMDCO ;CYLINDER = 0
1733 033304 012737 000000 001416 MOV #0,RMDAO ;TRACK = 0, SECTOR = 0
1734 033312 012702 001543 MOV #PUTINX,R2 ;WRITE REGISTER INDEX
1735 033316 112722 000024 MOVVB #RMMR1,(R2)+ ;TABLE
1736 033322 112722 000006 MOVVB #RMDA,(R2)+
1737 033326 112722 000034 MOVVB #RMDC,(R2)+
1738 033332 112722 000000 MOVVB #RMCS1,(R2)+
1739 033336 112722 000200 MOVVB #200,(R2)+
1740 033342 004737 042624 JSR PC,GETSTS ;SETUP FOR STATUS
1741 033346 004737 043160 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
033352 000404 BR 3$ ;GO TO 3$ IF NO ERROR
033354 000240 NOP ;RETURN HERE IF ERROR
033356 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
033360 000137 033516 JMP 8$ ;GO TO 8$ IF ERROR
1742 033364 3$: JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
1743 033364 004737 042710 BR 4$ ;GO TO 4$ IF NO ERROR
033370 000404 NOP ;RETURN HERE IF ERROR
033372 000240 EMT ;ERROR # DEFINED BY GET SUBROUTINE
033374 104000 JMP 8$ ;GO TO 8$ IF ERROR
1744 033402 4$: JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
1745 033402 004737 043706 BR 5$ ;GO TO 5$ IF NO ERROR
033406 000405 NOP ;RETURN HERE IF ERROR
033410 000240 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
033412 104000 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
033414 004736 JMP 8$ ;GO TO 8$ IF ERROR
1746 033422 032737 010000 001376 5$: BIT #IVC,RMER2I ;DID INVALID COMMAND SET??
1747 033430 001012 BNE 6$ ;YES!!
1748 033432 012737 010000 001140 MOV #IVC,$GDDAT ;GOOD DATA FOR TYPEOUT
1749 033440 013737 001376 001142 MOV RMER2I,$BDDAT ;BAD DATA FOR TYPEOUT
1750 033446 042737 167777 001142 BIC #^CIVC,$BDDAT
1751 033454 104064 EMT 64
1752 033456 6$: JSR PC,CMPEERRSTS ;CHECK ANY ERRORS NOT MASKED
1753 033456 004737 050400 .WORD NDTMSK ;MASK FOR RMER1
033462 115760 .WORD IVC!DPE ;MASK FOR RMER2
033464 010010 BR 7$ ;GO TO 7$ IF NO ERROR
033466 000405 NOP ;RETURN HERE IF ERROR
033470 000240 EMT ;ERROR # DEFINED BY CMPEERRSTS SUBROUTINE
033472 104000 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
033474 004736 JMP 8$ ;GO TO 8$ IF ERROR
1754 033502 7$: JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
1755 033502 004737 044540 BR 8$ ;GO TO 8$ IF NO ERROR
033506 000403 NOP ;RETURN HERE IF ERROR
033510 000240 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
033512 104000 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
033514 004736
1756 033516 8$:
1757
1758

```

 *TEST 54 ABORT SEEK TEST

```

033516
033516 000004
033520 000240
SCOPE
NOP
;SCOPE CALL
;START OF TEST

```

033522	012706	001100			MOV	#STACK,SP	:INITIALIZE STACK POINTER
033526	013700	001276			MOV	\$BASE,R0	:R0 = UNIBUS ADDRESS
033532	013701	001464			MOV	TSTQUE,R1	:(R1) = DEVICE BEING TESTED
033536	012737	000054	001226		MOV	#54,\$TESTN	::SET TEST NUMBER IN APT MAIL BOX
1759							
1760	033544	004737	041650		JSR	PC,\$TSTPRP	:PREPARE DEVICE FOR TEST
	033550	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 1\$ IF NO ERROR
							:RETURN HERE IF ERROR
							:ERROR # DEFINED BY TSTPRP SUBROUTINE
							:GO TO 9\$ IF ERROR
	033552	000404			BR	1\$	
	033554	000240			NOP		
	033556	104000			EMT		
	033560	000137	034060		JMP	9\$	
1761	033564			1\$:			
1762	033564	012737	040000	001424	MOV	#UNS,RMER10	:SET UNSAFE ERROR
1763	033572	012737	000000	001444	MOV	#0,RMDCO	:CYLINDER = 0
1764	033600	012737	000000	001416	MOV	#0,RMDAO	:TRACK = 0, SECTOR = 0
1765	033606	012737	000005	001410	MOV	#SEEK!GO,RMCS10	:SEEK COMMAND
1766	033614	012702	001543		MOV	#PUTINX,R2	:SETUP REGISTER INDEX TABLE
1767	033620	112722	000006		MOVB	#RMDA,(R2)+	:AND OUTPUT BUFFER
1768	033624	112722	000034		MOVB	#RMDC,(R2)+	
1769	033630	112722	000014		MOVB	#RMER1,(R2)+	
1770	033634	112722	000000		MOVB	#RMCS1,(R2)+	
1771	033640	112722	000200		MOVB	#200,(R2)+	
1772	033644	004737	042624		JSR	PC,\$GETSTS	:SETUP FOR STATUS FETCH
1773	033650	004737	043160		JSR	PC,\$PUT	:GO WRITE REGISTER(S) WITH PUT SUBROUTINE
	033654	000404			BR	2\$:GO TO 2\$ IF NO ERROR
	033656	000240			NOP		:RETURN HERE IF ERROR
	033660	104000			EMT		:ERROR # DEFINED BY PUT SUBROUTINE
	033662	000137	034060		JMP	9\$:GO TO 9\$ IF ERROR
1774	033666			2\$:			
1775	033666	004737	042710		JSR	PC,\$GET	:GO READ REGISTER(S) WITH GET SUBROUTINE
	033672	000404			BR	3\$:GO TO 3\$ IF NO ERROR
	033674	000240			NOP		:RETURN HERE IF ERROR
	033676	104000			EMT		:ERROR # DEFINED BY GET SUBROUTINE
	033700	000137	034060		JMP	9\$:GO TO 9\$ IF ERROR
1776	033704	032737	020000	001346	3\$:	BIT	#PIP,RMDSI
1777	033712	001411			BEQ	4\$:DID DRIVE START SEEK??
1778	033714	013737	001346	001142	MOV	RMDSI,\$BDDAT	:NO!!
1779	033722	042737	157777	001142	BIC	#^CPIP,\$BDDAT	:BAD DATA FOR TYPEOUT
1780	033730	005037	001140		CLR	\$GDDAT	:GOOD DATA FOR TYPEOUT
1781	033734	104065			EMT	6\$	
1782	033736	004737	043522		4\$:	JSR	PC,\$TIMOUT
1783							:WAIT FOR GO TO RESET
1784	033742	004737	042710		JSR	PC,\$GET	:GO READ REGISTER(S) WITH GET SUBROUTINE
	033746	000404			BR	5\$:GO TO 5\$ IF NO ERROR
	033750	000240			NOP		:RETURN HERE IF ERROR
	033752	104000			EMT		:ERROR # DEFINED BY GET SUBROUTINE
	033754	000137	034060		JMP	9\$:GO TO 9\$ IF ERROR
1785	033760			5\$:			
1786	033760	004737	043706		JSR	PC,\$PRIERR	:GO CHECK FOR PRIMARY ERRORS
	033764	000405			BR	6\$:GO TO 6\$ IF NO ERROR
	033766	000240			NOP		:RETURN HERE IF ERROR

```

154 ABORT SEEK TEST
033770 104000 EMT
033772 004736 JSR PC,@(SP)+ ;ERROR # DEFINED BY PRIERR SUBROUTINE
033774 000137 034060 JMP 9$ ;GO BACK FOR MORE ERROR CHECKS
1787 034000 6$: ;GO TO 9$ IF ERROR
034000 004737 057766 JSR PC,STCDRVSTS ;GO CHECK FOR CHANGES IN DRIVE STATUS
034004 000405 BR 7$ ;GO TO 7$ IF NO ERROR
034006 000240 NOP ;RETURN HERE IF ERROR
034010 104000 EMT ;ERROR # DEFINED BY STCDRVSTS SUBROUTINE
034012 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
034014 000137 034060 JMP 9$ ;GO TO 9$ IF ERROR
1789 034020 7$:
1790 034020 004737 050400 JSR PC,CMPERRSTS ;CHECK ANY ERRORS NOT MASKED
034024 155760 .WORD ND1MSK(UN) ;MASK FOR RMER1
034026 000010 .WORD DPE ;MASK FOR RMER2
034030 000405 BR 8$ ;GO TO 8$ IF NO ERROR
034032 000240 NOP ;RETURN HERE IF ERROR
034034 104000 EMT ;ERROR # DEFINED BY CMPERRSTS SUBROUTINE
034036 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
034040 000137 034060 JMP 9$ ;GO TO 9$ IF ERROR
1791 034044 8$:
1792 034044 004737 044540 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
034050 000403 BR 9$ ;GO TO 9$ IF NO ERROR
034052 000240 NOP ;RETURN HERE IF ERROR
034054 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
034056 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
1793 034060 9$:
1794
1795

```

 *TEST 55 SEEK AT OFFSET

 TST55:

```

034060 000004 SCOPE ;SCOPE CALL
034062 000240 NOP ;START OF TEST
034064 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
034070 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
034074 013701 001464 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
034100 012737 000055 001226 MOV #55,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
1796
1797 034106 1$:
1798 034106 004737 041650 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
034112 054130 .WORD 054130 ;TASK DESCRIPTOR AS FOLLOWS:
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF "SKI" OR "PIP" IS SET
;VERIFY RECALIBRATION
034114 000404 BR 2$ ;GO TO 2$ IF NO ERROR
034116 000240 NOP ;RETURN HERE IF ERROR
034120 104000 EMT ;ERROR # DEFINED BY TSTPRP SUBROUTINE
034122 000137 034404 JMP 10$ ;GO TO 10$ IF ERROR
1799 034126 2$:
1800 034126 012737 000000 001444 MOV #0,RMDC0 ;CYLINDER = 0
1801 034134 012737 000000 001416 MOV #0,RMDAO ;TRACK = 0, SECTOR = 0
1802 034142 012737 000015 001410 MOV #OFFSET,GO,RMCS10 ;LOAD OFFSET COMMAND IN BUFFER
1803 034150 012702 001543 MOV #PUTINX,R2 ;R2 POINTS TO REGISTER TABLE
1804 034154 112722 000034 MOVB #RMDC,(R2)+ ;WRITE REGISTER INDEX TABLE

```



```

1805 034160 112722 000006      MOVB   #RMDA,(R2)+
1806 034164 112722 000000      MOVB   #RMCS1,(R2)+
1807 034170 112722 000200      MOVB   #200,(R2)+
1808 034174 004737 043160      JSR    PC,PUT          ;WRITE TERMINATOR
      034200 000404          BR     3$             ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      034202 000240          NOP                    ;GO TO 3$ IF NO ERROR
      034204 104000          EMT                    ;RETURN HERE IF ERROR
      034206 000137 034404      JMP    10$           ;ERROR # DEFINED BY PUT SUBROUTINE
1809 034212 004737 042624      JSR    PC,GETSTS      ;GO TO 10$ IF ERROR
1810 034216 004737 043522      JSR    PC,TIMOUT     ;SETUP FOR STATUS FETCH
1811 034222          4$:                  ;WAIT FOR OFFSET TO COMPLETE
1812 034222 012737 000005 001410  MOV    #SEEK!GO,RMCS1D ;LOAD SEEK COMMAND
1813 034230 112737 000000 001543  MOVB   #RMCS1,PUTINX  ;LOAD REGISTER INDEX TABLE
1814 034236 112737 000200 001544  MOVB   #200,PUTINX+1
1815 034244 004737 043160      JSR    PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      034250 000404          BR     5$             ;GO TO 5$ IF NO ERROR
      034252 000240          NOP                    ;RETURN HERE IF ERROR
      034254 104000          EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      034256 000137 034404      JMP    10$           ;GO TO 10$ IF ERROR
1816 034262          5$:
1817
1818          ;WAIT FOR COMMAND TO COMPLETE
      034262 004737 043522      JSR    PC,TIMOUT     ;GO TO TIMOUT SUBROUTINE
1819
1820 034266 004737 042710      JSR    PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      034272 000404          BR     6$             ;GO TO 6$ IF NO ERROR
      034274 000240          NOP                    ;RETURN HERE IF ERROR
      034276 104000          EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      034300 000137 034404      JMP    10$           ;GO TO 10$ IF ERROR
1821 034304          6$:
1822 034304 004737 043706      JSR    PC,PRIERR     ;GO CHECK FOR PRIMARY ERRORS
      034310 000405          BR     7$             ;GO TO 7$ IF NO ERROR
      034312 000240          NOP                    ;RETURN HERE IF ERROR
      034314 104000          EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      034316 004736          JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      034320 000137 034404      JMP    10$           ;GO TO 10$ IF ERROR
1823 034324          7$:
1824 034324 004737 051024      JSR    PC,SEKSTS     ;GO VERIFY RESULTS OF SEEK OPERATION
      034330 000405          BR     8$             ;GO TO 8$ IF NO ERROR
      034332 000240          NOP                    ;RETURN HERE IF ERROR
      034334 104000          EMT                    ;ERROR # DEFINED BY SEKSTS SUBROUTINE
      034336 004736          JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      034340 000137 034404      JMP    10$           ;GO TO 10$ IF ERROR
1825 034344          8$:
1826 034344 004737 050400      JSR    PC,CMPEERRSTS ;CHECK ANY ERRORS NOT MASKED
      034350 115760          .WORD  NDTMSK        ;MASK FOR RMER1
      034352 000010          .WORD  DPE           ;MASK FOR RMER2
      034354 000405          BR     9$             ;GO TO 9$ IF NO ERROR
      034356 000240          NOP                    ;RETURN HERE IF ERROR
      034360 104000          EMT                    ;ERROR # DEFINED BY CMPEERRSTS SUBROUTINE
      034362 004736          JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      034364 000137 034404      JMP    10$           ;GO TO 10$ IF ERROR
1827 034370          9$:
1828 034370 004737 044540      JSR    PC,SECERR     ;GO CHECK FOR SECONDARY ERRORS
      034374 000403          BR     10$           ;GO TO 10$ IF NO ERROR
      034376 000240          NOP                    ;RETURN HERE IF ERROR
      034400 104000          EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
  
```

```

1829 034402 004736          JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
1830 034404          10$:
1831
:*****
:*TEST 56          LOOK AHEAD TEST
:*****
TST56:
034404          000004          SCOPE          ;SCOPE CALL
034406          000240          NOP            ;START OF TEST
034410          012706 001100  MOV    #STACK,SP ;INITIALIZE STACK POINTER
034414          013700 001276  MOV    $BASE,R0  ;R0 = UNIBUS ADDRESS
034420          013701 001464  MOV    TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
034424          012737 000056 001226  MOV    #56,$TESTN ;SET TEST NUMBER IN APT MAIL BOX

1832
1833 034432 004737 052264  JSR    PC,CNTCLR    ;GO ISSUE CONTROLLER CLEAR
034436          000404          BR      1$         ;GO TO 1$ IF NO ERROR
034440          000240          NOP            ;RETURN HERE IF ERROR
034442          104000          EMT          ;ERROR NUMBER DEFINED BY SUBROUTINE
034444          000137 035032  JMP    16$        ;GO TO 16$ IF ERROR

1834 034450          18$:
1835 034450          012703 000037  MOV    #31,R3     ;R3 = SAMPLE COUNT FOR 18 BIT MODE
1836 034454          012737 000000 001442  MOV    #0,RMOFO   ;START WITH 18 BIT MODE
1837 034462          012737 000000 001444  MOV    #0,RMDCO   ;CYLINDER = 0
1838 034470          012737 000000 001416  MOV    #0,RMDAO   ;SEARCH TRACK = 0, SECTOR = 0
1839 034476          012737 000031 001410  MOV    #SEARCH!GO,RMCS1C ;RECALIBRATE COMMAND
1840 034504          012702 001543  MOV    #PUTINX,R2
1841 034510          112722 000006  MOV    #RMDA,(R2)+
1842 034514          112722 000034  MOV    #RMDC,(R2)+
1843 034520          112722 000032  MOV    #RMOF,(R2)+
1844 034524          112722 000000  MOV    #RMCS1,(R2)+
1845 034530          112722 000200  MOV    #200,(R2)+
1846 034534          004737 042624  JSR    PC,GETSTS
1847 034540          2$:
1848 034540          004737 043160  JSR    PC,PUT      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
034544          000404          BR      3$         ;GO TO 3$ IF NO ERROR
034546          000240          NOP            ;RETURN HERE IF ERROR
034550          104000          EMT          ;ERROR # DEFINED BY PUT SUBROUTINE
034552          000137 035022  JMP    15$        ;GO TO 15$ IF ERROR
1849 034556          004737 043522  JSR    PC,TIMOUT   ;WAIT FOR COMPLETION
1850
1851 034562          004737 042710  JSR    PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
034566          000404          BR      4$         ;GO TO 4$ IF NO ERROR
034570          000240          NOP            ;RETURN HERE IF ERROR
034572          104000          EMT          ;ERROR # DEFINED BY GET SUBROUTINE
034574          000137 035022  JMP    15$        ;GO TO 15$ IF ERROR

1852 034600          4$:
1853 034600          004737 043706  JSR    PC,PRIERR   ;GO CHECK FOR PRIMARY ERRORS
034604          000405          BR      5$         ;GO TO 5$ IF NO ERROR
034606          000240          NOP            ;RETURN HERE IF ERROR
034610          104000          EMT          ;ERROR # DEFINED BY PRIERR SUBROUTINE
034612          004736          JSR    PC,@(SP)+  ;GO BACK FOR MORE ERROR CHECKS
034614          000137 035022  JMP    15$        ;GO TO 15$ IF ERROR

1854 034620          5$:
1855 034620          012704 103274  6$:
1856 034624          012705 177777  MOV    #BUFFER,R4 ;R4 = STORAGE ADDRESS
1857 034630          016014 000020  MOV    #-1,R5     ;R5 = GROSS TIMER
1858 034634          016002 000020  7$:
MOV    RMLA(R0),(R4) ;STORE RMLA SAMPLE
MOV    RMLA(R0),R2  ;GET ANOTHER LOOK
  
```

```

1859 034640 020214      CMP      R2,(R4)      ;ARE SAMPLES SAME??
1860 034642 001403      BEQ      8$           ;YES!!
1861 034644 005305      DEC      R5           ;TIMEOUT??
1862 034646 001370      BNE      7$           ;NO - TRY AGAIN
1863 034650 000411      BR       10$          ;PROGRAM TIMEOUT
1864 034652 062704 000002 8$:  ADD      #2,R4      ;ADVANCE STORAGE ADDRESS
1865 034656 005303      DEC      R3           ;ALL SAMPLES TAKEN??
1866 034660 001410      BEQ      11$          ;YES!!
1867 034662 026002 000020 9$:  CMP      RMLA(R0),R2 ;WAIT FOR CHANGE
1868 034666 001360      BNE      7$           ;YES!!
1869 034670 005305      DEC      R5           ;TIMEOUT??
1870 034672 001373      BNE      9$           ;NO
1871 034674 104352 10$:  EMT      352
1872 034676 000137 035032  JMP      16$
1873
1874
1875 034702 012702 003500 ;SET 18 BIT SAMPLE PARAMETERS
11$: MOV      #3500,R2    ;R2 = MAXIMUM SAMPLE
1876 034706 012703 000036    MOV      #30,R3       ;R3 = NUMBER OF COMPARES
1877 034712 012704 103274    MOV      #BUFFER,R4   ;R4 = BUFFER ADDRESS
1878 034716 032737 010000 001442 BIT      #FMT16,RMOFO ;IS SAMPLE FOR 18 BIT MODE ?
1879 034724 001404      BEQ      12$          ;BR IF YES
1880
1881 ;SET 16 BIT SAMPLE PARAMETERS
1882 034726 012702 003700    MOV      #3700,R2     ;R2 = MAXIMUM SAMPLE
1883 034732 012703 000040    MOV      #32,R3       ;R3 = NUMBER OF COMPARES
1884 034736 012405 12$:  MOV      (R4)+,R5     ;GET A SAMPLE AND INCREMENT
1885 034740 062705 000100    ADD      #100,R5      ;FOR EXPECTED VALUE OF NEXT
1886 034744 020205      CMP      R2,R5        ;SHOULD NEXT BE 0 ?
1887 034746 103001      BHIS    13$          ;NO!!
1888 034750 005005      CLR      R5           ;YES - CHANGE EXPECTED
1889 034752 020514 13$:  CMP      R5,(R4)     ;IS NEXT SAMPLE CORRECT??
1890 034754 001406      BEQ      14$          ;YES!!
1891 034756 010537 001140    MOV      R5,$GDDAT    ;EXPECTED VALUE
1892 034762 011437 001142    MOV      (R4),$BDDAT  ;RECEIVED VALUE
1893 034766 104353      EMT      353
1894 034770 000414      BR       15$
1895 034772 005303 14$:  DEC      R3           ;ALL SAMPLES CHECKED??
1896 034774 001360      BNE      12$          ;NO - TEST NEXT SAMPLE
1897
1898 034776 032737 010000 001442 BIT      #FMT16,RMOFO ;TEST DONE FOR 16 BIT MODE ?
1899 035004 001006      BNE      15$          ;BR IF YES
1900 035006 012737 010000 001442 MOV      #FMT16,RMOFO ;SET 16 BIT MODE AND
1901 035014 012703 000041    MOV      #33,R3       ;SET SAMPLE COUNT.
1902 035020 000647      BR       2$           ;TEST AGAIN
1903
1904 035022 15$:  MOV      (SP)+,ERRVEC+2 ;:POP STACK INTO ERRVEC+2
1905 035026 012637 000006    MOV      (SP)+,ERRVEC ;:POP STACK INTO ERRVEC
1906 035032 16$:
1907
1908

```

```

*****
*TEST 57 SEARCH ON CYLINDER
*****
TST57:
SCOPE
NOP ;SCOPE CALL
;START OF TEST

```

035032
035032 000004
035034 000240

```

035036 012706 001100      MOV      #STACK,SP      ;INITIALIZE STACK POINTER
035042 013700 001276      MOV      $BASE,R0       ;R0 = UNIBUS ADDRESS
035046 013701 001464      MOV      TSTQUE,R1      ;(R1) = DEVICE BEING TESTED
035052 012737 000057 001226  MOV      #57,$TESTN     ;SET TEST NUMBER IN APT MAIL BOX

1909
1910 035060 004737 041650      JSR      PC,TSTPRP      ;PREPARE DEVICE FOR TEST
035064 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 1$ IF NO ERROR
                                ;RETURN HERE IF ERROR
                                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                ;GO TO 13$ IF ERROR

035066 000404      BR      1$
035070 000240      NOP
035072 104000      EMT
035074 000137 035510      JMP     13$

1911 035100      1$:
1912 035100 012703 000035      MOV      #29,R3         ;R3 = MAXIMUM SECTOR ADDRESS
1913 035104 012737 000000 001444      MOV      #0,RMDCO       ;CYLINDER = 0
1914 035112 012737 000000 001416      MOV      #0,RMDAO       ;TRACK = 0, SECTOR = 0
1915 035120 012737 000000 001442      MOV      #0,RMOFO       ;18 BIT FORMAT
1916 035126 012737 000005 001410 2$:      MOV      #SEEK!GO,RMCS10 ;SEEK COMMAND
1917 035134 012702 001543      MOV      #PUTINX,R2     ;SETUP REGISTER INDEX TABLE
1918 035140 112722 000006      MOVB    #RMDA,(R2)+     ;FOR SEEK TO CYLINDER 0
1919 035144 112722 000034      MOVB    #RMDC,(R2)+
1920 035150 112722 000032      MOVB    #RMOF,(R2)+
1921 035154 112722 000000      MOVB    #RMCS1,(R2)+
1922 035160 112722 000200      MOVB    #200,(R2)+
1923 035164 004737 042624      JSR     PC,GETSTS      ;SETUP FOR STATUS FETCH
1924 035170 004737 043160      JSR     PC,PUT         ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
035174 000404      BR      3$
035176 000240      NOP
035200 104000      EMT
035202 000137 035510      JMP     13$
1925 035206 004737 043522      JSR     PC,TIMOUT     ;GO TO 13$ IF ERROR
1926
1927 035212 004737 042710      JSR     PC,GET        ;GO READ REGISTER(S) WITH GET SUBROUTINE
035216 000404      BR      4$
035220 000240      NOP
035222 104000      EMT
035224 000137 035510      JMP     13$
1928 035230      4$:
1929 035230 004737 043706      JSR     PC,PRIERR     ;GO CHECK FOR PRIMARY ERRORS
035234 000405      BR      5$
035236 000240      NOP
035240 104000      EMT
035242 004736      JSR     PC,@(SP)+     ;GO BACK FOR MORE ERROR CHECKS
035244 000137 035510      JMP     13$
1930 035250      5$:
1931 035250 004737 051024      JSR     PC,SEKSTS     ;GO VERIFY RESULTS OF SEEK OPERATION
035254 000405      BR      6$
035256 000240      NOP
035260 104000      EMT
035262 004736      JSR     PC,@(SP)+     ;GO BACK FOR MORE ERROR CHECKS
035264 000137 035510      JMP     13$
1932 035270 012737 000031 001410 6$:      MOV      #SEARCH!GO,RMCS10 ;STORE SEARCH COMMAND
    
```

```

1933 035276 004737 043160 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      035302 000404 BR 7$ ;GO TO 7$ IF NO ERROR
      035304 000240 NOP ;RETURN HERE IF ERROR
      035306 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
      035310 000137 035510 JMP 13$ ;GO TO 13$ IF ERROR
1934 035314 7$: JSR PC,TIMOUT ;WAIT FOR SEARCH TO COMPLETE
1935 035314 004737 043522
1936
1937 035320 004737 042710 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
      035324 000404 BR 8$ ;GO TO 8$ IF NO ERROR
      035326 000240 NOP ;RETURN HERE IF ERROR
      035330 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
      035332 000137 035510 JMP 13$ ;GO TO 13$ IF ERROR
1938 035336 8$: JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
1939 035336 004737 043706 BR 9$ ;GO TO 9$ IF NO ERROR
      035342 000405 NOP ;RETURN HERE IF ERROR
      035344 000240 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
      035346 104000 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      035350 004736 JMP 13$ ;GO TO 13$ IF ERROR
      035352 000137 035510
1940 035356 9$: JSR PC,SCHSTS ;GO VERIFY SEARCH OPERATION
1941 035356 004737 056422 BR 10$ ;GO TO 10$ IF NO ERROR
      035362 000405 NOP ;RETURN HERE IF ERROR
      035364 000240 EMT ;ERROR # DEFINED BY SCHSTS SUBROUTINE
      035366 104000 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      035370 004736 JMP 13$ ;GO TO 13$ IF ERROR
      035372 000137 035510
1942 035376 10$: JSR PC,COMPERRSTS ;CHECK ANY ERRORS NOT MASKED
1943 035376 004737 050400 .WORD NDTMSK ;MASK FOR RMER1
      035402 115760 .WORD DPE ;MASK FOR RMER2
      035404 000010 BR 11$ ;GO TO 11$ IF NO ERROR
      035406 000405 NOP ;RETURN HERE IF ERROR
      035410 000240 EMT ;ERROR # DEFINED BY COMPERRSTS SUBROUTINE
      035412 104000 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      035414 004736 JMP 13$ ;GO TO 13$ IF ERROR
      035416 000137 035510
1944 035422 11$: JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
1945 035422 004737 044540 BR 12$ ;GO TO 12$ IF NO ERROR
      035426 000405 NOP ;RETURN HERE IF ERROR
      035430 000240 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
      035432 104000 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      035434 004736 JMP 13$ ;GO TO 13$ IF ERROR
      035436 000137 035510
1946 035442 12$: INC RMDAO ;INCREMENT SECTOR ADDRESS
1947 035442 005237 001416 CMP R3,RMDAO ;DONE ALL SECTORS??
1948 035446 020337 001416 BHIS 6$ ;NO!!
1949 035452 103306
1950
1951 035454 032737 010000 001442 BIT #FMT16,RMOFO ;IS 16 BIT FORMAT DONE??
1952 035462 001012 BNE 13$ ;BR IF YES
1953 035464 012737 010000 001442 MOV #FMT16,RMOFO ;SET 16 BIT FORMAT
1954 035472 012703 000037 MOV #31,R3 ;R3 = MAXIMUM SECTOR ADDRESS
1955 035476 012737 000000 001416 MOV #0,RMDAO ;START AT TRACK = 0, SECTOR = 0
1956 035504 000137 035126 JMP 2$ ;GO TEST AGAIN
1957 035510
1958
1959

```

;*TEST 60 SEARCH OFF CYLINDER

TST60:

035510						SCOPE		:SCOPE CALL
035510	000004					NOP		:START OF TEST
035512	000240					MOV	#STACK,SP	:INITIALIZE STACK POINTER
035514	012706	001100				MOV	\$BASE,R0	:R0 = UNIBUS ADDRESS
035520	013700	001276				MOV	TSTQUE,R1	:(R1) = DEVICE BEING TESTED
035524	013701	001464				MOV	#60,\$TESTN	::SET TEST NUMBER IN APT MAIL BOX
035530	012737	000060	001226					
1960								
1961	035536	004737	041650			JSR	PC,TSTPRP	:PREPARE DEVICE FOR TEST
	035542	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 1\$ IF NO ERROR
								:RETURN HERE IF ERROR
								:ERROR # DEFINED BY TSTPRP SUBROUTINE
								:GO TO 14\$ IF ERROR
	035544	000404				BR	1\$	
	035546	000240				NOP		
	035550	104000				EMT		
	035552	000137	036214			JMP	14\$	
1962	035556				1\$:			
1963	035556	012737	000000	001416		MOV	#0,RMDAO	:START WITH 18 BIT MODE
1964	035564	012737	000000	001442		MOV	#0,RMFO	:R4 = MAXIMUM SECTOR ADDRESS
1965	035572	012704	000035			MOV	#29,R4	:R3 = SEARCH CYLINDER
1966	035576	012703	000632			MOV	#410,R3	:R5 = SEEK CYLINDER
1967	035602	012705	000633			MOV	#411,R5	:LOAD SEEK COMMAND
1968	035606	012737	000005	001410	2\$:	MOV	#SEEK!GO,RMCS10	:LOAD CYLINDER ADDRESS
1969	035614	010537	001444			MOV	R5,RMDCO	:R2 POINTS TO REGISTER TABLE
1970	035620	012702	001543			MOV	#PUTINX,R2	:SETUP REGISTER INDEX TABLE
1971	035624	112722	000032			MOVB	#RMOF,(R2)+	:FOR SEEK COMMAND
1972	035630	112722	000034			MOVB	#RMDC,(R2)+	
1973	035634	112722	000006			MOVB	#RMDA,(R2)+	
1974	035640	112722	000000			MOVB	#RMCS1,(R2)+	
1975	035644	112722	000200			MOVB	#200,(R2)+	:TERMINATE TABLE
1976	035650	004737	042624			JSR	PC,GETSTS	:SETUP FOR STATUS
1977	035654	004737	043160			JSR	PC,PUT	:GO WRITE REGISTER(S) WITH PUT SUBROUTINE
	035660	000404				BR	3\$:GO TO 3\$ IF NO ERROR
	035662	000240				NOP		:RETURN HERE IF ERROR
	035664	104000				EMT		:ERROR # DEFINED BY PUT SUBROUTINE
	035666	000137	036214			JMP	14\$:GO TO 14\$ IF ERROR
1978	035672				3\$:			
1979	035672	004737	043522			JSR	PC,TIMOUT	:WAIT FOR SEEK TO COMPLETE
1980								
1981	035676	004737	042710			JSR	PC,GET	:GO READ REGISTER(S) WITH GET SUBROUTINE
	035702	000404				BR	4\$:GO TO 4\$ IF NO ERROR
	035704	000240				NOP		:RETURN HERE IF ERROR
	035706	104000				EMT		:ERROR # DEFINED BY GET SUBROUTINE
	035710	000137	036214			JMP	14\$:GO TO 14\$ IF ERROR
1982	035714				4\$:			
1983	035714	004737	043706			JSR	PC,PRIERR	:GO CHECK FOR PRIMARY ERRORS
	035720	000405				BR	5\$:GO TO 5\$ IF NO ERROR
	035722	000240				NOP		:RETURN HERE IF ERROR
	035724	104000				EMT		:ERROR # DEFINED BY PRIERR SUBROUTINE
	035726	004736				JSR	PC,@(SP)+	:GO BACK FOR MORE ERROR CHECKS
	035730	000137	036214			JMP	14\$:GO TO 14\$ IF ERROR

1984	035734			5\$:			
1985	035734	004737	051024		JSR	PC,SEKSTS	:GO VERIFY RESULTS OF SEEK OPERATION
	035740	000405			BR	6\$:GO TO 6\$ IF NO ERROR
	035742	000240			NOP		:RETURN HERE IF ERROR
	035744	104000			EMT		:ERROR # DEFINED BY SEKSTS SUBROUTINE
	035746	004736			JSR	PC,@(SP)+	:GO BACK FOR MORE ERROR CHECKS
	035750	000137	036214		JMP	14\$:GO TO 14\$ IF ERROR
1986	035754			6\$:			
1987	035754	010337	001444		MOV	R3,RMDCO	:LOAD CYLINDER ADDRESS
1988	035760	012737	000031	001410	MOV	#SEARCH!GO,PCMS10	:LOAD SEARCH COMMAND
1989	035766	004737	043160		JSR	PC,PUT	:GO WRITE REGISTER(S) WITH PUT SUBROUTINE
	035772	000404			BR	7\$:GO TO 7\$ IF NO ERROR
	035774	000240			NOP		:RETURN HERE IF ERROR
	035776	104000			EMT		:ERROR # DEFINED BY PUT SUBROUTINE
	036000	000137	036214		JMP	14\$:GO TO 14\$ IF ERROR
1990	036004			7\$:			
1991	036004	004737	043522		JSR	PC,TIMOUT	:WAIT FOR SEARCH TO COMPLETE
1992							
1993	036010	004737	042710		JSR	PC,GET	:GO READ REGISTER(S) WITH GET SUBROUTINE
	036014	000404			BR	8\$:GO TO 8\$ IF NO ERROR
	036016	000240			NOP		:RETURN HERE IF ERROR
	036020	104000			EMT		:ERROR # DEFINED BY GET SUBROUTINE
	036022	000137	036214		JMP	14\$:GO TO 14\$ IF ERROR
1994	036026			8\$:			
1995	036026	004737	043706		JSR	PC,PRIERR	:GO CHECK FOR PRIMARY ERRORS
	036032	000405			BR	9\$:GO TO 9\$ IF NO ERROR
	036034	000240			NOP		:RETURN HERE IF ERROR
	036036	104000			EMT		:ERROR # DEFINED BY PRIERR SUBROUTINE
	036040	004736			JSR	PC,@(SP)+	:GO BACK FOR MORE ERROR CHECKS
	036042	000137	036214		JMP	14\$:GO TO 14\$ IF ERROR
1996	036046			9\$:			
1997	036046	004737	056422		JSR	PC,SCHSTS	:GO VERIFY SEARCH OPERATION
	036052	000405			BR	10\$:GO TO 10\$ IF NO ERROR
	036054	000240			NOP		:RETURN HERE IF ERROR
	036056	104000			EMT		:ERROR # DEFINED BY SCHSTS SUBROUTINE
	036060	004736			JSR	PC,@(SP)+	:GO BACK FOR MORE ERROR CHECKS
	036062	000137	036214		JMP	14\$:GO TO 14\$ IF ERROR
1998	036066			10\$:			
1999	036066	004737	050400		JSR	PC,COMPERRSTS	:CHECK ANY ERRORS NOT MASKED
	036072	115760			.WORD	NDTMSK	:MASK FOR RMER1
	036074	000010			.WORD	DPE	:MASK FOR RMER2
	036076	000405			BR	11\$:GO TO 11\$ IF NO ERROR
	036100	000240			NOP		:RETURN HERE IF ERROR
	036102	104000			EMT		:ERROR # DEFINED BY COMPERRSTS SUBROUTINE
	036104	004736			JSR	PC,@(SP)+	:GO BACK FOR MORE ERROR CHECKS
	036106	000137	036214		JMP	14\$:GO TO 14\$ IF ERROR
2000	036112			11\$:			
2001	036112	004737	044540		JSR	PC,SECERR	:GO CHECK FOR SECONDARY ERRORS
	036116	000405			BR	12\$:GO TO 12\$ IF NO ERROR
	036120	000240			NOP		:RETURN HERE IF ERROR
	036122	104000			EMT		:ERROR # DEFINED BY SECERR SUBROUTINE
	036124	004736			JSR	PC,@(SP)+	:GO BACK FOR MORE ERROR CHECKS
	036126	000137	036214		JMP	14\$:GO TO 14\$ IF ERROR
2002	036132			12\$:			
2003	036132	005303			DEC	R3	:DECREMENT SEARCH CYLINDER
2004	036134	005205			INC	R5	:INCREMENT SEEK CYLINDER
2005	036136	005237	001416		INC	RMDAO	:INCREMENT SECTOR ADDRESS

```

2006 036142 020437 001416      CMP      R4,RMDAO      :DONE ALL SECTORS??
2007 036146 103020      BHIS     13$          :NO - DO NEXT SECTOR
2008
2009 036150 032737 010000 001442      BIT      #FMT16,RMOFO  :DONE 16 BIT MODE YET ?
2010 036156 001016      BNE     14$          :BR IF YES
2011 036160 012737 010000 001442      MOV      #FMT16,RMOFO  :SET 16 BIT FORMAT
2012 036166 012704 000037      MOV      #31.,R4       :R4 = MAXIMUM SECTOR ADDRESS
2013 036172 012703 000632      MOV      #410.,R3     :R3 = SEARCH CYLINDER
2014 036176 012705 000633      MOV      #411.,R5     :R5 = SEEK CYLINDER
2015 036202 012737 000000 001416      MOV      #0,RMDAO     :START AT TRACK = 0, SECTOR = 0
2016 036210 000137 035606      MOV      #0,RMDAO     :GO TEST AGAIN
2017 036214      JMP      2$
2018
2019
:*****
:*TEST 61      SEARCH INVALID SECTOR
:*****
TST61:
036214      000004      SCOPE      :SCOPE CALL
036214      000240      NOP        :START OF TEST
036216      000240      NOP
036220      012706 001100      MOV      #STACK,SP    :INITIALIZE STACK POINTER
036224      013700 001276      MOV      $BASE,R0     :R0 = UNIBUS ADDRESS
036230      013701 001464      MOV      TSTQUE,R1    :(R1) = DEVICE BEING TESTED
036234      012737 000061 001226      MOV      #61,$TESTN   ;;SET TEST NUMBER IN APT MAIL BOX
2020
2021 036242 004737 041650      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 1$ IF NO ERROR
: RETURN HERE IF ERROR
: ERROR # DEFINED BY TSTPRP SUBROUTINE
: GO TO 12$ IF ERROR
036250      000404      BR        1$
036252      000240      NOP
036254      104000      EMT
036256      000137 036614      JMP      12$
2022 036262      012737 000000 001442      MOV      #0,RMOFO     :SET 18 BIT FORMAT
2023 036262      012737 000000 001444      MOV      #0,RMDCO     :CYLINDER = 0
2024 036270      012737 000036 001416      MOV      #30.,RMDAO   :FIRST INVALID SECTOR = 30.
2025 036276      012737 000031 001410      MOV      #SEARCH!GO,RMCS10 :SEARCH COMMAND
2026 036304      012737 000031 001410      MOV      #PUTINX,R2   :WRITE REGISTER TABLE FOR
2027 036312      012702 001543      MOV      #RMDC,(R2)+  :COMMAND
2028 036316      112722 000034      MOV      #RMDC,(R2)+
2029 036322      112722 000032      MOV      #RMOF,(R2)+
2030 036326      112722 000006      MOV      #RMDA,(R2)+
2031 036332      112722 000000      MOV      #RMCS1,(R2)+
2032 036336      112722 000200      MOV      #200,(R2)+
2033 036342      004737 042624      MOV      #200,(R2)+
2034 036346      004737 052264      JSR      PC,GETSTS   :TERMINATE TABLE
2035 036346      004737 052264      JSR      PC,CNTCLR   :SETUP STATUS FETCH
036352      000404      BR        3$
036354      000240      BR        3$
036356      104000      EMT
036360      000137 036614      JMP      12$
2036 036364      004737 043160      JSR      PC,PUT      :GO ISSUE CONTROLLER CLEAR
2037 036370      000404      BR        4$         :GO TO 3$ IF NO ERROR
: RETURN HERE IF ERROR
: ERROR NUMBER DEFINED BY SUBROUTINE
: GO TO 12$ IF ERROR
: GO WRITE REGISTER(S) WITH PUT SUBROUTINE
: GO TO 4$ IF NO ERROR
    
```


	036372	000240			NOP				:RETURN HERE IF ERROR
	036374	104000			EMT				:ERROR # DEFINED BY PUT SUBROUTINE
	036376	000137	036614		JMP	12\$:GO TO 12\$ IF ERROR
2038	036402			4\$:					
2039	036402	004737	043522		JSR	PC,TIMOUT			:WAIT FOR GO TO RESET
2040									
2041	036406	004737	042710		JSR	PC,GET			:GO READ REGISTER(S) WITH GET SUBROUTINE
	036412	000404			BR	5\$:GO TO 5\$ IF NO ERROR
	036414	000240			NOP				:RETURN HERE IF ERROR
	036416	104000			EMT				:ERROR # DEFINED BY GET SUBROUTINE
	036420	000137	036614		JMP	12\$:GO TO 12\$ IF ERROR
2042	036424			5\$:					
2043	036424	004737	043706		JSR	PC,PRIERR			:GO CHECK FOR PRIMARY ERRORS
	036430	000405			BR	6\$:GO TO 6\$ IF NO ERROR
	036432	000240			NOP				:RETURN HERE IF ERROR
	036434	104000			EMT				:ERROR # DEFINED BY PRIERR SUBROUTINE
	036436	004736			JSR	PC,@(SP)+			:GO BACK FOR MORE ERROR CHECKS
	036440	000137	036614		JMP	12\$:GO TO 12\$ IF ERROR
2044	036444			6\$:					
2045	036444	004737	056422		JSR	PC,SCHSTS			:GO VERIFY SEARCH OPERATION
	036450	000405			BR	7\$:GO TO 7\$ IF NO ERROR
	036452	000240			NOP				:RETURN HERE IF ERROR
	036454	104000			EMT				:ERROR # DEFINED BY SCHSTS SUBROUTINE
	036456	004736			JSR	PC,@(SP)+			:GO BACK FOR MORE ERROR CHECKS
	036460	000137	036614		JMP	12\$:GO TO 12\$ IF ERROR
2046	036464			7\$:					
2047	036464	004737	057766		JSR	PC,STCDRVSTS			:GO CHECK FOR CHANGES IN DRIVE STATUS
	036470	000405			BR	8\$:GO TO 8\$ IF NO ERROR
	036472	000240			NOP				:RETURN HERE IF ERROR
	036474	104000			EMT				:ERROR # DEFINED BY STCDRVSTS SUBROUTINE
	036476	004736			JSR	PC,@(SP)+			:GO BACK FOR MORE ERROR CHECKS
	036500	000137	036614		JMP	12\$:GO TO 12\$ IF ERROR
2048	036504			8\$:					
2049	036504	004737	050400		JSR	PC,CMPERRSTS			:CHECK ANY ERRORS NOT MASKED
	036510	117760			.WORD	IAE!NDTMSK			:MASK FOR RMER1
	036512	000010			.WORD	DPE			:MASK FOR RMER2
	036514	000405			BR	9\$:GO TO 9\$ IF NO ERROR
	036516	000240			NOP				:RETURN HERE IF ERROR
	036520	104000			EMT				:ERROR # DEFINED BY CMPERRSTS SUBROUTINE
	036522	004736			JSR	PC,@(SP)+			:GO BACK FOR MORE ERROR CHECKS
	036524	000137	036614		JMP	12\$:GO TO 12\$ IF ERROR
2050	036530			9\$:					
2051	036530	004737	044540		JSR	PC,SECERR			:GO CHECK FOR SECONDARY ERRORS
	036534	000405			BR	10\$:GO TO 10\$ IF NO ERROR
	036536	000240			NOP				:RETURN HERE IF ERROR
	036540	104000			EMT				:ERROR # DEFINED BY SECERR SUBROUTINE
	036542	004736			JSR	PC,@(SP)+			:GO BACK FOR MORE ERROR CHECKS
	036544	000137	036614		JMP	12\$:GO TO 12\$ IF ERROR
2052	036550			10\$:					
2053	036550	005237	001416		INC	RMDAO			:INCREMENT SECTOR ADDRESS
2054	036554	023727	001416	000100	CMP	RMDAO,#64.			:DONE YET ?
2055	036562	101412			BLOS	11\$:BR IF NO
2056									
2057	036564	032737	010000	001442	BIT	#FMT16,RMOFO			:TEST FOR 16 BIT MODE YET ?
2058	036572	001010			BNE	12\$:BR IF YES
2059	036574	012737	010000	001442	MOV	#FMT16,RMOFO			:SET 16 BIT MODE
2060	036602	012737	000040	001416	MOV	#32.,RMDAO			:SET INVALID SECTOR = 32.

2061 036610 000137 036346
 2062 036614
 2063
 2064

11\$: JMP 2\$;TEST NEXT SECTOR
 12\$:

 :TEST 62 SEARCH INVALID TRACK

 TST62:

036614
 036614 000004
 036616 000240
 036620 012706 001100
 036624 013700 001276
 036630 013701 001464
 036634 012737 000062 001226
 2065
 2066 036642 004737 041650
 036646 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 #62,\$TESTN ;:SET TEST NUMBER IN APT MAIL BOX

036650 000404
 036652 000240
 036654 104000
 036656 000137 037224
 2067 036662
 2068 036662 012737 000031 001410
 2069 036670 012737 000000 001444
 2070 036676 013737 001332 001416
 2071 036704 105237 001417
 2072 036710 012737 000000 001442
 2073 036716 012702 001543
 2074 036722 112722 000006
 2075 036726 112722 000034
 2076 036732 112722 000032
 2077 036736 112722 000000
 2078 036742 112722 000200
 2079 036746 004737 042624

BR 1\$;GO TO 1\$ IF NO ERROR
 NOP ;RETURN HERE IF ERROR
 EMT ;ERROR # DEFINED BY TSTPRP SUBROUTINE
 JMP 12\$;GO TO 12\$ IF ERROR

2080 036752
 2081 036752 004737 052264
 036756 000404
 036760 000240
 036762 104000
 036764 000137 037224
 2082 036770
 2083 036770 004737 043160
 036774 000404
 036776 000240
 037000 104000
 037002 000137 037224
 2084 037006
 2085 037006 004737 043522
 2086
 2087 037012 004737 042710
 037016 000404
 037020 000240

1\$: MOV #SEARCH!GO,PMCS10 ;SEARCH COMMAND
 MOV #0,RMDC0 ;CYLINDER = 0
 MOV LSTRK,RMDAO ;LAST TRACK, SECTOR = 0
 INCB RMDAO+1 ;SETUP FIRST INVALID TRACK
 MOV #0,RMFO ;SET 18 BIT FORMAT
 MOV #PUTINX,R2 ;WRITE REGISTER INDEX TABLE
 MOVB #RMDA,(R2)+
 MOVB #RMDC,(R2)+
 MOVB #RMOF,(R2)+
 MOVB #RMCS1,(R2)+
 MOVB #200,(R2)+ ;TERMINATE TABLE
 JSR PC,GETSTS

2\$: JSR PC,CNTCLR ;GO ISSUE CONTROLLER CLEAR
 BR 3\$;GO TO 3\$ IF NO ERROR
 NOP ;RETURN HERE IF ERROR
 EMT ;ERROR NUMBER DEFINED BY SUBROUTINE
 JMP 12\$;GO TO 12\$ IF ERROR

3\$: JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
 BR 4\$;GO TO 4\$ IF NO ERROR
 NOP ;RETURN HERE IF ERROR
 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
 JMP 12\$;GO TO 12\$ IF ERROR

4\$: JSR PC,TIMOUT ;WAIT FOR GO TO RESET

5\$: JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
 BR 5\$;GO TO 5\$ IF NO ERROR
 NOP ;RETURN HERE IF ERROR

```

037022 104000 EMT
037024 000137 037224 JMP 12$ ;ERROR # DEFINED BY GET SUBROUTINE
2088 037030 5$: JSR PC,PRIERR ;GO TO 12$ IF ERROR
2089 037030 004737 043706 BR 6$ ;GO CHECK FOR PRIMARY ERRORS
037034 000405 ;GO TO 6$ IF NO ERROR
037036 000240 NOP ;RETURN HERE IF ERROR
037040 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
037042 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
037044 000137 037224 JMP 12$ ;GO TO 12$ IF ERROR
2090 037050 6$: JSR PC,SCHSTS ;GO VERIFY SEARCH OPERATION
2091 037050 004737 056422 BR 7$ ;GO TO 7$ IF NO ERROR
037054 000405 ;RETURN HERE IF ERROR
037056 000240 NOP ;ERROR # DEFINED BY SCHSTS SUBROUTINE
037060 104000 EMT ;GO BACK FOR MORE ERROR CHECKS
037062 004736 JSR PC,@(SP)+ ;GO TO 12$ IF ERROR
037064 000137 037224 JMP 12$
2092 037070 7$: JSR PC,STCDRVSTS ;GO CHECK FOR CHANGES IN DRIVE STATUS
2093 037070 004737 057766 BR 8$ ;GO TO 8$ IF NO ERROR
037074 000405 ;RETURN HERE IF ERROR
037076 000240 NOP ;ERROR # DEFINED BY STCDRVSTS SUBROUTINE
037100 104000 EMT ;GO BACK FOR MORE ERROR CHECKS
037102 004736 JSR PC,@(SP)+ ;GO TO 12$ IF ERROR
037104 000137 037224 JMP 12$
2094 037110 8$: JSR PC,CMPERRSTS ;CHECK ANY ERRORS NOT MASKED
2095 037110 004737 050400 .WORD NDTMSK!IAE ;MASK FOR RMER1
037114 117760 .WORD DPE ;MASK FOR RMER2
037116 000010 BR 9$ ;GO TO 9$ IF NO ERROR
037120 000405 NOP ;RETURN HERE IF ERROR
037122 000240 EMT ;ERROR # DEFINED BY CMPERRSTS SUBROUTINE
037124 104000 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
037126 004736 JMP 12$ ;GO TO 12$ IF ERROR
037130 000137 037224
2096 037134 9$: JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
2097 037134 004737 044540 BR 10$ ;GO TO 10$ IF NO ERROR
037140 000405 ;RETURN HERE IF ERROR
037142 000240 NOP ;ERROR # DEFINED BY SECERR SUBROUTINE
037144 104000 EMT ;GO BACK FOR MORE ERROR CHECKS
037146 004736 JSR PC,@(SP)+ ;GO TO 12$ IF ERROR
037150 000137 037224 JMP 12$
2098 037154 10$: INCB RMDAO+1 ;INCREMENT TRACK ADDRESS
2099 037154 105237 001417 CMPB RMDAO+1,#128. ;DONE ALL TRACKS ?
2100 037160 123727 001417 000200 BLOS 11$ ;NO!!
2101 037166 101414
2102
2103 037170 032737 010000 001442 BIT #FMT16,RMOFO ;DONE 16 BIT MODE ?
2104 037176 001012 BNE 12$ ;YES!!
2105 037200 012737 010000 001442 MOV #FMT16,RMOFO ;SET 16 BIT FORMAT
2106 037206 013737 001332 001416 MOV LSTRK,RMDAO ;LAST TRACK, SECTOR = 0
2107 037214 105237 001417 INCB RMDAO+1 ;SETUP FIRST INVALID TRACK ADDRESS
2108 037220 000137 036752 11$: JMP 2$ ;TEST NEXT TRACK
2109
2110 037224 12$:
2111
2112

```

```

*****
*TEST 63 SEARCH INVALID CYLINDER
*****
TST63:

```

037224

037224	000004			SCOPE		:SCOPE CALL
037226	000240			NOP		:START OF TEST
037230	012706	001100		MOV	#STACK,SP	:INITIALIZE STACK POINTER
037234	013700	001276		MOV	\$BASE,R0	:R0 = UNIBUS ADDRESS
037240	013701	001464		MOV	TSTQUE,R1	:(R1) = DEVICE BEING TESTED
037244	012737	000063	001226	MOV	#63,\$TESTN	:SET TEST NUMBER IN APT MAIL BOX
2113						
2114	037252	004737	041650	JSR	PC,TSTPRP	:PREPARE DEVICE FOR TEST
	037256	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 1\$ IF NO ERROR
						:RETURN HERE IF ERROR
						:ERROR # DEFINED BY TSTPRP SUBROUTINE
						:GO TO 12\$ IF ERROR
037260	000404			BR	1\$	
037262	000240			NOP		
037264	104000			EMT		
037266	000137	037624		JMP	12\$	
2115	037272		1\$:			
2116	037272	012737	000031	MOV	#SEARCH,GO,RMCS10	:SEARCH COMMAND
2117	037300	012737	001467	MOV	#823.,RMDCO	:START AT FIRST INVALID CYLINDER
2118	037306	012737	000000	MOV	#0,RMDAO	:TRACK = 0, SECTOR = 0
2119	037314	012737	000000	MOV	#0,RMOFO	:SET 18 BIT FORMAT
2120	037322	012702	001543	MOV	#PUTINX,R2	:WRITE REGISTER INDEX TABLE
2121	037326	112722	000032	MOVB	#RMOF,(R2)+	
2122	037332	112722	000034	MOVB	#RMDC,(R2)+	
2123	037336	112722	000006	MOVB	#RMDA,(R2)+	
2124	037342	112722	000000	MOVB	#RMCS1,(R2)+	
2125	037346	112722	000200	MOVB	#200,(R2)+	
2126	037352	004737	042624	JSR	PC,GETSTS	:SETUP STATUS FETCH
2127	037356					
2128	037356	004737	052264	JSR	PC,CNTCLR	:GO ISSUE CONTROLLER CLEAR
	037362	000404		BR	3\$:GO TO 3\$ IF NO ERROR
	037364	000240		NOP		:RETURN HERE IF ERROR
	037366	104000		EMT		:ERROR NUMBER DEFINED BY SUBROUTINE
	037370	000137	037624	JMP	12\$:GO TO 12\$ IF ERROR
2129	037374					
2130	037374	004737	043160	JSR	PC,PUT	:GO WRITE REGISTER(S) WITH PUT SUBROUTINE
	037400	000404		BR	4\$:GO TO 4\$ IF NO ERROR
	037402	000240		NOP		:RETURN HERE IF ERROR
	037404	104000		EMT		:ERROR # DEFINED BY PUT SUBROUTINE
	037406	000137	037624	JMP	12\$:GO TO 12\$ IF ERROR
2131	037412					
2132	037412	004737	043522	JSR	PC,TIMOUT	
2133						
2134	037416	004737	042710	JSR	PC,GET	:GO READ REGISTER(S) WITH GET SUBROUTINE
	037422	000404		BR	5\$:GO TO 5\$ IF NO ERROR
	037424	000240		NOP		:RETURN HERE IF ERROR
	037426	104000		EMT		:ERROR # DEFINED BY GET SUBROUTINE
	037430	000137	037624	JMP	12\$:GO TO 12\$ IF ERROR
2135	037434					
2136	037434	004737	043706	JSR	PC,PRIERR	:GO CHECK FOR PRIMARY ERRORS
	037440	000405		BR	6\$:GO TO 6\$ IF NO ERROR
	037442	000240		NOP		:RETURN HERE IF ERROR
	037444	104000		EMT		:ERROR # DEFINED BY PRIERR SUBROUTINE
	037446	004736		JSR	PC,(GP)+	:GO BACK FOR MORE ERROR CHECKS

```

2137 037450 000137 037624          JMP      12$          :GO TO 12$ IF ERROR
2138 037454 004737 056422          JSR      PC,SCHSTS   :GO VERIFY SEARCH OPERATION
037460 000405          BR       7$          :GO TO 7$ IF NO ERROR
037462 000240          NOP      :RETURN HERE IF ERROR
037464 104000          EMT     :ERROR # DEFINED BY SCHSTS SUBROUTINE
037466 004736          JSR     :GO BACK FOR MORE ERROR CHECKS
037470 000137 037624          JMP      12$          :GO TO 12$ IF ERROR
2139 037474          7$:
2140 037474 004737 057766          JSR      PC,STCDRVSTS :GO CHECK FOR CHANGES IN DRIVE STATUS
037500 000405          BR       8$          :GO TO 8$ IF NO ERROR
037502 000240          NOP      :RETURN HERE IF ERROR
037504 104000          EMT     :ERROR # DEFINED BY STCDRVSTS SUBROUTINE
037506 004736          JSR     :GO BACK FOR MORE ERROR CHECKS
037510 000137 037624          JMP      12$          :GO TO 12$ IF ERROR
2141 037514          8$:
2142 037514 004737 050400          JSR      PC,CMPERRSTS :CHECK ANY ERRORS NOT MASKED
037520 117760          .WORD   IAEINDTMSK   :MASK FOR RMER1
037522 000010          .WORD   DPE          :MASK FOR RMER2
037524 000405          BR       9$          :GO TO 9$ IF NO ERROR
037526 000240          NOP      :RETURN HERE IF ERROR
037530 104000          EMT     :ERROR # DEFINED BY CMPERRSTS SUBROUTINE
037532 004736          JSR     :GO BACK FOR MORE ERROR CHECKS
037534 000137 037624          JMP      12$          :GO TO 12$ IF ERROR
2143 037540          9$:
2144 037540 004737 044540          JSR      PC,SECERR    :GO CHECK FOR SECONDARY ERRORS
037544 000405          BR       10$         :GO TO 10$ IF NO ERROR
037546 000240          NOP      :RETURN HERE IF ERROR
037550 104000          EMT     :ERROR # DEFINED BY SECERR SUBROUTINE
037552 004736          JSR     :GO BACK FOR MORE ERROR CHECKS
037554 000137 037624          JMP      12$          :GO TO 12$ IF ERROR
2145 037560          10$:
2146 037560 005237 001444          INC      RMD00        :INCREMENT CYLINDER ADDRESS
2147 037564 023727 001444 002000    CMP      RMD00,#1024. :DONE ALL CYLINDERS ?
2148 037572 002412          BLT     :NO!
2149
2150 037574 032737 010000 001442          BIT     #FMT16,RMOF0  :SET 16 BIT FORMAT
2151 037602 001010          BNE     12$          :YES!
2152 037604 012737 010000 001442          MOV     #FMT16,RMOF0  :SET 16 BIT FORMAT
2153 037612 012737 001467 001444          MOV     #823.,RMD00   :START AT FIRST INVALID CYLINDER
2154 037620 000137 037356          JMP      2$
2155
2156 037624          11$:
2157          12$:
2158

```

 TEST 64 IVC SEARCH TEST

```

037624          TST64:
037624 000004          SCOPE          :SCOPE CALL
037626 000240          NOP           :START OF TEST
037630 012708 000100          MOV           :INITIALIZE STACK POINTER
037634 013700 001276          MOV           $BASE,R0 :R0 = UNIBUS ADDRESS
037640 013701 001464          MOV           TSTQUE,R1 : (R1) = DEVICE BEING TESTED
037644 012737 000064 001226          MOV           #64,$TESTN :SET TEST NUMBER IN APT MAIL BOX
2159
2160 037652 004737 041650          JSR      PC,TSTPRP   :PREPARE DEVICE FOR TEST
037656 054130          .WORD   054130     :TASK DESCRIPTOR AS FOLLOWS:

```

```

037660 000404 BR 1$
037662 000240 NOP
037664 104000 EMT
037666 000137 040204 JMP 8$
2161 037672 112737 000024 001543 1$: MOV B #RMMR1,PUTINX ;SETUP PUT INDEX TABLE
2162 037700 112737 000200 001544 MOV B #200,PUTINX+1 ;SET TERMINATOR BYTE
037706 012737 000001 001434 MOV #DMD,RMMR10 ;SET RMMR1 OUTPUT BUFFER = DMD
037714 004737 043160 JSR PC,PUT ;GO WRITE RMMR1 VIA PUT SUBROUTINE
037720 000404 BR 2$ ;GO TO 2$ IF NO ERROR
037722 000240 NOP ;RETURN HERE IF ERROR
037724 104000 EMT ;ERROR DEFINED BY PUT SUBROUTINE
037726 000137 040204 JMP 8$ ;GO TO 8$ IF ERROR
2163 037732 012737 000000 001434 2$: MOV #0,RMMR10 ;RESET DIAGNOSTIC MODE
2164 037740 012737 000000 001444 MOV #0,RMDC0 ;CYLINDER = 0
2165 037746 012737 000000 001416 MOV #0,RMDA0 ;TRACK = 0, SECTOR = 0
2166 037754 012737 000000 001442 MOV #0,RMFO0 ;18 BIT FORMAT
2167 037762 012737 000031 001410 MOV #SEARCH!GO,RMCS10 ;SEARCH COMMAND
2168 037770 012702 001543 MOV #PUTINX,R2 ;LOAD INDEX TABLE
2169 037774 112722 000024 MOV B #RMMR1,(R2)+
2170 040000 112722 000034 MOV B #RMDC,(R2)+
2171 040004 112722 000006 MOV B #RMDA,(R2)+
2172 040010 112722 000032 MOV B #RMDF,(R2)+
2173 040014 112722 000000 MOV B #RMCS1,(R2)+
2174 040020 112722 000200 MOV B #200,(R2)+
2175 040024 004737 042624 JSR PC,GETSTS ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
2176 040030 004737 043160 JSR PC,PUT ;GO TO 3$ IF NO ERROR
040034 000404 BR 3$ ;RETURN HERE IF ERROR
040036 000240 NOP ;ERROR # DEFINED BY PUT SUBROUTINE
040040 104000 EMT ;GO TO 8$ IF ERROR
040042 000137 040204 JMP 8$
2177 040046 004737 043522 3$: JSR PC,TIMOUT ;WAIT FOR GO TO RESET
2178 040046 004737 042710 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
2179 040052 004737 042710 BR 4$ ;GO TO 4$ IF NO ERROR
040056 000404 BR 4$ ;RETURN HERE IF ERROR
040060 000240 NOP ;ERROR # DEFINED BY GET SUBROUTINE
040062 104000 EMT ;GO TO 8$ IF ERROR
040064 000137 040204 JMP 8$
2181 040070 004737 043706 4$: JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
2182 040074 000405 BR 5$ ;GO TO 5$ IF NO ERROR
040076 000240 NOP ;RETURN HERE IF ERROR
040100 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
040102 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
040104 000137 040204 JMP 8$ ;GO TO 8$ IF ERROR
2183 040110 032737 010000 001376 5$: BIT #IVC,RMER21 ;DID IVC SET??
2184 040110 001012 BNE 6$ ;YES!!
2185 040116 012737 010000 001140 MOV #IVC,$GDDAT ;EXPECTED STATUS
2186 040120 013737 001376 001142 MOV RMER21,$BDDAT ;RECEIVED STATUS

```

```

;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 1$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 8$ IF ERROR

;SETUP PUT INDEX TABLE
;SET TERMINATOR BYTE
;SET RMMR1 OUTPUT BUFFER = DMD
;GO WRITE RMMR1 VIA PUT SUBROUTINE
;GO TO 2$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR DEFINED BY PUT SUBROUTINE
;GO TO 8$ IF ERROR
;RESET DIAGNOSTIC MODE
;CYLINDER = 0
;TRACK = 0, SECTOR = 0
;18 BIT FORMAT
;SEARCH COMMAND
;LOAD INDEX TABLE

;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
;GO TO 3$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY PUT SUBROUTINE
;GO TO 8$ IF ERROR

;WAIT FOR GO TO RESET

;GO READ REGISTER(S) WITH GET SUBROUTINE
;GO TO 4$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY GET SUBROUTINE
;GO TO 8$ IF ERROR

;GO CHECK FOR PRIMARY ERRORS
;GO TO 5$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY PRIERR SUBROUTINE
;GO BACK FOR MORE ERROR CHECKS
;GO TO 8$ IF ERROR

;DID IVC SET??
;YES!!
;EXPECTED STATUS
;RECEIVED STATUS

```



```
2212 040346 104000 EMT          :ERROR # DEFINED BY PUT SUBROUTINE
      040350 000137 040550 JMP          9$          :GO TO 9$ IF ERROR
2213 040354 004737 042710 2$: JSR PC,GET      :GO READ REGISTER(S) WITH GET SUBROUTINE
      040360 000404 BR           3$          :GO TO 3$ IF NO ERROR
      040362 000240 NOP          :RETURN HERE IF ERROR
      040364 104000 EMT          :ERROR # DEFINED BY GET SUBROUTINE
      040366 000137 040550 JMP          9$          :GO TO 9$ IF ERROR
2214 040372 032737 020000 3$: BIT #PIP,RMSI    :IS PIP SET??
2215 040372 001412 BEQ         4$          :NO!!
2216 040400 012737 000000 001140 MOV #0,$GDDAT :EXPECTED STATUS
2217 040402 013737 001346 001142 MOV RMSI,$BDDAT :RECEIVED STATUS
2218 040410 042737 157777 001142 BIC #^CPIP,$BDDAT
2219 040416 104261 EMT         261
2220 040424 004737 043522 4$: JSR PC,TIMOUT   :WAIT FOR GO TO RESET
2221 040426 004737 042710 JSR PC,GET      :GO READ REGISTER(S) WITH GET SUBROUTINE
2222 040432 000404 BR           5$          :GO TO 5$ IF NO ERROR
2223 040440 000240 NOP          :RETURN HERE IF ERROR
2224 040442 104000 EMT          :ERROR # DEFINED BY GET SUBROUTINE
2225 040444 000137 040550 JMP          9$          :GO TO 9$ IF ERROR
2226 040450 004737 043706 5$: JSR PC,PRIERR   :GO CHECK FOR PRIMARY ERRORS
      040454 000405 BR           6$          :GO TO 6$ IF NO ERROR
      040456 000240 NOP          :RETURN HERE IF ERROR
      040460 104000 EMT          :ERROR # DEFINED BY PRIERR SUBROUTINE
      040462 004736 JSR PC,@(SP)+ :GO BACK FOR MORE ERROR CHECKS
      040464 000137 040550 JMP          9$          :GO TO 9$ IF ERROR
2227 040470 004737 057766 6$: JSR PC,STCDRVSTS :GO CHECK FOR CHANGES IN DRIVE STATUS
2228 040474 000405 BR           7$          :GO TO 7$ IF NO ERROR
      040476 000240 NOP          :RETURN HERE IF ERROR
      040500 104000 EMT          :ERROR # DEFINED BY STCDRVSTS SUBROUTINE
      040502 004736 JSR PC,@(SP)+ :GO BACK FOR MORE ERROR CHECKS
      040504 000137 040550 JMP          9$          :GO TO 9$ IF ERROR
2229 040510 004737 050400 7$: JSR PC,CMPEERRSTS :CHECK ANY ERRORS NOT MASKED
2230 040514 155760 .WORD NDTMSK!UNS :MASK FOR RMER1
      040516 000010 .WORD DPE          :MASK FOR RMER2
      040520 000405 BR           8$          :GO TO 8$ IF NO ERROR
      040522 000240 NOP          :RETURN HERE IF ERROR
      040524 104000 EMT          :ERROR # DEFINED BY CMPEERRSTS SUBROUTINE
      040526 004736 JSR PC,@(SP)+ :GO BACK FOR MORE ERROR CHECKS
      040530 000137 040550 JMP          9$          :GO TO 9$ IF ERROR
2231 040534 004737 044540 8$: JSR PC,SECERR   :GO CHECK FOR SECONDARY ERRORS
2232 040540 000403 BR           9$          :GO TO 9$ IF NO ERROR
      040542 000240 NOP          :RETURN HERE IF ERROR
      040544 104000 EMT          :ERROR # DEFINED BY SECERR SUBROUTINE
2233 040546 004736 JSR PC,@(SP)+ :GO BACK FOR MORE ERROR CHECKS
2234 040550
2235
```

```
*****
:TEST 66 SEARCH AT OFFSET
*****
```



```

040550          000004          TST66:          SCOPE          :SCOPE CALL
040550          000240          NOP          :START OF TEST
040554          012706          001100          MOV          #STACK,SP  :INITIALIZE STACK POINTER
040560          013700          001276          MOV          $BASE,R0   :R0 = UNIBUS ADDRESS
040564          013701          001464          MOV          TSTQUE,R1  :(R1) = DEVICE BEING TESTED
040570          012737          000066          001226          MOV          #66,$TESTN :SET TEST NUMBER IN APT MAIL BOX

2236          040576          004737          041650          JSR          PC,TSTPRP  :PREPARE DEVICE FOR TEST
2237          040602          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
040604          000404          BR          1$         :GO TO 1$ IF NO ERROR
040606          000240          NOP          :RETURN HERE IF ERROR
040610          104000          EMT          :ERROR # DEFINED BY TSTPRP SUBROUTINE
040612          000137          041106          JMP          9$         :GO TO 9$ IF ERROR

2238          040616          000137          041106          1$:          MOV          #0,RMDCO   :CYLINDER = 0
2239          040616          012737          000000          001444          MOV          #0,RMDAO   :TRACK = 0, SECTOR = 0
2240          040624          012737          000000          001416          MOV          #0,RMDAO   :TRACK = 0, SECTOR = 0
2241          040632          012737          010000          001442          MOV          #FMT16,RMOFO :16 BIT FORMAT
2242          040640          012737          000015          001410          MOV          #OFFSET!GO,RMCS10 :OFFSET COMMAND
2243          040646          012702          001543          MOV          #PUTINX,R2  :SETUP REGISTER INDEX TABLE
2244          040652          112722          000006          MOV          #RMDA,(R2)+ :FOR SEEK TO CYLINDER 0
2245          040656          112722          000034          MOV          #RMDC,(R2)+
2246          040662          112722          000032          MOV          #RMOF,(R2)+
2247          040666          112722          000000          MOV          #RMCS1,(R2)+
2248          040672          112722          000200          MOV          #200,(R2)+
2249          040676          004737          043160          JSR          PC,PUT     :GO WRITE REGISTER(S) WITH PUT SUBROUTINE
040702          000404          BR          2$         :GO TO 2$ IF NO ERROR
040704          000240          NOP          :RETURN HERE IF ERROR
040706          104000          EMT          :ERROR # DEFINED BY PUT SUBROUTINE
040710          000137          041106          JMP          9$         :GO TO 9$ IF ERROR

2250          040714          000137          041106          2$:          JSR          PC,GETSTS  :SETUP FOR STATUS FETCH
2251          040714          004737          042624          JSR          PC,TIMOUT
2252          040720          004737          043522          JSR          PC,TIMOUT
2253          040724          000137          041106          3$:          MOV          #RMCS1,PUTINX :LOAD REGISTER INDEX TABLE
2254          040724          112737          000000          001543          MOV          #200,PUTINX+1
2255          040732          112737          000200          001544          MOV          #SEARCH!GO,RMCS10 :STORE SEARCH COMMAND
2256          040740          012737          000031          001410          MOV          #SEARCH!GO,RMCS10
2257          040746          004737          043160          JSR          PC,PUT     :GO WRITE REGISTER(S) WITH PUT SUBROUTINE
040752          000404          BR          4$         :GO TO 4$ IF NO ERROR
040754          000240          NOP          :RETURN HERE IF ERROR
040756          104000          EMT          :ERROR # DEFINED BY PUT SUBROUTINE
040760          000137          041106          JMP          9$         :GO TO 9$ IF ERROR

2258          040764          000137          041106          4$:          JSR          PC,TIMOUT  :WAIT FOR SEARCH TO COMPLETE
2259          040764          004737          043522          JSR          PC,TIMOUT
2260          040770          004737          042710          JSR          PC,GET     :GO READ REGISTER(S) WITH GET SUBROUTINE
040774          000404          BR          5$         :GO TO 5$ IF NO ERROR
040776          000240          NOP          :RETURN HERE IF ERROR
041000          104000          EMT          :ERROR # DEFINED BY GET SUBROUTINE
041002          000137          041106          JMP          9$         :GO TO 9$ IF ERROR

2262          041006          000137          041106          5$:          JMP          9$

```

```

2263 041006 004737 043706 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
      041012 000405 BR 6$ ;GO TO 6$ IF NO ERROR
      041014 000240 NOP ;RETURN HERE IF ERROR
      041016 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
      041020 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      041022 000137 JMP 9$ ;GO TO 9$ IF ERROR
2264 041026 004737 056422 6$: JSR PC,SCHSTS ;GO VERIFY SEARCH OPERATION
2265 041032 000405 BR 7$ ;GO TO 7$ IF NO ERROR
      041034 000240 NOP ;RETURN HERE IF ERROR
      041036 104000 EMT ;ERROR # DEFINED BY SCHSTS SUBROUTINE
      041040 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      041042 000137 JMP 9$ ;GO TO 9$ IF ERROR
2266 041046 004737 050400 7$: JSR PC,CMPERRSTS ;CHECK ANY ERRORS NOT MASKED
2267 041052 115760 .WORD NDIMSK ;MASK FOR RMER1
      041054 000010 .WORD DPE ;MASK FOR RMER2
      041056 000405 BR 8$ ;GO TO 8$ IF NO ERROR
      041060 000240 NOP ;RETURN HERE IF ERROR
      041062 104000 EMT ;ERROR # DEFINED BY CMPERRSTS SUBROUTINE
      041064 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      041066 000137 JMP 9$ ;GO TO 9$ IF ERROR
2268 041072 004737 044540 8$: JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
2269 041076 000403 BR 9$ ;GO TO 9$ IF NO ERROR
      041100 000240 NOP ;RETURN HERE IF ERROR
      041102 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
      041104 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
2270 041106 000004 001206 9$:
2271
2279

```

```

*****
*TEST 67 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.
*
*****

```

```

041106 000004 TST67: SCOPE ;SCOPE CALL
041106 000240 NOP ;START OF TEST
041110 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
041112 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
041116 013701 001464 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
041122 012737 000001 001206 MOV #1,$TIMES ;DO 1 ITERATION
041126 012737 000067 001226 MOV #67,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
2280
2281 041142 004737 041650 1$: JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
2282 041146 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
041150 000404 BR 2$ ;GO TO 2$ IF NO ERROR

```

	041152	000240			NOP			:RETURN HERE IF ERROR
	041154	104000			EMT			:ERROR # DEFINED BY TSTPRP SUBROUTINE
	041156	000137	041374		JMP	8\$:GO TO 8\$ IF ERROR
2283	041162			2\$:				
2284	041162	012737	000365	001444	MOV	#245, RMDCO		:CYLINDER = 245
2285	041170	012737	000000	001416	MOV	#0, RMDAO		:TRACK = 0, SECTOR = 0
2286	041176	012737	000005	001410	MOV	#SEEK!GO, RMCS10		:LOAD SEEK COMMAND IN BUFFER
2287	041204	012702	001543		MOV	#PUTINX, R2		:R2 POINTS TO REGISTER TABLE
2288	041210	112722	000034		MOV	#RMDC, (R2)+		:WRITE REGISTER INDEX TABLE
2289	041214	112722	000006		MOV	#RMDA, (R2)+		
2290	041220	112722	000000		MOV	#RMCS1, (R2)+		
2291	041224	112722	000200		MOV	#200, (R2)+		:WRITE TERMINATOR
2292	041230	004737	043160		JSR	PC, PUT		:GO WRITE REGISTER(S) WITH PUT SUBROUTINE
	041234	000404			BR	3\$:GO TO 3\$ IF NO ERROR
	041236	000240			NOP			:RETURN HERE IF ERROR
	041240	104000			EMT			:ERROR # DEFINED BY PUT SUBROUTINE
	041242	000137	041374		JMP	8\$:GO TO 8\$ IF ERROR
2293	041246	004737	042624	3\$:	JSR	PC, GETSTS		:SETUP FOR STATUS FETCH
2294	041252	004737	043522		JSR	PC, TIMEOUT		:WAIT FOR SEEK TO COMPLETE
2295								
2296	041256	004737	042710		JSR	PC, GET		:GO READ REGISTER(S) WITH GET SUBROUTINE
	041262	000404			BR	4\$:GO TO 4\$ IF NO ERROR
	041264	000240			NOP			:RETURN HERE IF ERROR
	041266	104000			EMT			:ERROR # DEFINED BY GET SUBROUTINE
	041270	000137	041374		JMP	8\$:GO TO 8\$ IF ERROR
2297	041274			4\$:				
2298	041274	004737	043706		JSR	PC, PRIERR		:GO CHECK FOR PRIMARY ERRORS
	041300	000405			BR	5\$:GO TO 5\$ IF NO ERROR
	041302	000240			NOP			:RETURN HERE IF ERROR
	041304	104000			EMT			:ERROR # DEFINED BY PRIERR SUBROUTINE
	041306	004736			JSR	PC, @(SP)+		:GO BACK FOR MORE ERROR CHECKS
	041310	000137	041374		JMP	8\$:GO TO 8\$ IF ERROR
2299	041314			5\$:				
2300	041314	004737	051024		JSR	PC, SEKSTS		:GO VERIFY RESULTS OF SEEK OPERATION
	041320	000405			BR	6\$:GO TO 6\$ IF NO ERROR
	041322	000240			NOP			:RETURN HERE IF ERROR
	041324	104000			EMT			:ERROR # DEFINED BY SEKSTS SUBROUTINE
	041326	004736			JSR	PC, @(SP)+		:GO BACK FOR MORE ERROR CHECKS
	041330	000137	041374		JMP	8\$:GO TO 8\$ IF ERROR
2301	041334			6\$:				
2302	041334	004737	050400		JSR	PC, CMPERRSTS		:CHECK ANY ERRORS NOT MASKED
	041340	115760			.WORD	NDTMSK		:MASK FOR RMER1
	041342	000010			.WORD	DPE		:MASK FOR RMER2
	041344	000405			BR	7\$:GO TO 7\$ IF NO ERROR
	041346	000240			NOP			:RETURN HERE IF ERROR
	041350	104000			EMT			:ERROR # DEFINED BY CMPERRSTS SUBROUTINE
	041352	004736			JSR	PC, @(SP)+		:GO BACK FOR MORE ERROR CHECKS
	041354	000137	041374		JMP	8\$:GO TO 8\$ IF ERROR
2303	041360			7\$:				
2304	041360	004737	044540		JSR	PC, SECERR		:GO CHECK FOR SECONDARY ERRORS
	041364	000403			BR	8\$:GO TO 8\$ IF NO ERROR
	041366	000240			NOP			:RETURN HERE IF ERROR
	041370	104000			EMT			:ERROR # DEFINED BY SECERR SUBROUTINE
	041372	004736			JSR	PC, @(SP)+		:GO BACK FOR MORE ERROR CHECKS
2305	041374			8\$:				

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.

041374	000004			\$EOSP:	SCOPE		
041376	000240				NOP		
041400	013700	001464			MOV	TSTQUE,R0	:GET POINTER TO TSTQUE
041404	062700	000002			ADD	#2,R0	:ADJUST POINTER TO NEXT DEVICE
041410	010037	001464			MOV	R0,TSTQUE	:SAVE POINTER TO TSTQUE
041414	005710				TST	(R0)	:ANY MORE DEVICES FOR TEST ?
041416	001402				BEQ	1\$:BR IF NO
041420	000137	007660			JMP	READY	:YES, JUMP TO 'READY' ROUTINE
041424	012737	001466	001464	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 YYYYY'
:*WHERE XXXXX AND YYYYY ARE DECIMAL NUMBERS
:*IF THERES A MONITOR GO TO IT
:*IF THERE ISN'T JUMP TO READY

041432				\$EOP:	NOP		
041432	000240				CLR	\$STNM	::ZERO THE TEST NUMBER
041434	005037	001116			CLR	\$TIMES	::ZERO THE NUMBER OF ITERATIONS
041440	005037	001206			INC	\$PASS	::INCREMENT THE PASS NUMBER
041444	005237	001230			BIC	#100000,\$PASS	::DON'T ALLOW A NEG. NUMBER
041450	042737	100000	001230		DEC	(PC)+	::LOOP?
041456	005327			\$EOPCT:	.WORD	1	
041460	000001				BGT	\$DOAGN	::YES
041462	003066				MOV	(PC)+,@(PC)+	::RESTORE COUNTER
041464	012737			\$ENDCT:	.WORD	1	
041466	000001				\$EOPCT		
041470	041460				TYPE	,65\$::TYPE ASCIZ STRING
041472	104401	041500			BR	64\$::GET OVER THE ASCIZ
041476	000407			::65\$:	.ASCIZ	<12><15>/END PASS #/	
041516				64\$:	MOV	\$PASS,-(SP)	::SAVE \$PASS FOR TYPEOUT
041516	013746	001230					::TYPE PASS NUMBER
041522	104405				TYPDS		::GO TYPE--DECIMAL ASCII WITH SIGN
041524	005737	001126			TST	\$ERTTL	::SEE IF ANY ERRORS THIS PASS
041530	001431				BEQ	\$GT42P	::BR IF NO ERRORS TO REPORT
041532	104401	041540			TYPE	,67\$::TYPE ASCIZ STRING
041536	000421				BR	66\$::GET OVER THE ASCIZ
041602				::67\$:	.ASCIZ	/ TOTAL ERRORS SINCE LAST REPORT /	
041602	013746	001126		66\$:	MOV	\$ERTTL,-(SP)	::SAVE \$ERTTL FOR TYPEOUT
041606	104405						::TOTAL NUMBER OF ERRORS
041610	005037	001126			TYPDS		::GO TYPE--DECIMAL ASCII WITH SIGN
					CLR	\$ERTTL	::CLEAR ERROR TOTAL

END OF PASS ROUTINE

041614	104401	001217		\$GT42P: TYPE	,\$SRLF	::TYPE CARRIAGE RETURN, LINE FEED
041620	013700	000042		\$GET42: MOV	@#42,R0	::GET MONITOR ADDRESS
041624	001405			BEQ	\$DOAGN	::BRANCH IF NO MONITOR
041626	000005			RESET		::CLEAR THE WORLD
041630	004710			\$ENDAD: JSR	PC,(R0)	::GO TO MONITOR
041632	000240			NOP		::SAVE ROOM
041634	000240			NOP		::FOR
041636	000240			NOP		::ACT11
041640				\$DOAGN:		
041640	000137			JMP	@(PC)+	::RETURN
041642	007660			\$RTNAD: .WORD	READY	
041644	377	377	000	\$ENULL: .BYTE	-1,-1,0	::NULL CHARACTER STRING
				.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 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
      .WORD  NN'NNN      TASK/VERIFY DESCRIPTOR
BR     ??              RETURN HERE IF NO ERROR
      NOP              RETURN HERE IF ERROR
      ERROR           ERROR DEFINED BY MODULE
    
```

: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 OR SKI ERROR
      BIT 10 = 1      RECALIBRATE DRIVE
      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

```

41 041650 017637 000000 042620  MOV    @($P),39$      ;STORE DESCRIPTOR
42 041656 062716 000006          ADD    #6,($P)        ;MOVE SP TO USERS ERROR CALL
43 041662 105076 000000          CLRB  @($P)          ;CLEAR ERROR CALL
44 041666 162716 000004          SUB    #4,($P)        ;MOVE SP TO NO ERROR RETURN
46 041672 004737 042624          JSR    PC,GETSTS     ;SETUP TO READ ALL REGISTERS
47 041676 004737 042710          JSR    PC,GET        ;GET RMER2
48 041702 000411          BR     2$           ;BR IF NO ERROR DETECTED
49 041704 000401          BR     1$           ;GET OVER ERROR NUMBER
50 041706 000000          .WORD 0             ;ERROR DEFINED BY GET SUBROUTINE
51 041710 062716 000004 1$:    ADD    #4,($P)        ;XFER ERROR TO USER AND
52 041714 113776 041706 000000  MOVB  1$-2,@($P)     ;GET ERROR NUMBER.
53 041722 000137 042610          JMP    37$
55 041726 013737 001376 042622 2$:    MOV    RMER2I,40$    ;GET RMER2 AND SAVE FOR LATER
    
```

::*****

```

58                                     ;SELECT DEVICE AND VERIFY DEVICE AVAILABLE IF BIT 15 SET IN TASK
59 041734 005737 042620                TST      39$                ;SELECT DEVICE??
60 041740 100014                        BPL      4$                ;NO!!
61
62 041742 004737 050612                JSR      PC,DEVSEL        ;GO SELECT DEVICE
63 041746 000411                        BR       4$                ;NO ERROR - CONTINUE
64 041750 000401                        BR       3$
65 041752 000000                        .WORD   0                  ;ERROR NUMBER FROM DEVSEL
66 041754 062716 000004 000000 3$:  ADD     #4,(SP)          ;TRANSFER ERROR TO USER
67 041760 113776 041752 000000        MOVB    3$,2,@(SP)
68 041766 000137 042610                JMP     37$
69
70                                     ;*****
71                                     ;CLEAR CONTROLLER IF BIT 14 IS SET IN TASK
72 041772                                4$:
73 041772 032737 040000 042620        BIT     #BIT14,39$        ;CLEAR CONTROLLER??
74 042000 001451                        BEQ     13$                ;NO!!
75
76 042002 004737 052264                JSR      PC,CNTCLR        ;GO CLEAR CONTROLLER
77 042006 000411                        BR       7$                ;CONTINUE - NO ERROR
78 042010 000401                        BR       6$
79 042012 000000                        .WORD   0                  ;ERROR NUMBER FROM CNTCLR
80 042014 062716 000004 000000 5$:  ADD     #4,(SP)          ;TRANSFER ERROR TO USER
81 042020 113776 042012 000000 6$:  MOVB    5$,@ (SP)
82 042026 000137 042610                JMP     37$
83
84                                     ;*****
85                                     ;VERIFY CONTROLLER CLEAR IF BIT6 SET IN TASK
86 042032                                7$:
87 042032 032737 000100 042620        BIT     #BIT6,39$        ;VERIFY??
88 042040 001431                        BEQ     13$                ;NO!!
89
90 042042 004737 042710                JSR      PC,GET           ;GO GET STATUS
91 042046 000411                        BR       10$              ;NO ERROR GETTING STATUS
92 042050 000401                        BR       9$
93 042052 000000                        .WORD   0                  ;ERROR FROM GETTING STATUS
94 042054 062716 000004 000000 8$:  ADD     #4,(SP)          ;TRANSFER ERROR TO USER
95 042060 113776 042052 000000 9$:  MOVB    8$,@ (SP)
96 042066 000137 042610                JMP     37$
97
98 042072 004737 052402                10$:  JSR     PC,CLRSTS        ;GO VERIFY STATUS CLEAR
99 042076 000412                        BR       13$              ;NO ERROR IN CLEAR
100 042100 000401                        BR       12$
101 042102 000000                        .WORD   0                  ;ERROR IN STATUS CLEAR
102 042104 005726                        11$:  TST     (SP)+           ;STRIP RETURN ADDRESS TO
103 042106 062716 000004 000000 12$:  ADD     #4,(SP)          ;SUBROUTINE AND TRANSFER
104 042112 113776 042102 000000        MOVB    11$,@ (SP)        ;ERROR TO USER
105 042120 000137 042610                JMP     37$
106
107                                     ;*****
108                                     ;EXECUTE PACK ACKNOWLEDGE IF BIT12 SET IN TASK AND VOLUME IS
109                                     ;NOT VALID
110 042124                                13$:
111 042124 032737 010000 042620        BIT     #BIT12,39$        ;PACK ACKNOWLEDGE??
112 042132 001501                        BEQ     25$                ;NO!!
113
114 042134 004737 042710                JSR     PC,GET

```

```

115 042140 000411 BR 16$ ;NO ERROR GETTING RMDS
116 042142 000401 BR 15$
117 042144 000000 14$: .WORD 0
118 042146 062716 000004 15$: ADD #4,(SP) ;TRANSFER ERROR TO USER
119 042152 113776 042144 000000 MOVB 14$,@ (SP)
120 042160 000137 042610 JMP 37$
121
122 042164 032737 000100 001346 16$: BIT #VV,RMDSI ;IS VOLUME VALID??
123 042172 001061 BNE 25$ ;YES!!
124
125 042174 012737 000023 001410 MOV #PAKACK!GO,RMCS10 ;LOAD PACK ACK COMMAND
126 042202 112737 000000 001543 MOVB #RMCS1,PUTINX ;SETUP REGISTER INDEX TABLE
127 042210 112737 000200 001544 MOVB #200,PUTINX+1
128 042216 004737 043160 JSR PC,PUT ;GO WRITE COMMAND
129 042222 000410 BR 19$ ;NO ERROR LOADING REGISTER
130 042224 000401 BR 18$
131 042226 000000 17$: .WORD 0 ;ERROR FROM PUT SUB
132 042230 062716 000004 18$: ADD #4,(SP) ;TRANSFER ERROR TO USER
133 042234 113776 042226 000000 MOVB 17$,@ (SP)
134 042242 000562 BR 37$
135
136 042244 004737 043522 19$: JSR PC,TIMOUT ;WAIT FOR COMMAND TO COMPLETE
137
138
139 *****
140 042250 032737 000020 042620 :VERIFY PACK ACKNOWLEDGE IF #BIT4 SET IN TASK
141 042256 001427 BIT #BIT4,39$ ;VERIFY PACK ACKNOWLEDGE??
142 BEQ 25$ ;NO!!
143 042260 004737 042710 JSR PC,GET ;GO GET STATUS
144 042264 000410 BR 22$ ;NO ERROR GETTING STATUS
145 042266 000401 BR 21$
146 042270 000000 20$: .WORD 0 ;ERROR FROM GET SUB
147 042272 062716 000004 21$: ADD #4,(SP) ;TRANSFER ERROR TO USER
148 042276 113776 042270 000000 MOVB 20$,@ (SP)
149 042304 000541 BR 37$
150
151 042306 004737 053262 22$: JSR PC,ACKSTS ;GO CHECK ACKNOWLEDGE
152 042312 000411 BR 25$ ;NO ERROR
153 042314 000401 BR 24$
154 042316 000000 23$: .WORD 0 ;PACK ACKNOWLEDGE ERROR
155 042320 005726 24$: TST (SP)+ ;STRIP RETURN TO SUB AND
156 042322 062716 000004 ADD #4,(SP) ;TRANSFER ERROR TO USER
157 042326 113776 042316 000000 MOVB 23$,@ (SP)
158 042334 000525 BR 37$
159
160 *****
161 :RECALIBRATE DRIVE IF BIT 11 IS SET IN TASK AND "SKI" IS SET
162 :OR "PIP" IS ACTIVE.
163 :RECALIBRATE DRIVE IF BIT 10 IS SET
164 042336 25$:
165 042336 032737 002000 042620 BIT #BIT10,39$ ;RECALIBRATE DRIVE ?
166 042344 001027 BNE 29$ ;YES!!
167 042346 032737 004000 042620 BIT #BIT11,39$ ;RECALIBRATE??
168 042354 001517 BEQ 38$ ;NO!!
169
170 042356 004737 042710 JSR PC,GET ;GO GET RMDS
171 042362 000410 BR 28$ ;NO ERROR GETTING RMDS

```



```

172 042364 000401
173 042366 000000
174 042370 062716 000004
175 042374 113776 042366 000000
176 042402 000502
177
178 042404 032737 040000 042622 28$: BIT #SKI,40$ ;WAS SKI SET ?
179 042412 001004 BNE 29$ ;YES, GO RECALIBRATE
180 042414 032737 020000 001346 BIT #PIP,RMDSI ;IS PIP ACTIVE??
181 042422 001474 BEQ 38$ ;NO!!
182
183 042424 005037 001444 29$: CLR RMDCO ;CLEAR CYLINDER ADDRESS
184 042430 005037 001416 CLR RMDAO ;CLEAR TRACK/SECTOR ADDRESS
185 042434 012737 000007 001410 MOV #RECAL!GO,RMCS10 ;LOAD RECALIBRATE COMMAND
186 042442 112737 000034 001543 MOVVB #RMDC,PUTINX ;MAKE CYLINDER ADDRESS AND
187 042450 112737 000006 001544 MOVVB #RMDA,PUTINX+1 ;TRK/SEC ADDRESS LOOK LIKE SEEK TO ZERO
188 042456 112737 000000 001545 MOVVB #RMCS1,PUTINX+2 ;AND REGISTER INDEX
189 042464 112737 000200 001546 MOVVB #200,PUTINX+3 ;SET TERMINATOR
190 042472 004737 043160 JSR PC,PUT ;GO ISSUE RECALIBRATE
191 042476 000410 BR 31$ ;NO ERROR
192 042500 000401 BR 30$
193 042502 000000
194 042504 062716 000004 30$: .WORD 0 ;ERROR IN REGISTER TRANSFER
195 042510 113776 042502 000000 ADD #4,(SP) ;TRANSFER ERROR TO USER
196 042516 000434 MOVVB 30$-2,@(SP)
197 BR 37$
198 042520 004737 043522 31$: JSR PC,TIMOUT ;WAIT FOR COMPLETION
199
200
201 ;*****
202 042524 032737 000010 042620 ;VERIFY RECALIBRATE IF BIT 3 SET IN TASK
203 042532 001430 BIT #BIT3,39$ ;VERIFY RECALIBRATE??
204 BEQ 38$ ;NO!!
205 042534 004737 042710 JSR PC,GET ;GO GET STATUS
206 042540 000410 BR 34$ ;NO ERROR GETTING STATUS
207 042542 000401 BR 33$
208 042544 000000
209 042546 062716 000004 32$: .WORD 0 ;ERROR FROM GET
210 042552 113776 042544 000000 33$: ADD #4,(SP) ;TRANSFER ERROR TO USER
211 042560 000413 MOVVB 32$,@(SP)
212 BR 37$
213 042562 004737 054056 34$: JSR PC,RCLSTS ;GO CHECK RECALIBRATE
214 042566 000412 BR 38$ ;NO ERROR DURING RECALIBRATE
215 042570 000401 BR 36$
216 042572 000000 35$: .WORD 0 ;ERROR DURING RECALIBRATE
217 042574 005726 36$: TST (SP)+ ;STRIP RETURN TO SUB AND
218 042576 062716 000004 ADD #4,(SP) ;TRANSFER ERROR TO USER
219 042602 113776 042572 000000 MOVVB 35$,@(SP)
220 042610 162716 000002 37$: SUB #2,(SP) ;MOVE SP BACK BEFORE ERROR
221 042614 000240 38$: NOP
222 042616 000207 RTS PC ;RETURN TO USER
223
224 042620 000000 39$: .WORD 0 ;TASK/VERIFY DESCRIPTOR
225 042622 000000 40$: .WORD 0 ;CONTAINS RMER2

```

1
2
3
4
5
6
7
8
9
10 042624
11 042624 010046
12 042630 010246
13 042632 012700 001514
14 042636 012701 001404
15 042642 012702 000046
16 042646 110220
17 042650 005041
18 042652 162702 000002
19 042656 100405
20 042660 022702 000022
21 042664 001370
22 042666 005041
23 042670 000770
24 042672 112720 000200
25 042676 012602
26 042700 012601
27 042702 012600
28 042704 000240
29 042706 000207

.SBTTL GET STATUS SUBROUTINE

:THIS SUBROUTINE SETS UP THE "GET INDEX TABLE" AND THE "GET
:BUFFER" FOR READING ALL SUBSYSTEM REGISTERS VIA THE GET SUBROUTINE
:AND THEN RETURNS TO THE USER.

:CALL: JSR PC,GETSTS
: ??? RETURN HERE

GETSTS:
MOV R0,-(SP) ;;PUSH R0 ON STACK
MOV R1,-(SP) ;;PUSH R1 ON STACK
MOV R2,-(SP) ;;PUSH R2 ON STACK
MOV #GETINX,R0 ;R0 = ADDRESS OF INDEX TABLE
MOV #RMEC2I+2,R1 ;R1 = ADDRESS OF GET BUFFER
MOV #RMEC2,R2 ;R2 = REGISTER INDE
2\$: MOV B R2,(R0)+ ;WRITE REGISTER INDEX IN TABLE
CLR -(R1) ;CLEAR CORRESPONDING LOCATION
3\$: SUB #2,R2 ;DECREMENT TO NEXT INDEX
BMI 4\$;BRANCH OUT IF DONE
CMP #RMDB,R2 ;DONT WRITE RMDB INDEX
BNE 2\$
CLR -(R1)
BR 3\$
4\$: MOV B #200,(R0)+ ;WRITE TERMINATOR
MOV (SP)+,R2 ;;POP STACK INTO R2
MOV (SP)+,R1 ;;POP STACK INTO R1
MOV (SP)+,R0 ;;POP STACK INTO R0
NOP
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

.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 'RMB4' 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
      CLRB  @ (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
      MOVB #112,@16(SP)     ;ERROR CALL
      BR   7$
3$:   TSTB  (R4)             ;DONE??
      BMI  9$              ;YES!!
      MOVB (R4),R1          ;R1 = REGISTER ADDRESS
      BIC  #^CIDXMSK,R1     ;CLEAR ANY SIGN EXTENSION
    
```

```

31 042710 000240
32 042712 062716 000004
33 042716 105076 000000
34 042722 162716 000004
35 042726 010046
   042730 010146
   042732 010246
   042734 010346
   042736 010446
   042740 013746 000004
   042744 013746 000006
36 042750 013700 001276
37 042754 012702 001334
38 042760 012704 001514
39 042764 012737 043072 000004
40 042772 012737 000300 000006
41 043000 016037 000010 001174
42 043006 016037 000000 001176
43 043014 032737 004000 001176
44 043022 001007
45 043024 062766 000004 000016
46 043032 112776 000112 000016
47 043040 000423
48 043042 105714
49 043044 100433
50 043046 111401
51 043050 042701 177700
    
```

GET SUBROUTINE

```

52 043054 060001          ADD    R0,R1
53 043056 112403          MOVB  (R4)+,R3      ;R3 = STORAGE ADDRESS FOR REGISTER
54 043060 042703 177700    BIC   #^CIDXMSK,R3 ;CLEAR ANY SIGN EXTENSION
55 043064 060203          ADD    R2,R3
56 043066 011113          MOV   (R1),(R3)    ;READ REGISTER
57 043070 000764          BR    3$
58
59 043072 022626          5$:   CMP   (SP)+,(SP)+  ;RESTORE STACK
60 043074 062766 000004 000016  ADD   #4,16(SP)    ;WRITE ERROR NUMBER IN
61 043102 112776 000007 000016  MOVB  #7,@16(SP)  ;USER'S ERROR CALL
62 043110 162766 000002 000016  7$:   SUB   #2,16(SP)
63 043116 105714          8$:   TSTB  (R4)        ;DONE CLEARING??
64 043120 100405          BMI   9$          ;YES!!
65 043122 005003          CLR   R3         ;CLEAR REMAINING STORAGE
66 043124 112403          MOVB  (R4)+,R3    ;LOCATIONS
67 043126 060203          ADD   R2,R3
68 043130 005013          CLR   (R3)
69 043132 000771          BR    8$
70 043134
70 043134 012637 0C0006  9$:   MOV   (SP)+,ERRVEC+2  ;:POP STACK INTO ERRVEC+2
70 043140 012637 000004  MOV   (SP)+,ERRVEC  ;:POP STACK INTO ERRVEC
70 043144 012604  MOV   (SP)+,R4      ;:POP STACK INTO R4
70 043146 012603  MOV   (SP)+,R3      ;:POP STACK INTO R3
70 043150 012602  MOV   (SP)+,R2      ;:POP STACK INTO R2
70 043152 012601  MOV   (SP)+,R1      ;:POP STACK INTO R1
70 043154 012600  MOV   (SP)+,R0      ;:POP STACK INTO R0
71 043156 000207          RTS   PC          ;RETURN

```

PUT SUBROUTINE

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$: MOV RMCS2(R0),$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
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
```

```
000240
010046
010146
010246
010346
010446
013746 000004
013746 000006
013700 001276
012702 001410
012704 001543
012737 043330 000004
012737 000300 000006
016037 000010 001174
016037 000000 001176
032737 004000 001176
001007
062766 000004 000016
112776 000112 000016
000424
105714
100425
111401
042701 177700
060001
111403
042703 177700
060203
011311
105724
```

```

PUT SUBROUTINE
52 043326 000763          BR      3$
53
54 043330 022626          5$:    CMP      (SP)+,(SP)+      ;ADJUST STACK
55 043332 062766 000004 000016  ADD      #4,16(SP)      ;WRITE ERROR NUMBER IN
56 043340 112776 000007 000016  MOVB    #7,@16(SP)      ;USER'S ERROR CALL
57 043346 162766 000002 000016  7$:    SUB      #2,16(SP)
58
59 043354          9$:
  043354 012637 000006  MOV      (SP)+,ERRVEC+2      ;;POP STACK INTO ERRVEC+2
  043360 012637 000004  MOV      (SP)+,ERRVEC      ;;POP STACK INTO ERRVEC
  043364 012604  MOV      (SP)+,R4          ;;POP STACK INTO R4
  043366 012603  MOV      (SP)+,R3          ;;POP STACK INTO R3
  043370 012602  MOV      (SP)+,R2          ;;POP STACK INTO R2
  043372 012601  MOV      (SP)+,R1          ;;POP STACK INTO R1
  043374 012600  MOV      (SP)+,R0          ;;POP STACK INTO R0
60 043376 000207  RTS      PC          ;RETURN
    
```

```

1          .SBTTL  SIZE CLOCK SUBROUTINE
2
3          SIZCLK:
4 043400   013746  000004   MOV      ERRVEC,-(SP)      ;;PUSH ERRVEC ON STACK
5 043404   013746  000006   MOV      ERRVEC+2,-(SP)   ;;PUSH ERRVEC+2 ON STACK
6 043410   012737  043446   000004   MOV      #1$,ERRVEC      ;SET UP FOR BUS TIMEOUT
7 043416   012737  000300   000006   MOV      #PR6,ERRVEC+2
8 043424   012737  177546   001510   MOV      #177546,CLKADR  ;LOAD ADDRESSES FOR KW11-L
9 043432   012737  000100   001512   MOV      #100,CLKVCT
10 043440   005777  136044   TST      @CLKADR         ;TEST FOR KW11-L PRESENT
11 043444   000421   BR      3$              ;YES - KW11-L IS PRESENT
12 043446   022626   1$:  CMP      (SP)+,(SP)+   ;RESTORE SP
13 043450   012737  043500   000004   MOV      #2$,ERRVEC      ;SET UP FOR BUS TIMEOUT
14 043456   012737  172540   001510   MOV      #172540,CLKADR  ;LOAD ADDRESSES FOR KW11-P CLOCK
15 043464   012737  000104   001512   MOV      #104,CLKVCT
16 043472   005777  136012   TST      @CLKADR         ;TEST FOR KW11-P PRESENT
17 043476   000404   BR      3$              ;YES - KW11-P IS PRESENT
18 043500   022626   2$:  CMP      (SP)+,(SP)+   ;RESTORE SP
19 043502   062766   000002   000004   ADD      #2,4(SP)        ;MOVE RETURN TO ERROR
20 043510   012637  000006   3$:  MOV      (SP)+,ERRVEC+2   ;;POP STACK INTO ERRVEC+2
21 043514   012637  000004   MOV      (SP)+,ERRVEC    ;;POP STACK INTO ERRVEC
22 043520   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      ;      TIMEOUT:
9
10     043522      010046      000004      000004      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
11     043524      010146      000000      000000      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
12     043526      010246      000000      000000      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
13     043530      013746      000004      000004      MOV      ERRVEC,-(SP)  ;;PUSH ERRVEC ON STACK
14     043534      013746      000006      000006      MOV      ERRVEC+2,-(SP) ;;PUSH ERRVEC+2 ON STACK
15     043540      012737      043642      000004      MOV      #4$,ERRVEC    ;SETUP FOR BUS TIMEOUT - 04 TRAP
16     043546      012737      00030C      000006      MOV      #PR6,ERRVEC+2
17     043554      013700      001276      000000      MOV      $BASE,R0      ;R0=BASE ADDRESS
18     043560      013701      001510      000000      MOV      CLKADR,R1     ;R1=CLOCK ADDRESS
19     043564      012702      000036      000000      MOV      #30.,R2      ;R2=NUMBER OF CLOCK CYCLES
20     043570      020127      172540      000000      1$:     CMP      R1,#172540    ;KW11-P CLOCK??
21     043574      001003      000000      000000      BNE     2$            ;NO!!
22     043576      012761      000001      000002      MOV      #1,2(R1)     ;SET COUNTER
23     043604      012711      000005      000000      2$:     MOV      #BIT2!BIT0,(R1) ;START COUNTER
24
25     043610      016046      000000      000000      3$:     MOV      RMCS1(R0),-(SP) ;GET STATUS
26     043614      042716      177576      000000      BIC     #^C<RDY!GO>,(SP)
27     043620      022726      000200      000000      CMP     #RDY,(SP)+    ;RDY=1,GO=0??
28     043624      001420      000000      000000      BEQ     5$            ;YES!!
29     043626      032711      000200      000000      BIT     #BIT7,(R1)    ;TIMER DONE??
30     043632      001766      000000      000000      BEQ     3$            ;NO!!
31     043634      005302      000000      000000      DEC     R2            ;DEC NUMBER OF CYCLES
32     043636      001354      000000      000000      BNE     1$            ;CONTINUE IF NOT DONE
33     043640      000412      000000      000000      BR      5$            ;'RDY' DID NOT SET OR 'GO' DID NOT RESET
34
35     043642      022626      000004      000012      4$:     CMP     (SP)+,(SP)+    ;WITHIN 500 MSEC AFTER THE COMMAND WAS ISSUED.
36     043644      062766      000004      000012      ADD     #4,12(SP)     ;ADJUST STACK
37     043652      112776      000007      000012      MOV     #7,@12(SP)   ;MOVE SP TO USER'S CALL
38     043660      162766      000002      000012      SUB     #2,12(SP)    ;WRITE ERROR NUMBER
39
40     043666      012637      000006      000004      5$:     MOV     (SP)+,ERRVEC+2 ;;POP STACK INTO ERRVEC+2
41     043672      012637      000004      000004      MOV     (SP)+,ERRVEC  ;;POP STACK INTO ERRVEC
42     043676      012602      000000      000000      MOV     (SP)+,R2     ;;POP STACK INTO R2
43     043700      012601      000000      000000      MOV     (SP)+,R1     ;;POP STACK INTO R1
44     043702      012600      000000      000000      MOV     (SP)+,R0     ;;POP STACK INTO R0
45     043704      000207      000000      000000      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
:      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

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
:YES!!
BEQ    1$
ADD    #4,(SP)
MOVB   #1,@(SP)       :ERROR 1

```

```

41 043706
42
43
44 043706 062716 000004
45 043712 105076 000000
46 043716 162716 000004
47
48
49 043722 013737 001344 001142
50 043730 042737 177770 001142
51 043736 013737 001234 001140
52 043744 042737 177770 001140
53 043752 123737 001140 001142
54
55 043760 001415
56 043762 062716 000004
57 043766 112776 000001 000000

```

```

58 043774 162716 000002          SUB      #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
59 044000 004736                JSR      PC,@(SP)+        ;REPORT WRONG UNIT SELECTED
60 044002 162716 000010          SUB      #10,(SP)         ;RESTORE (SP)
61 044006 000240                NOP
62 044010 000137 044530          JMP      10$              ;SKIP OTHER CHECKS
63 044014
64
65                                ;REPORT AN ERROR IF THE DEVICE IS NOT AVAILABLE OR IF
66                                ;THE DEVICE IS NONEXISTANT
67 044014 032737 004000 001334    BIT      #DVA,RMCS1I      ;DEVICE AVAILABLE??
68 044022 001045                BNE      5$                ;YES!!
69 044024 013737 001334 001140    MOV      RMCS1I,$GDDAT    ;EXPECTED STATUS
70 044032 052737 004000 001140    BIS      #DVA,$GDDAT
71 044040 013737 001334 001142    MOV      RMCS1I,$BDDAT    ;RECEIVED STATUS
72 044046 062716 000004                ADD      #4,(SP)
73 044052 112776 000002 000000    MOV      #2,@(SP)         ;ERROR #2
74 044060 032737 010000 001344    BIT      #NED,RMCS2I      ;WAS NED SET??
75 044066 001414                BEQ      2$                ;NO!!
76 044070 013737 001344 001140    MOV      RMCS2I,$GDDAT    ;EXPECTED STATUS
77 044076 013737 001344 001142    MOV      RMCS2I,$BDDAT    ;RECEIVED STATUS
78 044104 042737 010000 001140    BIC      #NED,$GDDAT
79 044112 112776 000003 000000    MOV      #3,@(SP)         ;YES - CHANGE ERROR NUMBER
80 044120 162716 000002          SUB      #2,(SP)         ;MOVE SP TO RETURN FOR ERROR
81 044124 004736                JSR      PC,@(SP)+        ;REPORT DEVICE NOT AVAILABLE
82 044126 162716 000010          SUB      #10,(SP)         ;RESTORE (SP)
83 044132 000240                NOP
84 044134 000575                BR       10$              ;SKIP OTHER CHECKS
85 044136
86
87                                ;REPORT AN ERROR IF MASSBUS CONTROLLER IS NOT READY
88 044136 032737 000200 001334    BIT      #RDY,RMCS1I      ;CONTROLLER READY??
89 044144 001030                BNE      7$                ;YES!!
90 044146 013737 001334 001140    MOV      RMCS1I,$GDDAT    ;EXPECTED STATUS
91 044154 052737 000200 001140    BIS      #RDY,$GDDAT
92 044162 042737 160001 001140    BIC      #SC!TRE!MCPE!GO,$GDDAT
93 044170 013737 001334 001142    MOV      RMCS1I,$BDDAT    ;RECEIVED STATUS
94 044176 062716 000004                ADD      #4,(SP)
95 044202 112776 000004 000000    MOV      #4,@(SP)         ;ERROR #4
96 044210 162716 000002          SUB      #2,(SP)         ;MOVE SP TO RETURN FOR ERROR
97 044214 004736                JSR      PC,@(SP)+        ;REPORT CONTROLLER NOT READY
98 044216 162716 000010          SUB      #10,(SP)         ;RESTORE (SP)
99 044222 000240                NOP
100 044224 000541                BR       10$              ;SKIP OTHER CHECKS
101 044226
102
103                                ;REPORT AN ERROR IF GO IS NOT ZERO AND DRY IS NOT ONE
104 044226 032737 000001 001334    BIT      #GO,RMCS1I       ;GO RESET??
105 044234 001431                BEQ      8$                ;YES!!
106 044236 032737 000200 001346    BIT      #DRY,RMDSI       ;DRIVE READY??
107 044244 001025                BNE      8$                ;YES!!
108 044246 013737 001334 001140    MOV      RMCS1I,$GDDAT    ;EXPECTED STATUS
109 044254 042737 160001 001140    BIC      #SC!TRE!MCPE!GO,$GDDAT
110 044262 013737 001334 001142    MOV      RMCS1I,$BDDAT    ;RECEIVED STATUS
111 044270 062716 000004                ADD      #4,(SP)
112 044274 112776 000005 000000    MOV      #5,@(SP)         ;ERROR #5
113 044302 162716 000002          SUB      #2,(SP)         ;MOVE SP TO RETURN FOR ERROR
114 044306 004736                JSR      PC,@(SP)+        ;REPORT DRIVE NOT READY
    
```

```

115 044310 162716 000010          SUB    #10,(SP)          ;RESTORE (SP)
116 044314 000240          NOP
117 044316 000504          BR     10$
118 044320          8$:
119
120          ;REPORT AN ERROR IF THE RH CONTROLLER DETECTED BAD
121          ;PARITY ON THE MASSBUS CONTROL BUS
122 044320 032737 020000 001334    BIT    #MCPE,RMCS1I    ;PARITY ERROR ??
123 044326 001425          BEQ    9$              ;NO!!
124 044330 013737 001334 001140    MOV    RMCS1I,$GDDAT   ;EXPECTED STATUS
125 044336 042737 160001 001140    BIC    #SC!TRÉ!MCPE!GO,$GDDAT
126 044344 013737 001334 001142    MOV    RMCS1I,$BDDAT   ;RECEIVED STATUS
127 044352 062716 000004          ADD    #4,(SP)         ;MOVE STACK TO USER'S ERROR
128 044356 112776 000013 000000    MOVB   #13,@(SP)      ;ERROR #47
129 044364 162716 000002          SUB    #2,(SP)         ;MOVE SP TO RETURN FOR ERROR
130 044370 004736          JSR    PC,@(SP)+      ;REPORT ERROR VIA USER
131 044372 162716 000010          SUB    #10,(SP)       ;RESTORE STACK
132 044376 000240          NOP
133 044400 000453          BR     10$
134 044402          9$:
135
136          ;REPORT AN ERROR IF DETECTED A CONTROL BUS PARITY ERROR
137 044402 032737 000010 001350    BIT    #PAR,RMER1I    ;WAS THERE A PARITY ERROR??
138 044410 001451          BEQ    11$            ;NO!!
139 044412 032737 000010 001376    BIT    #DPE,RMER2I    ;WAS IT THE CONTROL BUS??
140 044420 001045          BNE    11$            ;NOT SURE!!
141 044422 032737 000010 001424    BIT    #PAR,RMER1O    ;DID TEST SET PAR ??
142 044430 001413          BEQ    93$           ;NO!!
143 044432 010046          MOV    R0,-(SP)       ;:PUSH R0 ON STACK
144 044434 012700 001543          MOV    #PUTINX,R0     ;R0 POINTS TO INDEX TABLE
145 044440 122710 000014          91$: CMPB   #RMER1,(R0)   ;SEARCH TABLE FOR RMER1
146 044444 001002          BNE    92$           ;:POP STACK INTO R0
147 044446 012600          MOV    (SP)+,R0       ;PAR WAS SET BY TEST
148 044450 000431          BR     11$           ;END OF TABLE??
149 044452 105720          92$: TSTB   (R0)+         ;NO!!
150 044454 100371          BPL    91$           ;:POP STACK INTO R0
151 044456 012600          MOV    (SP)+,R0       ;EXPECTED STATUS
152 044460 013737 001350 001140    93$: MOV    RMER1I,$GDDAT
153 044466 042737 000010 001140    BIC    #PAR,$GDDAT
154 044474 013737 001350 001142    MOV    RMER1I,$BDDAT   ;RECEIVED STATUS
155 044502 062716 000004          ADD    #4,(SP) ;MOVE SP TO USER'S ERROR CALL
156 044506 112776 000050 000000    MOVB   #50,@(SP)     ;WRITE THE ERROR NUMBER
157 044514 162716 000002          SUB    #2,(SP)         ;MOVE SP TO RETURN FOR ERROR
158 044520 004736          JSR    PC,@(SP)+      ;REPORT THE ERROR
159 044522 162716 000010          SUB    #10,(SP)       ;MOVE SP TO NO ERROR RETURN
160 044526 000240          NOP
161 044530 062716 000010          10$: ADD    #10,(SP)
162 044534 000240          11$: NOP
163 044536 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
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
      ERROR          RETURN HERE TO REPORT AN ERROR
      JSR    PC,@(SP)+   ERROR NUMBER DEFINED BY SUB
      ???          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.

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$: BIT    #GO,RMCS1I    :GO BIT RESET??
    BEQ    10$          :YES!!
    MOV    RMCS1I,$BDDAT :BAD DATA FOR TYPEOUT
    BIC    #^CGO,$BDDAT
    CLR    $GDDAT       :GOOD DATA FOR TYPEOUT
    ADD    #4,(SP)
    MOVB   #11,@(SP)    :ERROR NUMBER
```

```
044540
044540 013737 001410 050374
044546 042737 177701 050374
044554 062716 000004
044560 105076 000000
044564 162716 000004
044570 032737 000200 001346
044576 001024
044600 013737 001346 001142
044606 042737 177577 001142
044614 012737 000200 001140
044622 062716 000004
044626 112776 000010 000000
044634 162716 000002
044640 004736
044642 162716 000010
044646 000240
044650 032737 000001 001334
044656 001423
044660 013737 001334 001142
044666 042737 177776 001142
044674 005037 001140
044700 062716 000004
044704 112776 000011 000000
```

```

58 044712 162716 000002          SUB    #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
59 044716 004736                JSR    PC,@(SP)+      ;REPORT DEVICE NOT AVAILABLE
60 044720 162716 000010          SUB    #10,(SP)       ;RESTORE (SP)
61 044724 000240                NOP
62
63                                ;REPORT ERROR IF FUNCTION CODE READ FROM DEVICE IS NOT CORRECT
64 044726 013737 001334 001142 10$:  MOV    RMCS1I,$BDDAT ;IS FUNCTION CODE CORRECT??
65 044734 042737 177701 001142    BIC    #^C76,$BDDAT
66 044742 013737 050374 001140    MOV    515$,$GDDAT   ;EXPECTED FUNCTION CODE
67 044750 023737 001142 001140    CMP    $BDDAT,$GDDAT
68 044756 001413                BEQ    15$            ;YES!!
69 044760 062716 000004          ADD    #4,(SP)
70 044764 112776 000012 000000    MOVB  #12,@(SP)      ;ERROR NUMBER
71 044772 162716 000002          SUB    #2,(SP)       ;MOVE SP TO RETURN FOR ERROR
72 044776 004736                JSR    PC,@(SP)+      ;REPORT WRONG FUNCTION CODE
73 045000 162716 000010          SUB    #10,(SP)       ;RESTORE (SP)
74 045004 000240                NOP
75 045006
76                                15$:
77                                ;REPORT AN ERROR IF COMPOSITE ERROR IS SET AND NO OTHER
78                                ;ERRORS ARE SET, OR IF COMPOSITE ERROR IS NOT SET AND
79                                ;OTHER ERRORS ARE SET
80 045012 005037 001140          CLR    $GDDAT        ;EXPECT 'ERR' = 0
81 045016 001003 001350          TST    RMER1I        ;IS RMER1 = 0??
82 045020 005737 001376          BNE    20$           ;NO!!
83 045024 001403                TST    RMER2I        ;IS RMERZ = 0??
84 045026 052737 040000 001140 20$:  BEQ    25$           ;YES!!
85 045034 013737 001346 001142 25$:  BIS    #ERR,$GDDAT   ;'ERR' SOULD BE SET
86 045042 042737 137777 001142    MOV    RMDSI,$BDDAT
87 045050 023737 001140 001142    BIC    #^CERR,$BDDAT
88 045056 001412                CMP    $GDDAT,$BDDAT ;IS 'ERR' OK??
89 045060 062716 000004          BEQ    30$           ;YES!!
90 045064 112776 000047 000000    ADD    #4,(SP)       ;MOVE SP TO USER'S ERROR
91 045072 162716 000002          MOVB  #47,@(SP)      ;WRITE ERROR NUMBER
92 045076 004736                SUB    #2,(SP)       ;MOVE SP TO ERROR RETURN
93 045100 162716 000010          JSR    PC,@(SP)+      ;REPORT INVALID COMP ERROR
94                                SUB    #10,(SP)
95                                ;REPORT AN ERROR IF 'TRE' IS SET AND NONE OF THE BITS WHICH SET
96                                ;TRE IS SET, OR IF TRE IS NOT SET AND ONE OR MORE BITS WHICH
97                                ;SET TRE IS SET
98 045104 005037 001140          CLR    $GDDAT        ;EXPECT 'TRE' = 0
99 045110 013746 001344          MOV    RMCS2I,-(SP)  ;WAS DLT, WCE, UPE, NED, NEM
100 045114 042726 000377          BIC    #377,(SP)+    ;PGE, MXF OR MDPE SET
101 045120 001010                BNE    35$           ;YES!!
102 045122 032737 040000 001346    BIT    #ERR,RMDSI    ;WAS EXCEPTION RECEIVED??
103 045130 001407                BEQ    40$           ;NO!!
104 045132 022737 000030 050374    CMP    #SEARCH,515$ ;WAS DATA TRANSFERRED??
105 045140 103003                BHIS  40$            ;NO!!
106 045142 052737 040000 001140 35$:  BIS    #TRE,$GDDAT   ;'TRE' SHOULD BE SET
107 045150 013737 001334 001142 40$:  MOV    RMCS1I,$BDDAT ;BAD DATA FOR TYPEOUT
108 045156 042737 137777 001142    BIC    #^CTRE,$BDDAT
109 045164 023737 001140 001142    CMP    $GDDAT,$BDDAT ;IS 'TRE' OK??
110 045172 001413                BEQ    45$           ;YES!!
111 045174 062716 000004          ADD    #4,(SP)       ;MOVE SP TO USER'S ERROR CALL
112 045200 112776 000014 000000    MOVB  #14,@(SP)      ;WRITE ERROR NUMBER
113 045206 162716 000002          SUB    #2,(SP)       ;MOVE SP TO RETURN FOR ERROR
114 045212 004736                JSR    PC,@(SP)+      ;REPORT TRE ERROR
    
```

```

115 045214 162716 000010          SUB    #10,(SP)      ;RESTORE (SP)
116 045220 000240          NOP
117 045222          45$:
118
119          ;*****
120          ;USING THE FUNCTION CODE TABLE, CHECK FOR THE FOLLOWING ERRORS:
121          ;      .STATUS BITS NOT SET THAT SHOULD BE SET, E.G., ATA AND ILF
122          ;      .STATUS BITS SET THAT SHOULD NOT BE SET, E.G., WCE AND ECH
123          ;NOTE THAT SOME ERROR BITS ARE CONDITIONAL ON THE COMMAND AND OTHER
124          ;STATUS CONDITIONS, E.G., WRITE LOCK ERROR SHOULD ONLY BE SET IF
125          ;WRITE LOCK IS ON AND THE COMMAND IS A WRITE.
126
127          ;GET AND STORE THE ENTRY FROM THE FUNCTION CODE TABLE
128 045222 010046          MOV    R0,-(SP)      ;:PUSH R0 ON STACK
129 045224 013700 050374      MOV    515$,R0      ;:GET FUNCTION CODE
130 045230 016037 067136 050366  MOV    FNCDTB(R0),500$ ;:STORE ENTRY
131 045236 012600          MOV    (SP)+,R0     ;:POP STACK INTO R0
132
133          ;REPORT AN ERROR IF AN UNEXPECTED ATTENTION OCCURRED OR IF
134          ;ATA IS NOT SET AND SHOULD BE SET.
135 045240 013737 050366 001140  MOV    500$, $GDDAT ;:GET EXPECTED ATA STATUS
136 045246 032737 040000 001346  BIT    #ERR,RMSI    ;:IS COMPOSITE ERROR SET ??
137 045254 001403          BEQ    50$         ;:NO !!
138 045256 052737 100000 001140  BIS    #ATA,$GDDAT ;:EXPECT AN ATTENTION
139 045264 042737 077777 001140 50$: BIC    #^CATA,$GDDAT ;:STRIP DONT CARES
140 045272 013737 001346 001142  MOV    RMSI,$BDDAT ;:GET RECEIVED ATA
141 045300 042737 077777 001142  BIC    #^CATA,$BDDAT ;:STRIP DONT CARES
142 045306 023737 001140 001142  CMP    $GDDAT,$BDDAT ;:IS ATA OK ??
143 045314 001413          BEQ    55$         ;:YES !!
144 045316 062716 000004          ADD    #4,(SP)     ;:MOVE SP TO USERS ERROR CALL
145 045322 112776 000006 000000  MOVB   #6,@(SP)    ;:LOAD ERROR # IN CALL
146 045330 162716 000002          SUB    #2,(SP)     ;:MOVE SP TO ERROR RETURN
147 045334 004736          JSR    PC,@(SP)+   ;:REPORT ERROR
148 045336 162716 000010          SUB    #10,(SP)   ;:RESTORE SP
149 045342 000240          NOP
150 045344          55$:
151
152          ;REPORT ERROR IF ILF IS INCORRECT, I.E., IF ILF DOES NOT COMPARE
153          ;WITH FUNCTION CODE TABLE
154 045344 013737 050366 001140  MOV    500$, $GDDAT ;:GET EXPECTED ILF
155 045352 042737 177776 001140  BIC    #^CILF,$GDDAT ;:CLEAR ALL OTHER BITS
156 045360 013737 001350 001142  MOV    RMER1I,$BDDAT ;:GET RECEIVED ILF
157 045366 042737 177776 001142  BIC    #^CILF,$BDDAT ;:CLEAR ALL OTHER BITS
158 045374 023737 001140 001142  CMP    $GDDAT,$BDDAT ;:IS ILF OK ??
159 045402 001412          BEQ    60$         ;:YES !!
160 045404 062716 000004          ADD    #4,(SP)     ;:MOVE SP TO USERS ERROR CALL
161 045410 112776 000254 000000  MOVB   #254,@(SP)  ;:WRITE ERROR NUMBER IN CALL
162 045416 162716 000002          SUB    #2,(SP)     ;:MOVE SP TO ERROR RETURN
163 045422 004736          JSR    PC,@(SP)+   ;:REPORT ERROR AND RETURN
164 045424 162716 000010          SUB    #10,(SP)   ;:MOVE SP TO NO ERROR
165 045430 005037 001140 60$: CLR    $GDDAT      ;:CLEAR EXPECTED STATUS
166
167          ;REPORT AN ERROR IF WCE IS SET AND SHOULD NOT BE SET
168 045434 013746 050366          MOV    500$,-(SP) ;:GET WCE STATUS ENABLE
169 045440 052716 137777          BIS    #^CWCE,(SP) ;:SET ALL OTHER BITS
170 045444 013737 001344 001142  MOV    RMCS2I,$BDDAT ;:RECEIVED STATUS
171 045452 042637 001142          BIC    (SP)+,$BDDAT ;:CLEAR WCE IF ENABLED
    
```

```

172 045456 001412          BEQ      90$          ;BRANCH IF WCE OK
173 045460 062716 000004    ADD      #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
174 045464 112776 000026 000000    MOVB    #26,@(SP)    ;WRITE ERROR NUMBER
175 045472 162716 000002    SUB      #2,(SP)      ;MOVE SP TO ERROR RETURN
176 045476 004736          JSR      PC,@(SP)+    ;REPORT ERROR
177 045500 162716 000010    SUB      #10,(SP)     ;RESTORE ERROR
178 045504          90$:
179
180          ;REPORT ERROR IF OPI STATUS IS SET AND SHOULD NOT BE SET
181 045504 013746 050366    MOV      500$,-(SP)   ;GET OPI STATUS ENABLE
182 045510 052716 157777    BIS      #^COPI,(SP)  ;SET ALL OTHER BITS
183 045514 013737 001350 001142    MOV      RMER1I,$BDDAT ;GET RECEIVED STATUS
184 045522 042637 001142    BIC      (SP)+,$BDDAT ;CLEAR OPI IF ENABLED
185 045526 001412          BEQ      100$         ;BRANCH IF OPI OK
186 045530 062716 000004    ADD      #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
187 045534 112776 000164 000000    MOVB    #164,@(SP)   ;WRITE ERROR NUMBER IN CALL
188 045542 162716 000002    SUB      #2,(SP)      ;MOVE SP TO ERROR RETURN
189 045546 004736          JSR      PC,@(SP)+    ;REPORT ERROR
190 045550 162716 000010    SUB      #10,(SP)     ;RESTORE SP
191 045554          100$:
192
193          ;REPORT ERROR IF IVC IS SET AND IS NOT ENABLED OR IF IVC IS
194          ;SET AND VV IS NOT RESET
195 045554 013746 050366    MOV      500$,-(SP)   ;GET IVC STATUS ENABLE
196 045560 032737 000100 001346    BIT      #VV,RMDSI    ;IS VV SET
197 045566 001402          BEQ      105$         ;NO !!
198 045570 042716 010000    BIC      #IVC,(SP)    ;YES - IVC SHOULD BE 0
199 045574 052716 167777    BIS      #^CIVC,(SP)  ;SET ALL OTHER BITS
200 045600 013737 001376 001142    MOV      RMER2I,$BDDAT ;GET RECEIVED STATUS
201 045606 042637 001142    BIC      (SP)+,$BDDAT ;CLEAR IVC IF ENABLED
202 045612 001412          BEQ      110$         ;BRANCH IF IVC OK
203 045614 062716 000004    ADD      #4,(SP)      ;MOVE SP TO USERS ERROR CALL
204 045620 112776 000165 000000    MOVB    #165,@(SP)   ;WRITE ERROR NUMBER IN CALL
205 045626 162716 000002    SUB      #2,(SP)      ;MOVE SP TO ERROR RETURN
206 045632 004736          JSR      PC,@(SP)+    ;REPORT ERROR
207 045634 162716 000010    SUB      #10,(SP)     ;RESTORE SP TO NO ERROR
208 045640          110$:
209
210          ;BIT 11 (WLE) OF THE FUNCTION CODE TABLE IS THE ENABLING BIT FOR
211          ; ALL WRITE ERRORS, I.E.,
212          ; RMER1 - WLE, WCF
213          ; RMER2 - DPE
214          ; RMCS2 - UPE.
215          ;EACH OF THESE ERRORS IS CHECKED TO SEE IF AN ERROR IS SET WHEN THE
216          ;WRITE ERROR ENABLE BIT IS RESET.
217
218          ;REPORT AN ERROR IF WLE IS SET AND WRITE ERRORS ARE NOT ENABLED, OR IF
219          ;THE DRIVE IS NOT WRITE PROTECTED
220 045640 012746 177777    MOV      #-1,-(SP)    ;ASSUME WRITE ERRORS ENABLED
221 045644 032737 004000 050366    BIT      #WLE,500$    ;ARE WRITE ERRORS ENABLED ??
222 045652 001404          BEQ      115$         ;NO !!
223 045654 032737 004000 001346    BIT      #WRL,RMDSI    ;IS THE DRIVE WRITE PROTECTED ??
224 045662 001002          BNE      120$         ;YES !!
225 045664 042716 004000 115$:    BIC      #WLE,(SP)    ;RESET WLE ENABLE
226 045670 013737 001350 001142 120$:    MOV      RMER1I,$BDDAT ;GET RECEIVED STATUS
227 045676 042637 001142    BIC      (SP)+,$BDDAT ;CLEAR WLE IF ENABLED
228 045702 001412          BEQ      125$         ;BRANCH IF WLE OK
    
```

```

229 045704 062716 000004      ADD      #4,(SP)      ;MOVE SP TO USERS ERROR CALL
230 045710 112776 000023 000000      MOV      #23,@(SP)   ;WRITE ERROR NUMBER IN CALL
231 045716 162716 000002      SUB      #2,(SP)     ;MOVE SP TO ERROR RETURN
232 045722 004736          JSR      PC,@(SP)+   ;REPORT ERROR AND RETURN
233 045724 162716 000010      SUB      #10,(SP)    ;RESTORE SP TO NO ERROR
234 045730          125$:
235
236          ;REPORT ERROR IF WCF IS SET AND WRITE ERRORS ARE NOT ENABLED
237 045730 012746 177777          MOV      #-1,-(SP)   ;ASSUME WRITE ERRORS ENABLED
238 045734 032737 004000 050366      BIT      #WLE,500$   ;ARE WRITE ERRORS ENABLED ??
239 045742 001002          BNE      130$        ;YES !!
240 045744 042716 000040          BIC      #WCF,(SP)   ;DISABLE WCF ERROR
241 045750 013737 001350 001142 130$:      MOV      RMER1I,$BDDAT ;GET RECEIVED STATUS
242 045756 042637 001142          BIC      (SP)+,$BDDAT ;RESET WCF IF ENABLED
243 045762 001412          BEQ      135$        ;BRANCH IF WCF OK
244 045764 062716 000004      ADD      #4,(SP)     ;MOVE SP TO USERS ERROR CALL
245 045770 112776 000025 000000      MOV      #25,@(SP)   ;WRITE ERROR NUMBER IN CALL
246 045776 162716 000002      SUB      #2,(SP)     ;MOVE SP TO ERROR RETURN
247 046002 004736          JSR      PC,@(SP)+   ;REPORT ERROR
248 046004 162716 000010      SUB      #10,(SP)    ;RESTORE SP TO NO ERROR
249 046010          135$:
250
251          ;REPORT ERROR IF DPE IS SET AND WRITE ERRORS ARE NOT ENABLED
252 046010 012746 177777          MOV      #-1,-(SP)   ;ASSUME WRITE ERRORS ARE ENABLED
253 046014 032737 004000 050366      BIT      #WLE,500$   ;ARE WRITE ERRORS ENABLED ??
254 046022 001002          BNE      140$        ;YES !!
255 046024 042716 000010          BIC      #DPE,(SP)   ;RESET DPE ENABLE
256 046030 013737 001376 001142 140$:      MOV      RMER2I,$BDDAT ;GET RECEIVED STATUS
257 046036 042637 001142          BIC      (SP)+,$BDDAT ;RESET DPE IF ENABLED
258 046042 001412          BEQ      145$        ;BRANCH IF DPE OK
259 046044 062716 000004      ADD      #4,(SP)     ;MOVE SP TO USERS ERROR CALL
260 046050 112776 000040 000000      MOV      #40,@(SP)   ;WRITE ERROR NUMBER IN CALL
261 046056 162716 000002      SUB      #2,(SP)     ;MOVE SP TO ERROR RETURN
262 046062 004736          JSR      PC,@(SP)+   ;REPORT ERROR
263 046064 162716 000010      SUB      #10,(SP)    ;RESTORE SP TO NO ERROR
264 046070          145$:
265
266          ;REPORT AN ERROR IF UPE IS SET AND WRITE ERRORS ARE NOT ENABLED
267 046070 012746 177777          MOV      #-1,-(SP)   ;ASSUME WRITE ERRORS ARE ENABLED
268 046074 032737 004000 050366      BIT      #WLE,500$   ;ARE WRITE ERRORS ENABLED ??
269 046102 001002          BNE      150$        ;YES !!
270 046104 042716 020000          BIC      #UPE,(SP)   ;DISABLE UPE ERROR
271 046110 013737 001344 001142 150$:      MOV      RMCS2I,$BDDAT ;GET RECEIVED STATUS
272 046116 042637 001142          BIC      (SP)+,$BDDAT ;RESET UPE IF ENABLED
273 046122 001412          BEQ      155$        ;BRANCH IF UPE OK
274 046124 062716 000004      ADD      #4,(SP)     ;MOVE SP TO USERS ERROR CALL
275 046130 112776 000024 000000      MOV      #24,@(SP)   ;WRITE ERROR NUMBER IN CALL
276 046136 162716 000002      SUB      #2,(SP)     ;MOVE SP TO ERROR RETURN
277 046142 004736          JSR      PC,@(SP)+   ;REPORT ERROR AND RETURN
278 046144 162716 000010      SUB      #10,(SP)    ;MOVE SP TO NO ERROR
279 046150          155$:
280
281          ;REPORT AN ERROR IF IAE IS SET AND IS NOT ENABLED
282 046150 013746 050366          MOV      500$,-(SP)  ;GET IAE ENABLE
283 046154 052716 175777          BIS      #^IAE,(SP)  ;SET ALL OTHER BITS
284 046160 013737 001350 001142          MOV      RMER1I,$BDDAT ;GET RECEIVED STATUS
285 046166 042637 001142          BIC      (SP)+,$BDDAT ;CLEAR IAE IF ENABLED
    
```



```

286 046172 001412          BEQ      160$          ;BRANCH IF IAE IS OK
287 046174 062716 000004  ADD      #4,(SP)        ;MOVE SP TO USERS ERROR CALL
288 046200 112776 000166 000000  MOVVB   #166,@(SP)     ;WRITE ERROR NUMBER
289 046206 162716 000002          SUB      #2,(SP)        ;MOVE SP TO ERROR RETURN
290 046212 004736          JSR     PC,@(SP)+      ;REPORT ERROR AND RETURN
291 046214 162716 000010  SUB      #10,(SP)      ;MOVE SP TO NO ERROR
292 046220          160$:
293
294          ;BIT 09 (AOE) OF THE FUNCTION CODE TABLE IS THE ENABLING BIT FOR
295          ; ALL READ/WRITE ERRORS, I.E.,
296          ;
297          ; RMCS1 - TRE
298          ; RMCS2 - DLT,NEM,MXF
299          ; RMDS - LBT
300          ; RMER1 - AOE
301          ;NOTE:
302          ; LBT IS NOT CHECKED BECAUSE IT ONLY RESETS WHEN THE DESIRED
303          ;CYLINDER REGISTER IS WRITTEN
304          ;NOTE:
305          ; AOE CANNOT BE SET IF LBT IS NOT ALSO SET
306          ;NOTE:
307          ; TRE IS CHECKED AS A FUNCTION OF OTHER ERROR CONDITONS ABOVE
308
309          ;REPORT AN ERROR IF DLT IS SET AND READ/WRITE ERRORS ARE NOT ENABLED
310 046220 012746 177777          MOV      #-1,-(SP)     ;ASSUME ERRORS ARE ENABLED
311 046224 032737 001000 050366  BIT      #AOE,500$    ;ARE ERRORS ENABLED ??
312 046232 001002          BNE     165$          ;YES !!
313 046234 042716 100000          BIC     #DLT,(SP)     ;RESET DLT ENABLE
314 046240 013737 001344 001142 165$: MOV      RMCS2I,$BDDAT ;GET RECEIVED STATUS
315 046246 042637 001142          BIC     (SP)+,$BDDAT ;CLEAR DLT IF ENABLED
316 046252 001412          BEQ     170$          ;BRANCH IF DLT IS OK
317 046254 062716 000004  ADD      #4,(SP)        ;MOVE SP TO USERS ERROR CALL
318 046260 112776 000032 000000  MOVVB   #32,@(SP)     ;WRITE ERROR NUMBER IN CALL
319 046266 162716 000002          SUB      #2,(SP)        ;MOVE SP TO ERROR RETURN
320 046272 004736          JSR     PC,@(SP)+      ;REPORT ERROR AND RETURN
321 046274 162716 000010  SUB      #10,(SP)      ;MOVE SP TO NO ERROR
322 046300          170$:
323
324          ;REPORT ERROR IF NEM IS SET AND READ/WRITE ERRORS ARE NOT ENABLED
325 046300 012746 177777          MOV      #-1,-(SP)     ;ASSUME ERRORS ARE ENABLED
326 046304 032737 001000 050366  BIT      #AOE,500$    ;ARE ERRORS ENABLED ??
327 046312 001002          BNE     175$          ;YES !!
328 046314 042716 004000          BIC     #NEM,(SP)     ;DISABLE NEM
329 046320 013737 001344 001142 175$: MOV      RMCS2I,$BDDAT ;GET RECEIVED STATUS
330 046326 042637 001142          BIC     (SP)+,$BDDAT ;CLEAR NEM IF ENABLED
331 046332 001412          BEQ     180$          ;BRANCH IF NEM IS OK
332 046334 062716 000004  ADD      #4,(SP)        ;MOVE SP TO USERS ERROR CALL
333 046340 112776 000167 000000  MOVVB   #167,@(SP)     ;WRITE ERROR NUMBER IN CALL
334 046346 162716 000002          SUB      #2,(SP)        ;MOVE SP TO ERROR RETURN
335 046352 004736          JSR     PC,@(SP)+      ;REPORT ERROR AND RETURN
336 046354 162716 000010  SUB      #10,(SP)      ;MOVE SP TO NO ERROR
337 046360          180$:
338
339          ;REPORT ERROR IF MXF IS SET AND READ/WRITE ERRORS ARE NOT ENABLED
340 046360 012746 177777          MOV      #-1,-(SP)     ;ASSUME ERRORS ARE ENABLED
341 046364 032737 001000 050366  BIT      #AOE,500$    ;ARE DATA ERRORS ENABLED ??
342 046372 001002          BNE     185$          ;YES !!
    
```

```

343 046374 042716 001000          BIC      #MXF,(SP)      ;DISABLE MXF ERROR
344 046400 013737 001344 001142 185$: MOV      RMCS2I,$BDDAT  ;GET RECEIVED STATUS
345 046406 042637 001142          BIC      (SP)+,$BDDAT  ;CLEAR MXF IF ENABLED
346 046412 001412          BEQ      190$          ;BRANCH IF MXF IS OK
347 046414 062716 000004          ADD      #4,(SP)       ;MOVE SP TO USERS ERROR CALL
348 046420 112776 000033 000000  MOVB     #33,@(SP)     ;WRITE ERROR NUMBER IN CALL
349 046426 162716 000002          SUB      #2,(SP)       ;MOVE SP TO ERROR RETURN
350 046432 004736          JSR      PC,@(SP)+    ;REPORT ERROR AND RETURN
351 046434 162716 000010          SUB      #10,(SP)     ;MOVE SP TO NO ERROR
352 046440          190$:
353
354          ;REPORT ERROR IF AOE IS SET AND DATA ERRORS ARE NOT ENABLED
355 046440 012746 177777          MOV      #-1,-(SP)    ;ASSUME DATA ERRORS ARE ENABLED
356 046444 032737 001000 050366  BIT      #AOE,500$    ;ARE DATA ERRORS EANBLED ??
357 046452 001404          BEQ      191$          ;NO !!
358 046454 032737 002000 001346  BIT      #LBT,RMDSI   ;IS LBT ALSO SET ??
359 046462 001002          BNE      195$          ;YES !!
360 046464 042716 001000 191$: BIC      #AOE,(SP)    ;DISABLE AOE
361 046470 013737 001350 001142 195$: MOV      RMER1I,$BDDAT ;GET RECEIVED STATUS
362 046476 042637 001142          BIC      (SP)+,$BDDAT ;CLEAR AOE IF ENABLED
363 046502 001412          BEQ      200$          ;BRANCH IF AOE IS OK
364 046504 062716 000004          ADD      #4,(SP)       ;MOVE SP TO USERS ERROR CALL
365 046510 112776 000020 000000  MOVB     #20,@(SP)    ;WRITE ERROR NUMBER
366 046516 162716 000002          SUB      #2,(SP)       ;MOVE SP TO ERROR RETURN
367 046522 004736          JSR      PC,@(SP)+    ;REPORT ERROR AND RETURN
368 046524 162716 000010          SUB      #10,(SP)     ;MOVE SP TO NO ERROR
369 046530          200$:
370
371          ;BIT 07 (HCE) OF THE FUNCTION CODE TABLE IS THE ENABLING BIT FOR
372          ;HEADER ERRORS, I.E..
373          ;
374          ;   RMER1 - HCRC,HCE,FER
375          ;   RMER2 - BSE
376          ;RESET THE ENABLING BIT (HCE) IF HEADER COMPARE INHIBIT IS SET
377 046530 032737 002000 001366  BIT      #HCI,RMOFI   ;IS HCI SET ??
378 046536 001403          BEQ      201$          ;NO !!
379 046540 042737 000200 050366  BIC      #HCE,500$    ;YES - DISABLE ALL HEADER ERRORS
380 046546          201$:
381
382          ;REPORT AN ERROR IF HCRC IS SET AND HEADER ERRORS ARE NOT ENABLED
383 046546 012746 177777          MOV      #-1,-(SP)    ;ASSUME ERRORS ENABLED
384 046552 032737 000200 050366  BIT      #HCE,500$    ;ARE HEADER ERRORS ENABLED ??
385 046560 001002          BNE      205$          ;YES !!
386 046562 042716 000400          BIC      #HCRC,(SP)   ;DISABLE HCRC
387 046566 013737 001350 001142 205$: MOV      RMER1I,$BDDAT ;GET RECEIVED STATUS
388 046574 042637 001142          BIC      (SP)+,$BDDAT ;RESET HCRC IF ENABLED
389 046600 001412          BEQ      210$          ;BRANCH IF HCRC IS OK
390 046602 062716 000004          ADD      #4,(SP)       ;MOVE SP TO USERS ERROR CALL
391 046606 112776 000035 000000  MOVB     #35,@(SP)    ;WRITE ERROR NUMBER IN CALL
392 046614 162716 000002          SUB      #2,(SP)       ;MOVE SP TO ERROR RETURN
393 046620 004736          JSR      PC,@(SP)+    ;REPORT ERROR AND RETURN
394 046622 162716 000010          SUB      #10,(SP)     ;MOVE SP TO NO ERROR
395 046626          210$:
396
397          ;REPORT ERROR IF HCE IS SET AND HEADER ERRORS ARE NOT ENABLED
398 046626 012746 177777          MOV      #-1,-(SP)    ;ASSUME ERRORS ENABLED
399 046632 032737 000200 050366  BIT      #HCE,500$    ;ARE ERRORS ENABLED ??
    
```

```

400 046640 001002          BNE      215$          ;YES !!
401 046642 042716 000200    BIC      #HCE,(SP)      ;DISABLE HCE
402 046646 013737 001350 001142 215$:  MOV      RMER1I,$BDDAT  ;GET RECEIVED STATUS
403 046654 042637 001142    BIC      (SP)+,$BDDAT  ;CLEAR HCE IF ENABLED
404 046660 001412          BEQ      220$          ;BRANCH IF HCE IS OK
405 046662 062716 000004    ADD      #4,(SP)        ;MOVE SP TO USERS ERROR CALL
406 046666 112776 000036 000000    MOVVB   #36,@(SP)      ;WRITE ERROR NUMBER IN CALL
407 046674 162716 000002    SUB      #2,(SP)        ;MOVE SP TO ERROR RETURN
408 046700 004736          JSR      PC,@(SP)+      ;REPORT ERROR AND RETURN
409 046702 162716 000010    SUB      #10,(SP)       ;MOVE SP TO NO ERROR
410 046706          220$:
411
412          ;REPORT ERROR IF FER IS SET AND HEADER ERRORS ARE NOT ENABLED
413 046706 012746 177777    MOV      #-1,-(SP)      ;ASSUME FER IS ENABLED
414 046712 032737 000200 050366    BIT      #HCE,500$      ;ARE HEADER ERRORS ENABLED ??
415 046720 001002          BNE      225$          ;YES !!
416 046722 042716 000020    BIC      #FER,(SP)      ;DISABLE FER
417 046726 013737 001350 001142 225$:  MOV      RMER1I,$BDDAT  ;GET RECEIVED STATUS
418 046734 042637 001142    BIC      (SP)+,$BDDAT  ;RESET FER IF ENABLED
419 046740 001412          BEQ      230$          ;BRANCH IF FER OK
420 046742 062716 000004    ADD      #4,(SP)        ;MOVE SP TO USERS ERROR CALL
421 046746 112776 000037 000000    MOVVB   #37,@(SP)      ;WRITE ERROR NUMBER IN CALL
422 046754 162716 000002    SUB      #2,(SP)        ;MOVE SP TO ERROR RETURN
423 046760 004736          JSR      PC,@(SP)+      ;REPORT ERROR AND RETURN
424 046762 162716 000010    SUB      #10,(SP)       ;MOVE SP TO NO ERROR
425 046766          230$:
426
427          ;REPORT ERROR IF BSE IS SET AND HEADER ERRORS ARE NOT ENABLED
428 046766 012746 177777    MOV      #-1,-(SP)      ;ASSUME ERRORS ENABLED
429 046772 032737 000200 050366    BIT      #HCE,500$      ;ARE THEY ENABLED ??
430 047000 001002          BNE      235$          ;YES !!
431 047002 042716 100000    BIC      #BSE,(SP)      ;DISABLE BSE
432 047006 013737 001376 001142 235$:  MOV      RMER2I,$BDDAT  ;GET RECEIVED STATUS
433 047014 042637 001142    BIC      (SP)+,$BDDAT  ;CLEAR BSE IF ENABLED
434 047020 001412          BEQ      240$          ;BRANCH IF BSE OK
435 047022 062716 000004    ADD      #4,(SP)        ;MOVE SP TO USERS ERROR CALL
436 047026 112776 000354 000000    MOVVB   #354,@(SP)     ;WRITE ERROR NUMBER
437 047034 162716 000002    SUB      #2,(SP)        ;MOVE SP TO ERROR RETURN
438 047040 004736          JSR      PC,@(SP)+      ;REPORT ERROR AND RETURN
439 047042 162716 000010    SUB      #10,(SP)       ;MOVE SP TO NO ERROR
440 047046          240$:
441
442          ;BIT 06 OF THE FUNCTION CODE TABLE IS THE ENABLING BIT FOR DATA
443          ;FIELD ERRORS, I.E.,
444          ;      RMCS2 - MDPE
445          ;      RMER1 - DCK,ECH
446          ;NOTE:
447          ;      ECH CANNOT SET UNLESS IT IS ENABLED AND ECI IS RESET AND
448          ;      DCK IS SET.
449
450          ;REPORT ERROR IF MDPE IS SET AND IS NOT ENABLED
451 047046 012746 177777    MOV      #-1,-(SP)      ;ASSUME ENABLED
452 047052 032737 000100 050366    BIT      #ECH,500$      ;ARE DATA FIELD ERRORS ENABLED ??
453 047060 001002          BNE      245$          ;YES !!
454 047062 042716 000400    BIC      #MDPE,(SP)     ;DISBALE MDPE
455 047066 013737 001344 001142 245$:  MOV      RMCS2I,$BDDAT  ;GET RECEIVED STATUS
456 047074 042637 001142    BIC      (SP)+,$BDDAT  ;CLEAR MDPE IF ENABLED
    
```

```

457 047100 001412          BEQ      250$          ;BRANCH IF MDPE OK
458 047102 062716 000004    ADD      #4,(SP)        ;MOVE SP TO USERS ERROR CALL
459 047106 112776 000027 000000  MOVB    #27,@(SP)      ;WRITE ERROR NUMBER IN CALL
460 047114 162716 000002    SUB      #2,(SP)        ;MOVE SP TO ERROR RETURN
461 047120 004736          JSR     PC,@(SP)+      ;REPORT ERROR AND RETURN
462 047122 162716 000010    SUB      #10,(SP)       ;MOVE SP TO NO ERROR
463 047126          250$:
464
465          ;REPORT ERROR IF DCK IS SET AND DATA FIELD ERRORS ARE NOT ENABLED
466 047126 012746 177777    MOV      #-1,-(SP)      ;ASSUME ENABLED
467 047132 032737 000100 050366  BIT     #ECH,500$      ;ARE THEY ENABLED ??
468 047140 001002          BNE     255$          ;YES !!
469 047142 042716 100000    BIC     #DCK,(SP)      ;DISABLE DCK
470 047146 013737 001350 001142 255$:  MOV     RMER1I,$BDDAT  ;GET RECEIVED STATUS
471 047154 042637 001142    BIC     (SP)+,$BDDAT   ;CLEAR DCK IF ENABLED
472 047160 001412          BEQ     260$          ;BRANCH IF DCK IS OK
473 047162 062716 000004    ADD      #4,(SP)        ;MOVE SP TO USERS ERROR CALL
474 047166 112776 000030 000000  MOVB    #30,@(SP)      ;WRITE ERROR NUMBER IN CALL
475 047174 162716 000002    SUB      #2,(SP)        ;MOVE SP TO ERROR RETURN
476 047200 004736          JSR     PC,@(SP)+      ;REPORT ERROR AND RETURN
477 047202 162716 000010    SUB      #10,(SP)       ;MOVE SP TO NO ERROR
478 047206          260$:
479
480          ;REPORT ERROR IF ECH IS SET AND
481          ;DATA FIELD ERRORS ARE NOT ENABLED, OR
482          ;ECI IS SET, OR
483          ;DCK IS NOT SET.
484 047206 012746 177777    MOV      #-1,-(SP)      ;ASSUME ENABLED
485 047212 032737 000100 050366  BIT     #ECH,500$      ;ARE ERRORS ENABLED ??
486 047220 001410          BEQ     265$          ;NO !!
487 047222 032737 004000 001366  BIT     #ECI,RMOFI     ;IS ECI SET ??
488 047230 001004          BNE     265$          ;YES !!
489 047232 032737 100000 001350  BIT     #DCK,RMER1I    ;IS DCK ALSO SET ??
490 047240 001002          BNE     270$          ;YES !!
491 047242 042716 000100 265$:  BIC     #ECH,(SP)      ;DISABLE ECH
492 047246 013737 001350 001142 270$:  MOV     RMER1I,$BDDAT  ;GET RECEIVED STATUS
493 047254 042637 001142    BIC     (SP)+,$BDDAT   ;CLEAR ECH IF ENABLED
494 047260 001412          BEQ     275$          ;BRANCH IF ECH IS OK
495 047262 062716 000004    ADD      #4,(SP)        ;MOVE SP TO USERS ERROR CALL
496 047266 112776 000031 000000  MOVB    #31,@(SP)      ;WRITE ERROR NUMBER IN CALL
497 047274 162716 000002    SUB      #2,(SP)        ;MOVE SP TO ERROR RETURN
498 047300 004736          JSR     PC,@(SP)+      ;REPORT ERROR AND RETURN
499 047302 162716 000010    SUB      #10,(SP)       ;MOVE SP TO NO ERROR
500 047306          275$:
    
```

```

1
2
3
4
5 047306 022737 000030 050374      CMP      #SEARCH,515$      ;WAS DATA TRANSFERRED ?
6 047314 103402      BLO      280$              ;BR IF YES
7 047316 000137 050340      JMP      355$              ;NO - EXIT
8
9
10 047322 013737 001336 001142  ;REPORT ERROR IF RMWC NOT ZERO AND TRE IS ZERO
11 047330 001421      280$:  MOV      RMWCI,$BDDAT      ;WORD COUNT ZERO??
12 047332 032737 040000 001334      BEQ      285$              ;YES
13 047340 001015      BIT      #TRE,RMCS1I      ;TRANSFER ERROR DETECTED??
14 047342 062716 000004      BNE      285$              ;YES!!
15 047346 112776 000015 000000      ADD      #4,(SP)
16 047354 005037 001140      MOV      #15,@(SP)        ;ERROR NUMBER
17 047360 162716 000002      CLRB    $GDDAT            ;GOOD DATA FOR TYPEOUT
18 047364 004736      SUB      #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
19 047366 162716 000010      JSR      PC,@(SP)+        ;REPORT WORD COUNT NOT ZERO
20 047372 000240      SUB      #10,(SP)        ;RESTORE (SP)
21  NOP
22
23 047374 013737 001336 001140  ;REPORT ERROR IF RMBA IS NOT CORRECT
24 047402 163737 001412 001140      285$:  MOV      RMWCI,$GDDAT      ;GET WORD COUNT AT END OF TRANSFER AND
25 047410 006337 001140      SUB      RMWCO,$GDDAT      ;SUBTRACT STARTING WORD COUNT.
26 047414 063737 001414 001140      ASL      $GDDAT            ;* 2
27  ADD      RMBAO,$GDDAT      ;ADD STARTING BUS ADDRESS
28 047422 032737 000010 001344      BIT      #BAI,RMCS2I      ;WAS BUS ADDRESS INHIBIT (BAI) SET ??
29 047430 001403      BEQ      290$              ;NO !!
30 047432 013737 001414 001140      MOV      RMBAO,$GDDAT      ;ADDRESS SHOULD NOT HAVE CHANGED
31
32 047440 023737 001140 001340      290$:  CMP      $GDDAT,RMBAI      ;BUS ADDRESS OK??
33 047446 001416      BEQ      295$              ;YES!!
34 047450 013737 001340 001142      MOV      RMBAI,$BDDAT      ;BAD DATA FOR TYPEOUT
35 047456 062716 000004      ADD      #4,(SP)
36 047462 112776 000016 000000      MOV      #16,@(SP)        ;ERROR NUMBER
37 047470 162716 000002      SUB      #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
38 047474 004736      JSR      PC,@(SP)+        ;REPORT UNEXPECTED ADDRESS
39 047476 162716 000010      SUB      #10,(SP)        ;RESTORE (SP)
40 047502 000240      NOP
41
42
43 047504 005046      ;COMPUTE NUMBER OF SECTORS TRANSFERRED FROM WORD COUNT
44 047506 013746 001336      295$:  CLR      -(SP)            ;NUMBER OF SECTORS TRANSFERRED
45 047512 163716 001412      MOV      RMWCI,-(SP)      ;GET WORD COUNT AT END OF TRANSFER AND
46  SUB      RMWCO,(SP)      ;SUBTRACT STARTING WORD COUNT.
47 047516 012746 000400      MOV      #256,-(SP)      ;ASSUME 256. WORDS PER SECTOR
48 047522 032737 000002 001410      BIT      #BIT1,RMCS10     ;HEADER & DATA COMMAND ??
49 047530 001402      BEQ      300$              ;NO !!
50 047532 062716 000002      ADD      #2,(SP)          ;CHANGE TO 258. WORDS PER SECTOR
51
52 047536 005266 000004      300$:  INC      4(SP)            ;INCREMENT SECTOR COUNT
53 047542 161666 000002      SUB      (SP),2(SP)       ;SUBTRACT ONE SECTOR'S WORTH
54 047546 003373      BGT      300$              ;CONTINUE IF NOT DONE
55 047550 022626      CMP      (SP)+,(SP)+      ;RESTORE STACK
56
57  ;COMPUTE EXPECTED SECTOR, TRACK AND CYLINDER ADDRESS FROM
    
```

```

58
59 047552 013737 001444 050366 ;NUMBER OF SECTORS
60 047560 013737 001416 050370 MOV RMDCO,500$ ;STORE ORIGINAL CYLINDER
61 047566 013737 001416 050372 MOV RMDAO,505$ ;STORE ORIGINAL TRACK
62 047574 013737 001332 050376 MOV LSTRK,520$ ;STORE ORIGINAL SECTOR
63 047602 000337 050376 SWAB 520$ ;STORE LAST TRACK
64 047606 005237 050376 INC 520$ ;GET TRACK ADDRESS TO LO BYTE AND
65 ;INCREMENT TO GET TOTL # OF TRACKS.
66 047612 042737 000377 050370 BIC #^C<TADMSK>,505$ ;SAVE TRACK ADDRESS BITS AND
67 047620 000337 050370 SWAB 505$ ;SWAP TRACK ADDRESS TO LOW BYTE.
68 047624 042737 177400 050372 BIC #^C<SADMSK>,510$ ;SAVE SECTOR ADDRESS BITS
69 047632 062637 050372 ADD (SP)+,510$
70
71 047636 023727 050372 000040 310$: CMP 510$,#32. ;SECTOR OVEFLOWED??
72 047644 103406 BLO 315$ ;NO!!
73 047646 005237 050370 INC 505$ ;INCREMENT TRACK
74 047652 162737 000040 050372 SUB #32.,510$ ;ADJUST SECTOR
75 047660 000766 BR 310$ ;TRY AGAIN
76
77 047662 023737 050370 050376 315$: CMP 505$,520$ ;TRACK OVEFLOWED??
78 047670 103407 BLO 320$ ;NO!!
79 047672 005237 050366 INC 500$ ;INCREMENT CYLINDER
80 047676 163737 050376 050370 SUB 520$,505$ ;ADJUST TRACK
81 047704 000766 BR 315$ ;TRY AGAIN
82 047706 000240 NOP
83
84 ;REPORT ERROR IF 'LBT' IS NOT CORRECT
85 047710 320$:
86 047710 005037 001140 CLR $GDDAT ;SET GOOD DATA FOR LBT = 0
87 047714 023727 050366 001466 CMP 500$,#822. ;SHOULD LBT BE SET??
88 047722 101407 BLOS 325$ ;NO!!
89 047724 032737 002000 001350 BIT #IAE,RMER1I ;WAS IAE SET ??
90 047732 001003 BNE 325$ ;YES - LBT SHOULD NOT BE SET
91 047734 012737 002000 001140 MOV #LBT,$GDDAT ;SET GOOD DATA FOR LBT = 1
92 047742 013737 001346 001142 325$: MOV RMDSI,$BDDAT ;BAD DATA FOR TYPEOUT
93 047750 042737 175777 001142 BIC #^CLBT,$BDDAT
94 047756 023737 001140 001142 CMP $GDDAT,$BDDAT ;IS LBT CORRECT??
95 047764 001413 BEQ 330$ ;YES!!
96 047766 062716 000004 ADD #4,(SP)
97 047772 112776 000017 000000 MOVB #17,@(SP) ;ERROR NUMBER
98 050000 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
99 050004 004736 JSR PC,@(SP)+ ;REPORT LBT IS WRONG
100 050006 162716 000010 SUB #10,(SP) ;RESTORE (SP)
101 050012 000240 NOP
102
103 ;REPORT ERROR IF 'AOE' IS INCORRECT
104 050014 005037 001140 330$: CLR $GDDAT ;SET FOR AOE = 0
105 050020 032737 002000 001350 BIT #IAE,RMER1I ;WAS 'IAE' DETECTED??
106 050026 001031 BNE 340$ ;YES-'AOE' SHOULD BE ZERO
107 050030 023727 050366 001466 CMP 500$,#822. ;SHOULD AOE BE SET??
108 050036 101425 BLOS 340$ ;NO!!
109 050040 005737 050370 TST 505$ ;MAYBE
110 050044 001012 BNE 335$ ;YES
111 050046 005737 050372 TST 510$
112 050052 001007 BNE 335$ ;YES !!
113 050054 032737 000010 050374 BIT #F2,515$ ;WAS THIS READ OR WRITE CHECK ??
114 050062 001413 BEQ 340$ ;NO !!
    
```

```

115 050064 005737 001336          TST      RMWCI          :WAS ALL DATA TRANSFERRED ??
116 050070 001410          BEQ      340$          :YES !!
117 050072 012737 001000 001140 335$: MOV      #AOE,$GDDAT   :SET FOR AOE = 1
118 050100 005037 050370          CLR      505$          :CLEAR EXPECTED TRACK
119 050104 012737 000001 050372  MOV      #1,510$        :EXPECT SECTOR = 1
120 050112 013737 001350 001142 340$: MOV      RMER1I,$BDDAT  :BAD DATA FOR TYPEOUT
121 050120 042737 176777 001142  BIC      #^CAOE,$BDDAT
122 050126 023737 001140 001142  CMP      $GDDAT,$BDDAT :IS AOE CORRECTY??
123 050134 001413          BEQ      345$          :YES!!
124 050136 062716 000004          ADD      #4,(SP)
125 050142 112776 000020 000000  MOVVB   #20,@(SP)      :ERROR NUMBER
126 050150 162716 000002          SUB      #2,(SP)      :MOVE SP TO RETURN FOR ERROR
127 050154 004736          JSR     PC,@(SP)+     :REPORT AOE IS WRONG
128 050156 162716 000010          SUB      #10,(SP)     :RESTORE (SP)
129 050162 000240          NOP
130
131
132 050164 032737 002000 001350  :REPORT ERROR IF RMDA IS NOT CORRECT
133 050172 001062          345$: BIT      #IAE,RMER1I :WAS THERE AN IAE ERROR ??
134 050174 013737 050370 001140  BNE     355$          :YES - DONT CHECK RMDA,RMDC
135 050202 000337 001140          MOV      505$,$GDDAT  :SETUP EXPECTED DISK ADDRESS
136 050206 113737 050372 001140  SWAB    $GDDAT
137 050214 013737 001342 001142  MOVVB   510$,$GDDAT
138 050222 023737 001140 001142  MOV      RMDAI,$BDDAT  :SETUP RECEIVED DISK ADDRESS
139 050230 001413          CMP      $GDDAT,$BDDAT :COMPARE EXPECTED & RECEIVED
140 050232 062716 000004          BEQ     350$          :BRANCH IF EQUAL
141 050236 112776 000021 000000  ADD      #4,(SP)
142 050244 162716 000002          MOVVB   #21,@(SP)    :ERROR NUMBER
143 050250 004736          SUB      #2,(SP)      :MOVE SP TO RETURN FOR ERROR
144 050252 162716 000010  JSR     PC,@(SP)+     :REPORT BAD DISK ADDRESS
145 050256 000240          SUB      #10,(SP)    :RESTORE (SP)
146
147
148 050260 013737 050366 001140  :REPORT ERROR IF RMDC IS INCORRECT
149 050266 042737 176000 001140  350$: MOV      500$,$GDDAT  :SETUP EXPECTED CYLINDER
150 050274 013737 001370 001142  BIC     #^C1777,$GDDAT
151 050302 023737 001140 001142  MOV      RMDCI,$BDDAT  :SETUP RECEIVED CYLINDER
152 050310 001413          CMP     $GDDAT,$BDDAT :COMPARE CYLINDERS
153 050312 062716 000004          BEQ     355$          :BRANCH IF EQUAL
154 050316 112776 000022 000000  ADD      #4,(SP)
155 050324 162716 000002          MOVVB   #22,@(SP)    :ERROR NUMBER
156 050330 004736          SUB     #2,(SP)      :MOVE SP TO RETURN FOR ERROR
157 050332 162716 000010  JSR     PC,@(SP)+     :REPORT BAD CYLINDER
158 050336 000240          SUB     #10,(SP)    :RESTORE (SP)
159
160 050340 062716 000004          355$: ADD     #4,(SP)      :MOVE (SP) TO ERROR CALL
161 050344 105776 000000          TSTB   @(SP)         :WAS ERROR FOUND??
162 050350 001403          BEQ     360$
163 050352 062716 000004          ADD     #4,(SP)      :MOVE (SP) TO ERROR RETURN
164 050356 000402          BR      365$
165 050360 162716 000004  360$: SUB     #4,(SP)    :MOVE (SP) TO NO ERROR RETURN
166 050364 000207  365$: RTS      PC
167
168 050366 000000          500$: .WORD  0         :CYLINDER
169 050370 000000          505$: .WORD  0         :TRACK
170 050372 000000          510$: .WORD  0         :SECTOR
171 050374 000000          515$: .WORD  0         :FUNCTION CODE
    
```

CZRMBO RM05/3/2 FCTNL TST 1 MACRO V04.00 4-APR-81 11:43:28 PAGE 23-3
SECONDARY ERROR CHECK SUBROUTINE

D 16

SEQ 0198

.172 050376 000000

520\$: .WORD 0

;TOTAL # OF TRACKS = LAST TRACK +1

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 COMPOSITE ERROR CHECK SUBROUTINE

;THIS SUBROUTINE CHECKS THE STORED CONTENTS OF RMER1 AND
;RMER2 AFTER MASKING EACH REGISTER WORD WITH THE USER'S STATUS
;MASKS AND REPORTS AN ERROR IF ANY BITS ARE LEFT ON AFTER
;THE MASKS ARE APPLIED.

;CALL:
;(1) JSR PC,CMPERRSTS
      .WORD MASK FOR ERROR REGISTER 1
      .WORD MASK FOR ERROR REGISTER 2
      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: BITS TO BE MASKED SHOULD BE ONE; BITS TO BE TESTED SHOULD
;BE ZERO

CMPERRSTS:

;MASK AND STORE THE CONTENTS OF RMER1 AND RMER2
MOV RMER1I,$TMP1 ;STORE RMER1 AT TEMP STORAGE
BIC @(SP),$TMP1 ;MASK RMER1
ADD #2,(SP) ;MOVE SP TO NEXT MASK
MOV RMER2I,$TMP2 ;STORE RMER2 AT TEMP STORAGE
BIC @(SP),$TMP2 ;MASK RMER2

;CLEAR USER'S ERROR CALL
ADD #6,(SP) ;MOVE SP TO USER'S ERROR CALL
CLRB @(SP) ;CLEAR ERROR NUMBER
SUB #4,(SP) ;LEAVE SP AT NO ERROR RETURN

;SEE IF THERE WERE ANY ERRORS IN RMER1, I.E., $TMP1
TST $TMP1 ;ANY ERRORS TO REPORT??
BEQ 5$ ;NO !!
MOV $TMP1,$BDDAT ;RECEIVED STATUS FOR TYPEOUT
CLR $GDDAT ;EXPECTED STATUS FOR TYPEOUT
ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
MOVB #6,@(SP) ;CORRECTABLE DATA CHECK ERROR #
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

5$:

;SEE IF THERE ARE ANY ERRORS TO REPORT IN RMER2 ($TMP2)
TST $TMP2 ;ANY ERRORS IN RMER2?
BEQ 10$ ;NO!!

MOV $TMP2,$BDDAT ;RECEIVED STATUS FOR TYPEOUT
CLR $GDDAT ;EXPECTED STATUS FOR TYPEOUT
ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
MOVB #6,@(SP) ;WRITE ERROR NUMBER IN USER'S CALL
    
```

050400				
050400	013737	001350	001176	
050406	047637	000000	001176	
050414	062716	000002		
050420	013737	001376	001200	
050426	047637	000000	001200	
050434	062716	000006		
050440	105076	000000		
050444	162716	000004		
050450	005737	001176		
050454	001420			
050456	013737	001176	001142	
050464	005037	001140		
050470	062716	000004		
050474	112776	000066	000000	
050502	162716	000002		
050506	004736			
050510	162716	000010		
050514	000240			
050516	005737	001200		
050522	001420			
050524	013737	001200	001142	
050532	005037	001140		
050536	062716	000004		
050542	112776	000067	000000	

58	050550	162716	000002	SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
59	050554	004736		JSR	PC,@(SP)+	:REPORT ERROR VIA USER
60	050556	162716	000010	SUB	#10,(SP)	:MOVE SP TO NO ERROR RETURN
61	050562	000240		NOP		
62	050564			10\$:		
63						
64						
65	050564	062716	000004			:AUGMENT THE RETURN ADDRESS IF ANY ERROR WAS DETECTED
66	050570	105776	000000	ADD	#4,(SP)	:MOVE SP TO USER'S ERROR CALL
67	050574	001403		TSTB	@(SP)	:WAS THERE AN ERROR CALLED??
68	050576	062716	000004	BEQ	20\$:NO!!
69	050602	000402		ADD	#4,(SP)	:YES - MOVE SP TO ERROR RETURN
70	050604	162716	000004	BR	30\$	
71	050610	000207		20\$:	SUB #4,(SP)	:MOVE SP TO NO ERROR RETURN
				30\$:	RTS PC	:RETURN TO USER

```

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 050612  DEVSEL:
13
14     ;CLEAR USER'S ERROR CALL
15 050612 062716 000004      ADD      #4,(SP)          ;MOVE SP TO USER'S ERROR
16 050616 105076 000000      CLRB   @10(SP)         ;CLEAR LOW ORDER BYTE OF CALL
17 050622 162716 000004      SUB      #4,(SP)          ;MOVE SP BACK
18
19     ;SAVE USER'S INFORMATION AND SETUP REGISTERS
20 050626 013746 000004      MOV      ERRVEC,-(SP)    ;;PUSH ERRVEC ON STACK
21 050632 013746 000006      MOV      ERRVEC+2,-(SP) ;;PUSH ERRVEC+2 ON STACK
22 050636 010046             MOV      R0,-(SP)        ;;PUSH R0 ON STACK
23 050640 010146             MOV      R1,-(SP)        ;;PUSH R1 ON STACK
24 050642 012737 050762 000004  MOV      #20$,ERRVEC    ;SETUP FOR BUS TIMEOUT
25 050650 012737 000300 000006  MOV      #PR6,ERRVEC+2
26 050656 013700 001276      MOV      $BASE,R0      ;R0 = UNIBUS ADDRESS
27 050662 013701 001464      MOV      TSTQUE,R1     ;R1 POINTS TO DEVICE NUMBER
28
29     ;SELECT DEVICE AND VERIFY THAT DEVICE IS AVAILABLE
30
31 050666 111160 000010      MOVVB   (R1),RMCS2(R0)  ;WRITE UNIT SELECT BITS
32 050672 016037 000000 001176  MOV      RMCS1(R0),$TMP1 ;GET 'DVA' STATUS
33 050700 016037 000010 001174  MOV      RMCS2(R0),$TMP0 ;GET 'NED' STATUS
34
35 050706 032737 010000 001174  BIT      #NED,$TMP0     ;IS DEVICE NONEXISTENT ?
36 050714 001407             BEQ      10$            ;NO!!
37 050716 062766 000004 000010  ADD      #4,10(SP)      ;MOVE SP TO USERS ERROR CALL
38 050724 112776 000111 000010  MOVVB   #111,@10(SP)   ;WRITE ERROR NUMBER
39 050732 000422             BR       30$
40
41 050734 032737 004000 001176 10$: BIT      #DVA,$TMP1     ;IS DEVICE AVAILABLE ?
42 050742 001021             BNE      35$            ;YES!!
43 050744 062766 000004 000010  ADD      #4,10(SP)      ;MOVE SP TO USERS ERROR CALL
44 050752 112776 000112 000010  MOVVB   #112,@10(SP)   ;WRITE ERRGR NUMBER
45 050760 000407             BR       30$
46
47     ;HANDLE BUS TIMEOUT
48 20$: CMP      (SP)+,(SP)+  ;ADJUST SP
49 050764 062766 000004 000010  ADD      #4,10(SP)      ;MOVE SP TO USERS ERROR CALL
50 050772 112776 000113 000010  MOVVB   #113,@10(SP)   ;WRITE BUS TIMEOUT ERROR NUMBER
51 051000 162766 000002 000010 30$: 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 35$: MOV      (SP)+,R1     ;;POP STACK INTO R1
56     MOV      (SP)+,R0     ;;POP STACK INTO R0
57     MOV      (SP)+,ERRVEC+2 ;;POP STACK INTO ERRVEC+2
    
```

051016 012637 000004
52 051022 000207

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

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19 051024
20
21
22 051024 000240
23 051026 062716 000004
24 051032 105076 000000
25 051036 162716 000004
26 051042 005037 052262
27
28
29
30 051046 032737 000010 001350
31 051054 001424
32 051056 032737 000010 001376
33 051064 001020
34
35
36 051066 005037 001140
37 051072 013737 001350 001142
38 051100 062716 000004
39 051104 112776 000050 000000
40 051112 162716 000002
41 051116 004736
42 051120 162716 000010
43 051124 000437
44
45
46
47 051126 012737 002000 001140
48 051134 052737 040000 052262
49 051142 023727 001444 001466
50 051150 101025
51 051152 042737 040000 052262
52
53 051160 123737 001417 001333
54 051166 101016
55
56 051170 123727 001416 000035
57 051176 101410

```

.SBTTL  SEEK STATUS CHECK SUBROUTINE

:THIS SUBROUTINE VERIFIES THE RESULTS OF SEEK TESTS USING STATUS
:STORED IN THE GET BUFFER AND TEST PARAMETERS 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 'ERROR' TRAP
:OF THE CALLING ROUTINE.  SEEK STATUS IS CHECKED AS FOLLOWS:

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

SEKSTS:

:CLEAR USERS' ERROR CALL
      NOP
      ADD    #4,(SP)      :MOVE (SP) TO ERROR CALL
      CLRB   @(SP)        :CLEAR ERROR NUMBER
      SUB    #4,(SP)      :MOVE (SP) TO NO ERROR RETURN
      CLR    300$         :CLEAR ERROR FLAGS

:TEST FOR MASSBUS CONTROL BUS PARITY ERROR WHEN WRITING
:LOCAL REGISTERS, I.E., 'PAR' = 1 AND 'DPE' = 0
      BIT    #PAR,RMER1I  :WAS PARITY ERROR DETECTED??
      BEQ    1$           :NO!!
      BIT    #DPE,RMER2I  :WAS IT DUE TO CONTROL BUS??
      BNE    1$           :NOT SURE!!

:REPORT REGISTER PARITY ERROR VIA USER'S ERROR CALL
      CLR    $GDDAT       :EXPECTED STATUS
      MOV    RMER1I,$BDDAT :RECEIVED STATUS
      ADD    #4,(SP)      :MOVE STACK TO USER'S ERROR
      MOVB   #50,@(SP)    :ERROR #50
      SUB    #2,(SP)      :MOVE SP TO RETURN FOR ERROR
      JSR    PC,@(SP)+
      SUB    #10,(SP)     :RESTORE STACK
      BR     3$           :IAE SHOULD BE ZERO

:DETERMINE THE VALUE OF 'IAE' STATUS BASED ON TRACK, SECTOR AND CYLINDER
:ALSO, SET 'SKI' IF CYLINDER ADDRESS IS TOO LARGE.
1$:  MOV    #IAE,$GDDAT   :SETUP FOR IAE = 1
      BIS    #SKI,300$    :SETUP FOR SKI = 1
      CMP    RMDCO,#822.  :GREATER THAN LAST CYLINDER?
      BHI    3$           :YES - CYLINDER IS INVALID
      BIC    #SKI,300$    :CLEAR SKI ERROR FLAG

      CMPB   RMDAO+1,LSTRK+1 :GREATER THAN LAST TRACK?
      BHI    3$           :YES - TRACK IS INVALID

      CMPB   RMDAO,#29.   :SECTOR > 29. ?
      BLOS  2$           :BR IF NO
  
```

```

58 051200 032737 010000 001442      BIT      #FMT16,RMOFO      ;18 BIT FORMAT ?
59 051206 001406                BEQ      3$              ;YES - SECTOR IS INVALID FOR 18 BIT MODE
60 051210 123727 001416 000037      CMPB    RMDAO,#31.      ;SECTOR > 31. ?
61 051216 101002                BHI      3$              ;YES - SECTOR IS INVALID
62
63 051220 005037 001140      2$:     CLR      $GDDAT      ;'IAE' SHOULD = 0
64
65                                ;COMPARE EXPECTED AND RECIEVED 'IAE' STATUS
66 051224 013737 001350 001142      3$:     MOV      RMER1I,$BDDAT ;IS IAE OK??
67 051232 042737 175777 001142      BIC     #^CIAE,$BDDAT   ;SAVE IAE BIT FOR COMPARE
68 051240 023737 001140 001142      CMP     $GDDAT,$BDDAT   ;CORRECT 'IAE' STATUS ?
69 051246 001004                BNE     35$             ;BR IF NO
70 051250 042737 040000 052262      BIC     #SKI,300$       ;CLEAR SKI FLAG
71 051256 000413                BR      5$              ;GO CHECK NEXT ERROR
72 051260
73                                35$:
74 051260 062716 000004                ;REPORT INCORRECT 'IAE' STATUS VIA USER'S ERROR CALL
75 051264 112776 000051 000000      ADD     #4,(SP)
76 051272 162716 000002      MOVB   #51,@(SP)      ;ERROR 51
77 051276 004736                SUB     #2,(SP)        ;MOVE SP TO RETURN FOR ERROR
78 051300 162716 000010      JSR    PC,@(SP)+      ;REPORT INCORRECT IAE
79 051304 000240                SUB     #10,(SP)       ;RESTORE (SP)
80 051306
81
82                                5$:
83                                ;REPORT ANY IVC ERROR AS
84                                ; IVC ERROR WITH VOLUME VALID ZERO
85 051306 032737 010000 001376      ; ERRONEOUS IVC ERROR, VOLUME VALID IS SET
86 051314 001427                BIT     #IVC,RMER2I     ;IVC ERROR??
87 051316 005037 001140      BEQ     52$            ;NO!!
88 051322 013737 001376 001142      CLR     $GDDAT        ;EXPECTED STATUS
89 051330 062716 000004                MOV     RMER2I,$BDDAT  ;RECEIVED STATUS
90 051334 112776 000060 000000      ADD     #4,(SP)        ;MOVE SP TO USER'S ERROR
91 051342 032737 000100 001346      MOVB   #60,@(SP)      ;ERROR 60 IF VV = 0
92 051350 001403                BIT     #VV,RMDSI
93 051352 112776 000061 000000      BEQ     51$            ;ERROR 61 IF VV = 1
94 051360 162716 000002      51$:   SUB     #2,(SP)        ;MOVE SP TO RETURN FOR ERROR
95 051364 004736                JSR    PC,@(SP)+      ;REPORT ERROR VIA USER
96 051366 162716 000010      SUB     #10,(SP)       ;RESTORE SP
97 051372 000240                NOP
98
99 051374 013737 001376 001142      52$:   MOV     RMER2I,$BDDAT ;RECEIVED STATUS
100 051402 042737 137777 001142      BIC     #^CSKI,$BDDAT ;CLEAR DONT CARES
101 051410 013737 052262 001140      MOV     300$,$GDDAT   ;GET EXPECTED SKI STATUS
102 051416 042737 137777 001140      BIC     #^CSKI,$GDDAT ;CLEAR DONT CARES
103 051424 001417                BEQ     53$            ;BRANCH IF 0 EXPECTED
104
105                                ;REPORT ERROR IF SKI IS NOT SET (IAE WAS NOT DETECTED)
106 051426 032737 040000 001142      BIT     #SKI,$BDDAT   ;WAS SKI DETECTED ??
107 051434 001032                BNE     54$            ;YES !!
108 051436 062716 000004                ADD     #4,(SP)        ;MOVE SP TO USERS ERROR CALL
109 051442 112776 000267 000000      MOVB   #267,@(SP)    ;WRITE ERROR NUMBER
110 051450 162716 000002      SUB     #2,(SP)        ;MOVE SP TO ERROR RETURN
111 051454 004736                JSR    PC,@(SP)+      ;REPORT ERROR AND RETURN
112 051456 162716 000010      SUB     #10,(SP)       ;MOVE SP TO NO ERROR
113 051462 000443                BR      6$              ;GO TO NEXT ERROR CHECK
114 051464                                53$:

```

```

115
116
117 051464 032737 040000 001142 ;REPORT ERROR IF SKI IS SET
118 051472 001413 BIT #SKI,$BDDAT ;IS SKI SET ??
119 051474 062716 000004 BEQ 54$ ;NO - SKI IS OK
120 051500 112776 000054 000000 ADD #4,(SP) ;MOVE (SP) TO ERROR
121 051506 162716 000002 MOVB #54,@(SP) ;LOAD ERROR NUMBER
122 051512 004736 JSR PC,@(SP)+ ;MOVE SP TO RETURN FOR ERROR
123 051514 162716 000010 SUB #2,(SP) ;REPORT SEEK ERROR
124 051520 000240 NOP ;RESTORE (SP)
125
126
127 051522 032737 000200 001376 ;REPORT ANY DEVICE CHECK
128 051530 001420 54$: BIT #DVC,RMER2I ;WAS THERE DVC DURING SEEK??
129 051532 005037 001140 BEQ 6$ ;NO!!
130 051536 013737 001376 001142 CLR $GDDAT ;EXPECTED STATUS
131 051544 062716 000004 MOV RMER2I,$BDDAT ;RECEIVED STATUS
132 051550 112776 000055 000000 ADD #4,(SP)
133 051556 162716 000002 MOVB #55,@(SP) ;ERROR #55
134 051562 004736 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
135 051564 162716 000010 JSR PC,@(SP)+ ;REPORT ERROR VIA USER
136 051570 000240 SUB #10,(SP) ;RESTORE SP
137
138
139 ;REPORT ANY 'OPI' ERROR AS OPI WITH MOL = 0, OR OPI
;BECAUSE ON CYLINDER LATCH DIDN'T RESET
140 051572 032737 020000 001350 6$: BIT #OPI,RMER1I ;'OPI' ERROR??
141 051600 001427 BEQ 8$ ;NO!!
142 051602 005037 001140 CLR $GDDAT ;EXPECTED STATUS
143 051606 013737 001350 001142 MOV RMER1I,$BDDAT ;RECEIVED STATUS
144 051614 062716 000004 ADD #4,(SP) ;MOVE (SP) TO ERROR
145 051620 112776 000052 000000 MOVB #52,@(SP) ;LOAD ERROR NUMBER
146 051626 032737 010000 001346 BIT #MOL,RMDSI ;WAS MEDIUM ON LINE??
147 051634 001403 BEQ 7$ ;NO!!
148 051636 112776 000053 000000 MOVB #53,@(SP) ;YES - CHANGE ERROR NUMBER
149 051644 162716 000002 7$: SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
150 051650 004736 JSR PC,@(SP)+ ;REPORT 'OPI' ERROR
151 051652 162716 000010 SUB #10,(SP) ;RESTORE (SP)
152 051656 000240 NOP
153
154
155 051660 013746 001346 8$: ;SEE IF 'PIP' = 0, AND 'ATA', 'MOL' AND 'VV' = 1
156 051664 042716 047677 MOV RMDSI,-(SP)
157 051670 022726 110100 BIC #^C<ATA!PIP!MOL!VV>,(SP)
158 051674 001002 CMP #ATA!MOL!VV,(SP)+
159 051676 000137 052232 BNE 9$ ;ERROR IN RMDS
160 JMP 14$ ;RMDS IS OK
161
162 051702 032737 010000 001346 9$: ;REPORT ERROR IF MOL = 0 AND OPI = 0
163 051710 001030 BIT #MOL,RMDSI ;IS MOL RESET??
164 051712 032737 020000 001350 BNE 10$ ;NO - MOL IS SET
165 051720 001024 BIT #OPI,RMER1I ;WAS OPI SET
166 051722 013737 001346 001140 BNE 10$ ;YES - DONT REPORT ERROR
167 051730 052737 010000 001140 MOV RMDSI,$GDDAT ;EXPECTED STATUS
168 051736 013737 001346 001142 BIS #MOL,$GDDAT
169 051744 062716 000004 MOV RMDSI,$BDDAT ;RECEIVED STATUS
170 051750 112776 000062 000000 ADD #4,(SP)
171 051756 162716 000002 MOVB #62,@(SP) ;MOVE SP TO RETURN FOR ERROR
SUB #2,(SP)

```

```

172 051762 004736          JSR      PC,@(SP)+      ;REPORT ERROR VIA USER
173 051764 162716 000010  SUB      #10,(SP)
174 051770 000240          NOP
175
176
177 051772 032737 020000 001346 ;REPORT AN ERROR IF 'PIP' IS STIL SET AND SKI NOT SET
10$: BIT      #PIP,RMDSI      ;IS 'PIP' STILL SET??
    BEQ      11$             ;NO!!
178 052000 001430          BIT      #SKI,RMER2I      ;WAS 'SKI' SET??
179 052002 032737 040000 001376 BNE      11$             ;YES-DONT REPORT PIP
180 052010 001024          MOV      RMDSI,$GDDAT     ;EXPECTED STATUS
181 052012 013737 001346 001140 BIC      #PIP,$BDDAT
182 052020 042737 020000 001142 MOV      RMDSI,$BDDAT     ;RECEIVED STATUS
183 052026 013737 001346 001142 ADD      #4,(SP)         ;MOVE (SP) TO ERROR
184 052034 062716 000004          MOVVB   #56,@(SP)        ;LOAD ERROR NUMBER
185 052040 112776 000056 000000 SUB      #2,(SP)         ;MOVE SP TO RETURN FOR ERROR
186 052046 162716 000002          JSR      PC,@(SP)+      ;REPORT 'PIP' SET AFTER SEEK
187 052052 004736          SUB      #10,(SP)      ;RESTORE (SP)
188 052054 162716 000010  NOP
189 052060 000240
190
191
192 052062 032737 100000 001346 ;REPORT AN ERROR IF 'ATA' IS NOT SET
11$: BIT      #ATA,RMDSI      ;WAS 'ATA' SET ??
    BNE      13$             ;YES!!
193 052070 001024          MOV      RMDSI,$GDDAT     ;EXPECTED STATUS
194 052072 013737 001346 001140 BIS      #ATA!MOL!DPR!DRY,$GDDAT
195 052100 052737 110600 001140 MOV      RMDSI,$BDDAT     ;RECEIVED STATUS
196 052106 013737 001346 001142 ADD      #4,(SP)         ;MOVE (SP) TO ERROR
197 052114 062716 000004          MOVVB   #57,@(SP)        ;LOAD ERROR NUMBER
198 052120 112776 000057 000000 SUB      #2,(SP)         ;MOVE SP TO RETURN FOR ERROR
199 052126 162716 000002          JSR      PC,@(SP)+      ;REPORT ATTENTION NOT SET DURING
200 052132 004736          SUB      #10,(SP)      ;SEEK TEST
201
202 052134 162716 000010  NOP          ;RESTORE (SP)
203 052140 000240
204
205
206 052142 032737 000100 001346 ;REPORT ERROR IF VOLUME VALID IS RESET AND IVC IS ZERO
13$: BIT      #VV,RMDSI      ;IS VV = 0 ??
    BNE      14$             ;NO!!
207 052150 001030          BIT      #IVC,RMER2I      ;IS IVC ALSO 0 ??
208 052152 032737 010000 001376 BNE      14$             ;NO - IVC IS SET
209 052160 001024          MOV      RMDSI,$GDDAT     ;EXPECTED STATUS
210 052162 013737 001346 001140 BIS      #VV,$GDDAT
211 052170 052737 000100 001140 MOV      RMDSI,$BDDAT     ;RECEIVED STATUS
212 052176 013737 001346 001142 ADD      #4,(SP)
213 052204 062716 000004          MOVVB   #64,@(SP)        ;ERROR #64
214 052210 112776 000064 000000 SUB      #2,(SP)         ;MOVE SP TO RETURN FOR ERROR
215 052216 162716 000002          JSR      PC,@(SP)+
216 052222 004736          SUB      #10,(SP)
217 052224 162716 000010  NOP
218 052230 000240
219 052232
220
221
222 052232 000240          ;MODIFY THE RETURN ADDRESS IF AN ERROR WAS DETECTED
223 052234 062716 000004  NOP
224 052240 105776 000000  ADD      #4,(SP)         ;MOVE (SP) TO ERROR CALL
225 052244 001403          TSTB   @ (SP)           ;WAS ERROR CALLED??
226 052246 062716 000004  BEQ      15$             ;NO!!
227 052252 000402          ADD      #4,(SP)         ;MOVE TO ERROR RETURN
228

```


CZRMBO RM05/3/2 FCTNL TST 1
SEEK STATUS CHECK SUBROUTINE

MACRO V04.00 4-APR-81 11:43:28 PAGE 26-4

M 16

SEQ 0207

229 052254 162716 000004
230 052260 000207
231
232 052262 000000

15\$: SUB #4,(SP)
16\$: RTS PC

:MOVE (SP) TO NO ERROR RETURN
: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      ???          RETURN HERE IF NO ERROR FOUND
8              NOP
9              ERROR          RETURN HERE IF ANY ERROR FOUND
10             ???          SUB DEFINES ERROR NUMBER
11
12 CNTCLR:
13 052264      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
14 052264      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
15 052270      MOV      ERRVEC,-(SP)  ;;PUSH ERRVEC ON STACK
16 052274      MOV      ERRVEC+2,-(SP) ;;PUSH ERRVEC+2 ON STACK
17 052300      MOV      #10$,ERRVEC   ;SETUP FOR BUS TIMEOUT
18 052306      MOV      #PR6,ERRVEC+2
19 052314      MOV      $BASE,R0      ;R0 = UNIBUS ADDRESS
20 052320      MOV      #CLR, RMCS2(R0) ;CLEAR MASSBUS
21 052326      MOV      TSTQUE,R1     ;GET DEVICE UNDER TEST
22 052332      MOV      (R1),RMCS2(R0) ;SELECT DEVICE
23 052336      BR      20$
24
25 10$:      CMP      (SP)+,(SP)+     ;ADJUST STACK
26 052340      ADD      #4,10(SP)     ;MOVE SP TO USER'S ERROR CALL
27 052342      MOV      #7,@10(SP)    ;WRITE THE ERROR NUMBER
28 052350      SUB      #2,10(SP)     ;ADJUST SP TO RETURN TO ERROR
29 052356
30 20$:      MOV      (SP)+,ERRVEC+2   ;;POP STACK INTO ERRVEC+2
31 052364      MOV      (SP)+,ERRVEC   ;;POP STACK INTO ERRVEC
32 052370      MOV      (SP)+,R1      ;;POP STACK INTO R1
33 052374      MOV      (SP)+,R0      ;;POP STACK INTO R0
34 052376      MOV      (SP)+,R0
35 052400      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          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

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)
```

052402				
052402	062716	000004		
052406	105076	000000		
052412	162716	000004		
052416	013737	001334	001142	
052424	042737	100000	001142	
052432	012737	004200	001140	
052440	023737	001140	001142	
052446	001413			
052450	062716	000004		
052454	112776	000126	000000	
052462	162716	000002		
052466	004736			
052470	162716	000010		
052474	000240			
052476	005037	001140		
052502	013737	001340	001142	
052510	001413			
052512	062716	000004		
052516	112776	000127	000000	
052524	162716	000002		
052530	004736			
052532	162716	000010		
052536	000240			
052540	013737	001344	001142	
052546	010146			
052550	005046			
052552	013701	001464		
052556	111116			

```

58 052560 052716 000100      BIS      #IR,(SP)
59 052564 012637 001140      MOV      (SP)+,$GDDAT
60 052570 012601              MOV      (SP)+,R1          ;;POP STACK INTO R1
61 052572 023737 001140 001142      CMP      $GDDAT,$BDDAT    ;;COMPARE EXPECTED & RECEIVED
62 052600 001413              BEQ      9$               ;;BRANCH IF EQUAL
63 052602 062716 000004              ADD      #4,(SP)          ;;MOVE SP TO USER'S ERROR CALL
64 052606 112776 000130 000000      MOVVB   #130,@(SP)        ;;WRITE ERROR NUMBER IN CALL
65 052614 162716 000002              SUB      #2,(SP)          ;;MOVE SP TO RETURN FOR ERROR
66 052620 004736              JSR      PC,@(SP)+        ;;REPORT ERROR VIA USER
67 052622 162716 000010              SUB      #10,(SP)         ;;MOVE SP BACK TO NO ERROR
68 052626 000240              NOP
69                                ;REPORT ERROR IF RMER1 NOT RESET-IGNORE UNS
70 052630 005037 001140 001142      9$:    CLR      $GDDAT          ;;VERIFY RMER1
71 052634 013737 001350 001142      MOV      RMER1I,$BDDAT
72 052642 042737 040000 001142      BIC      #UNS,$BDDAT      ;;IGNORE UNSAFE
73 052650 001413              BEQ      13$             ;;BRANCH IF ZERO
74 052652 062716 000004              ADD      #4,(SP)          ;;MOVE SP TO USER'S ERROR CALL
75 052656 112776 000131 000000      MOVVB   #131,@(SP)        ;;WRITE ERROR NUMBER IN CALL
76 052664 162716 000002              SUB      #2,(SP)          ;;MOVE SP TO RETURN FOR ERROR
77 052670 004736              JSR      PC,@(SP)+        ;;REPORT ERROR VIA USER
78 052672 162716 000010              SUB      #10,(SP)         ;;MOVE SP BACK TO NO ERROR
79 052676 000240              NOP
80                                ;REPORT ERROR IF RMMR1 NOT INITIALIZED-IGNORE WC,LS,LST
81 052700 013737 001360 001142      13$:   MOV      RMMR1I,$BDDAT    ;;VERIFY RMMR
82 052706 042737 000046 001142      BIC      #WC!LS!LST,$BDDAT ;;IGNORE WORD CLOCK, SCT, TRK
83 052714 012737 000010 001140      MOV      #MWD,$GDDAT      ;;EXPECT WRITE DATA BIT
84 052722 023737 001140 001142      CMP      $GDDAT,$BDDAT    ;;COMPARE EXPECTED AND RECEIVED
85 052730 001413              BEQ      17$             ;;BRANCH IF 0
86 052732 062716 000004              ADD      #4,(SP)          ;;MOVE SP TO USER'S ERROR CALL
87 052736 112776 000133 000000      MOVVB   #133,@(SP)        ;;WRITE ERROR NUMBER IN CALL
88 052744 162716 000002              SUB      #2,(SP)          ;;MOVE SP TO RETURN FOR ERROR
89 052750 004736              JSR      PC,@(SP)+        ;;REPORT ERROR VIA USER
90 052752 162716 000010              SUB      #10,(SP)         ;;MOVE SP BACK TO NO ERROR
91 052756 000240              NOP
92                                ;REPORT AN ERROR IF RMEC2 IS NOT RESET
93 052760 005037 001140 001142      17$:   CLR      $GDDAT          ;;EXPECT ZEROS
94 052764 013737 001402 001142      MOV      RMEC2I,$BDDAT    ;;VERIFY RMEC2 = 0
95 052772 001413              BEQ      19$
96 052774 062716 000004              ADD      #4,(SP)          ;;MOVE SP TO USER'S ERROR CALL
97 053000 112776 000135 000000      MOVVB   #135,@(SP)        ;;WRITE ERROR NUMBER IN CALL
98 053006 162716 000002              SUB      #2,(SP)          ;;MOVE SP TO RETURN FOR ERROR
99 053012 004736              JSR      PC,@(SP)+        ;;REPORT ERROR VIA USER
100 053014 162716 000010              SUB      #10,(SP)         ;;MOVE SP BACK TO NO ERROR
101 053020 000240              NOP
102                                ;REPORT ERROR IF RMMR2 NOT INITIALIZED-IGNORE RQA,RQB
103 053022 013737 001374 001142      19$:   MOV      RMMR2I,$BDDAT    ;;VERIFY RMMR2
104 053030 042737 140000 001142      BIC      #RQA!RQB,$BDDAT
105 053036 012737 011777 001140      MOV      #TST!1777,$GDDAT ;;EXPECT TEST BIT ON
106 053044 023737 001140 001142      CMP      $GDDAT,$BDDAT
107 053052 001413              BEQ      21$
108 053054 062716 000004              ADD      #4,(SP)          ;;MOVE SP TO USER'S ERROR CALL
109 053060 112776 000136 000000      MOVVB   #136,@(SP)        ;;WRITE ERROR NUMBER IN CALL
110 053066 162716 000002              SUB      #2,(SP)          ;;MOVE SP TO RETURN FOR ERROR
111 053072 004736              JSR      PC,@(SP)+        ;;REPORT ERROR VIA USER
112 053074 162716 000010              SUB      #10,(SP)         ;;MOVE SP BACK TO NO ERROR
113 053100 000240              NOP
114                                ;REPORT ERROR IF RMER2 NOT RESE1-IGNORE SKI,DVC
    
```

```

115 053102 005037 001140      21$: CLR    $GDDAT      ;EXPECT ALL ZEROS
116 053106 013737 001376 001142  MOV    RMER2I,$BDDAT ;VERIFY RMER2
117 053114 042737 040200 001142  BIC    #SKI!DVC,$BDDAT ;IGNORE DEVICE ERRORS
118 053122 001413                BEQ    215$         ;BRANCH IF OTHER BITS 0
119 053124 062716 000004                ADD    #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
120 053130 112776 000174 000000  MOVB   #174,@(SP)   ;WRITE ERROR NUMBER IN CALL
121 053136 162716 000002                SUB    #2,(SP)     ;MOVE SP TO RETURN FOR ERROR
122 053142 004736                JSR    PC,@(SP)+   ;REPORT ERROR VIA USER
123 053144 162716 000010                SUB    #10,(SP)    ;MOVE SP BACK TO NO ERROR
124 053150 000240                NOP
125                                ;REPORT ERROR IF RMDS NOT INITIALIZED
126 053152 013737 001346 001142 215$: MOV    RMDSI,$BDDAT ;TEST DRIVE STATUS REGISTER
127 053160 042737 177177 001142  BIC    #^C<DRY!DPR>,$BDDAT
128 053166 012737 000600 001140  MOV    #DPR!DRY,$GDDAT ;EXPECTED DRIVE STATUS
129 053174 023737 001140 001142  CMP    $GDDAT,$BDDAT ;COMPARE EXPECTED & RECEIVED
130 053202 001413                BEQ    22$         ;BRANCH IF EQUAL
131 053204 062716 000004                ADD    #4,(SP)     ;MOVE SP TO USER'S ERROR CALL
132 053210 112776 000134 000000  MOVB   #134,@(SP)   ;WRITE ERROR NUMBER
133 053216 162716 000002                SUB    #2,(SP)     ;MOVE SP TO RETURN FOR ERROR
134 053222 004736                JSR    PC,@(SP)+   ;REPORT ERROR TO USER
135 053224 162716 000010                SUB    #10,(SP)    ;MOVE SP BACK TO NO ERROR
136 053230 000240                NOP
137 053232 062716 000004      22$: ADD    #4,(SP)     ;MOVE SP TO ERROR CALL
138 053236 105776 000000                TSTB   @(SP)       ;WAS AN ERROE DETECTED??
139 053242 001403                BEQ    23$         ;NO!!
140 053244 062716 000004                ADD    #4,(SP)     ;YES - MOVE TO ERROR RETURN
141 053250 000402                BR     24$
142 053252 162716 000004      23$: SUB    #4,(SP)     ;MOVE SP TO NO ERROR RETURN
143 053256 000240      24$: NOP
144 053260 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
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 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,RMER1I
BEQ 7$
    
```

```

: REPORT AN ERROR IF 'UNS' IS SET
BIT #UNS,RMER1I
BEQ 3$
MOV RMER1I,$BDDAT
    
```

: WAS UNS SET??
 : NO!!
 : RECEIVED STATUS

```

15 053262
18 053262 062716 000004
19 053266 105076 000000
20 053272 162716 000004
23 053276 032737 000100 001346
24 053304 001024
25 053306 013737 001346 001140
26 053314 052737 000100 001140
27 053322 013737 001346 001142
28 053330 062716 000004
29 053334 112776 000155 000000
30 053342 162716 000002
31 053346 004736
32 053350 162716 000010
33 053354 000240
34 053356
37 053356 032737 010000 001346
38 053364 001024
39 053366 013737 001346 001140
40 053374 052737 010000 001140
41 053402 013737 001346 001142
42 053410 062716 000004
43 053414 112776 000041 000000
44 053422 162716 000002
45 053426 004736
46 053430 162716 000010
47 053434 000240
48 053436
51 053436 032737 060007 001350
52 053444 001570
55 053446 032737 040000 001350
56 053454 001424
57 053456 013737 001350 001142
    
```

58	053464	013737	001350	001140	MOV	RMER1I,\$GDDAT	;EXPECTED STATUS
59	053472	042737	040000	001140	BIC	#UNS,\$GDDAT	
60	053500	062716	000004		ADD	#4,(SP)	;MOVE SP TO ERROR CALL
61	053504	112776	000042	000000	MOVB	#42,@(SP)	;WRITE NUMBER OF ERROR IN CALL
62	053512	162716	000002		SUB	#2,(SP)	;MOVE SP TO RETURN FOR ERROR
63	053516	004736			JSR	PC,@(SP)+	;REPORT THE ERROR VIA USER
64	053520	162716	000010		SUB	#10,(SP)	;MOVE SP TO NO ERROR RETURN
65	053524	000240			NOP		
66	053526					3\$:	
67							
68						;REPORT ANY OPI ERROR	
69	053526	032737	020000	001350	BIT	#OPI,RMER1I	;WAS OPI SET ??
70	053534	001424			BEQ	4\$;NO!!
71	053536	013737	001350	001142	MOV	RMER1I,\$BDDAT	;RECEIVED STATUS
72	053544	013737	001350	001140	MOV	RMER1I,\$GDDAT	;EXPECTED STATUS
73	053552	042737	020000	001140	BIC	#OPI,\$GDDAT	
74	053560	062716	000004		ADD	#4,(SP)	;MOVE SP TO ERROR CALL
75	053564	112776	000043	000000	MOVB	#43,@(SP)	;WRITE NUMBER OF ERROR IN CALL
76	053572	162716	000002		SUB	#2,(SP)	;MOVE SP TO RETURN FOR ERROR
77	053576	004736			JSR	PC,@(SP)+	;REPORT THE ERROR VIA USER
78	053600	162716	000010		SUB	#10,(SP)	;MOVE SP TO NO ERROR RETURN
79	053604	000240			NOP		
80	053606					4\$:	
81							
82						;REPORT ANY RMR ERROR	
83	053606	032737	000004	001350	BIT	#RMR,RMER1I	;WAS RMR SET??
84	053614	001424			BEQ	5\$;NO!!
85	053616	013737	001350	001142	MOV	RMER1I,\$BDDAT	;RECEIVED STATUS
86	053624	013737	001350	001140	MOV	RMER1I,\$GDDAT	;EXPECTED STATUS
87	053632	042737	000004	001140	BIC	#RMR,\$GDDAT	
88	053640	062716	000004		ADD	#4,(SP)	;MOVE SP TO ERROR CALL
89	053644	112776	000044	000000	MOVB	#44,@(SP)	;WRITE NUMBER OF ERROR IN CALL
90	053652	162716	000002		SUB	#2,(SP)	;MOVE SP TO RETURN FOR ERROR
91	053656	004736			JSR	PC,@(SP)+	;REPORT THE ERROR VIA USER
92	053660	162716	000010		SUB	#10,(SP)	;MOVE SP TO NO ERROR RETURN
93	053664	000240			NOP		
94	053666					5\$:	
95							
96						;REPORT ANY ILR ERROR	
97	053666	032737	000002	001350	BIT	#ILR,RMER1I	;WAS ILR SET??
98	053674	001424			BEQ	6\$;NO!!
99	053676	013737	001350	001142	MOV	RMER1I,\$BDDAT	;RECEIVED STATUS
100	053704	013737	001350	001140	MOV	RMER1I,\$GDDAT	;EXPECTED STATUS
101	053712	042737	000002	001140	BIC	#ILR,\$GDDAT	
102	053720	062716	000004		ADD	#4,(SP)	;MOVE SP TO ERROR CALL
103	053724	112776	000045	000000	MOVB	#45,@(SP)	;WRITE NUMBER OF ERROR IN CALL
104	053732	162716	000002		SUB	#2,(SP)	;MOVE SP TO RETURN FOR ERROR
105	053736	004736			JSR	PC,@(SP)+	;REPORT THE ERROR VIA USER
106	053740	162716	000010		SUB	#10,(SP)	;MOVE SP TO NO ERROR RETURN
107	053744	000240			NOP		
108	053746					6\$:	
109							
110						;REPORT ANY ILF ERROR	
111	053746	032737	000001	001350	BIT	#ILF,RMER1I	;WAS ILF SET??
112	053754	001424			BEQ	7\$;NO!!
113	053756	013737	001350	001142	MOV	RMER1I,\$BDDAT	;RECEIVED STATUS
114	053764	013737	001350	001140	MOV	RMER1I,\$GDDAT	;EXPECTED STATUS

115	053772	042737	000001	001140	BIC	#1LF,\$GDDAT	
116	054000	062716	000004		ADD	#4,(SP)	:MOVE SP TO ERROR CALL
117	054004	112776	000046	000000	MOVB	#46,@(SP)	:WRITE NUMBER OF ERROR IN CALL
118	054012	162716	000002		SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
119	054016	004736			JSR	PC,@(SP)+	:REPORT THE ERROR VIA USER
120	054020	162716	000010		SUB	#10,(SP)	:MOVE SP TO NO ERROR RETURN
121	054024	000240			NOP		
122	054026			7\$:			
123							
124							
125	054026	062716	000004		ADD	#4,(SP)	:MOVE SP TO ERROR CALL
126	054032	105776	000000		TSTB	@(SP)	:WAS ERROR FOUND??
127	054036	001403			BEQ	8\$:NO!!
128	054040	062716	000004		ADD	#4,(SP)	:YES - MOVE TO ERROR RETURN
129	054044	000402			BR	9\$	
130	054046	162716	000004	8\$:	SUB	#4,(SP)	:MOVE SP TO NO ERROR RETURN
131	054052	000240		9\$:	NOP		
132	054054	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
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)
CLRB @(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
MOVB #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
MOVB #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\$:

```

15 054056
18 054056 062716 000004
19 054062 105076 000000
20 054066 162716 000004
24 054072 032737 022011 001350
25 054100 001553
29 054102 032737 000010 001350
30 054110 001430
31 054112 032737 000010 001376
32 054120 001024
33 054122 013737 001350 001140
34 054130 042737 000010 001140
35 054136 013737 001350 001142
36 054144 062716 000004
37 054150 112776 000050 000000
38 054156 162716 000002
39 054162 004736
40 054164 162716 000010
41 054170 000240
42 054172
45 054172 032737 000001 001350
46 054200 001424
47 054202 013737 001350 001140
48 054210 042737 000001 001140
49 054216 013737 001350 001142
50 054224 062716 000004
51 054230 112776 000071 000000
52 054236 162716 000002
53 054242 004736
54 054244 162716 000010
55 054250 000240
56 054252
    
```

```

58      ;REPORT ANY 'OPI' ERROR AS
59      ;
60      ; . OPI DUE TO 'MOL' = 0
61      ; . OPI BECAUSE ON CYLINDER LATCH DIDN'T RESET
61 054252 032737 020000 001350      BIT      #OPI,RMER1I      ;WAS OPI SET??
62 054260 001433                      BEQ      31$              ;NO!!
63 054262 013737 001350 001140      MOV      RMER1I,$GDDAT    ;EXPECTED STATUS
64 054270 042737 020000 001140      BIC      #OPI,$GDDAT
65 054276 013737 001350 001142      MOV      RMER1I,$BDDAT    ;RECEIVED STATUS
66 054304 062716 000004              ADD      #4,(SP)          ;MOVE SP TO USER'S ERROR CALL
67 054310 112776 000072 000000      MOVVB   #72,@(SP)        ;WRITE ERROR NUMBER IN USER'S CALL
68 054316 032737 010000 001346      BIT      #MOL,RMDSI      ;WAS 'MOL' = 0??
69 054324 001403                      BEQ      3$              ;YES!!
70 054326 112776 000073 000000      MOVVB   #73,@(SP)        ;NO - CHANGE ERROR NUMBER
71 054334 162716 000002      3$:    SUB      #2,(SP)        ;MOVE SP TO RETURN FOR ERROR
72 054340 004736                      JSR      PC,@(SP)+        ;REPORT ERROR VIA USER
73 054342 162716 000010      SUB      #10,(SP)        ;MOVE SP BACK TO BRANCH
74 054346 000240                      NOP
75 054350      31$:
76
77      ;REPORT AN ERROR IF 'IAE' IS SET
78 054350 032737 002000 001350      BIT      #IAE,RMER1I     ;IS 'IAE' SET??
79 054356 001424                      BEQ      4$              ;NO!!
80 054360 013737 001350 001140      MOV      RMER1I,$GDDAT    ;EXPECTED STATUS
81 054366 042737 002000 001140      BIC      #IAE,$GDDAT
82 054374 013737 001350 001142      MOV      RMER1I,$BDDAT    ;RECEIVED STATUS
83 054402 062716 000004              ADD      #4,(SP)          ;MOVE SP TO ERROR CALL
84 054406 112776 000070 000000      MOVVB   #70,@(SP)        ;WRITE ERROR NUMBER IN USER'S CALL
85 054414 162716 000002      SUB      #2,(SP)        ;MOVE SP TO RETURN FOR ERROR
86 054420 004736                      JSR      PC,@(SP)+        ;REPORT ERROR
87 054422 162716 000010      SUB      #10,(SP)        ;MOVE SP BACK TO NO ERROR RETURN
88 054426 000240                      NOP
89 054430      4$:
90
91      ;SEE IF 'SKI' OR 'IVC' OR 'DVC' IS SET
92 054430 032737 050200 001376      BIT      #SKI!IVC!DVC,RMER2I
93 054436 001517                      BEQ      8$              ;NONE OF THE BITS ARE SET
94
95
96      ;REPORT ANY 'IVC' ERROR AS
97      ;
98      ; . IVC WITH VV = 0
99      ; . ERRONEOUS IVC ERROR
99 054440 032737 010000 001376      BIT      #IVC,RMER2I     ;WAS IVC SET??
100 054446 001433                      BEQ      6$              ;NO!!
101 054450 013737 001376 001140      MOV      RMER2I,$GDDAT    ;EXPECTED STATUS
102 054456 042737 010000 001140      BIC      #IVC,$GDDAT
103 054464 013737 001376 001142      MOV      RMER2I,$BDDAT    ;RECEIVED STATUS
104 054472 062716 000004              ADD      #4,(SP)          ;MOVE SP TO USER'S ERROR CALL
105 054476 112776 000074 000000      MOVVB   #74,@(SP)        ;WRITE ERROR NUMBER IN CALL
106 054504 032737 000100 001346      BIT      #VV,RMDSI       ;WAS VV = 0??
107 054512 001403                      BEQ      5$              ;YES!!
108 054514 112776 000075 000000      MOVVB   #75,@(SP)        ;NO - CHANGE ERROR NUMBER
109 054522 162716 000002      5$:    SUB      #2,(SP)        ;MOVE SP TO RETURN FOR ERROR
110 054526 004736                      JSR      PC,@(SP)+        ;REPORT ERROR VIA USER
111 054530 162716 000010      SUB      #10,(SP)        ;MOVE SP BACK TO BRANCH
112 054534 000240                      NOP
113 054536      6$:
114

```

```

115 ;REPORT ANY "SKI" ERROR
116 054536 032737 040000 001376 BIT #SKI,RMER2I ;WAS SKI SET??
117 054544 001424 BEQ 7$ ;NO!!
118 054546 013737 001376 001140 MOV RMER2I,$GDDAT ;EXPECTED STATUS
119 054554 042737 040000 001140 BIC #SKI,$GDDAT
120 054562 013737 001376 001142 MOV RMER2I,$BDDAT ;RECEIVED STATUS
121 054570 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
122 054574 112776 000076 000000 MOVB #76,@(SP) ;WRITE ERROR NUMBER
123 054602 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
124 054606 004736 JSR PC,@(SP)+ ;REPORT ERROR VIA USER
125 054610 162716 000010 SUB #10,(SP) ;MOVE SP TO BRANCH
126 054614 000240 NOP
127 054616 7$:
128
129 ;REPORT ANY "DVC" ERROR
130 054616 032737 000200 001376 BIT #DVC,RMER2I ;WAS "DVC" SET??
131 054624 001424 BEQ 8$ ;NO!!
132 054626 013737 001376 001140 MOV RMER2I,$GDDAT ;EXPECTED STATUS
133 054634 042737 000200 001140 BIC #DVC,$GDDAT
134 054642 013737 001376 001142 MOV RMER2I,$BDDAT ;RECEIVED STATUS
135 054650 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
136 054654 112776 000077 000000 MOVB #77,@(SP) ;WRITE ERROR NUMBER
137 054662 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
138 054666 004736 JSR PC,@(SP)+ ;REPORT ERROR VIA USER
139 054670 162716 000010 SUB #10,(SP) ;MOVE SP TO USER'S BRANCH
140 054674 000240 NOP
141 054676 8$:
142
143 ;SEE IF "PIP" AND "OM" ARE 0, AND "ATA","MOL" AND "VV" ARE 1
144 054676 013746 001346 MOV RMDSI,-(SP) ;PUT RMDS ON STACK
145 054702 042716 047676 BIC #^C<PIP!MOL!VV!OM!ATA>,(SP)
146 054706 022726 110100 CMP #ATA!MOL!VV,(SP)+
147 054712 001002 BNE 85$
148 054714 000137 055330 JMP 13$
149 054720 85$:
150
151 ;REPORT AN ERROR IF MOL = 0 AND OPI = 0, I.E., MEDIUM WENT OFF
152 ;LINE AFTER RECALIBRATE WAS INITIATED
153 054720 032737 010000 001346 BIT #MOL,RMDSI ;DID MOL DROP??
154 054726 001030 BNE 9$ ;NO!!
155 054730 032737 020000 001350 BIT #OPI,RMER1I ;WAS OPI ERROR REPORTED??
156 054736 001024 BNE 9$ ;YES - DON'T REPORT MOL = 0
157 054740 013737 001346 001140 MOV RMDSI,$GDDAT ;EXPECTED STATUS
158 054746 052737 010000 001140 BIS #MOL,$GDDAT
159 054754 013737 001346 001142 MOV RMDSI,$BDDAT ;RECEIVED STATUS
160 054762 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
161 054766 112776 000100 000000 MOVB #100,@(SP) ;WRITE ERROR NUMBER
162 054774 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
163 055000 004736 JSR PC,@(SP)+ ;REPORT ERROR VIA USER
164 055002 162716 000010 SUB #10,(SP) ;MOVE SP BACK TO USER'S BRANCH
165 055006 000240 NOP
166 055010 9$:
167
168 ;REPORT AN ERROR IF "VV" = 0 AND "IVC" = 0
169 055010 032737 000100 001346 BIT #VV,RMDSI ;DID "VV" DROP??
170 055016 001030 BNE 10$ ;NO!!
171 055020 032737 010000 001376 BIT #IVC,RMER2I ;WAS THERE A IVC ERROR??
    
```

```

172 055026 001024          BNE      10$          ;YES - DONT REPORT VV = 0
173 055030 013737 001346 001140  MOV      RMDSI,$GDDAT ;EXPECTED STATUS
174 055036 013737 001346 001142  MOV      RMDSI,$BDDAT ;RECEIVED STATUS
175 055044 052737 000100 001140  BIS      #VV,$GDDAT
176 055052 062716 000004          ADD      #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
177 055056 112776 000101 000000  MOVB    #101,@(SP)   ;WRITE ERROR NUMBER IN CALL
178 055064 162716 000002          SUB      #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
179 055070 004736          JSR     PC,@(SP)+
180 055072 162716 000010  SUB      #10,(SP)    ;MOVE SP BACK TO USER'S BRANCH
181 055076 000240  NOP
182 055100          10$:
183
184          ;REPORT AN ERROR IF ATA IS NOT SET
185 055100 032737 100000 001346  BIT      #ATA,RMDSI   ;WAS ATA SET DURING RECALIBRATE??
186 055106 001024          BNE      11$          ;YES!!
187 055110 013737 001346 001140  MOV      RMDSI,$GDDAT ;EXPECTED STATUS
188 055116 052737 100000 001140  BIS      #ATA,$GDDAT
189 055124 013737 001346 001142  MOV      RMDSI,$BDDAT ;RECEIVED STATUS
190 055132 062716 000004          ADD      #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
191 055136 112776 000102 000000  MOVB    #102,@(SP)   ;WRITE ERROR NUMBER IN CALL
192 055144 162716 000002          SUB      #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
193 055150 004736          JSR     PC,@(SP)+
194 055152 162716 000010  SUB      #10,(SP)    ;MOVE SP TO USER'S BRANCH
195 055156 000240  NOP
196
197 055160          11$:
198
199          ;REPORT AN ERROR IF 'OM' IS NOT ZERO BECAUSE RECALIBRATE SHOULD
200          ;ALWAYS CLEAR OFFSET MODE
201 055160 032737 000001 001346  BIT      #OM,RMDSI   ;WAS 'OM' RESET??
202 055166 001424          BEQ     12$          ;YES!!
203 055170 013737 001346 001140  MOV      RMDSI,$GDDAT ;EXPECTED STATUS
204 055176 042737 000001 001140  BIC      #OM,$GDDAT
205 055204 013737 001346 001142  MOV      RMDSI,$BDDAT ;RECEIVED STATUS
206 055212 062716 000004          ADD      #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
207 055216 112776 000103 000000  MOVB    #103,@(SP)   ;WRITE ERROR NUMBER
208 055224 162716 000002          SUB      #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
209 055230 004736          JSR     PC,@(SP)+
210 055232 162716 000010  SUB      #10,(SP)    ;REPORT ERROR VIA USER
211 055236 000240  NOP          ;MOVE SP TO USER'S BRANCH
212 055240          12$:
213
214          ;REPORT AN ERROR IF 'PIP' IS STIL ON, I.E., DRIVE NOT ON
215          ;CYLINDER
216 055240 032737 020000 001346  BIT      #PIP,RMDSI   ;IS DRIVE OFF CYLINDER??
217 055246 001430          BEQ     13$          ;NO!!
218 055250 032737 040000 001376  BIT      #SKI,RMER2I  ;WAS 'SKI' DETECTED??
219 055256 001024          BNE      13$          ;YES-DONT REPORT 'PIP'
220 055260 013737 001346 001140  MOV      RMDSI,$GDDAT ;EXPECTED STATUS
221 055266 042737 020000 001140  BIC      #PIP,$GDDAT
222 055274 013737 001346 001142  MOV      RMDSI,$BDDAT ;RECEIVED STATUS
223 055302 062716 000004          ADD      #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
224 055306 112776 000104 000000  MOVB    #104,@(SP)   ;WRITE ERROR NUMBER
225 055314 162716 000002          SUB      #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
226 055320 004736          JSR     PC,@(SP)+
227 055322 162716 000010  SUB      #10,(SP)    ;MOVE SP BACK TO USER'S BRANCH
228 055326 000240  NOP
    
```

```

229 055330      13$:
230
231              ;SEE IF 'ILR' OR 'RMR' OR 'UNS' IS SET
232 055330      ^32737 040006 001350      BIT      #ILR!RMR!UNS,RMER1I
233 055336      001514              BEQ      16$
234
235              ;REPORT AN ERROR IF 'ILR' IS SET
236 055340      032737 000002 001350      BIT      #ILR,RMER1I      ;WAS ILR SET DURING RECALIBRATE??
237 055346      001424              BEQ      14$      ;NO!!
238 055350      013737 001350 001140      MOV      RMER1I,$GDDAT      ;EXPECTED STATUS
239 055356      042737 000002 J01140      BIC      #ILR,$GDDAT
240 055364      013737 001350 001142      MOV      RMER1I,$BDDAT      ;RECEIVED STATUS
241 055372      062716 000004      ADD      #4,(SP) ;MOVE SP TO USER'S ERROR CALL
242 055376      112776 000105 000000      MOVSB   #105,@(SP) ;WRITE ERROR NUMBER IN CALL
243 055404      162716 000002      SUB      #2,(SP) ;MOVE SP TO RETURN FOR ERROR
244 055410      004736              JSR      PC,@(SP)+
245 055412      162716 000010      SUB      #10,(SP) ;MOVE SP TO USER'S BRANCH
246 055416      000240              NOP
247 055420
248
249              ;REPORT AN ERROR IF 'RMR' IS SET
250 055420      032737 000004 001350      BIT      #RMR,RMER1I      ;WAS RMR SET??
251 055426      001424              BEQ      15$      ;NO!!
252 055430      013737 001350 001140      MOV      RMER1I,$GDDAT      ;EXPECTED STATUS
253 055436      042737 000004 001140      BIC      #RMR,$GDDAT
254 055444      013737 001350 001142      MOV      RMER1I,$BDDAT      ;RECEIVED STATUS
255 055452      062716 000004      ADD      #4,(SP) ;MOVE SP TO USER'S ERROR CALL
256 055456      112776 000106 000000      MOVSB   #106,@(SP) ;WRITE ERROR NUMBER IN USER'S CALL
257 055464      162716 000002      SUB      #2,(SP) ;MOVE SP TO RETURN FOR ERROR
258 055470      004736              JSR      PC,@(SP)+ ;REPORT ERROR VIA USER
259 055472      162716 000010      SUB      #10,(SP) ;MOVE SP TO USER'S BRANCH
260 055476      000240              NOP
261 055500
262
263              ;REPORT AN ERROR IF 'UNS' IS SET AND 'DVC' IS 0
264 055500      032737 040000 001350      BIT      #UNS,RMER1I      ;WAS UNSAFE ON??
265 055506      001430              BEQ      16$      ;NO!!
266 055510      032737 000200 001376      BIT      #DVC,RMER2I      ;WAS THERE A DEVICE CHECK??
267 055516      001024              BNE      16$      ;YES - DON'T REPORT UNSAFE
268 055520      013737 001350 001140      MOV      RMER1I,$GDDAT      ;EXPECTED STATUS
269 055526      042737 040000 001140      BIC      #UNS,$GDDAT
270 055534      013737 001350 001142      MOV      RMER1I,$BDDAT      ;RECEIVED STATUS
271 055542      062716 000004      ADD      #4,(SP) ;MOVE SP TO USER'S ERROR CALL
272 055546      112776 000107 000000      MOVSB   #107,@(SP) ;WRITE ERROR NUMBER
273 055554      162716 000002      SUB      #2,(SP) ;MOVE SP TO RETURN FOR ERROR
274 055560      004736              JSR      PC,@(SP)+ ;REPORT ERROR VIA USER
275 055562      162716 000010      SUB      #10,(SP) ;MOVE SP BACK TO USER'S BRANCH
276 055566      000240              NOP
277 055570
278
279              ;AUGMENT THE RETURN ADDRESS IF ANY ERROR WAS DETECTED
280 055570      062716 000004      ADD      #4,(SP) ;MOVE SP TO USER'S ERROR CALL
281 055574      105776 000000      TSTB   @(SP) ;WAS AN ERROR REPORTED??
282 055600      001403              BEQ      17$      ;NO!!
283 055602      062716 000004      ADD      #4,(SP) ;YES - AUGMENT SP RETURN
284 055606      000402              BR      18$
285 055610      162716 000004      17$:   SUB      #4,(SP) ;NO ERROR - RETURN SP TO BRANCH
    
```

286 055614 000240
287 055616 000207

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          RETURN HERE TO REPORT AN ERROR
5          :      ERROR      ERROR NUMBER DEFINED BY SUB
6          :      JSR      PC,@(SP)+    GO BACK TO SUB FOR MORE ERROR CHECKS
7          :      ???          RETURN HERE IF NO MORE ERRORS
8
9 055620   DRVSTS:
10
11          :CLEAR USER'S ERROR CALL
12 055620   062716   000004          ADD      #4,(SP)          ;MOVE SP TO ERROR CALL
13 055624   105076   000000          CLRB    @(SP)          ;CLEAR ERROR CALL
14 055630   162716   000004          SUB      #4,(SP)          ;MOVE SP TO USER'S BRANCH
15
16 055634   013737   001334   001142   :REPORT ERROR IF RMCS1 NOT INITIALIZED
17 055642   042737   173700   001142   4$:  MOV      RMCS1I,$BDDAT      ;CHECK RMCS1
18 055650   012737   004010   001140   BIC      #^C<DVA!FNCMSK>,$BDDAT ;CLEAR DONT CARES
19 055656   023737   001140   001142   MOV      #DVA!DRVCLR,$GDDAT      ;EXPECT DVA
20 055664   001443          BEQ      6$              ;COMPARE EXPECTED & RECEIVED
21 055666   062716   000004          ADD      #4,(SP)          ;BRANCH IF EQUAL
22 055672   112776   000141   000000   MOVVB   #141,@(SP)        ;MOVE SP TO ERROR CALL
23 055700   162716   000002          SUB      #2,(SP)          ;WRITE NUMBER OF ERROR IN CALL
24 055704   004736          JSR      PC,@(SP)+        ;MOVE SP TO RETURN FOR ERROR
25 055706   162716   000010          SUB      #10,(SP)         ;REPORT THE ERROR VIA USER
26 055712   000240          NOP          ;MOVE SP TO NO ERROR RETURN
27
28 055714   013737   001346   001142   :REPORT ERROR IF RMDS NOT INITIALIZED
29 055722   042737   021101   001142   5$:  MOV      RMDSI,$BDDAT      ;CHECK RMDS
30 055730   012737   010600   001140   BIC      #PGM!OM!VV!PIP,$BDDAT    ;CLEAR DONT CARES
31 055736   023737   001140   001142   MOV      #MOL!DPR!DRY,$GDDAT      ;EXPECT DRY & DPR
32 055744   001413          BEQ      6$              ;COMPARE EXPECTED & RECEIVED
33 055746   062716   000004          ADD      #4,(SP)          ;BRANCH IF EQUAL
34 055752   112776   000142   000000   MOVVB   #142,@(SP)        ;MOVE SP TO ERROR CALL
35 055760   162716   000002          SUB      #2,(SP)          ;WRITE NUMBER OF ERROR IN CALL
36 055764   004736          JSR      PC,@(SP)+        ;MOVE SP TO RETURN FOR ERROR
37 055766   162716   000010          SUB      #10,(SP)         ;REPORT THE ERROR VIA USER
38 055772   000240          NOP          ;MOVE SP TO NO ERROR RETURN
39
40 055774   005037   001140          :REPORT ERROR IF RMER1 NOT INITIALIZED
41 056000   013737   001350   001142   6$:  CLR      $GDDAT          ;EXPECT 0'S
42 056006   001413          MOV      RMER1I,$BDDAT        ;CHECK RMER1
43 056010   062716   000004          BEQ      8$              ;BRANCH IF EQUAL
44 056014   112776   000143   000000   ADD      #4,(SP)          ;MOVE SP TO ERROR CALL
45 056022   162716   000002          MOVVB   #143,@(SP)        ;WRITE NUMBER OF ERROR IN CALL
46 056026   004736          SUB      #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
47 056030   162716   000010          JSR      PC,@(SP)+        ;REPORT THE ERROR VIA USER
48 056034   000240          SUB      #10,(SP)         ;MOVE SP TO NO ERROR RETURN
49
50 056036   013737   001352   001142   :REPORT ERROR IF ATA NOT INITIALIZED
51 056044   010146          8$:  MOV      RMASI,$BDDAT        ;CHECK ATTENTION BIT
52 056046   010246          MOV      R1,-(SP)          ;;PUSH R1 ON STACK
53 056050   013701   001464          MOV      R2,-(SP)          ;;PUSH R2 ON STACK
54 056054   116102   000001          MOV      TSTQUE,R1
55 056060   042702   177400          MOVVB   1(R1),R2
56 056064   005102          BIC      #^CATNMSK,R2
57 056066   040237   001142          COM      R2
          BIC      R2,$BDDAT
    
```

```

58 056072 012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
59 056074 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
60 056076 005737 001142  TST      $BDDAT      ;;IS ATTENTION CLEARED??
61 056102 001413      BEQ      9$           ;;BRANCH IF ATTENTION CLEARED
62 056104 062716 000004  ADD      #4,(SP)      ;;MOVE SP TO ERROR CALL
63 056110 112776 000144 000000  MOVVB   #144,@(SP)    ;;WRITE NUMBER OF ERROR IN CALL
64 056116 162716 000002      SUB      #2,(SP)      ;;MOVE SP TO RETURN FOR ERROR
65 056122 004736      JSR      PC,@(SP)+    ;;REPORT THE ERROR VIA USER
66 056124 162716 000010      SUB      #10,(SP)     ;;MOVE SP TO NO ERROR RETURN
67 056130 000240      NOP
68
69 056132 013737 001360 001142  :REPORT  ERROR IF RMMR1 NOT INITIALIZED
9$:      MOV      RMMR1I,$BDDAT  ;;CHECK RMMR
70 056140 042737 000046 001142  BIC      #WC!LS!LST,$BDDAT  ;;CLEAR DONT CARES
71 056146 012737 000010 001140  MOV      #MWD,$GDDAT      ;;EXPECT WRITE DATA ON
72 056154 023737 001140 001142  CMP      $GDDAT,$BDDAT    ;;COMPARE EXPECTED AND RECEIVED
73 056162 001413      BEQ      11$         ;;BRANCH IF ZERO
74 056164 062716 000004  ADD      #4,(SP)      ;;MOVE SP TO ERROR CALL
75 056170 112776 000145 000000  MOVVB   #145,@(SP)    ;;WRITE NUMBER OF ERROR IN CALL
76 056176 162716 000002      SUB      #2,(SP)      ;;MOVE SP TO RETURN FOR ERROR
77 056202 004736      JSR      PC,@(SP)+    ;;REPORT THE ERROR VIA USER
78 056204 162716 000010      SUB      #10,(SP)     ;;MOVE SP TO NO ERROR RETURN
79 056210 000240      NOP
80
81 056212 013737 001374 001142  :REPORT  ERROR IF RMMR2 NOT INITIALIZED
11$:     MOV      RMMR2I,$BDDAT  ;;CHECK RMMR2
82 056220 042737 140000 001142  BIC      #RQA!RQB,$BDDAT  ;;CLEAR REQA, REQB
83 056226 012737 011777 001140  MOV      #TST!1777,$GDDAT  ;;EXPECT TEST BIT ON
84 056234 023737 001140 001142  CMP      $GDDAT,$BDDAT    ;;COMPARE EXPECTED & RECEIVED
85 056242 001413      BEQ      15$         ;;BRANCH IF EQUAL
86 056244 062716 000004  ADD      #4,(SP)      ;;MOVE SP TO ERROR CALL
87 056250 112776 000146 000000  MOVVB   #146,@(SP)    ;;WRITE NUMBER OF ERROR IN CALL
88 056256 162716 000002      SUB      #2,(SP)      ;;MOVE SP TO RETURN FOR ERROR
89 056262 004736      JSR      PC,@(SP)+    ;;REPORT THE ERROR VIA USER
90 056264 162716 000010      SUB      #10,(SP)     ;;MOVE SP TO NO ERROR RETURN
91 056270 000240      NOP
92 056272 005037 001140 15$:     CLR      $GDDAT      ;;EXPECT ZEROS
93
94 056276 013737 001402 001142  :REPORT  ERROR IF RMEC2 NOT RESET
MOV      RMEC2I,$BDDAT  ;;CHECK RMEC2
95 056304 001413      BEQ      17$         ;;BRANCH IF 0
96 056306 062716 000004  ADD      #4,(SP)      ;;MOVE SP TO ERROR CALL
97 056312 112776 000150 000000  MOVVB   #150,@(SP)    ;;WRITE NUMBER OF ERROR IN CALL
98 056320 162716 000002      SUB      #2,(SP)      ;;MOVE SP TO RETURN FOR ERROR
99 056324 004736      JSR      PC,@(SP)+    ;;REPORT THE ERROR VIA USER
100 056326 162716 000010      SUB      #10,(SP)     ;;MOVE SP TO NO ERROR RETURN
101 056332 000240      NOP
102
103 056334 013737 001376 001142  :REPORT  ERROR IF RMER2 NOT RESET
17$:     MOV      RMER2I,$BDDAT  ;;CHECK RMER2
104 056342 001413      BEQ      18$         ;;BRANCH IF NO ERROR
105 056344 062716 000004  ADD      #4,(SP)      ;;MOVE SP TO ERROR CALL
106 056350 112776 000147 000000  MOVVB   #147,@(SP)    ;;WRITE NUMBER OF ERROR IN CALL
107 056356 162716 000002      SUB      #2,(SP)      ;;MOVE SP TO RETURN FOR ERROR
108 056362 004736      JSR      PC,@(SP)+    ;;REPORT THE ERROR VIA USER
109 056364 162716 000010      SUB      #10,(SP)     ;;MOVE SP TO NO ERROR RETURN
110 056370 000240      NOP
111 056372      18$:
112
113 056372      19$:
114

```



```
115                                     :AUGMENT RETURN ADDRESS IF ANY ERROR WAS FOUND
116 056372 062716 000004                ADD #4,(SP) ;MOVE SP TO ERROR CALL
117 056376 105776 000000                TSTB @ (SP) ;WAS AN ERROR DETECTED??
118 056402 001403                       BEQ 21$ ;NO!!
119 056404 062716 000004                ADD #4,(SP) ;YES - MOVE SP TO ERROR RETURN
120 056410 000402                       BR 23$
121 056412 162716 000004                21$: SUB #4,(SP) ;MOVE SP BACK TO NO ERROR RETURN
122 056416 000240                       23$: NOP
123 056420 000207                       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,RMER1I ;WAS PARITY ERROR DETECTED??
BEQ 10$ ;NO!!
BIT #DPE,RMER2I ;WAS IT CONTROL BUS ERROR??
BNE 10$ ;PROBABLY NOT!!
;REPORT CONTROL BUS PARITY ERROR VIA USER'S ERROR CALL
MOV RMER1I,$GDDAT ;EXPECTED STATUS
BIC #PAR,$GDDAT
```

40	056422			
43	056422	062716	000004	
44	056426	105076	000000	
45	056432	162716	000004	
46	056436	005037	057764	
50	056442	032737	000010	001350
51	056450	001431		
52	056452	032737	000010	001376
53	056460	001025		
56	056462	013737	001350	001140
57	056470	042737	000010	001140

```

58 056476 013737 001350 001142      MOV      RMER11,$BDDAT      ;RECEIVED STATUS
59 056504 062716 000004                ADD      #4,(SP)           ;MOVE SP TO USER'S ERROR CALL
60 056510 112776 000050 000000      MOVVB   #50,@(SP)         ;WRITE ERROR NUMBER IN CALL
61 056516 162716 000002                SUB      #2,(SP)           ;MOVE SP TO RETURN IF ERROR
62 056522 004736                JSR      PC,@(SP)+         ;REPORT ERROR
63 056524 162716 000010      SUB      #10,(SP)         ;RESTORE SP TO NO ERROR
64 056530 000240                NOP
65 056532 000430                BR       15$              ;SKIP FURTHER ERROR CHECKS
66 056534
67
68                                     ;TEST FOR MASSBUS CONTROL BUS PARITY ERROR WHEN READING REMOTE
69                                     ;REGISTERS, I.E., 'MCPE' = 1.
70 056534 032737 020000 001334      BIT      #MCPE,RMCS11     ;WAS PARITY ERROR DETECTED??
71 056542 001426                BEQ      20$              ;NO!!
72
73                                     ;REPORT CONTROL BUS PARITY ERROR VIA USER'S ERROR CALL.
74 056544 013737 001334 001140      MOV      RMCS11,$GDDAT    ;EXPECTED STATUS
75 056552 042737 020000 001140      BIC      #MCPE,$GDDAT
76 056560 013737 001334 001142      MOV      RMCS11,$BDDAT    ;RECEIVED STATUS
77 056566 062716 000004                ADD      #4,(SP)           ;MOVE SP TO USER'S ERROR CALL
78 056572 112776 000013 000000      MOVVB   #13,@(SP)         ;WRITE ERROR NUMBER
79 056600 162716 000002                SUB      #2,(SP)           ;MOVE SP TO RETURN IF ERROR
80 056604 004736                JSR      PC,@(SP)+         ;REPORT ERROR AND RETURN
81 056606 162716 000010      SUB      #10,(SP)         ;RESTORE SP TO NO ERROR
82 056612 000240                NOP
83 056614 000137 057736      15$:   JMP      150$         ;OMIT STATUS CHECKING
84 056620      20$:
85
86                                     ;DETERMINE THE VALUE OF "IAE" STATUS BASED ON TRACK, SECTOR AND CYLINDER
87                                     ;ALSO SET "SKI" FLAG IF CYLINDER ADDRESS IS TOO LARGE.
88 056620 012737 002000 001140      MOV      #IAE,$GDDAT     ;SETUP FOR IAE = 1
89 056626 052737 040000 057764      BIS      #SKI,200$        ;SETUP FOR SKI = 1
90 056634 023727 001444 001466      CMP      RMDCO,#822.     ;GREATER THAN LAST CYLINDER ?
91 056642 101025                BHI      30$              ;YES - CYLINDER IS INVALID
92 056644 042737 040000 057764      BIC      #SKI,200$        ;"SKI" SHOULD BE ZERO
93
94 056652 123737 001417 001333      CMPB    RMDAO+1,LSTRK+1  ;GREATER THAN LAST TRACK ?
95 056660 101016                BHI      30$              ;YES - TRACK IS INVALID
96
97 056662 123727 001416 000035      CMPB    RMDAO,#29.       ;SECTOR > 29. ?
98 056670 101410                BLOS    25$              ;NO!!
99 056672 032737 010000 001442      BIT      #FMT16,RMOFO    ;18 BIT FORMAT ?
100 056700 001406                BEQ     30$              ;YES - SECTOR IS INVALID FOR 18 BIT MODE
101 056702 123727 001416 000037      CMPB    RMDAO,#31.       ;SECTOR > 31. ?
102 056710 101002                BHI     30$              ;YES - SECTOR IS INVALID
103
104 056712 005037 001140      25$:   CLR      $GDDAT         ;"IAE" SHOULD = 0
105
106                                     ;COMPARE EXPECTED AND RECIVED "IAE" STATUS
107 056716 013737 001350 001142      30$:   MOV      RMER11,$BDDAT    ;GET RECEIVED IAE
108 056724 042737 175777 001142      BIC      #^CIAE,$BDDAT
109 056732 023737 001140 001142      CMP      $GDDAT,$BDDAT   ;IS IAE CORRECT??
110 056740 001004                BNE     40$              ;NO!!
111 056742 042737 040000 057764      BIC      #SKI,200$        ;SKI SHOULD BE ZERO
112 056750 000413                BR       50$
113 056752      40$:
114

```

```

115 ;REPORT INCORRECT IAE STATUS VIA USER'S ERROR CALL
116 056752 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
117 056756 112776 000257 000000 MOVB #257,@(SP) ;WRITE ERROR NUMBER IN CALL
118 056764 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
119 056770 004736 JSR PC,@(SP)+ ;REPORT ERROR AND RETURN
120 056772 162716 000010 SUB #10,(SP) ;RETURN SP TO NO ERROR
121 056776 000240 NOP
122 057000 50$:
123
124 ;SEE IF 'SKI' OR 'DVC' OR 'IVC' IS SET
125 057000 032737 050200 001376 BIT #SKI!DVC!IVC,RMER2I ;ARE ANY BITS SET??
126 057006 001531 BEQ 90$ ;NO!!
127
128 ;REPORT ERROR IF 'SKI' IS SET AND 'SKI' FLAG IS NOT SET
129 057010 013737 001376 001142 MOV RMER2I,$BDDAT ;RECEIVED 'SKI' STATUS
130 057016 042737 137777 001142 BIC #^CSKI,$BDDAT
131 057024 013737 057764 001140 MOV 200$,$GDDAT ;EXPECTED 'SKI' STATUS
132 057032 042737 137777 001140 BIC #^CSKI,$GDDAT
133 057040 023737 001140 001142 CMP $GDDAT,$BDDAT ;IS 'SKI' OK??
134 057046 001422 BEQ 60$ ;YES-CHANGE ERROR NUMBER
135 057050 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
136 057054 112776 000263 000000 MOVB #263,@(SP) ;WRITE ERROR NUMBER
137 057062 032737 040000 001140 BIT #SKI,$GDDAT ;SHOULD 'SKI' BE SET??
138 057070 001403 BEQ 55$ ;NO!!
139 057072 112776 000267 000000 MOVB #267,@(SP) ;YES-CHANGE ERROR NUMBER
140 057100 162716 000002 55$: SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
141 057104 004736 JSR PC,@(SP)+ ;REPORT ERROR AND RETURN
142 057106 162716 000010 SUB #10,(SP) ;RESTORE SP TO NO ERROR
143 057112 000240 NOP
144 057114 60$:
145
146 ;REPORT 'IVC' ERROR AS
147 : .IVC DUE TO LOSS OF VOLUME VALID
148 : .ERRONEOUS IVC WITH VOLUME VALID SET
149 057114 032737 010000 001376 BIT #IVC,RMER2I ;IS IVC SET??
150 057122 001433 BEQ 80$ ;NO!!
151 057124 013737 001376 001140 MOV RMER2I,$GDDAT ;EXPECTED STATUS
152 057132 042737 010000 001140 BIC #IVC,$GDDAT
153 057140 013737 001376 001142 MOV RMER2I,$BDDAT ;RECEIVED STATUS
154 057146 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
155 057152 112776 000264 000000 MOVB #264,@(SP) ;WRITE ERROR NUMBER IN CALL
156 057160 032737 000100 001346 BIT #VV,RMDSI ;WAS VOLUME VALID??
157 057166 001403 BEQ 70$ ;NO!!
158 057170 112776 000265 000000 MOVB #265,@(SP) ;YES - CHANGE ERROR NUMBER
159 057176 162716 000002 70$: SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
160 057202 004736 JSR PC,@(SP)+ ;REPORT ERROR AND RETURN
161 057204 162716 000010 SUB #10,(SP) ;RESTORE SP TO NO ERROR
162 057210 000240 NOP
163 057212 80$:
164
165 ;REPORT ANY DEVICE FAULT, I.E., 'DVC' = 1
166 057212 032737 000200 001376 BIT #DVC,RMER2I ;WAS THERE A DEVICE FAULT??
167 057220 001424 BEQ 90$ ;NO!!
168 057222 013737 001376 001140 MOV RMER2I,$GDDAT ;EXPECTED STATUS
169 057230 042737 000200 001140 BIC #DVC,$GDDAT
170 057236 013737 001376 001142 MOV RMER2I,$BDDAT ;RECEIVED STATUS
171 057244 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL

```

```
172 057250 112776 000266 000000      MOVB    #266,@(SP)      ;WRITE ERROR NUMBER IN CALL
173 057256 162716 000002              SUB     #2,(SP)        ;MOVE SP TO RETURN IF ERROR
174 057262 004736                    JSR    PC,@(SP)+      ;REPORT ERROR AND RETURN
175 057264 162716 000010              SUB     #10,(SP)       ;RESTORE SP TO NO ERROR
176 057270 000240                    NOP
177 057272                      90$:
178
179
180      ;REPORT ANY "OPI" ERROR AS
181      ;"OPI" BECAUSE MEDIUM IS NOT ONLINE
182      ;"OPI" BECAUSE "ON CYLINDER" DIDN'T DROP
182 057272 032737 020000 001350      BIT     #OPI,RMER1I    ;WAS OPI SET??
183 057300 001433                    BEQ    110$            ;NO!!
184 057302 013737 001350 001140      MOV     RMER1I,$GDDAT  ;EXPECTED STATUS
185 057310 042737 020000 001140      BIC    #OPI,$GDDAT
186 057316 013737 001350 001142      MOV     RMER1I,$BDDAT  ;RECEIVED STATUS
187 057324 062716 000004              ADD     #4,(SP)        ;MOVE SP TO USER'S ERROR CALL
188 057330 112776 000270 000000      MOVB   #270,@(SP)     ;SETUP ERROR FOR MOL = 0
189 057336 032737 010000 001346      BIT     #MOL,RMDSI    ;WAS MOL 0??
190 057344 001403                    BEQ    105$            ;YES!!
191 057346 112776 000271 000000      MOVB   #271,@(SP)     ;MOL WAS 1 - CHANGE ERROR
192 057354 162716 000002 105$:      SUB     #2,(SP)        ;MOVE SP TO RETURN IF ERROR
193 057360 004736                    JSR    PC,@(SP)+      ;REPORT ERROR AND RETURN
194 057362 162716 000010              SUB     #10,(SP)       ;RESTORE SP TO NO ERROR
195 057366 000240                    NOP
196 057370                      110$:
197
198      ;SEE IF "ATA" = "MOL" = "VV" = 1 AND "PIP" = 0
199 057370 013746 001346 001346      MOV     RMDSI,-(SP)    ;GET DRIVE STATUS
200 057374 042716 047677              BIC    #^C<ATA!PIP!MOL!VV>,(SP)
201 057400 022726 110100              CMP    #ATA!MOL!VV,(SP)+ ;IS DRIVE STATUS CORRECT??
202 057404 001554                    BEQ    150$            ;YES!!
203
204      ;REPORT AN ERROR IF MOL = 0 AND OPI ERROR WAS NOT REPORTED, I.E., OPI = 0
205 057406 032737 010000 001346      BIT     #MOL,RMDSI    ;WAS MEDIUM OFF LINE??
206 057414 001030                    BNE    120$            ;NO!!
207 057416 013737 001346 001140      MOV     RMDSI,$GDDAT  ;EXPECTED STATUS
208 057424 052737 010000 001140      BIS    #MOL,$GDDAT
209 057432 013737 001346 001142      MOV     RMDSI,$BDDAT  ;RECEIVED STATUS
210 057440 032737 020000 001350      BIT     #OPI,RMER1I    ;WAS OPI REPORTED BEFORE??
211 057446 001013                    BNE    120$            ;YES - DON'T REPORT MOL = 0
212 057450 062716 000004              ADD     #4,(SP)        ;MOVE SP TO USER'S ERROR CALL
213 057454 112776 000272 000000      MOVB   #272,@(SP)     ;WRITE ERROR NUMBER IN CALL
214 057462 162716 000002              SUB     #2,(SP)        ;MOVE SP TO RETURN IF ERROR
215 057466 004736                    JSR    PC,@(SP)+      ;REPORT ERROR AND RETURN
216 057470 162716 000010              SUB     #10,(SP)       ;RESTORE SP TO NO ERROR
217 057474 000240                    NOP
218 057476                      120$:
219
220      ;REPORT AN ERROR IF PIP IS STIL SET AND SKI IS RESET
221 057476 032737 020000 001346      BIT     #PIP,RMDSI    ;IS POSITIONING IN PROGRESS??
222 057504 001430                    BEQ    130$            ;NO!!
223 057506 032737 040000 001376      BIT     #SKI,RMER2I    ;WAS "SKI" DETECTED??
224 057514 001024                    BNE    130$            ;YES-DONT REPORT PIP
225 057516 013737 001346 001140      MOV     RMDSI,$GDDAT  ;EXPECTED STATUS
226 057524 042737 020000 001140      BIC    #PIP,$GDDAT
227 057532 013737 001346 001142      MOV     RMDSI,$BDDAT  ;RECEIVED STATUS
228 057540 062716 000004              ADD     #4,(SP)        ;MOVE SP TO USER'S ERROR CALL
```

```

SEARCH STATUS CHECK SUBROUTINE

229 057544 112776 000273 000000      MOVB   #273,@(SP)      ;WRITE ERROR NUMBER IN CALL
230 057552 162716 000002              SUB    #2,(SP)         ;MOVE SP TO RETURN IF ERROR
231 057556 004736                    JSR    PC,@(SP)+       ;REPORT ERROR AND RETURN
232 057560 162716 000010      SUB    #10,(SP)       ;RESTORE SP TO NO ERROR
233 057564 000240                    NOP
234 057566                    130$:
235
236                    ;REPORT AN ERROR IF VOLUME IS NOT VALID AND IVC = 0
237 057566 032737 000100 001346      BIT    #VV,RMDSI      ;IS VOLUME VALID??
238 057574 001030                    BNE    140$           ;YES!!
239 057576 032737 010000 001376      BIT    #IVC,RMER2I    ;WAS IVC DETECTED??
240 057604 001024                    BNE    140$           ;YES - DON'T REPORT VV = 0
241 057606 013737 001346 001140      MOV    RMDSI,$GDDAT   ;EXPECTED STATUS
242 057614 052737 000100 001140      BIS    #VV,$GDDAT
243 057622 013737 001346 001142      MOV    RMDSI,$BDDAT   ;RECEIVED STATUS
244 057630 062716 000004                    ADD    #4,(SP)        ;MOVE SP TO USERS ERROR CALL
245 057634 112776 000350 000000      MOVB   #350,@(SP)     ;WRITE ERROR NUMBER IN CALL
246 057642 162716 000002              SUB    #2,(SP)         ;MOVE SP TO RETURN IF ERROR
247 057646 004736                    JSR    PC,@(SP)+       ;REPORT ERROR AND RETURN
248 057650 162716 000010      SUB    #10,(SP)       ;RESTORE SP TO NO ERROR
249 057654 000240                    NOP
250 057656                    140$:
251
252                    ;REPORT AN ERROR IF ATTENTION IS NOT SET
253 057656 032737 100000 001346      BIT    #ATA,RMDSI     ;IS ATA ON??
254 057664 001024                    BNE    150$           ;YES!!
255 057666 013737 001346 001140      MOV    RMDSI,$GDDAT   ;EXPECTED STATUS
256 057674 052737 100000 001140      BIS    #ATA,$GDDAT
257 057702 013737 001346 001142      MOV    RMDSI,$BDDAT   ;RECEIVED STATUS
258 057710 062716 000004                    ADD    #4,(SP)        ;MOVE SP TO USER'S ERROR CALL
259 057714 112776 000351 000000      MOVB   #351,@(SP)     ;WRITE ERROR NUMBER IN CALL
260 057722 162716 000002              SUB    #2,(SP)         ;MOVE SP TO RETURN IF ERROR
261 057726 004736                    JSR    PC,@(SP)+       ;REPORT ERROR AND RETURN
262 057730 162716 000010      SUB    #10,(SP)       ;RESTORE SP TO NO ERROR
263 057734 000240                    NOP
264 057736                    150$:
265
266                    ;ARGUMENT THE RETURN ADDRESS IF AN ERROR WAS DETECTED
267 057736 062716 000004                    ADD    #4,(SP)        ;MOVE SP TO USER'S ERROR CALL
268 057742 105776 000000                    TSTB   @(SP)          ;WAS ERROR FOUND??
269 057746 001403                    BEQ    160$           ;NO!!
270 057750 062716 000004                    ADD    #4,(SP)        ;YES - CHANGE RETURN
271 057754 000402                    BR     170$
272 057756 162716 000004      160$: SUB    #4,(SP)     ;NO ERROR FOUND
273 057762 000207      170$: RTS    PC      ;RETURN TO USER
274
275 057764 000000      200$: .WORD    ;STORAGE FOR FLAGS

```

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 STATIC DRIVE STATUS CHECK SUBROUTINE

:THIS SUBROUTINE LOOKS FOR UNEXPECTED CHANGES IN DRIVE
 :STATUS, SUCH AS THE DRIVE LOSING VOLUME VALID. THE SUBROUTINE
 :CAN BE USED BY HOUSEKEEPING AND OTHER COMMANDS DURING WHICH THERE
 :SHOULD NOT BE ANY DRIVE ERRORS OR CHANGES IN STATE.

:THE FOLLOWING CONDITIONS ARE TESTED AND REPORTED AS ERRORS
 :IF TRUE:

- : .MOL = 0, INDICATES DRIVE WENT OFFLINE, NOTE
- : THAT MOL IS ASSUMED TO HAVE BEEN SET
- : .VV = 0, INDICATES THE DRIVE LOST VOLUME VALID
- : .PIP = 1, INDICATES THAT THE DRIVE IS OFF CYLINDER
- : .SKI = 1, INDICATES THE DRIVE HAS AN UNEXPECTED SKI ERROR
- : .DVC = 1, INDICATES AN UNEXPECTED DEVICE FAULT

:THE SUBROUTINE IS CALLED AFTER STORING STATUS IN THE GET BUFFER.

```

:(1) JSR PC,STCDRVSTS
: 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
    
```

STCDRVSTS:

```

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

```

: SEE IF 'MOL' = 'VV' = 1, AND 'PIP' = 0
: MOV RMDSI,-(SP) ;PUT DRIVE STATUS ON STACK
: BIC #^C<PIP!MOL!VV>,(SP)
: CMP #MOL!VV,(SP)+ ;ARE MOL,VV AND PIP O.K.??
: BEQ 30$ ;YES!!
    
```

```

: REPORT AN ERROR IF MOL = 0 AND 'OPI' = 0
: BIT #MOL,RMDSI ;IS MOL ON ??
: BNE 10$ ;YES!!
: BIT #OPI,RMER11 ;WAS 'OPI' SET??
: BNE 10$ ;YES-DONT REPORT 'MOL' = 0
: MOV RMDSI,$GDDAT ;EXPECTED STATUS
: BIS #MOL,$GDDAT
: MOV RMDSI,$BDDAT ;RECEIVED STATUS
: ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
: MCVB #207,@(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 RETURN
: NOP
    
```

10\$:

```

: REPORT AN ERROR IF VOLUME VALID IS NOW ZERO AND 'IVC' = 0
: BIT #VV,RMDSI ;IS 'VV' = 0??
: BNE 20$ ;NO!!
    
```

057766				
057766	062716	000004		
057772	105076	000000		
057776	162716	000004		
060002	013746	001346		
060006	042716	147677		
060012	022726	010100		
060016	001524			
060020	032737	010000	001346	
060026	001030			
060030	032737	020000	001350	
060036	001024			
060040	013737	001346	001140	
060046	052737	010000	001140	
060054	013737	001346	001142	
060062	062716	000004		
060066	112776	000207	000000	
060074	162716	000002		
060100	004736			
060102	162716	000010		
060106	000240			
060110				
060110	032737	000100	001346	
060116	001030			

```

58 060120 032737 010000 001376      BIT      #IVC,RMER2I      ;WAS 'IVC' SET??
59 060126 001024                      BNE      20$            ;YES-DONT REPORT 'VV' = 0
60 060130 013737 001346 001140      MOV      RMDSI,$GDDAT   ;EXPECTED STATUS
61 060136 052737 000100 001346      BIS      #VV,RMDSI
62 060144 013737 001346 001142      MOV      RMDSI,$BDDAT   ;RECEIVED STATUS
63 060152 062716 000004                      ADD      #4,(SP) ;MOVE SP TO USER'S ERROR CALL
64 060156 112776 000210 000000      MOVVB   #210,@(SP)     ;WRITE ERROR NUMBER IN CALL
65 060164 162716 000002                      SUB      #2,(SP)       ;MOVE SP TO RETURN FOR ERROR
66 060170 004736                      JSR      PC,@(SP)+     ;REPORT ERROR VIA USER
67 060172 162716 000010                      SUB      #10,(SP)      ;MOVE SP BACK TO NO ERROR
68 060176 000240                      NOP
69 060200                                20$:
70
71                                ;REPORT AN ERROR IF DRIVE IS OFF CYLINDER AND 'SKI' = 0
72 060200 032737 020000 001346      BIT      #PIP,RMDSI     ;IS DRIVE OFF CYLINDER??
73 060206 001430                      BEQ      30$            ;NO!!
74 060210 032737 040000 001376      BIT      #SKI,RMER2I    ;WAS 'SKI' SET??
75 060216 001024                      BNE      30$            ;YES-DONT REPORT 'PIP' = 1
76 060220 013737 001346 001140      MOV      RMDSI,$GDDAT   ;EXPECTED STATUS
77 060226 042737 020000 001140      BIC      #PIP,$GDDAT
78 060234 013737 001346 001142      MOV      RMDSI,$BDDAT   ;RECEIVED STATUS
79 060242 062716 000004                      ADD      #4,(SP)       ;MOVE SP TO USER'S ERROR CALL
80 060246 112776 000211 000000      MOVVB   #211,@(SP)     ;WRITE ERROR NUMBER IN USER'S CALL
81 060254 162716 000002                      SUB      #2,(SP)       ;MOVE SP TO RETURN FOR ERROR
82 060260 004736                      JSR      PC,@(SP)+     ;REPORT ERROR VIA USER
83 060262 162716 000010                      SUB      #10,(SP)      ;MOVE SP TO NO ERROR RETURN
84 060266 000240                      NOP
85 060270                                30$:
86
87                                ;SEE IF 'SKI' = 'DVC' = 0
88 060270 013746 001376                      MOV      RMER2I,-(SP)   ;PUT ERROR REG 2 ON STACK
89 060274 042726 137577                      BIC      #^C<SKI!DVC>,(SP)+
90 060300 001460                      BEQ      60$            ;BRANCH IF NO ERROR
91 060302                                40$:
92
93                                ;REPORT AN ERROR IF THERE IS A DEVICE FAULT
94 060302 032737 000200 001376      BIT      #DVC,RMER2I    ;ANY DEVICE FAULT??
95 060310 001424                      BEQ      50$            ;NO!!
96 060312 013737 001376 001140      MOV      RMER2I,$GDDAT  ;EXPECTED STATUS
97 060320 042737 000200 001140      BIC      #DVC,$GDDAT
98 060326 013737 001376 001142      MOV      RMER2I,$BDDAT  ;RECEIVED STATUS
99 060334 062716 000004                      ADD      #4,(SP)       ;MOVE SP TO USER'S CALL
100 060340 112776 000212 000000     MOVVB   #212,@(SP)     ;WRITE NUMBER OF ERROR IN CALL
101 060346 162716 000002                      SUB      #2,(SP)       ;MOVE SP TO RETURN FOR ERROR
102 060352 004736                      JSR      PC,@(SP)+     ;REPORT ERROR VIA USER
103 060354 162716 000010                      SUB      #10,(SP)      ;MOVE SP BACK TO NO ERROR
104 060360 000240                      NOP
105 060362                                50$:
106
107                                ;REPORT AN ERROR IF 'SKI' = 1
108 060362 032737 040000 001376      BIT      #SKI,RMER2I    ;IS THERE A SEEK INCOMPLETE ERROR
109 060370 001424                      BEQ      60$            ;NO!!
110 060372 013737 001376 001140      MOV      RMER2I,$GDDAT  ;EXPECTED STATUS
111 060400 042737 040000 001140      BIC      #SKI,$GDDAT
112 060406 013737 001376 001142      MOV      RMER2I,$BDDAT  ;RECEIVED STATUS
113 060414 062716 000004                      ADD      #4,(SP)       ;MOVE SP TO USER'S ERROR CALL
114 060420 112776 000213 000000     MOVVB   #213,@(SP)     ;WRITE ERROR NUMBER IN USER'S ERROR CALL
    
```



```

115 060426 162716 000002          SUB    #2,(SP)      :MOVE SP TO RETURN FOR ERROR
116 060432 004736                JSR    PC,@(SP)+   :REPORT ERROR VIA USER
117 060434 162716 000010          SUB    #10,(SP)    :MOVE SP BACK TO NO ERROR
118 060440 000240                NOP
119 060442                60$:
120
121                ;AUGMENT THE RETURN ADDRESS IF ANY ERROR WAS DETECTED
122 060442 062716 000004          ADD    #4,(SP)     :MOVE SP TO USER'S ERROR CALL
123 060446 105776 000000          TSTB  @(SP)       :WAS AN ERROR DETECTED??
124 060452 001403                BEQ    70$         :NO!!
125 060454 062716 000004          ADD    #4,(SP)     :YES - MOVE SP TO USER'S ERROR RETURN
126 060460 000402                BR     80$
127 060462 162716 000004          70$: SUB    #4,(SP)    :NO - MOVE SP TO NO ERROR RETURN
128 060466 000240          80$: NOP
129 060470 000207                RTS    PC          :RETURN TO USER

```

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
    
```

```

060472
060472 010046
060474 010146
060476 010246
060500 010346
060502 010446
060504 010546
060506 016646 000022
060512 016646 000022
060516 016646 000022
060522 016646 000022
060526 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
    
```

```

060530
060530 012666 000022
060534 012666 000022
060540 012666 000022
060544 012666 000022
060550 012605
060552 012604
060554 012603
060556 012602
060560 012601
060562 012600
060564 000002
    
```

```

:*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
    
```

.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
*
060566 010146          $TYPBN: MOV      R1,-(SP)      ;;SAVE R1 ON THE STACK
060570 016601 000006  MOV      6(SP),R1      ;;GET THE INPUT NUMBER
060574 000261          SEC                          ;;SET 'C' SO CAN KEEP TRACK OF THE NUMBER OF BITS
060576 112737 000060 060640 1$:  MOVVB  #'0,$BIN      ;;SET CHARACTER TO AN ASCII '0'.
060604 006101          ROL      R1                ;;GET THIS BIT
060606 001406          BEQ      2$                ;;DONE?
060610 105537 060640  ADCB     $BIN          ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
060614 104401 060640  TYPE     ,$BIN          ;;GO TYPE THIS BIT
060620 000241          CLC                          ;;CLEAR 'C' SO CAN KEEP TRACK OF BITS
060622 000765          BR       1$                ;;GO DO THE NEXT BIT
060624 012601          MOV      (SP)+,R1          ;;POP THE STACK INTO R1
060626 016666 000002 000004 2$:  MOV      2(SP),4(SP)      ;;ADJUST THE STACK
060634 012616          MOV      (SP)+,(SP)
060636 000002          RTI                          ;;RETURN TO USER
060640      000      000  $BIN:  .BYTE  0,0      ;;STORAGE FOR ASCII CHAR. AND TERMINATOR
    
```

.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

060642				\$TYPDS:	MOV	R0,-(SP)	::PUSH R0 ON STACK
060642	010046				MOV	R1,-(SP)	::PUSH R1 ON STACK
060644	010146				MOV	R2,-(SP)	::PUSH R2 ON STACK
060646	010246				MOV	R3,-(SP)	::PUSH R3 ON STACK
060650	010346				MOV	R5,-(SP)	::PUSH R5 ON STACK
060652	010546				MOV	#20200,-(SP)	::SET BLANK SWITCH AND SIGN
060654	012746	020200			MOV	20(SP),R5	::GET THE INPUT NUMBER
060660	016605	000020			BPL	1\$::BR IF INPUT IS POS.
060664	100004				NEG	R5	::MAKE THE BINARY NUMBER POS.
060666	005405				MOVB	#'-,1(SP)	::MAKE THE ASCII NUMBER NEG.
060670	112766	000055	000001	1\$:	CLR	R0	::ZERO THE CONSTANTS INDEX
060676	005000				MOV	#\$DBLK,R3	::SETUP THE OUTPUT POINTER
060700	012703	061056			MOVB	#' ,(R3)+	::SET THE FIRST CHARACTER TO A BLANK
060704	112723	000040		2\$:	CLR	R2	::CLEAR THE BCD NUMBER
060710	005002				MOV	\$DTBL(R0),R1	::GET THE CONSTANT
060712	016001	061046		3\$:	SUB	R1,R5	::FORM THIS BCD DIGIT
060716	160105				BLT	4\$::BR IF DONE
060720	002402				INC	R2	::INCREASE THE BCD DIGIT BY 1
060722	005202				BR	3\$	
060724	000774				ADD	R1,R5	::ADD BACK THE CONSTANT
060726	060105			4\$:	TST	R2	::CHECK IF BCD DIGIT=0
060730	005702				BNE	5\$::FALL THROUGH IF 0
060732	001002				TSTB	(SP)	::STILL DOING LEADING 0'S?
060734	105716				BMI	7\$::BR IF YES
060736	100407				ASLB	(SP)	::MSD?
060740	106316			5\$:	BCC	6\$::BR IF NO
060742	103003				MOVB	1(SP),-1(R3)	::YES--SET THE SIGN
060744	116663	000001	177777	6\$:	BIS	#'0,R2	::MAKE THE BCD DIGIT ASCII
060752	052702	000060		7\$:	BIS	#' ,R2	::MAKE IT A SPACE IF NOT ALREADY A DIGIT
060756	052702	000040			MOVB	R2,(R3)+	::PUT THIS CHARACTER IN THE OUTPUT BUFFER
060762	110223				TST	(R0)+	::JUST INCREMENTING
060764	005720				CMP	R0,#10	::CHECK THE TABLE INDEX
060766	020027	000010			BLT	2\$::GO DO THE NEXT DIGIT
060772	002746				BGT	8\$::GO TO EXIT
060774	003002				MOV	R5,R2	::GET THE LSD
060776	010502				BR	6\$::GO CHANGE TO ASCII
061000	000764			8\$:	TSTB	(SP)+	::WAS THE LSD THE FIRST NON-ZERO?
061002	105726				BPL	9\$::BR IF NO
061004	100003				MOVB	-1(SP),-2(R3)	::YES--SET THE SIGN FOR TYPING
061006	116663	177777	177776	9\$:	CLRB	(R3)	::SET THE TERMINATOR
061014	105013				MOV	(SP)+,R5	::POP STACK INTO R5
061016	012605				MOV	(SP)+,R3	::POP STACK INTO R3
061020	012603				MOV	(SP)+,R2	::POP STACK INTO R2
061022	012602				MOV	(SP)+,R1	::POP STACK INTO R1
061024	012601						

061026	012600			MOV	(SP)+,R0	::POP STACK INTO R0
061030	104401	061056		TYPE	,SDBLK	::NOW TYPE THE NUMBER
061034	016666	000002	000004	MOV	2(SP),4(SP)	::ADJUST THE STACK
061042	012616			MOV	(SP)+,(SP)	
061044	000002			RTI		::RETURN TO USER
061046	023420		\$DTBL:	10000.		
061050	001750			1000.		
061052	000144			100.		
061054	000012			10.		
061056			\$DBLK:	.BLKW 4		

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

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

```

```

061066 017646 000000
061072 116637 000001 061311
061100 112637 061313
061104 062716 000002
061110 000406
061112 112737 000001 061311
061120 112737 000006 061313
061126 112737 000005 061310
061134 010346
061136 010446
061140 010546
061142 113704 061313
061146 005404
061150 062704 000006
061154 110437 061312
061160 113704 061311
061164 016605 000012
061170 005003
061172 006105 1$:
061174 000404 BR 3$
061176 006105 2$:
061200 006105 ROL R5
061202 006105 ROL R5
061204 010503 MOV R5,R3
061206 006103 3$:
061210 105337 061312 ROL R3
061214 100016 DECB $OMODE
061216 042703 177770 BPL 7$
061222 001002 BIC #177770,R3
061224 005704 BNE 4$
061226 001403 TST R4
061230 005204 BEQ 5$
                                4$: INC R4
                                5$:

```

```

$TYPOS: MOV @ (SP),-(SP) ;;PICKUP THE MODE
        MOVB 1(SP),$OFILL ;;LOAD ZERO FILL SWITCH
        MOVB (SP)+,$OMODE+1 ;;NUMBER OF DIGITS TO TYPE
        ADD #2,(SP) ;;ADJUST RETURN ADDRESS
        BR $TYPON
$TYPOC: MOVB #1,$OFILL ;;SET THE ZERO FILL SWITCH
        MOVB #6,$OMODE+1 ;;SET FOR SIX(6) DIGITS
$TYPON: MOVB #5,$OCNT ;;SET THE ITERATION COUNT
        MOV R3,-(SP) ;;SAVE R3
        MOV R4,-(SP) ;;SAVE R4
        MOV R5,-(SP) ;;SAVE R5
        MOVB $OMODE+1,R4 ;;GET THE NUMBER OF DIGITS TO TYPE
        NEG R4
        ADD #6,R4 ;;SUBTRACT IT FOR MAX. ALLOWED
        MOVB R4,$OMODE ;;SAVE IT FOR USE
        MOVB $OFILL,R4 ;;GET THE ZERO FILL SWITCH
        MOV 12(SP),R5 ;;PICKUP THE INPUT NUMBER
        CLR R3 ;;CLEAR THE OUTPUT WORD
1$: ROL R5 ;;ROTATE MSB INTO 'C'
    BR 3$ ;;GO DO MSB
2$: ROL R5 ;;FORM THIS DIGIT
    ROL R5
    MOV R5,R3
3$: ROL R3 ;;GET LSB OF THIS DIGIT
    DECB $OMODE ;;TYPE THIS DIGIT?
    BPL 7$ ;;BR IF NO
    BIC #177770,R3 ;;GET RID OF JUNK
    BNE 4$ ;;TEST FOR 0
    TST R4 ;;SUPPRESS THIS 0?
    BEQ 5$ ;;BR IF YES
4$: INC R4 ;;DON'T SUPPRESS ANYMORE 0'S
5$:

```

061232	052703	000060		BIS	#'0,R3	::MAKE THIS DIGIT ASCII
061236	052703	000040	5\$:	BIS	#' ,R3	::MAKE ASCII IF NOT ALREADY
061242	110337	061306		MOVB	R3,8\$::SAVE FOR TYPING
061246	104401	061306		TYPE	.8\$::GO TYPE THIS DIGIT
061252	105337	061310	7\$:	DECB	\$OCNT	::COUNT BY 1
061256	003347			BGT	2\$::BR IF MORE TO DO
061260	002402			BLT	6\$::BR IF DONE
061262	005204			INC	R4	::INSURE LAST DIGIT ISN'T A BLANK
061264	000744			BR	2\$::GO DO THE LAST DIGIT
061266	012605		6\$:	MOV	(SP)+,R5	::RESTORE R5
061270	012604			MOV	(SP)+,R4	::RESTORE R4
061272	012603			MOV	(SP)+,R3	::RESTORE R3
061274	016666	000002 000004		MOV	2(SP),4(SP)	::SET THE STACK FOR RETURNING
061302	012616			MOV	(SP)+,(SP)	
061304	000002			RTI		::RETURN
061306	000		8\$:	.BYTE	0	::STORAGE FOR ASCII DIGIT
061307	000			.BYTE	0	::TERMINATOR FOR TYPE ROUTINE
061310	000		\$OCNT:	.BYTE	0	::OCTAL DIGIT COUNTER
061311	000		\$OFILL:	.BYTE	0	::ZERO FILL SWITCH
061312	000000		\$OMODE:	.WORD	0	::NUMBER OF DIGITS TO TYPE

.SBTTL TYPE ROUTINE

```

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

```

```

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

```

```

061314 105737 001173 $TYPE: TSTB $TPFLG      ;; IS THERE A TERMINAL?
061320 100002          BPL 1$          ;; BR IF YES
061322 000000          HALT          ;; HALT HERE IF NO TERMINAL
061324 000430          BR 3$          ;; LEAVE
061326 010046 1$: MOV RO,-(SP)      ;; SAVE RO
061330 017600 000002 MOV @2(SP),RO      ;; GET ADDRESS OF ASCIZ STRING
061334 122737 000001 001242 CMPB #APTENV,$ENV      ;; RUNNING IN APT MODE
061342 001011          BNE 62$       ;; NO, GO CHECK FOR APT CONSOLE
061344 132737 000100 001243 BITB #APTSPOOL,$ENVM      ;; SPOOL MESSAGE TO APT
061352 001405          BEQ 62$       ;; NO, GO CHECK FOR CONSOLE
061354 010037 061364 MOV RO,61$      ;; SETUP MESSAGE ADDRESS FOR APT
061360 004737 066066 JSR PC,$ATY3    ;; SPOOL MESSAGE TO APT
061364 000000          61$: .WORD 0      ;; MESSAGE ADDRESS
061366 132737 000040 001243 62$: BITB #APTCSUP,$ENVM      ;; APT CONSOLE SUPPRESSED
061374 001003          BNE 60$       ;; YES, SKIP TYPE OUT
061376 112046 2$: MOVB (RO)+,-(SP)    ;; PUSH CHARACTER TO BE TYPED ONTO STACK
061400 001005          BNE 4$         ;; BR IF IT ISN'T THE TERMINATOR
061402 005726          TST (SP)+      ;; IF TERMINATOR POP IT OFF THE STACK
061404 012600 60$: MOV (SP)+,RO      ;; RESTORE RO
061406 062716 000002 3$: ADD #2,(SP)   ;; ADJUST RETURN PC
061412 000002          RTI          ;; RETURN
061414 122716 000011 4$: CMPB #HT,(SP)   ;; BRANCH IF <HT>
061420 001430          BEQ 8$         ;;
061422 122716 000200 CMPB #CRLF,(SP)   ;; BRANCH IF NOT <CRLF>
061426 001006          BNE 5$         ;;
061430 005726          TST (SP)+      ;; POP <CR><LF> EQUIV
061432 104401          TYPE          ;; TYPE A CR AND LF
061434 001217          $CRLF
061436 105037 061644 CLRB $CHARCNT    ;; CLEAR CHARACTER COUNT
061442 000755          BR 2$         ;; GET NEXT CHARACTER
061444 004737 061526 5$: JSR PC,$TYPEC      ;; GO TYPE THIS CHARACTER
061450 123726 001172 6$: CMPB $FILLC,(SP)+    ;; IS IT TIME FOR FILLER CHARS.?
061454 001350          BNE 2$         ;; IF NO GO GET NEXT CHAR.
061456 013746 001170 MOV $NULL,-(SP)   ;; GET # OF FILLER CHARS. NEEDED
                                ;; AND THE NULL CHAR.
061462 105366 000001 7$: DECB 1(SP)      ;; DOES A NULL NEED TO BE TYPED?
061466 002770          BLT 6$         ;; BR IF NO--GO POP THE NULL OFF OF STACK
061470 004737 061526 JSR PC,$TYPEC      ;; GO TYPE A NULL
061474 105337 061644 DECB $CHARCNT     ;; DO NOT COUNT AS A COUNT
061500 000770          BR 7$         ;; LOOP

```


;HORIZONTAL TAB PROCESSOR

061502	112716	000040		8\$:	MOVB	#' (SP)	::REPLACE TAB WITH SPACE
061506	004737	061526		9\$:	JSR	PC,\$TYPEC	::TYPE A SPACE
061512	132737	000007	061644		BITB	#7,\$CHARCNT	::BRANCH IF NOT AT
061520	001372				BNE	9\$::TAB STOP
061522	005726				TST	(SP)+	::POP SPACE OFF STACK
061524	000724				BR	2\$::GET NEXT CHARACTER
061526				\$TYPEC:			
061526	105777	117426			TSTB	@\$TKS	::CHAR IN KYBD BUFFER?
061532	100022				BPL	10\$::BR IF NOT
061534	017746	117422			MOV	@\$TKB,-(SP)	::GET CHAR
061540	042716	177600			BIC	#177600,(SP)	::STRIP EXTRANEIOUS BITS
061544	122716	000023			CMPB	#\$XOFF,(SP)	::WAS CHAR XOFF
061550	001012				BNE	102\$::BR IF NOT
061552				101\$:			
061552	105777	117402			TSTB	@\$TKS	::WAIT FOR CHAR
061556	100375				BPL	101\$	
061560	117716	117376			MOVB	@\$TKB,(SP)	::GET CHAR
061564	042716	177600			BIC	#177600,(SP)	::STRIP IT
061570	122716	000021			CMPB	#\$XON,(SP)	::WAS IT XON?
061574	001366				BNE	101\$::BR IF NOT
061576				102\$:			
061576	005726				TST	(SP)+	::FIX STACK
061600				10\$:			
061600	105777	117360			TSTB	@\$TPS	::WAIT UNTIL PRINTER IS READY
061604	100375				BPL	10\$	
061606	116677	000002	117352		MOVB	2(SP),@\$TPB	::LOAD CHAR TO BE TYPED INTO DATA REG.
061614	122766	000015	000002		CMPB	#CR,2(SP)	::IS CHARACTER A CARRIAGE RETURN?
061622	001003				BNE	1\$::BRANCH IF NO
061624	105037	061644			CLRB	\$CHARCNT	::YES--CLEAR CHARACTER COUNT
061630	000406				BR	\$TYPEX	::EXIT
061632	122766	000012	000002	1\$:	CMPB	#LF,2(SP)	::IS CHARACTER A LINE FEED?
061640	001402				BEQ	\$TYPEX	::BRANCH IF YES
061642	105227				INCB	(PC)+	::COUNT THE CHARACTER
061644	000000			\$CHARCNT:	.WORD	0	::CHARACTER COUNT STORAGE
061646	000207			\$TYPEX:	RTS	PC	

.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
    
```

```

061650          $SCOPE:
061650 104410          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
061652 004737 062512  JSR          PC,STOP
061656 032777 040000 117270 1$:    BIT          #BIT14,@SWR          ;;LOOP ON PRESENT TEST?
061664 001402          BEQ          9$          ;;NO IF SW14=0
061666 000137 062316          JMP          $OVER          ;;JUMP OVER SCOPE ROUTINE
061672          9$:
:#####START OF CODE FOR THE XOR TESTER#####
061672 000416  $XTSTR: BR          6$          ;;IF RUNNING ON THE "XOR" TESTER CHANGE
:THIS INSTRUCTION TO A "NOP" (NOP=240)
061674 013746 000004          MOV          @#ERRVEC,-(SP)          ;;SAVE THE CONTENTS OF THE ERROR VECTOR
061700 012737 061720 000004          MOV          #5$,@#ERRVEC          ;;SET FOR TIMEOUT
061706 005737 177060          TST          @#177060          ;;TIME OUT ON XOR?
061712 012637 000004          MOV          (SP)+,@#ERRVEC          ;;RESTORE THE ERROR VECTOR
061716 000561          BR          $SVLAD          ;;GO TO THE NEXT TEST
061720 022626          5$:    CMP          (SP)+,(SP)+          ;;CLEAR THE STACK AFTER A TIME OUT
061722 012637 000004          MOV          (SP)+,@#ERRVEC          ;;RESTORE THE ERROR VECTOR
061726 000521          BR          7$          ;;LOOP ON THE PRESENT TEST
061730          6$:;#####END OF CODE FOR THE XOR TESTER#####
061730 032777 000400 117216          BIT          #BIT08,@SWR          ;;LOOP ON SPEC. TEST?
061736 001421          BEQ          2$          ;;BR IF NO
061740 005046          CLR          -(SP)          ;;CLEAR A TEMP. LOCATION
061742 117716 117206          MOVB         @SWR,(SP)          ;;PICKUP THE DESIRED TEST NUMBER
061746 001414          BEQ          8$          ;;BRANCH IF BAD TEST NUMBER IN SWR
061750 022716 000067          CMP          #67,(SP)          ;;CHECK THE NUMBER IN THE SWR
061754 002411          BLT          8$          ;;BRANCH IF TEST NUMBER IS OUT OF RANGE
061756 011637 001116          MOV          (SP),$TSTNM          ;;UPDATE THE TEST NUMBER
061762 005316          DEC          (SP)          ;;BACKUP BY ONE
061764 006316          ASL          (SP)          ;;SCALE THE TEST NUMBER AS AN INDEX
061766 062716 062334          ADD          #$$SW08TBL,(SP)          ;;FORM THE ADDRESS OF TEST POINTER
061772 013637 001122          MOV          @(SP)+,$LPADR          ;;SET LOOP ADDRESS TO DESIRED TEST
061776 000547          BR          $OVER          ;;GO LOOP ON THE TEST
062000 005726          8$:    TST          (SP)+          ;;CLEAN THE BAD TEST NUMBER OFF OF THE STACK
062002 105737 001117          2$:    TSTB         $ERFLG          ;;HAS AN ERROR OCCURRED?
062006 001502          BEQ          3$          ;;BR IF NO
062010 022737 177777 063130          CMP          #-1,CPSAVE          ;;SEE IF TIMEOUT WAS PREVIOUSLY RECORDED
062016 001455          BEQ          2003$          ;;KICK AROUND ROUTINE IF SO
062020 013746 000004          MOV          ERRVEC,-(SP)          ;;SAVE CONTENTS OF ERROR VECTOR
062024 012737 062042 000004          MOV          #2000$,ERRVEC          ;;SETUP 'TRAP' RETURN ADDRESS
062032 013737 177766 063130          MOV          177766,CPSAVE          ;;MOVE CPU ERROR REGISTER TO CPSAVE FOR TEST
062040 000406          BR          2001$
062042 012737 177777 063130 2000$: MOV          #-1,CPSAVE          ;;SET CPU ERROR REGISTER TIMEOUT INDICATOR
062050 012716 062056          MOV          #2001$,(SP)          ;;SETUP RETURN ADDRESS
    
```

```

SCOPE HANDLER ROUTINE

062054 000002
062056 012637 000004          2001$: RTI
                                MOV      (SP)+,ERRVEC      ;;RESTORE CONTENTS OF ERROR VECTOR

062062 022737 177777 063130 2002$: CMP      #-1,CPSAVE      ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
062070 001430                BEQ      2003$      ;;BRANCH IF SO
062072 032737 000001 063130 BIT      #BIT00,CPSAVE  ;;SEE IF THE POWER MONITOR BIT IS ON
062100 001424                BEQ      2003$      ;;BRANCH TO CONTINUE ROUTINE IF CLEAR
062102 042737 000001 177766 BIC      #BIT00,177766 ;;CLEAR THE BIT FOUND TO BE SET
062110 013746 001154        MOV      SWR,-(SP)      ;;SAVE SWR ADDRESS
062114 017646 000000        MOV      @ (SP),-(SP)   ;;SAVE SWR VALUE
062120 012737 000176 001154 MOV      #176,SWR      ;;GET SOFTWARE SWR ADDRESS
062126 011677 117022        MOV      (SP),@SWR     ;;GET CURRENT SWR VALUE
062132 042777 001000 117014 BIC      #BIT09,@SWR   ;;DON'T ALLOW LOOP ON ERROR ON THIS ERROR
062140 104177                EMT      177          ;;CALL SPECIAL POWER FAIL BIT ERROR CALL
062142 012676 000000        MOV      (SP)+,@(SP)   ;;RESTORE SWR TO ORIGINAL VALUE
062146 012637 001154        MOV      (SP)+,SWR     ;;RESTORE SWR ADDRESS
062152
062152 123737 001131 001117 2003$: CMPB     $ERMAX,$ERFLG  ;;MAX. ERRORS FOR THIS TEST OCCURRED?
062160 101015                BHI     3$          ;;BR IF NO
062162 032777 001000 116764 BIT      #BIT09,@SWR   ;;LOOP ON ERROR?
062170 001404                BEQ     4$          ;;BR IF NO
062172 013737 001124 001122 7$:  MOV     $LPERR,$LPADR  ;;SET LOOP ADDRESS TO LAST SCOPE
062200 000446                BR      $OVER
062202 105037 001117        4$:  CLRB    $ERFLG      ;;ZERO THE ERROR FLAG
062206 005037 001206        CLR     $TIMES      ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
062212 000415                BR      1$          ;;ESCAPE TO THE NEXT TEST
062214 032777 004000 116732 3$:  BIT     #BIT11,@SWR   ;;INHIBIT ITERATIONS?
062222 001011                BNE    1$          ;;BR IF YES
062224 005737 001230        TST     $PASS       ;;IF FIRST PASS OF PROGRAM
062230 001406                BEQ     1$          ;;INHIBIT ITERATIONS
062232 005237 001120        INC     $ICNT       ;;INCREMENT ITERATION COUNT
062236 023737 001206 001120 CMP     $TIMES,$ICNT  ;;CHECK THE NUMBER OF ITERATIONS MADE
062244 002024                BGE     $OVER       ;;BR IF MORE ITERATION REQUIRED
062246 012737 000001 001120 1$:  MOV     #1,$ICNT     ;;REINITIALIZE THE ITERATION COUNTER
062254 013737 062332 001206 MOV     $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
062262 105237 001116        $SVLAD: INCB   $TSTNM    ;;COUNT TEST NUMBERS
062266 113737 001116 001226 MOV     $TSTNM,$TESTN ;;SET TEST NUMBER IN APT MAILBOX
062274 011637 001122        MOV     (SP),$LPADR  ;;SAVE SCOPE LOOP ADDRESS
062300 011637 001124        MOV     (SP),$LPERR  ;;SAVE ERROR LOOP ADDRESS
062304 005037 001210        CLR     $ESCAPE     ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
062310 112737 000001 001131 MOV     #1,$ERMAX    ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
062316 013777 001116 116632 $OVER: MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
062324 013716 001122        MOV     $LPADR,(SP) ;;FUDGE RETURN ADDRESS
062330 000002                RTI
062332 000005        $MXCNT: 5.      ;;FIXES PS
062334                $SWOBTBL:  ;;MAX. NUMBER OF ITERATIONS
                                .REPT $TN-1
062334 010050                .WORD   TST1+2      ;;STARTING ADDRESS OF TEST 1
062336 010250                .WORD   TST2+2      ;;STARTING ADDRESS OF TEST 2
062340 011120                .WORD   TST3+2      ;;STARTING ADDRESS OF TEST 3
062342 011456                .WORD   TST4+2      ;;STARTING ADDRESS OF TEST 4
062344 011662                .WORD   TST5+2      ;;STARTING ADDRESS OF TEST 5
062346 0i2176                .WORD   TST6+2      ;;STARTING ADDRESS OF TEST 6
062350 012356                .WORD   TST7+2      ;;STARTING ADDRESS OF TEST 7
062352 013430                .WORD   TST10+2     ;;STARTING ADDRESS OF TEST 10
062354 013730                .WORD   TST11+2    ;;STARTING ADDRESS OF TEST 11
062356 014146                .WORD   TST12+2    ;;STARTING ADDRESS OF TEST 12

```

062360	014454	.WORD	TST13+2	:: STARTING ADDRESS OF TEST 13
062362	014764	.WORD	TST14+2	:: STARTING ADDRESS OF TEST 14
062364	015226	.WORD	TST15+2	:: STARTING ADDRESS OF TEST 15
062366	015504	.WORD	TST16+2	:: STARTING ADDRESS OF TEST 16
062370	015754	.WORD	TST17+2	:: STARTING ADDRESS OF TEST 17
062372	016204	.WORD	TST20+2	:: STARTING ADDRESS OF TEST 20
062374	016530	.WORD	TST21+2	:: STARTING ADDRESS OF TEST 21
062376	017150	.WORD	TST22+2	:: STARTING ADDRESS OF TEST 22
062400	017514	.WORD	TST23+2	:: STARTING ADDRESS OF TEST 23
062402	020106	.WORD	TST24+2	:: STARTING ADDRESS OF TEST 24
062404	020576	.WORD	TST25+2	:: STARTING ADDRESS OF TEST 25
062406	021144	.WORD	TST26+2	:: STARTING ADDRESS OF TEST 26
062410	021466	.WORD	TST27+2	:: STARTING ADDRESS OF TEST 27
062412	022050	.WORD	TST30+2	:: STARTING ADDRESS OF TEST 30
062414	022366	.WORD	TST31+2	:: STARTING ADDRESS OF TEST 31
062416	023152	.WORD	TST32+2	:: STARTING ADDRESS OF TEST 32
062420	023676	.WORD	TST33+2	:: STARTING ADDRESS OF TEST 33
062422	024466	.WORD	TST34+2	:: STARTING ADDRESS OF TEST 34
062424	025252	.WORD	TST35+2	:: STARTING ADDRESS OF TEST 35
062426	025604	.WORD	TST36+2	:: STARTING ADDRESS OF TEST 36
062430	026244	.WORD	TST37+2	:: STARTING ADDRESS OF TEST 37
062432	026464	.WORD	TST40+2	:: STARTING ADDRESS OF TEST 40
062434	027022	.WORD	TST41+2	:: STARTING ADDRESS OF TEST 41
062436	027310	.WORD	TST42+2	:: STARTING ADDRESS OF TEST 42
062440	027576	.WORD	TST43+2	:: STARTING ADDRESS OF TEST 43
062442	030110	.WORD	TST44+2	:: STARTING ADDRESS OF TEST 44
062444	030410	.WORD	TST45+2	:: STARTING ADDRESS OF TEST 45
062446	030670	.WORD	TST46+2	:: STARTING ADDRESS OF TEST 46
062450	031172	.WORD	TST47+2	:: STARTING ADDRESS OF TEST 47
062452	031476	.WORD	TST50+2	:: STARTING ADDRESS OF TEST 50
062454	032114	.WORD	TST51+2	:: STARTING ADDRESS OF TEST 51
062456	032542	.WORD	TST52+2	:: STARTING ADDRESS OF TEST 52
062460	033160	.WORD	TST53+2	:: STARTING ADDRESS OF TEST 53
062462	033520	.WORD	TST54+2	:: STARTING ADDRESS OF TEST 54
062464	034062	.WORD	TST55+2	:: STARTING ADDRESS OF TEST 55
062466	034406	.WORD	TST56+2	:: STARTING ADDRESS OF TEST 56
062470	035034	.WORD	TST57+2	:: STARTING ADDRESS OF TEST 57
062472	035512	.WORD	TST60+2	:: STARTING ADDRESS OF TEST 60
062474	036216	.WORD	TST61+2	:: STARTING ADDRESS OF TEST 61
062476	036616	.WORD	TST62+2	:: STARTING ADDRESS OF TEST 62
062500	037226	.WORD	TST63+2	:: STARTING ADDRESS OF TEST 63
062502	037626	.WORD	TST64+2	:: STARTING ADDRESS OF TEST 64
062504	040206	.WORD	TST65+2	:: STARTING ADDRESS OF TEST 65
062506	040552	.WORD	TST66+2	:: STARTING ADDRESS OF TEST 66
062510	041110	.WORD	TST67+2	:: STARTING ADDRESS OF TEST 67

:DROP PRIORITY TO ALLOW CONSOLE INTERRUPT

STOP:

MOV	#PR3,-(SP)	:: PUT NEW PS ON STACK
MOV	#64\$,-(SP)	:: PUT NEW PC ON STACK
RTI		:: POP NEW PC AND PS

64\$:

:RAISE PRIORITY TO INHIBIT CONSOLE INTERRUPT

MOV	#PR5,-(SP)	:: PUT NEW PS ON STACK
-----	------------	------------------------

2
3
4
5
6
7
8
9

062512		
062512	012746	000140
062516	012746	062524
062522	000002	
062524		
062524	012746	000240

062530 012746 062536
062534 000002
062536
10 062536 000207

MOV #658,-(SP) ;;PUT NEW PC ON STACK
RTI ;;POP NEW PC AND PS
658:
RTS PC .RETURN

.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 ERRYP 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

062540 105037 063132          $ERROR: CLRB      IBSAVE          ;;CLEAR THE ITEM BYTE SAVE LOCATION
062544 104410                CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
062546 105237 001117          7$:  INCB      $ERFLG          ;;SET THE ERROR FLAG
062552 001775                BEQ        7$          ;;DON'T LET THE FLAG GO TO ZERO
062554 013777 001116 116374    MOV        $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
062562 032777 002000 116364    BIT        #BIT10,@SWR      ;;BELL ON ERROR?
062570 001402                BEQ        1$          ;;NO - SKIP
062572 104401 001212                TYPE      , $BELL          ;;RING BELL
062576 005237 001126          1$:  INC        $ERTTL          ;;COUNT THE NUMBER OF ERRORS
062602 011637 001132                MOV        (SP), $ERRPC      ;;GET ADDRESS OF ERROR INSTRUCTION
062606 162737 000002 001132    SUB        #2, $ERRPC
062614 117737 116312 001130    MOVB      @ $ERRPC, $ITEMB   ;;STRIP AND SAVE THE ERROR ITEM CODE
062622 032777 001000 116324    BIT        #BIT09,@SWR      ;;SEE IF LOOP ON ERROR IS SET
062630 001060                BNE        1004$          ;;BRANCH AROUND ROUTINE IF SO
062632 122737 000177 001130    CMPB      #177, $ITEMB      ;;SEE IF THIS IS THE POWER FAIL CALL
062640 001454                BEQ        1004$          ;;BRANCH AROUND ROUTINE IF IT IS
062642 105737 063132                TSTB      IBSAVE          ;;SEE IF THIS IS THE 2ND ERROR CALL IN THIS ROUTINE
062646 001047                BNE        1003$          ;;BRANCH IF SO
062650 022737 177777 063130    CMP        #-1, CPSAVE      ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
062656 001445                BEQ        1004$          ;;BRANCH IF SO
062660 013746 000004                MOV        ERRVEC, -(SP)    ;;SAVE CONTENTS OF ERROR VECTOR
062664 012737 062702 000004    MOV        #1000$, ERRVEC   ;;SETUP 'TRAP' RETURN ADDRESS
062672 013737 177766 063130    MOV        177766, CPSAVE   ;;MOVE CPU ERROR REGISTER TO CPSAVE FOR TEST
062700 000406                BR        1001$
062702 012737 177777 063130 1000$: MOV        #-1, CPSAVE      ;;SET CPU ERROR REGISTER TIMEOUT INDICATOR
062710 012716 062716                MOV        #1001$, (SP)    ;;SETUP RETURN ADDRESS
062714 000002                RTI
062716 012637 000004          1001$: MOV        (SP)+, ERRVEC   ;;RESTORE CONTENTS OF ERROR VECTOR

062722 022737 177777 063130 1002$: CMP        #-1, CPSAVE      ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
062730 001420                BEQ        1004$          ;;BRANCH IF SO
062732 032737 000001 063130    BIT        #BIT00, CPSAVE   ;;SEE IF POWER MONITOR BIT IS SET IN CPU ERR REG
062740 001414                BEQ        1004$          ;;BRANCH IF OK
062742 042737 000001 177766    BIC        #BIT00, 177766   ;;CLEAR THE BIT FOUND SET
062750 113737 001130 063132    MOVB      $ITEMB, IBSAVE    ;;MAKE IBSAVE NON-ZERO FOR DUAL ERROR CALL
062756 112737 000177 001130    MOVB      #177, $ITEMB     ;;SET $ITEMB TO SPECIAL POWER FAIL POINTER
062764 000402                BR        1004$          ;;BRANCH OVER IBSAVE CLEARING

062766 105037 063132          1003$: CLRB      IBSAVE          ;;CLEAR IBSAVE SO 2ND TIME THROUGH EXITS
062772 032777 020000 116154 1004$: BIT        #BIT13,@SWR      ;;SKIP TYPEOUT IF SET
063000 001004                BNE        20$          ;;SKIP TYPEOUTS
063002 004737 063134          JSR        PC, ERRYP       ;;GO TO USER ERROR ROUTINE
```

```

063006 104401 001217          TYPE      , $CRLF
063012 122737 000001 001242 20$:      CMPB      #APTENV,$ENV      ;;RUNNING IN APT MODE
063020 001007          BNE        2$              ;;NO,SKIP APT ERROR REPORT
063022 113737 001130 063034  MOVB      $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
063030 004737 066076          JSR        PC,$ATY4      ;;REPORT FATAL ERROR TO APT
063034      000          .BYTE      0
063035      000          .BYTE      0
063036 000777          BR         22$              ;;APT ERROR LOOP
063040 105737 063132          TSTB      IBSAVE          ;;SEE IF IBSAVE IS LOADED
063044 001005          BNE        3$              ;;BRANCH IF NOT - NO HALT ON PWR MON BIT ERROR
063046 005777 116102          TST       @SWR           ;;HALT ON ERROR
063052 100002          BPL        3$              ;;SKIP IF CONTINUE
063054 000000          HALT
063056 104410          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
063060 032777 001000 116066 3$:      BIT       #BIT09,@SWR      ;;LOOP ON ERROR SWITCH SET?
063066 001402          BEQ        4$              ;;BR IF NO
063070 013716 001124          MOV       $LPERR,(SP)    ;;FUDGE RETURN FOR LOOPING
063074 005737 001210          TST      $ESCAPE        ;;CHECK FOR AN ESCAPE ADDRESS
063100 001402          BEQ        5$              ;;BR IF NONE
063102 013716 001210          MOV       $ESCAPE,(SP)  ;;FUDGE RETURN ADDRESS FOR ESCAPE
063106 022737 041630 000042 5$:      CMP       #SENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
063114 001001          BNE        6$              ;;BRANCH IF NO
063116 000000          HALT          ;;YES
063120 105737 063132          TSTB      IBSAVE          ;;SEE IF ITEM BYTE SAVE LOCATION HAS AN ERROR CALL
063124 001210          BNE        7$              ;;BRANCH BACK TO CALL ORIGINAL ERROR
063126 000002          RTI              ;;RETURN
063130 000000          CPSAVE: .WORD      0      ;;LOCATION TO SAVE CPU ERROR REG CONTENTS
063132 000000          IBSAVE: .WORD     0      ;;LOCATION TO SAVE ITEM BYTE

```

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

.SBTTL ERROR TYPEOUT ROUTINE

```

: *THE ERROR TYPEOUT ROUTINE ASSEMBLES AND PRINTS INFORMATION
: *REGARDING THE DETECTION OF AN ERROR AS FOLLOWS:
: *
: *   .UNIT NUMBER, DRIVE TYPE, TEST NUMBER, ERROR NUMBER AND
: *PROGRAM COUNTER ARE PRINTED ON THE FIRST LINE;
: *   .ERROR MESSAGE IS ASSEMBLED, FORMATTED AND PRINTED ON
: *ONE OR MORE SUCCEEDING LINES;
: *   .PAIRED LINES OF ERROR HEADERS AND ERROR DATA ARE PRINTED
: *AFTER THE ERROR MESSAGE.
  
```

```

ERRTYP: SAVREG
BIT #SW13,@SWR ;INHIBIT TYPEOUTS??
SEQ 1$ ;NO!!
JMP 27$ ;YES!!

:TYPE UNIT NUMBER, DRIVE TYPE, TEST NUMBER, ERROR NUMBER, AND
:PROGRAM COUNTER
1$: TYPE ,SCRLF ;TYPE 'DRV#'
TYPE ,ERTY00 ;:SAVE $UNIT FOR TYPEOUT
MOV $UNIT,-(SP) ;:TYPE DRIVE NUMBER
;:GO TYPE--OCTAL ASCII
TYPOS ;:TYPE 3 DIGIT(S)
.BYTE 3 ;:SUPPRESS LEADING ZEROS
.BYTE 0

:TYPE 'DRIVE TYPE' RM05, RM03 OR RM02 FOR UNIT UNDER TEST
MOV $BASE,R0 ;GET RM BASE ADDRESS
MOV RMDT(R0),R0 ;GET DRIVE TYPE REGISTER
BIC #177740,R0 ;SAVE DRIVE TYPE BITS AND
MOV #$RM03,3$ ;GET ASCII DRIVE TYPE
CMP #24,R0 ;IS DEVICE AN RM03 ?
BEQ 2$ ;YES !!

MOV #$RM02,3$ ;SAVE ASCII DRIVE TYPE
CMP #25,R0 ;IS DEVICE AN RM02 ?
BEQ 2$ ;YES !!

MOV #$RM05,3$ ;SAVE ASCII DRIVE TYPE
CMP #27,R0 ;IS DEVICE AN RM05 ?
BNE 4$ ;NO !!
2$: TYPE ,ERTY05 ;TYPE " - "
TYPE ;TYPE DRIVE TYPE
3$: .WORD 0 ;DRIVE TYPE MESSAGE IS STORED HERE

:TYPE TEST NUMBER, ERROR NUMBER AND PROGRAM COUNTER
4$: 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
MOV $ITEMB,ERRNMB ;TYPEOUT
  
```

```

13 063134 104414 .
14 063136 032777 020000 116010
15 063144 001402
16 063146 000137 063774
17
18
19
20 063152 104401 0C1217
21 063156 104401 064010
22 063162 013746 001234
    063166 104403
    063170 003
    063171 000
23
24
25 063172 013700 001276
26 063176 016000 000026
27 063202 042700 177740
28 063206 012737 066743 063260
29 063214 022700 000024
30 063220 001414
31
32 063222 012737 066736 063260
33 063230 022700 000025
34 063234 001406
35
36 063236 012737 066750 063260
37 063244 022700 000027
38 063250 001004
39 063252 104401 064045
40 063256 104401
41 063260 000000
42
43
44 063262 005037 064000
45 063266 013737 001226 064000
46 063274 104401 064015
47 063300 013746 064000
    063304 104403
    063306 003
    063307 000
48 063310 005037 064002
49 063314 113737 001130 064002
  
```



```

50 063322 001406          BEQ      5$          ;SKIP IF NO ERROR CALLED
51 063324 104401 064025   TYPE     ,ERTY02      ;TYPE 'ERR#'
52 063330 013746 064002   MOV      ERRNMB,-(SP) ;:SAVE ERRNMB FOR TYPEOUT
                                ;:TYPE ERROR NUMBER
                                ;:GO TYPE--OCTAL ASCII
                                ;:TYPE 3 DIGIT(S)
                                ;:SUPPRESS LEADING ZEROS
                                ;:TYPE 'PC='
                                ;:SAVE $ERRPC FOR TYPEOUT
                                ;:TYPE PROGRAM COUNTER
                                ;:GO TYPE--OCTAL ASCII
                                ;:TYPE 6 DIGIT(S)
                                ;:TYPE LEADING ZEROS
    063334 104403          TYPOS
    063336     003         .BYTE 3
    063337     000         .BYTE 0
53 063340 104401 064034   5$:    TYPE     ,ERTY03
54 063344 013746 001132   MOV      $ERRPC,-(SP)
                                ;:TYPE PROGRAM COUNTER
                                ;:GO TYPE--OCTAL ASCII
                                ;:TYPE 6 DIGIT(S)
                                ;:TYPE LEADING ZEROS
    063350 104403          TYPOS
    063352     006         .BYTE 6
    063353     001         .BYTE 1

55
56                                     ;GENERATE POINTER TO ERROR TABLE UNLESS ERROR NUMBER IS 0
57 063354 005737 064002   6$:    TST      ERRNMB  ;WAS AN ERROR CALLED?
58 063360 001002          BNE      7$          ;BR IF YES
59 063362 000137 063774   JMP      27$         ;NO--EXIT
60
61 063366 104401 001217   7$:    TYPE     , $CRLF ;YES-TYPE CRLF
62 063372 105037 064006   CLR     BOTFLG      ;CLEAR BOT FLAG
63 063376 105037 064007   CLR     CHRCNT      ;CLEAR CHARACTER COUNTER
64 063402 013700 064002   MOV     ERRNMB,R0   ;R0 POINTS TO FIRST OF
65 063406 122700 000177   CMP     #177,R0     ;SEE IF THIS ERROR CALL IS SPECIAL POWER FAIL CALL
66 063412 001003          BNE      8$          ;BRANCH IF NOT
67 063414 012700 064052   MOV     #PFECH,R0   ;MOVE POWER FAIL ERROR CALL TABLE TO R0
68 063420 000405          BR       9$
69 063422 006300          8$:    ASL     R0
70 063424 006300          ASL     R0
71 063426 006300          ASL     R0
72 063430 062700 001562   ADD     #$ERRTB-8.,R0
73 063434 011001          9$:    MOV     (R0),R1 ;R1 POINTS TO ERROR MESSAGE
74                                     ;TABLE
75 063436 001507          BEQ     19$         ;BRANCH IF NO ERROR MESSAGE
76
77                                     ;TYPE THE ERROR MESSAGE
78 063440 012102          10$:   MOV     (R1)+,R2   ;R2=ADDRESS OF MESSAGE STRING
79 063442 001505          BEQ     19$         ;BRANCH IF END OF MESSAGE
80 063444 010237 063612   MOV     R2,18$      ;LOAD ADDRESS OF STRING
81 063450 005037 064004   CLR     BOTADR      ;CLEAR BOT ADDRESS
82 063454 112203          11$:   MOV     (R2)+,R3   ;END OF STRING??
83 063456 001454          BEQ     17$         ;YES!!
84 063460 122703 000015   CMP     #CR,R3     ;CARRIAGE RETURN??
85 063464 001003          BNE     12$        ;NO!!
86 063466 105037 064007   CLR     CHRCNT      ;YES-CLEAR CHAR COUNT
87 063472 000770          BR      11$        ;GET NEXT CHARACTER
88 063474 122703 000012   12$:   CMP     #LF,R3   ;LINE FEED??
89 063500 001765          BEQ     11$        ;YES-GET NEXT CHARACTER
90 063502 122703 000011   CMP     #HT,R3     ;HORIZONTAL TAB??
91 063506 001007          BNE     14$        ;NO!!
92 063510 105237 064007   13$:   INCB   CHRCNT     ;ADJUST CHARACTER COUNT
93 063514 132737 000007 064007   BIT     #7,CHRCNT
94 063522 001372          BNE     13$
95 063524 000407          BR      15$
96 063526 105237 064007   14$:   INCB   CHRCNT     ;INCREMENT CHARACTER COUNT
97 063532 122703 000040   CMP     #' ,R3     ;SPACE??
98 063536 001002          BNE     15$        ;NO!!

```

```

99 063540 010237 064004      MOV      R2,BOTADR      ;SAVE ADDRESS OF SPACE
100 063544 122737 000100 064007 15$:  CMPB    #64.,CHRCNT    ;END OF LINE??
101 063552 103340          BHIS    11$            ;NO!!
102 063554 013704 064004      MOV      BOTADR,R4     ;GET ADDRESS OF LAST SPACE
103 063560 001007          BNE     16$            ;BRANCH IF SPACE DETECTED
104 063562 104401 001217      TYPE    ,SCLRF         ;TYPE CRLF
105 063566 105037 064007      CLRB    CHRCNT         ;CLEAR CHARACTER COUNT
106 063572 013702 063612      MOV      18$,R2        ;SET UP R2 FOR TESTING
107 063576 000726          BR      11$
108 063600 105044          CLRB    -(R4)          ;REPLACE SPACE
109 063602 112737 177777 064006 16$:  MOVB    #-1,BOTFLG     ;SET BOT FLAG
110 063610 104401          TYPE    ;TYPE ERROR MESSAGE STRING
111 063612 000000          .WORD   ;STRING ADDRESS GOES HERE
112 063614 105737 064006      TSTB    BOTFLG        ;WAS STRING TRUNCATED??
113 063620 001707          BEQ     10$            ;NO!!
114 063622 104401 001217      TYPE    ,SCLRF         ;YES-TYPE CRLF
115 063626 105037 064006      CLRB    BOTFLG        ;CLEAR BOT FLAG
116 063632 105037 064007      CLRB    CHRCNT         ;CLEAR CHARACTER COUNT
117 063636 013702 064004      MOV      BOTADR,R2     ;SETUP R2 FOR TESTING
118 063642 010237 063612      MOV      R2,18$        ;SETUP 18$ FOR TYPING
119 063646 112742 000040      MOVB    #' -(R2)       ;RESTORE SPACE
120 063652 105722          TSTB    (R2)+          ;RESTORE R2
121 063654 000677          BR      11$            ;TYPE REST OF STRING
122
123                                     ;TYPE ERROR HEADER AND ERROR DATA
124 063656 016001 000002      19$:  MOV      2(R0),R1      ;R1 POINTS TO ERROR HEADER TABLE
125 063662 001444          BEQ     27$            ;BRANCH IF NO HEADER
126 063664 104401 001217      TYPE    ,SCLRF         ;(ASSUME NO DATA)
127 063670 016002 000004      MOV      4(R0),R2      ;R2 POINTS TO DATA ADDRESS TABLE
128 063674 016003 000006      MOV      6(R0),R3      ;R3 POINTS TO FORMAT TABLE
129 063700 012137 063710      20$:  MOV      (R1)+,21$     ;PUT HEADER ADDRESS FOR TYPE
130 063704 001433          BEQ     27$            ;BRANCH IF END OF HEADERS
131                                     ;(ASSUME END OF DATA)
132 063706 104401          TYPE    ;HEADER ADDRESS GOES HERE
133 063710 000000          .WORD   0
134 063712 104401 001217      TYPE    ,SCLRF
135 063716 005702          TST     R2             ;DATA WITH HEADER??
136 063720 001767          BEQ     20$            ;NO!!
137 063722 012204          MOV     (R2)+,R4       ;R4 POINTS TO DATA ADDRESS
138 063724 012305          MOV     (R3)+,R5       ;R5 POINTS TO FORMAT
139 063726 105725          22$:  TSTB    (R5)+          ;WHAT KIND OF DATA??
140 063730 100407          BMI     24$            ;BINARY
141 063732 001403          BEQ     23$            ;OCTAL
142 063734 013446          MOV     @ (R4)+,-(SP)  ;DECIMAL
143 063736 104405          TYPDS
144 063740 000405          BR      25$
145 063742 013446          23$:  MOV     @ (R4)+,-(SP)
146 063744 104402          TYPOC
147 063746 000402          BR      25$
148 063750 013446          24$:  MOV     @ (R4)+,-(SP)
149 063752 104406          TYPBN
150 063754 005714          25$:  TST     (R4)           ;MORE DATA??
151 063756 001403          BEQ     26$            ;NO!!
152 063760 104401 064042      TYPE    ,ERTY04        ;YES-TYPE 2 SPACES
153 063764 000760          BR      22$            ;AND CONTINUE
154 063766 104401 001217      26$:  TYPE    ,SCLRF         ;TYPE ONE BLANK LINE
155 063772 000742          BR      20$            ;BEFORE NEXT HEADER

```

```

156 063774 104415
157 063776 000207
158
159 064000 000000
160 064002 000000
161 064004 000000
162 064006 000
163 064007 000
164
165 064010 104 122 126 ERTY00: .ASCIZ @DRV#@
166 064015 054 040 124 ERTY01: .ASCIZ @, TEST#@
167 064025 054 040 105 ERTY02: .ASCIZ @, ERR#@
168 064034 054 040 120 ERTY03: .ASCIZ @, PC=@
169 064042 040 040 000 ERTY04: .ASCIZ @ @
170 064045 040 055 040 ERTY05: .ASCIZ @ - @
171
172 064052 064062 064150 064166 PFECH: PFECH1,PFECH2,PFECH3,PFECH4 ;WORDS DEFINING TABLES BELOW
173 064062 064066 000000
174 064066 120 117 127 PFECH1: .+4,0
175 .ASCIZ ?POWER MONITOR BIT IN CPU ERROR REGISTER FOUND SET?
176 064150 064154 000000 PFECH2: .EVEN
177 064154 103 120 125 .+4,0
178 .ASCIZ ?CPUERREG?
179 064166 064170 PFECH3: .EVEN
180 064170 063130 000000 .+2
181 064174 064176 PFECH4: .WORD CPSAVE,0
182 064176 000 000 .+2
.BYTE 0,0

```

```

27$: RESREG
RTS PC
TSTNMB: .WORD 0 ;TEST NUMBER
ERRNMB: .WORD 0 ;ERROR NUMBER
BOTADR: .WORD 0 ;BEGINNING OF TEXT ADDRESS
BOTFLG: .BYTE 0 ;BOT FLAG
CHRCNT: .BYTE 0 ;CHARACTER COUNT

```

.SBTTL TTY INPUT ROUTINE

064200 000000
064202 000000
064204 000000
064206 064207

```

*****
ENABL  LSB
$TKCNT: .WORD 0      ;;NUMBER OF ITEMS IN QUEUE
$TKQIN: .WORD 0      ;;INPUT POINTER
$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
    
```

064210 005037 064200
064214 012737 064206 064202
064222 013737 064202 064204
064230 012737 064260 000060
064236 012737 000200 000062
064244 005777 114712
064250 012777 000100 114702
064256 000207

```

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

```

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

064260 117746 114676
064264 042716 177600
064270 021627 000021
064274 001002
064276 005726
064300 000002
064302
064302 021627 000003
064306 001007
064310 104401 065406
064314 004737 064210
064320 005726
064322 000137 065450
064326 021627 000007
064332 001004
064334 022737 000176 001154
064342 001500

064344
064344 022737 000001 064200
064352 001004
064354 104401 001212

```

$TKSRV: MOV      @#TKB,-(SP) ;;PICKUP THE CHARACTER
        BIC      #^C177,(SP) ;;STRIP THE JUNK
        CMP      (SP),#$XON  ;;IS IT A RANDOM XON?
        BNE      30$         ;;BRANCH IF NO
        TST      (SP)+       ;;CLEAN RANDOM XON OFF STACK
        RTI                    ;;RETURN
30$:
        CMP      (SP),#3     ;;IS IT A CONTROL C?
        BNE      1$         ;;BRANCH IF NO
        TYPE     ,%CNTLC     ;;TYPE A CONTROL-C (^C)
        JSR      PC,$TKINT  ;;INIT THE KEYBOARD
        TST      (SP)+       ;;CLEAN UP STACK
        JMP      SHUT       ;;CONTROL C RESTART
1$:
        CMP      (SP),#7     ;;IS IT A CONTROL G?
        BNE      2$         ;;BRANCH IF NO
        CMP      #SWREG,SWR  ;;IS SOFT-SWR SELECTED?
        BEQ      6$         ;;GO TO SWR CHANGE
2$:
        CMP      #1,$TKCNT  ;;IS THE QUEUE FULL?
        BNE      3$         ;;BRANCH IF NO
        TYPE     ,%BELL     ;;RING THE TTY BELL
    
```

```

064360 005726          TST      (SP)+      ;;CLEAN CHARACTER OFF OF STACK
064362 000451          BR        5$          ;;EXIT
064364 021627 000023 3$:      CMP      (SP),#23    ;;IS IT A CONTROL-S?
064370 001021          BNE      32$         ;;BRANCH IF NO
064372 005077 114562    CLR      @STKS       ;;DISABLE TTY KEYBOARD INTERRUPTS
064376 005726          TST      (SP)+      ;;CLEAN CHAR OFF STACK
064400 105777 114554 31$:    TSTB     @STKS       ;;WAIT FOR A CHAR
064404 100375          BPL      31$         ;;LOOP UNTIL ITS THERE
064406 117746 114550    MOVB    @STKB,-(SP)  ;;GET THE CHARACTER
064412 042716 177600    BIC     #'C177,(SP) ;;MAKE IT 7-BIT ASCII
064416 022627 000021    CMP     (SP)+,#21   ;;IS IT A CONTROL-Q?
064422 001366          BNE      31$         ;;BRANCH IF NO
064424 012777 000100 114526    MOV     #100,@STKS  ;;REENABLE TTY KEYBOARD INTERRUPTS
064432 000002          RTI             ;;RETURN
064434 005237 064200 32$:    INC     $TKCNT      ;;COUNT THIS CHARACTER
064440 021627 000140    CMP     (SP),#140   ;;IS IT UPPER CASE?
064444 002405          BLT     4$          ;;BRANCH IF YES
064446 021627 000175    CMP     (SP),#175   ;;IS IT A SPECIAL CHAR?
064452 003002          BGT     4$          ;;BRANCH IF YES
064454 042716 000040    BIC     #40,(SP)    ;;MAKE IT UPPER CASE
064460 112677 177516 4$:      MOVB    (SP)+,@STKQIN ;;AND PUT IT IN QUEUE
064464 005237 064202    INC     $TKQIN      ;;UPDATE THE POINTER
064470 023727 064202 064207    CMP     $TKQIN,#$TKQEND ;;GO OFF THE END?
064476 001003          BNE     5$          ;;BRANCH IF NO
064500 012737 064206 064202    MOV     #$TKQSRST,$TKQIN ;;RESET THE POINTER
064506 000002          RTI             ;;RETURN
    
```

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

```

064510 022737 000176 001154 $CKSWR: CMP     #SWREG,SWR    ;;IS THE SOFT-SWR SELECTED
064516 001124          BNE     15$         ;;EXIT IF NOT
064520 105777 114434    TSTB    @STKS       ;;IS A CHAR WAITING?
064524 100121          BPL     15$         ;;IF NOT, EXIT
064526 117746 114430    MOVB    @STKB,-(SP) ;;YES
064532 042716 177600    BIC     #'C177,(SP) ;;MAKE IT 7-BIT ASCII
064536 021627 000007    CMP     (SP),#7     ;;IS IT A CONTROL-G?
064542 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.

```

064544 123727 001150 000001 6$:      CMPB    $AUTOB,#1    ;;ARE WE RUNNING IN AUTO-MODE?
064552 001674          BEQ     2$          ;;BRANCH IF YES
064554 005726          TST     (SP)+      ;;CLEAR CONTROL-G OFF STACK
064556 004737 064210    JSR     PC,$TKINT   ;;FLUSH THE TTY INPUT QUEUE
064562 005077 114372    CLR     @STKS       ;;DISABLE TTY KEYBOARD INTERRUPTS
064566 112737 000001 001151    MOVB    #1,$INTAG   ;;SET INTERRUPT MODE INDICATOR

064574 104401 065420    TYPE    ,$CNTLG     ;;ECHO THE CONTROL-G (^G)
064600 104401 065425    $GTSWR: TYPE    ,$MSWR ;;TYPE CURRENT CONTENTS
064604 013746 000176    MOV     SWREG,-(SP) ;;SAVE SWREG FOR TYPEOUT
064610 104402    TYPOC    ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
    
```

064612	104401	065436			TYPE	,\$MNEW	::PROMPT FOR NEW SWR
064616	005046		19\$:		CLR	-(SP)	::CLEAR COUNTER
064620	005046				CLR	-(SP)	::THE NEW SWR
064622	105777	114332	7\$:		TSTB	@\$TKS	::CHAR THERE?
064626	100375				BPL	7\$::IF NOT TRY AGAIN
064630	117746	114326			MOVB	@\$TKB,-(SP)	::PICK UP CHAR
064634	042716	177600			BIC	#^C177,(SP)	::MAKE IT 7-BIT ASCII
064640	021627	000003			CMP	(SP),#3	::IS IT A CONTROL-C?
064644	001015				BNE	9\$::BRANCH IF NOT
064646	104401	065406			TYPE	,\$CNTLC	::YES, ECHO CONTROL-C (^C)
064652	062706	000006			ADD	#6,SP	::CLEAN UP STACK
064656	123727	001151	000001		CMPB	\$INTAG,#1	::REENABLE TTY KEYBOARD INTERRUPTS?
064664	001003				BNE	8\$::BRANCH IF NO
064666	012777	000100	114264		MOV	#100,@\$1KS	::ALLOW TTY KEYBOARD INTERRUPTS
064674	000137	065450	8\$:		JMP	SHUT	::CONTROL-C RESTART
064700	021627	000025	9\$:		CMP	(SP),#25	::IS IT A CONTROL-U?
064704	001005				BNE	10\$::BRANCH IF NOT
064706	104401	065413			TYPE	,\$CNTLU	::YES, ECHO CONTROL-U (^U)
064712	062706	000006	20\$:		ADD	#6,SP	::IGNORE PREVIOUS INPUT
064716	000737				BR	19\$::LET'S TRY IT AGAIN
064720	021627	000015	10\$:		CMP	(SP),#15	::IS IT A <CR>?
064724	001022				BNE	16\$::BRANCH IF NO
064726	005766	000004			TST	4(SP)	::YES, IS IT THE FIRST CHAR?
064732	001403				BEQ	11\$::BRANCH IF YES
064734	016677	000002	114212		MOV	2(SP),@SWR	::SAVE NEW SWR
064742	062706	000006	11\$:		ADD	#6,SP	::CLEAN UP STACK
064746	104401	001217	14\$:		TYPE	,\$CRLF	::ECHO <CR> AND <LF>
064752	123727	001151	000001		CMPB	\$INTAG,#1	::RE-ENABLE TTY KBD INTERRUPTS?
064760	001003				BNE	15\$::BRANCH IF NOT
064762	012777	000100	114170		MOV	#100,@\$TKS	::RE-ENABLE TTY KBD INTERRUPTS
064770	000002		15\$:		RTI		::RETURN
064772	004737	061526	16\$:		JSR	PC,\$TYPEC	::ECHO CHAR
064776	021627	000060			CMP	(SP),#60	::CHAR < 0?
065002	002420				BLT	18\$::BRANCH IF YES
065004	021627	000067			CMP	(SP),#67	::CHAR > 7?
065010	003015				BGT	18\$::BRANCH IF YES
065012	042726	000060			BIC	#60,(SP)+	::STRIP-OFF ASCII
065016	005766	000002			TST	2(SP)	::IS THIS THE FIRST CHAR
065022	001403				BEQ	17\$::BRANCH IF YES
065024	006316				ASL	(SP)	::NO, SHIFT PRESENT
065026	006316				ASL	(SP)	::CHAR OVER TO MAKE
065030	006316				ASL	(SP)	::ROOM FOR NEW ONE.
065032	005266	000002	17\$:		INC	2(SP)	::KEEP COUNT OF CHAR
065036	056616	177776			BIS	-2(SP),(SP)	::SET IN NEW CHAR
065042	000667				BR	7\$::GET THE NEXT ONE
065044	104401	001216	18\$:		TYPE	,\$QUES	::TYPE ?<CR><LF>
065050	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
;;*

065052	011646			\$RDCHR:	MOV	(SP),-(SP)	;;PUSH DOWN THE PC AND
065054	016666	000004	000002		MOV	4(SP),2(SP)	;;THE PS
065062	005066	000004			CLR	4(SP)	;;GET READY FOR A CHARACTER
065066	005046				CLR	-(SP)	;;PUT NEW PS ON STACK
065070	012746	065076			MOV	#64\$,-(SP)	;;PUT NEW PC ON STACK
065074	000002				RTI		;;POP NEW PC AND PS
065076				64\$:			
065076	005737	064200		1\$:	TST	\$TKCNT	;;WAIT ON A CHARACTER
065102	001775				BEQ	1\$	
065104	005337	064200			DEC	\$TKCNT	;;DECREMENT THE COUNTER
065110	117766	177070	000004		MOVB	@\$TKQOUT,4(SP)	;;GET ONE CHARACTER
065116	005237	064204			INC	\$TKQOUT	;;UPDATE THE POINTER
065122	023727	064204	064207		CMP	\$TKQOUT,#\$TKQEND	;;DID IT GO OFF OF THE END?
065130	001003				BNE	2\$;;BRANCH IF NO
065132	012737	064206	064204		MOV	#\$TKQSRRT,\$TKQOUT	;;RESET THE POINTER
065140	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
;;*

065142	010346			\$RDLIN:	MOV	R3,-(SP)	;;SAVE R3
065144	005046				CLR	-(SP)	;;CLEAR THE RUBOUT KEY
065146	012703	065376		1\$:	MOV	#\$TTYIN,R3	;;GET ADDRESS
065152	022703	065406		2\$:	CMP	#\$TTYIN+8.,R3	;;BUFFER FULL?
065156	101456				BLOS	4\$;;BR IF YES
065160	104411				RDCHR		;;GO READ ONE CHARACTER FROM THE TTY
065162	112613				MOVB	(SP)+,(R3)	;;GET CHARACTER
065164	122713	000177		10\$:	CMPB	#177,(R3)	;;IS IT A RUBOUT
065170	001022				BNE	5\$;;BR IF NO
065172	005716				TST	(SP)	;;IS THIS THE FIRST RUBOUT?
065174	001007				BNE	6\$;;BR IF NO
065176	112737	000134	065374		MOVB	#'\,9\$;;TYPE A BACK SLASH
065204	104401	065374			TYPE	,9\$	
065210	012716	177777			MOV	#-1,(SP)	;;SET THE RUBOUT KEY
065214	005303			6\$:	DEC	R3	;;BACKUP BY ONE
065216	020327	065376			CMP	R3,\$\$TTYIN	;;STACK EMPTY?
065222	103434				BLO	4\$;;BR IF YES
065224	111337	065374			MOVB	(R3),9\$;;SETUP TO TYPEOUT THE DELETED CHAR.
065230	104401	065374			TYPE	,9\$;;GO TYPE
065234	000746				BR	2\$;;GO READ ANOTHER CHAR.
065236	005716			5\$:	TST	(SP)	;;RUBOUT KEY SET?
065240	001406				BEQ	7\$;;BR IF NO
065242	112737	000134	065374		MOVB	#'\,9\$;;TYPE A BACK SLASH
065250	104401	065374			TYPE	,9\$	
065254	005016				CLR	(SP)	;;CLEAR THE RUBOUT KEY
065256	122713	000025		7\$:	CMPB	#25,(R3)	;;IS CHARACTER A CTRL U?
065262	001003				BNE	8\$;;BR IF NO

```

065264 104401 065413          TYPE      ,SCNTLU          ;;TYPE A CONTROL 'U'
065270 000726          BR          1$          ;;GO START OVER
065272 122713 000022      8$:      CMPB      #22,(R3)      ;;IS CHARACTER A '^R'?
065276 001011          BNE          3$          ;;BRANCH IF NO
065300 105013          CLR      (R3)          ;;CLEAR THE CHARACTER
065302 104401 001217      TYPE      ,SCRLF          ;;TYPE A 'CR' & 'LF'
065306 104401 065376      TYPE      ,STYIN          ;;TYPE THE INPUT STRING
065312 000717          BR          2$          ;;GO PICKUP ANOTHER CHACTER
065314 104401 001216      4$:      TYPE      ,SQUES          ;;TYPE A '?'
065320 000712          BR          1$          ;;CLEAR THE BUFFER AND LOOP
065322 111337 065374      3$:      MOV      (R3),9$          ;;ECHO THE CHARACTER
065326 104401 065374      TYPE      ,9$
065332 122723 000015      CMPB      #15,(R3)+      ;;CHECK FOR RETURN
065336 001305          BNE          2$          ;;LOOP IF NOT RETURN
065340 105063 177777      CLR      -1(R3)          ;;CLEAR RETURN (THE 15)
065344 104401 001220      TYPE      ,SLF          ;;TYPE A LINE FEED
065350 005726          TST      (SP)+          ;;CLEAN RUBOUT KEY FROM THE STACK
065352 012603          MOV      (SP)+,R3          ;;RESTORE R3
065354 011646          MOV      (SP),-(SP)          ;;ADJUST THE STACK AND PUT ADDRESS OF THE
065356 016666 000004 000002      MOV      4(SP),2(SP)          ;; FIRST ASCII CHARACTER ON IT
065364 012766 065376 000004      MOV      #STYIN,4(SP)
065372 000002          RTI          ;;RETURN
065374 000          9$:      .BYTE      0          ;;STORAGE FOR ASCII CHAR. TO TYPE
065375 000          .BYTE      0          ;;TERMINATOR
065376          $TTYIN: .BLKB      8          ;;RESERVE 8 BYTES FOR TTY INPUT
065406 136 103 015 $CNTLC: .ASCIZ /^C/<15><12> ;;CONTROL 'C'
065413 136 125 015 $CNTLU: .ASCIZ /^U/<15><12> ;;CONTROL 'U'
065420 136 107 015 $CNTLG: .ASCIZ /^G/<15><12> ;;CONTROL 'G'
065425 015 012 123 $MSWR: .ASCIZ <15><12>/SWR = /
065436 040 040 116 $MNEW: .ASCIZ / NEW = /
          .EVEN

2
3 065450 005737 000042      SHUT:   TST      @#42          ;;ANY MONITOR PRESENT ?
4 065454 001002          BNE          1$          ;;BR IF YES
5 065456 000137 005422      JMP      START          ;;GO TO START
6 065462 005037 001300      1$:     CLR      $DEVN          ;;FUDGE NO DRIVES IN MAP
7 065466 000137 041432      JMP      $EOP          ;;RETURN TO $EOP
    
```


.SBTTL READ AN OCTAL NUMBER FROM THE TTY

.....
 *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

065472	011646		\$RDOCT: MOV	(SP),-(SP)	::PROVIDE SPACE FOR THE
065474	016666	000004	MOV	4(SP),2(SP)	::INPUT NUMBER
065502	010046		MOV	R0,-(SP)	::PUSH R0 ON STACK
065504	010146		MOV	R1,-(SP)	::PUSH R1 ON STACK
065506	010246		MOV	R2,-(SP)	::PUSH R2 ON STACK
065510	104412		1\$: RDLIN		::READ AN ASCII LINE
065512	012600		MOV	(SP)+,R0	::GET ADDRESS OF 1ST CHARACTER
065514	005001		CLR	R1	::CLEAR DATA WORD
065516	005002		CLR	R2	
065520	112046		2\$: MOVB	(R0)+,-(SP)	::PICKUP THIS CHARACTER
065522	001412		BEQ	3\$::IF ZERO GET OUT
065524	006301		ASL	R1	::*2
065526	006102		ROL	R2	
065530	006301		ASL	R1	::*4
065532	006102		ROL	R2	
065534	006301		ASL	R1	::*8
065536	006102		ROL	R2	
065540	042716	177770	BIC	#*C7,(SP)	::STRIP THE ASCII JUNK
065544	062601		ADD	(SP)+,R1	::ADD IN THIS DIGIT
065546	000764		BR	2\$::LOOP
065550	005726		3\$: TST	(SP)+	::CLEAN TERMINATOR FROM STACK
065552	010166	000012	MOV	R1,12(SP)	::SAVE THE RESULT
065556	010237	065572	MOV	R2,\$HIOCT	
065562	012602		MOV	(SP)+,R2	::POP STACK INTO R2
065564	012601		MOV	(SP)+,R1	::POP STACK INTO R1
065566	012600		MOV	(SP)+,R0	::POP STACK INTO R0
065570	000002		RTI		::RETURN
065572	000000		\$HIOCT: .WORD	0	::HIGH ORDER BITS GO HERE

.SBTTL TRAP DECODER

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

065574	016646	000002	\$TRAP:	MOV	2(SP),-(SP)	::ASSUME THE STATUS OF
065600	042716	000020		BIC	#20,(SP)	:: THE CALLER--DO NOT ALLOW
065604	012746	065612		MOV	#1\$,-(SP)	:: T-BIT TRAPS
065610	000002			RTI		::SET THE NEW STATUS
065612	010046		1\$:	MOV	R0,-(SP)	::SAVE R0
065614	016600	000002		MOV	2(SP),R0	::GET TRAP ADDRESS
065620	005740			TST	-(R0)	::BACKUP BY 2
065622	111000			MOVB	(R0),R0	::GET RIGHT BYTE OF TRAP
065624	006300			ASL	R0	::POSITION FOR INDEXING
065626	016000	065646		MOV	\$TRPAD(R0),R0	::INDEX TO TABLE
065632	000200			RTS	R0	::GO TO ROUTINE

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

065634	011646		\$TRAP2:	MOV	(SP),-(SP)	::MOVE THE PC DOWN
065636	016666	000004		MOV	4(SP),2(SP)	::MOVE THE PSW DOWN
065644	000002	000002		RTI		::RESTORE THE PSW

.SBTTL TRAP TABLE

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

			:	ROUTINE	
			:	-----	
065646	065634		\$TRPAD:	.WORD	\$TRAP2
065650	061314			\$TYPE	::CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
065652	061112			\$TYPOC	::CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
065654	061066			\$TYPOS	::CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
065656	061126			\$TYPON	::CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
065660	060642			\$TYPDS	::CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
065662	060566			\$TYPBN	::CALL=TYPBN TRAP+6(104406) TYPE BINARY (ASCII) NUMBER
065664	064600			\$GTSWR	::CALL=GTSWR TRAP+7(104407) GET SOFT-SWR SETTING
065666	064510			\$CKSWR	::CALL=CKSWR TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
065670	065052			\$RDCHR	::CALL=RDCHR TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
065672	065142			\$RDLIN	::CALL=RDLIN TRAP+12(104412) TTY TYPEIN STRING ROUTINE
065674	065472			\$RDOCT	::CALL=RDOCT TRAP+13(104413) READ AN OCTAL NUMBER FROM TTY
065676	060472			\$SAVREG	::CALL=SAVREG TRAP+14(104414) SAVE R0-R5 ROUTINE
065700	060530			\$RESREG	::CALL=RESREG TRAP+15(104415) RESTORE R0-R5 ROUTINE

.SBTTL POWER DOWN AND UP ROUTINES

```

*****
:POWER DOWN ROUTINE
065702 012737 066042 000024 $PWRDN: MOV    #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
065710 012737 000340 000026      MOV    #340,@#PWRVEC+2 ;;PRIO:7
065716 010046      MOV    R0,-(SP)      ;;PUSH R0 ON STACK
065720 010146      MOV    R1,-(SP)      ;;PUSH R1 ON STACK
065722 010246      MOV    R2,-(SP)      ;;PUSH R2 ON STACK
065724 010346      MOV    R3,-(SP)      ;;PUSH R3 ON STACK
065726 010446      MOV    R4,-(SP)      ;;PUSH R4 ON STACK
065730 010546      MOV    R5,-(SP)      ;;PUSH R5 ON STACK
065732 017746 113216      MOV    @SWR,-(SP)    ;;PUSH @SWR ON STACK
065736 010637 066046      MOV    SP,$SAVR6    ;;SAVE SP
065742 012737 065754 000024      MOV    #$PWRUP,@#PWRVEC ;;SET UP VECTOR
065750 000000      HALT
065752 000776      BR     .-2          ;;HANG UP

*****
:POWER UP ROUTINE
065754 012737 066042 000024 $PWRUP: MOV    #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
065762 013706 066046      MOV    $SAVR6,SP    ;;GET SP
065766 005037 066046      CLR    $SAVR6      ;;WAIT LOOP FOR THE TTY
065772 005237 066046 1$: INC    $SAVR6      ;;WAIT FOR THE INC
065776 001375      BNE    1$          ;;OF WORD
066000 012677 113150      MOV    (SP)+,@SWR   ;;POP STACK INTO @SWR
066004 012605      MOV    (SP)+,R5    ;;POP STACK INTO R5
066006 012604      MOV    (SP)+,R4    ;;POP STACK INTO R4
066010 012603      MOV    (SP)+,3     ;;POP STACK INTO R3
066012 012602      MOV    (SP)+,R2    ;;POP STACK INTO R2
066014 012601      MOV    (SP)+,R1    ;;POP STACK INTO R1
066016 012600      MOV    (SP)+,R0    ;;POP STACK INTO R0
066020 012737 065702 000024      MOV    #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
066026 012737 000340 000026      MOV    #340,@#PWRVEC+2 ;;PRIO:7
066034 104401      TYPE                                ;;REPORT THE POWER FAILURE
066036 066050      $PWRMG: .WORD    $POWER              ;;POWER FAIL MESSAGE POINTER
066040 000002      RTI
066042 000000      $ILLUP: HALT                                ;;THE POWER UP SEQUENCE WAS STARTED
066044 000776      BR     .-2          ;; BEFORE THE POWER DOWN WAS COMPLETE
066046 000000      $SAVR6: 0                                  ;;PUT THE SP HERE
066050 015 012 120 $POWER: .ASCIIZ <15><12>'POWER'
.EVEN

```

.SBTTL APT COMMUNICATIONS ROUTINE

```

*****
066060 112737 000001 066324 $ATY1:  MOV  #1,$FFLG      ;;TO REPORT FATAL ERROR
066066 112737 000001 066322 $ATY3:  MOV  #1,$MFLG     ;;TO TYPE A MESSAGE
066074 000403
066076 112737 000001 066324 $ATY4:  MOV  #1,$FFLG     ;;TO ONLY REPORT FATAL ERROR
066104 $ATYC:
066104 010046      MOV  R0,-(SP)      ;;PUSH R0 ON STACK
066106 010146      MOV  R1,-(SP)      ;;PUSH R1 ON STACK
066110 105737 066322      TSTB $MFLG        ;;SHOULD TYPE A MESSAGE?
066114 001450      BEQ  5$           ;;IF NOT: BR
066116 122737 000001 001242      CMPB #APTENV,$ENV  ;;OPERATING UNDER APT?
066124 001031      BNE  3$           ;;IF NOT: BR
066126 132737 000100 001243      BITB #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
066134 001425      BEQ  3$           ;;IF NOT: BR
066136 017600 000004      MOV  @4(SP),R0     ;;GET MESSAGE ADDR.
066142 062766 000002 000004      ADD  #2,4(SP)     ;;BUMP RETURN ADDR.
066150 005737 001222      1$:  TST  $MSGTYPE    ;;SEE IF DONE W/ LAST XMISSION?
066154 001375      BNE  1$           ;;IF NOT: WAIT
066156 010037 001236      MOV  R0,$MSGAD    ;;PUT ADDR IN MAILBOX
066162 105720      2$:  TSTB (R0)+      ;;FIND END OF MESSAGE
066164 001376      BNE  2$
066166 163700 001236      SUB  $MSGAD,R0    ;;SUB START OF MESSAGE
066172 006200      ASR  R0           ;;GET MESSAGE LNGTH IN WORDS
066174 010037 001240      MOV  R0,$MSGGLT  ;;PUT LENGTH IN MAILBOX
066200 012737 000004 001222      MOV  #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
066206 000413      BR   5$
066210 017637 000004 066234 3$:  MOV  @4(SP),4$    ;;PUT MSG ADDR IN JSR LINKAGE
066216 062766 000002 000004      ADD  #2,4(SP)    ;;BUMP RETURN ADDRESS
066224 013746 177776      MOV  177776,-(SP) ;;PUSH 177776 ON STACK
066230 004737 061314      JSR  PC,$TYPE    ;;CALL TYPE MACRO
066234 000000      4$:  .WORD 0
066236      5$:
066236 105737 066324      10$: TSTB $FFLG        ;;SHOULD REPORT FATAL ERROR?
066242 001416      BEQ  12$         ;;IF NOT: BR
066244 005737 001242      TST  $ENV        ;;RUNNING UNDER APT?
066250 001413      BEQ  12$         ;;IF NOT: BR
066252 005737 001222      11$: TST  $MSGTYPE    ;;FINISHED LAST MESSAGE?
066256 001375      BNE  11$        ;;IF NOT: WAIT
066260 017637 000004 001224      MOV  @4(SP),$FATAL ;;GET ERROR #
066266 062766 000002 000004      ADD  #2,4(SP)    ;;BUMP RETURN ADDR.
066274 005237 001222      INC  $MSGTYPE    ;;TELL APT TO TAKE ERROR
066300 105037 066324      12$: CLRB $FFLG      ;;CLEAR FATAL FLAG
066304 105037 066323      CLRB $LFLG      ;;CLEAR LOG FLAG
066310 105037 066322      CLRB $MFLG      ;;CLEAR MESSAGE FLAG
066314 012601      MOV  (SP)+,R1    ;;POP STACK INTO R1
066316 012600      MOV  (SP)+,R0    ;;POP STACK INTO R0
066320 000207      RTS  PC         ;;RETURN
066322 000      $MFLG: .BYTE 0  ;;MESSG. FLAG
066323 000      $LFLG: .BYTE 0  ;;LOG FLAG
066324 000      $FFLG: .BYTE 0  ;;FATAL FLAG
      .EVEN
000200      APTSIZE = 200
000001      APTENV  = 001
000100      APTSPOOL= 100
000040      APTCSUP= 040

```

.SBTTL CONSOLE MESSAGES

2					
3	066326	075	000		EQUALS: .ASCIZ @=@
4	066330	101	114	114	ALL: .ASCIZ @ALL@<CRLF>
5	066335	040	077	040	QUES: .ASCIZ @ ? @
6	066341	054	040	000	COMMA: .ASCIZ @, @
7	066344	200	124	131	MSHELP: .ASCIZ <CRLF>@TYPE HELP TEXT (L) N ? @
8	066375	200	122	115	CNSLO1: .ASCIZ <CRLF>@RMCS1=@
9	066405	040	114	111	CNSLO2: .ASCIZ @ LIMITS - LO= 160000, HI= 17XXXX@<CRLF>
10	066447	122	115	126	CNSLO3: .ASCIZ @RMVEC=@
11	066456	040	114	111	CNSLO4: .ASCIZ @ LIMITS - LO= 0, HI= 1000@<CRLF><LF>
12	066512	200	124	131	CNSLO7: .ASCII <CRLF>@TYPE 'A' TO TEST ALL DRIVES, OR TYPE DRIVE NUMBER(S)@
13	066577	200	101	116	.ASCIZ <CRLF>@AND TERMINATE INPUT WITH A CARRIAGE RETURN.@
14	066654	200			CNSLO8: .ASCII <CRLF>
15	066655	040	077	111	CNSLO9: .ASCIZ @ ?ILLEGAL INPUT@<CRLF>
16	066676	200	104	122	MSDRVS: .ASCIZ <CRLF>/DRIVE(S): /
17	066712	104	122	111	MSGDRV: .ASCIZ /DRIVE/
18	066720	200	125	116	SYSTAT: .ASCIZ <CRLF>/UNIT STATUS:/
19	066736	122	115	060	\$RM02: .ASCIZ /RM02/
20	066743	122	115	060	\$RM03: .ASCIZ /RM03/
21	066750	122	115	060	\$RM05: .ASCIZ /RM05/
22	066755	040	116	117	NOTRM: .ASCIZ @ NOT AN RM05/3/2@
23	066776	040	114	117	LODEV: .ASCIZ / LOAD DEVICE/
24	067013	040	116	117	NOTPRS: .ASCIZ / NOT PRESENT/
25	067030	040	116	117	NOTAVL: .ASCIZ / NOT AVAILABLE/
26	067047	040	117	106	UNTOFF: .ASCIZ / OFFLINE/
27	067060	040	117	116	UNTON: .ASCIZ / ONLINE/
28	067070	200	104	122	DRIVES: .ASCIZ <CRLF>/DRIVE(S) TO BE TESTED/
29	067117	116	117	116	NONE: .ASCIZ /NONE/
30	067124	116	000		N: .ASCIZ /N/
31	067126	131	000		Y: .ASCIZ /Y/
32	067130	040			BLNKS4: .ASCII / /
33	067131	040			BLNKS3: .ASCII / /
34	067132	040			BLNKS2: .ASCII / /
35	067133	040	000		BLNKS1: .ASCIZ / /
36					.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 ;NOP

067136

067136 020000

FUNCTION CODE TABLE

58	067140	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (2)
59	067142	132000	.WORD	ATA!OPI!IVC!IAE	:SEEK
60	067144	130000	.WCRD	ATA!OPI!IVC	:RECALIBRATE
61	067146	020000	.WORD	OPI	:DRIVE CLEAR
62	067150	030000	.WORD	OPI!IVC	:RELEASE
63	067152	130000	.WORD	OPI!ATA!IVC	:OFFSET
64	067154	130000	.WORD	OPI!ATA!IVC	:RETURN TO CENTERLINE
65	067156	020000	.WORD	OPI	:READ IN PRESET
66	067160	020000	.WORD	OPI	:PACK ACKNOWLEDGE
67	067162	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (24)
68	067164	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (26)
69	067166	132000	.WORD	ATA!OPI!IVC!IAE	:SEARCH
70	067170	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (32)
71	067172	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (34)
72	067174	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (36)
73	067176	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (40)
74	067200	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (42)
75	067202	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (44)
76	067204	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (46)
77	067206	073300	.WORD	WCE!OPI!IVC!IAE!AOE!HCE!ECH	:WRITE CHECK DATA
78	067210	073300	.WORD	WCE!OPI!IVC!IAE!AOE!HCE!ECH	:WRITE CHECK HEADER AND DATA
79	067212	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (54)
80	067214	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (56)
81	067216	037200	.WORD	OPI!IVC!WLE!IAE!AOE!HCE	:WRITE DATA
82	067220	037000	.WORD	OPI!IVC!WLE!IAE!AOE	:WRITE HEADER AND DATA
83	067222	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (64)
84	067224	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (66)
85	067226	033300	.WORD	OPI!IVC!IAE!AOE!HCE!ECH	:READ DATA
86	067230	033300	.WORD	OPI!IVC!IAE!AOE!HCE!ECH	:READ HEADER AND DATA
87	067232	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (74)
88	067234	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (76)

1
2
3 067236 001
4 067237 002
5 067240 004
6 067241 010
7 067242 020
8 067243 040
9 067244 100
10 067245 200

.SBTTL ATTENTION (ATA) TABLE

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

				.SBTTL ERROR MESSAGE TABLE		
1						
2						
3	067246	073634	000000	EMT1:	.WORD	EMS1,0
4	067252	073703	073720	000000	EMT2:	.WORD EMS2,EMS3,0
5	067260	073703	073763	000000	EMT3:	.WORD EMS2,EMS4,0
6	067266	074026	074056	000000	EMT4:	.WORD EMS5,EMS6,0
7	067274	074026	074170	000000	EMT5:	.WORD EMS5,EMS10,0
8	067302	100631	076047	000000	EMT6:	.WORD EMS167,EMS64,0
9	067310	076615	100656	000000	EMT7:	.WORD EMS110,EMS170,0
10	067316	074123	000000	EMT10:	.WORD	EMS7,0
11	067322	074170	000000	EMT11:	.WORD	EMS10,0
12	067326	074232	074243	000000	EMT12:	.WORD EMS11,EMS12,0
13	067334	074304	074315	074326	EMT13:	.WORD EMS13,EMS14,EMS15,EMS16,0
14	067346	074400	076047	000000	EMT14:	.WORD EMS17,EMS64,0
15	067354	074232	074463	000000	EMT15:	.WORD EMS11,EMS21,0
16	067362	074232	074506	074635	EMT16:	.WORD EMS11,EMS22,EMS27,0
17	067372	074232	074522	074646	EMT17:	.WORD EMS11,EMS23,EMS30,0
18	067402	074232	074550	074646	EMT20:	.WORD EMS11,EMS24,EMS30,0
19	067412	074232	074577	074635	EMT21:	.WORD EMS11,EMS25,EMS27,0
20	067422	074232	074614	074635	EMT22:	.WORD EMS11,EMS26,EMS27,0
21	067432	074232	074656	074646	EMT23:	.WORD EMS11,EMS31,EMS30,0
22	067442	074232	074705	074646	EMT24:	.WORD EMS11,EMS32,EMS30,0
23	067452	074232	074734	074646	EMT25:	.WORD EMS11,EMS33,EMS30,0
24	067462	074232	074762	074646	EMT26:	.WORD EMS11,EMS34,EMS30,0
25	067472	074232	075033	074646	EMT27:	.WORD EMS11,EMS35,EMS30,0
26	067502	074232	075062	074646	EMT30:	.WORD EMS11,EMS36,EMS30,0
27	067512	074232	075111	074646	EMT31:	.WORD EMS11,EMS37,EMS30,0
28	067522	074232	075137	074646	EMT32:	.WORD EMS11,EMS40,EMS30,0
29	067532	074232	075166	074646	EMT33:	.WORD EMS11,EMS41,EMS30,0
30	067542	074232	075214	074646	EMT34:	.WORD EMS11,EMS42,EMS30,0
31	067552	074232	075243	074646	EMT35:	.WORD EMS11,EMS43,EMS30,0
32	067562	074232	075272	074646	EMT36:	.WORD EMS11,EMS44,EMS30,0
33	067572	074232	075345	074646	EMT37:	.WORD EMS11,EMS45,EMS30,0
34	067602	076133	074440	000000	EMT40:	.WORD EMS66,EMS20,0
35	067610	076321	100032	076327	EMT41:	.WORD EMS75,EMS141,EMS76,0
36	067620	100123	100133	076255	EMT42:	.WORD EMS144,EMS145,EMS72,EMS76,0
37	067632	075437	075553	076327	EMT43:	.WORD EMS47,EMS53,EMS76,0
38	067642	076360	075553	076327	EMT44:	.WORD EMS77,EMS53,EMS76,0
39	067652	076406	075553	076327	EMT45:	.WORD EMS100,EMS53,EMS76,0
40	067662	076434	075553	076327	EMT46:	.WORD EMS101,EMS53,EMS76,0
41	067672	076065	076047	000000	EMT47:	.WORD EMS65,EMS64,0
42	067700	074304	074326	076021	EMT50:	.WORD EMS13,EMS15,EMS63,0
43	067710	074232	075410	074646	EMT51:	.WORD EMS11,EMS46,EMS30,EMS67,0
44	067722	075437	075553	076203	EMT52:	.WORD EMS47,EMS53,EMS67,EMS115,EMS140,EMS141,0
45	067740	075437	075553	076203	EMT53:	.WORD EMS47,EMS53,EMS67,EMS115,EMS141,EMS164,0
46	067756	075466	075553	076203	EMT54:	.WORD EMS50,EMS53,EMS67,0
47	067766	100060	100075	075553	EMT55:	.WORD EMS142,EMS143,EMS53,EMS67,0
48	070000	075515	076255	076203	EMT56:	.WORD EMS51,EMS72,EMS67,EMS115,EMS50,EMS70,0
49	070016	100631	076265	076203	EMT57:	.WORD EMS167,EMS73,EMS67,0
50	070026	075636	076203	077015	EMT60:	.WORD EMS56,EMS67,EMS115,EMS150,EMS152,EMS70,0
51	070044	076242	075636	076203	EMT61:	.WORD EMS71,EMS56,EMS67,EMS115,EMS150,EMS152,EMS72,0
52	070064	100011	076203	077015	EMT62:	.WORD EMS140,EMS67,EMS115,EMS47,EMS70,0
53	070100	000000			EMT63:	.WORD
54	070102	100216	100261	076230	EMT64:	.WORD EMS150,EMS152,EMS70,EMS115,EMS56,EMS73,EMS67,0
55	070122	075543	101256	101437	EMT65:	.WORD EMS52,EMS205,EMS214,EMS206,EMS115,EMS51,EMS72,0
56	070142	100570	076511	076255	EMT66:	.WORD EMS165,EMS103,EMS72,EMS124,0
57	070154	100570	076511	076255	EMT67:	.WORD EMS165,EMS103,EMS72,EMS171,0

58	070166	075410	074440	100463	EMT70:	.WORD	EMS46,EMS20,EMS163,0
59	070176	076242	076434	100463	EMT71:	.WORD	EMS71,EMS101,EMS163,0
60	070206	075437	076255	100463	EMT72:	.WORD	EMS47,EMS72,EMS163,EMS115,EMS140,EMS141,0
61	070224	075437	076255	100463	EMT73:	.WORD	EMS47,EMS72,EMS163,EMS115,EMS141,EMS72,0
62	070242	075636	075553	100463	EMT74:	.WORD	EMS56,EMS53,EMS163,0
63	070252	076242	075636	100463	EMT75:	.WORD	EMS71,EMS56,EMS163,EMS115,EMS150,EMS152,EMS72,0
64	070272	075466	075553	100463	EMT76:	.WORD	EMS50,EMS53,EMS163,0
65	070302	100060	100075	075553	EMT77:	.WORD	EMS142,EMS143,EMS53,EMS163,0
66	070314	076321	100032	100463	EMT100:	.WORD	EMS75,EMS141,EMS163,EMS115,EMS47,EMS70,0
67	070332	076321	100216	100463	EMT101:	.WORD	EMS75,EMS150,EMS163,EMS115,EMS56,EMS73,0
68	070350	100631	076265	100463	EMT102:	.WORD	EMS167,EMS73,EMS163,0
69	070360	100201	100235	076302	EMT103:	.WORD	EMS147,EMS151,EMS74,EMS163,0
70	070372	075515	076302	100463	EMT104:	.WORD	EMS51,EMS74,EMS163,EMS115,EMS50,EMS70,0
71	070410	100570	076406	100463	EMT105:	.WORD	EMS165,EMS100,EMS163,0
72	070420	100570	076360	100463	EMT106:	.WORD	EMS165,EMS77,EMS163,0
73	070430	100123	100133	075553	EMT107:	.WORD	EMS144,EMS145,EMS53,EMS163,EMS115,EMS143,EMS70,0
74	070450	076615	076655	077020	EMT110:	.WORD	EMS110,EMS112,EMS116,EMS111,0
75	070462	076741	073763	000000	EMT111:	.WORD	EMS113,EMS4,0
76	070470	076741	073720	000000	EMT112:	.WORD	EMS113,EMS3,0
77	070476	076615	077015	077020	EMT113:	.WORD	EMS110,EMS115,EMS116,EMS117,EMS114,0
78	070512	076741	077072	000000	EMT114:	.WORD	EMS113,EMS120,0
79	070520	077113	077553	077577	EMT115:	.WORD	EMS121,EMS132,EMS133,0
80	070530	077156	077553	077577	EMT116:	.WORD	EMS122,EMS132,EMS133,0
81	070540	077213	077553	077577	EMT117:	.WORD	EMS123,EMS132,EMS133,0
82	070550	077256	077553	077577	EMT120:	.WORD	EMS124,EMS132,EMS133,0
83	070560	077310	077553	077577	EMT121:	.WORD	EMS125,EMS132,EMS133,0
84	070570	077353	077553	077577	EMT122:	.WORD	EMS126,EMS132,EMS133,0
85	070600	077754	077553	077577	EMT123:	.WORD	EMS137,EMS132,EMS133,0
86	070610	077455	077553	077577	EMT124:	.WORD	EMS130,EMS132,EMS133,0
87	070620	077513	077553	077577	EMT125:	.WORD	EMS131,EMS132,EMS133,0
88	070630	077113	077553	077622	EMT126:	.WORD	EMS121,EMS132,EMS134,EMS123,0
89	070642	077156	077553	077622	EMT127:	.WORD	EMS122,EMS132,EMS134,EMS123,0
90	070654	077213	077553	077622	EMT130:	.WORD	EMS123,EMS132,EMS134,EMS123,0
91	070666	077256	077553	077622	EMT131:	.WORD	EMS124,EMS132,EMS134,EMS123,0
92	070700	077310	077553	077622	EMT132:	.WORD	EMS125,EMS132,EMS134,EMS123,0
93	070712	077353	077553	077622	EMT133:	.WORD	EMS126,EMS132,EMS134,EMS123,0
94	070724	077754	077553	077622	EMT134:	.WORD	EMS137,EMS132,EMS134,EMS123,0
95	070736	077455	077553	077622	EMT135:	.WORD	EMS130,EMS132,EMS134,EMS123,0
96	070750	077513	077553	077622	EMT136:	.WORD	EMS131,EMS132,EMS134,EMS123,0
97	070762	077113	077553	077664	EMT137:	.WORD	EMS121,EMS132,EMS135,0
98	070772	077213	077553	077664	EMT140:	.WORD	EMS123,EMS132,EMS135,0
99	071002	077113	077553	077727	EMT141:	.WORD	EMS121,EMS132,EMS136,0
100	071012	077754	077553	077727	EMT142:	.WORD	EMS137,EMS132,EMS136,0
101	071022	077256	077553	077727	EMT143:	.WORD	EMS124,EMS132,EMS136,0
102	071032	077310	077553	077727	EMT144:	.WORD	EMS125,EMS132,EMS136,0
103	071042	077353	077553	077727	EMT145:	.WORD	EMS126,EMS132,EMS136,0
104	071052	077513	077553	077727	EMT146:	.WORD	EMS131,EMS132,EMS136,0
105	071062	100705	077553	077727	EMT147:	.WORD	EMS171,EMS132,EMS136,0
106	071072	077455	077553	077727	EMT150:	.WORD	EMS130,EMS132,EMS136,0
107	071102	100011	077015	100032	EMT151:	.WORD	EMS140,EMS115,EMS141,EMS70,0
108	071114	100060	077015	100075	EMT152:	.WORD	EMS142,EMS115,EMS143,EMS72,0
109	071126	100123	100133	100162	EMT153:	.WORD	EMS144,EMS145,EMS146,EMS115,EMS143,EMS72,0
110	071144	100123	100133	074440	EMT154:	.WORD	EMS144,EMS145,EMS20,EMS115,EMS143,EMS70,0
111	071162	100216	100261	100327	EMT155:	.WORD	EMS150,EMS152,EMS154,EMS153,0
112	071174	100201	100235	100327	EMT156:	.WORD	EMS147,EMS151,EMS154,EMS155,0
113	071206	100201	100235	100363	EMT157:	.WORD	EMS147,EMS151,EMS156,EMS157,0
114	071220	100433	100363	100416	EMT160:	.WORD	EMS161,EMS156,EMS160,0

115	071230	074577	074635	100363	EMT161:	.WORD	EMS25,EMS27,EMS156,EMS160,0
116	071242	074614	074635	100363	EMT162:	.WORD	EMS26,EMS27,EMS156,EMS160,0
117	071254	075437	100463	100011	EMT163:	.WORD	EMS47,EMS163,EMS140,0
118	071264	075437	074440	000000	EMT164:	.WORD	EMS47,EMS20,0
119	071272	075636	074440	000000	EMT165:	.WORD	EMS56,EMS20,0
120	071300	075410	074440	000000	EMT166:	.WORD	EMS46,EMS20,0
121	071306	101757	074440	000000	EMT167:	.WORD	EMS224,EMS20,0
122	071314	101215	100032	074646	EMT170:	.WORD	EMS203,EMS141,EMS30,EMS202,0
123	071326	100631	100327	100401	EMT171:	.WORD	EMS167,EMS154,EMS157,0
124	071336	100631	100327	100343	EMT172:	.WORD	EMS167,EMS154,EMS155,0
125	071346	100705	077553	077577	EMT173:	.WORD	EMS171,EMS132,EMS133,0
126	071356	100705	077553	077622	EMT174:	.WORD	EMS171,EMS132,EMS134,EMS123,0
127	071370	076741	101060	000000	EMT175:	.WORD	EMS113,EMS177,0
128	071376	101102	101117	000000	EMT176:	.WORD	EMS200,EMS201,0
129	071404	000000			EMT177:	.WORD	
130	071406	101215	075515	074646	EMT200:	.WORD	EMS203,EMS51,EMS30,EMS202,0
131	071420	101215	101230	074646	EMT201:	.WORD	EMS203,EMS204,EMS30,EMS202,0
132	071432	101215	075466	074646	EMT202:	.WORD	EMS203,EMS50,EMS30,EMS202,0
133	071444	101215	100075	074646	EMT203:	.WORD	EMS203,EMS143,EMS30,EMS202,0
134	071456	100216	076302	101165	EMT204:	.WORD	EMS150,EMS74,EMS202,EMS115,EMS152,EMS72,0
135	071474	075466	076511	076255	EMT205:	.WORD	EMS50,EMS103,EMS72,0
136	071504	076520	076265	101165	EMT206:	.WORD	EMS104,EMS73,EMS202,0
137	071514	076321	100032	077015	EMT207:	.WORD	EMS75,EMS141,EMS115,EMS140,0
138	071526	076321	100216	000000	EMT210:	.WORD	EMS75,EMS150,0
139	071534	075515	076255	077015	EMT211:	.WORD	EMS51,EMS72,EMS115,EMS50,EMS70,0
140	071550	100060	075553	077015	EMT212:	.WORD	EMS142,EMS53,EMS115,EMS143,EMS72,0
141	071564	075466	076511	075553	EMT213:	.WORD	EMS50,EMS103,EMS53,0
142	071574	075543	101256	100604	EMT214:	.WORD	EMS52,EMS205,EMS166,EMS206,EMS115,EMS51,EMS72,0
143	071614	075543	077050	075636	EMT215:	.WORD	EMS52,EMS117,EMS56,EMS163,0
144	071626	075636	074646	076047	EMT216:	.WORD	EMS56,EMS30,EMS64,0
145	071636	075410	074646	076047	EMT217:	.WORD	EMS46,EMS30,EMS64,0
146	071646	074656	074646	076047	EMT220:	.WORD	EMS31,EMS30,EMS64,0
147	071656	075437	074646	076047	EMT221:	.WORD	EMS47,EMS30,EMS64,0
148	071666	075543	077050	076360	EMT222:	.WORD	EMS52,EMS117,EMS77,0
149	071676	075543	077050	076021	EMT223:	.WORD	EMS52,EMS117,EMS63,0
150	071706	000000			EMT224:	.WORD	
151	071710	000000			EMT225:	.WORD	
152	071712	000000			EMT226:	.WORD	
153	071714	000000			EMT227:	.WORD	
154	071716	000000			EMT230:	.WORD	
155	071720	000000			EMT231:	.WORD	
156	071722	000000			EMT232:	.WORD	
157	071724	000000			EMT233:	.WORD	
158	071726	000000			EMT234:	.WORD	
159	071730	000000			EMT235:	.WORD	
160	071732	000000			EMT236:	.WORD	
161	071734	000000			EMT237:	.WORD	
162	071736	000000			EMT240:	.WORD	
163	071740	000000			EMT241:	.WORD	
164	071742	000000			EMT242:	.WORD	
165	071744	000000			EMT243:	.WORD	
166	071746	000000			EMT244:	.WORD	
167	071750	000000			EMT245:	.WORD	
168	071752	100631	077553	101315	EMT246:	.WORD	EMS167,EMS132,EMS207,0
169	071762	100631	077553	101342	EMT247:	.WORD	EMS167,EMS132,EMS210,EMS125,0
170	071774	100631	101353	101342	EMT250:	.WORD	EMS167,EMS211,EMS210,EMS207,EMS206,0
171	072010	101371	101414	000000	EMT251:	.WORD	EMS212,EMS213,0

172	072016	100570	101371	000000	EMT252: .WORD	EMS165,EMS212,0
173	072024	100201	100235	100363	EMT253: .WORD	EMS147,EMS151,EMS156,EMS210,EMS26,EMS27,0
174	072042	101215	076434	074646	EMT254: .WORD	EMS203,EMS101,EMS30,0
175	072052	101215	100631	074646	EMT255: .WORD	EMS203,EMS167,EMS30,0
176	072062	101215	076406	074646	EMT256: .WORD	EMS203,EMS100,EMS30,0
177	072072	074232	075410	074646	EMT257: .WORD	EMS11,EMS46,EMS30,EMS102,0
178	072104	075543	077050	075636	EMT260: .WORD	EMS52,EMS117,EMS56,EMS102,0
179	072116	075543	101256	101561	EMT261: .WORD	EMS52,EMS205,EMS220,EMS206,EMS115,EMS51,EMS72,0
180	072136	076520	076265	101165	EMT262: .WORD	EMS104,EMS73,EMS202,0
181	072146	075466	075553	076462	EMT263: .WORD	EMS50,EMS53,EMS102,0
182	072156	075636	075553	076462	EMT264: .WORD	EMS56,EMS53,EMS102,EMS115,EMS150,EMS152,EMS70,0
183	072176	076242	075636	076462	EMT265: .WORD	EMS71,EMS56,EMS102,EMS115,EMS150,EMS152,EMS72,0
184	072216	100060	100075	075553	EMT266: .WORD	EMS142,EMS143,EMS53,EMS102,0
185	072230	075466	100162	077015	EMT267: .WORD	EMS50,EMS146,EMS115,EMS52,EMS117,EMS46,0
186	072246	075437	075553	076462	EMT270: .WORD	EMS47,EMS53,EMS102,EMS115,EMS140,0
187	072262	075437	075553	076462	EMT271: .WORD	EMS47,EMS53,EMS102,EMS115,EMS141,EMS72,0
188	072300	076321	100032	076462	EMT272: .WORD	EMS75,EMS141,EMS102,EMS115,EMS47,EMS73,0
189	072316	075515	076302	076462	EMT273: .WORD	EMS51,EMS74,EMS102,EMS115,EMS50,EMS70,0
190	072334	076021	075553	075713	EMT274: .WORD	EMS63,EMS53,EMS57,EMS115,EMS41,EMS146,0
191	072352	077020	075553	075166	EMT275: .WORD	EMS116,EMS53,EMS41,EMS57,0
192	072364	075437	075553	075713	EMT276: .WORD	EMS47,EMS53,EMS57,EMS115,EMS140,0
193	072400	075437	075553	075713	EMT277: .WORD	EMS47,EMS53,EMS57,EMS115,EMS141,EMS72,0
194	072416	075636	075553	075713	EMT300: .WORD	EMS56,EMS53,EMS57,EMS115,EMS150,EMS152,EMS70,0
195	072436	076242	075636	075553	EMT301: .WORD	EMS71,EMS56,EMS53,EMS57,EMS115,EMS150,EMS152,EMS72,0
196	072460	100570	076406	076511	EMT302: .WORD	EMS165,EMS100,EMS103,EMS57,0
197	072472	100570	076434	076511	EMT303: .WORD	EMS165,EMS101,EMS103,EMS57,0
198	072504	100570	076360	076511	EMT304: .WORD	EMS165,EMS77,EMS103,EMS57,0
199	072516	075410	074646	076047	EMT305: .WORD	EMS46,EMS30,EMS64,EMS57,0
200	072530	075466	075553	075713	EMT306: .WORD	EMS50,EMS53,EMS57,0
201	072540	075466	100162	077015	EMT307: .WORD	EMS50,EMS146,EMS115,EMS52,EMS117,EMS46,EMS57,0
202	072560	100060	100075	075553	EMT310: .WORD	EMS142,EMS143,EMS53,EMS57,0
203	072572	076520	076511	075553	EMT311: .WORD	EMS104,EMS103,EMS53,EMS57,0
204	072604	076547	076511	075553	EMT312: .WORD	EMS105,EMS103,EMS53,EMS57,0
205	072616	100123	100133	076511	EMT313: .WORD	EMS144,EMS145,EMS103,EMS57,EMS115,EMS143,EMS70,0
206	072636	075214	076511	075553	EMT314: .WORD	EMS42,EMS103,EMS53,EMS57,0
207	072650	074656	076511	075553	EMT315: .WORD	EMS31,EMS103,EMS53,EMS57,0
208	072662	076242	074656	076511	EMT316: .WORD	EMS71,EMS31,EMS103,EMS57,0
209	072674	075243	076511	075713	EMT317: .WORD	EMS43,EMS103,EMS57,0
210	072704	075345	076511	075713	EMT320: .WORD	EMS45,EMS103,EMS57,0
211	072714	075272	076511	075713	EMT321: .WORD	EMS44,EMS103,EMS57,0
212	072724	076576	074440	000000	EMT322: .WORD	EMS106,EMS20,0
213	072732	075062	076511	075713	EMT323: .WORD	EMS36,EMS103,EMS57,0
214	072742	100760	075062	076511	EMT324: .WORD	EMS173,EMS36,EMS103,EMS57,0
215	072754	100740	075062	076511	EMT325: .WORD	EMS172,EMS36,EMS103,EMS57,0
216	072766	074304	100775	074326	EMT326: .WORD	EMS13,EMS174,EMS15,EMS35,EMS53,EMS175,0
217	073004	100201	100235	076302	EMT327: .WORD	EMS147,EMS151,EMS74,EMS175,0
218	073016	076133	075553	101003	EMT330: .WORD	EMS66,EMS53,EMS175,0
219	073026	074734	076511	075553	EMT331: .WORD	EMS33,EMS103,EMS53,EMS175,0
220	073040	075137	076511	075553	EMT332: .WORD	EMS40,EMS103,EMS53,EMS57,0
221	073052	075515	076302	075713	EMT333: .WORD	EMS51,EMS74,EMS57,EMS115,EMS50,EMS70,0
222	073070	076321	100032	075713	EMT334: .WORD	EMS75,EMS141,EMS57,EMS115,EMS47,EMS73,0
223	073106	076321	100216	100261	EMT335: .WORD	EMS75,EMS150,EMS152,EMS57,EMS115,EMS56,EMS73,0
224	073126	075741	075754	076003	EMT336: .WORD	EMS60,EMS61,EMS62,0
225	073136	077020	077050	074762	EMT337: .WORD	EMS116,EMS117,EMS34,0
226	073146	074762	075553	075565	EMT340: .WORD	EMS34,EMS53,EMS54,EMS111,0
227	073160	075607	074762	000000	EMT341: .WORD	EMS55,EMS34,0
228	073166	075543	077050	075636	EMT342: .WORD	EMS52,EMS117,EMS56,EMS57,0

ERROR MESSAGE TABLE

229	073200	075543	077050	075345	EMT343: .WORD	EMS52,EMS117,EMS45,EMS57,0
230	073212	075543	077050	075272	EMT344: .WORD	EMS52,EMS117,EMS44,EMS57,0
231	073224	075543	077050	101601	EMT345: .WORD	EMS52,EMS117,EMS221,0
232	073234	076576	074440	077015	EMT346: .WORD	EMS106,EMS20,EMS115,EMS223,EMS72,0
233	073250	075543	101256	101651	EMT347: .WORD	EMS52,EMS205,EMS222,EMS206,0
234	073262	076321	100216	076462	EMT350: .WORD	EMS75,EMS150,EMS102,EMS115,EMS56,EMS73,0
235	073300	100631	076265	076462	EMT351: .WORD	EMS167,EMS73,EMS102,0
236	073310	101455	000000		EMT352: .WORD	EMS215,0
237	073314	101526	101215	101476	EMT353: .WORD	EMS217,EMS203,EMS216,0
238	073324	101601	074440	000000	EMT354: .WORD	EMS221,EMS20,0

1	073332	102031	102627	102704	EHT11: .WORD	EH1,STSH1,STSH2,STSH4,0
2	073344	102627	102704	103031	EHT2: .WORD	STSH1,STSH2,STSH4,0
3						
4	073354	102050	000000		EHT110: .WORD	EH110,0
5	073360	102057	000000		EHT111: .WORD	EH111,0
6						
7	073364	102076	000000		EHT114: .WORD	EH114,0
8	073370	102125	102627	102704	EHT223: .WORD	EH223,STSH1,STSH2,STSH4,0
9	073402	102153	102627	102704	EHT256: .WORD	EH256,STSH1,STSH2,STSH4,0
10						
11	073414	102227	102627	102704	EHT336: .WORD	EH336,STSH1,STSH2,STSH4,0
12	073426	102266	102627	102704	EHT337: .WORD	EH337,STSH1,STSH2,STSH4,0
13	073440	102423	102627	102704	EHT344: .WORD	EH344,STSH1,STSH2,STSH4,0
14						
15	073452	102561	000000		EHT353: .WORD	EH353,0

ERROR MESSAGE TABLE

1	073456	103070	103164	103202	EDT1: .WORD	ED1,STSD1,STSD2,STSD4
2	073466	103164	103202	103234	EDT2: .WORD	STSD1,STSD2,STSD4
3						
4	073474	103076			EDT110: .WORD	ED110
5	073476	103102			EDT111: .WORD	ED111
6						
7	073500	103110			EDT114: .WORD	ED114
8	073502	103120	103164	103202	EDT223: .WORD	ED223,STSD1,STSD2,STSD4
9						
10	073512	103130	103164	103202	EDT336: .WORD	ED336,STSD1,STSD2,STSD4
11	073522	103142	103164	103202	EDT337: .WORD	ED337,STSD1,STSD2,STSD4
12	073532	103142	103164	103202	EDT344: .WORD	ED337,STSD1,STSD2,STSD4,0
13						
14	073544	103154			EDT353: .WORD	ED353

1	073546	103247	103265	103265	EFT11: .WORD	EF111,STSF,STSF,STSF
2	073556	103265	103265	103265	EFT2: .WORD	STSF,STSF,STSF
3						
4	073564	103246			EFT110: .WORD	EF110
5	073566	103247			EFT111: .WORD	EF111
6						
7	073570	103251			EFT114: .WORD	EF114
8	073572	103251	103265	103265	EFT223: .WORD	EF114,STSF,STSF,STSF
9						
10	073602	103254	103265	103265	EFT336: .WORD	EF336,STSF,STSF,STSF
11	073612	103254	103265	103265	EFT337: .WORD	EF336,STSF,STSF,STSF
12	073622	103254	103265	103265	EFT344: .WORD	EF336,STSF,STSF,STSF
13						
14	073632	103251			EFT353: .WORD	EF114

				.SBTTL ERROR MESSAGE STRINGS	
1					
2					
3	073634	127	122	117	EMS1: .ASCIZ @WRONG UNIT SELECTED (RMCS2, BITS 0-2) @
4	073703	104	105	126	EMS2: .ASCIZ @DEVICE WENT @
5	073720	125	116	101	EMS3: .ASCIZ @UNAVAILABLE 'DVA' (RMCS1, BIT 11) @
6	073763	116	117	116	EMS4: .ASCIZ @NONEXISTENT 'NED' (RMCS2, BIT 12) @
7	074026	103	117	115	EMS5: .ASCIZ @COMMAND NOT COMPLETED, @
8	074056	103	117	116	EMS6: .ASCIZ @CONTROLLER NOT READY (RMCS1, BIT 7) @
9	074123	104	122	111	EMS7: .ASCIZ @DRIVE NOT READY 'DRY' (RMDS, BIT 7) @
10	074170	107	117	040	EMS10: .ASCIZ @GO NOT RESET 'GO' (RMCS1, BIT 0) @
11	074232	111	116	126	EMS11: .ASCIZ @INVALID @
12	074243	106	125	116	EMS12: .ASCIZ @FUNCTION CODE (RMCS1, BITS 1-5) @
13	074304	115	101	123	EMS13: .ASCIZ @MASSBUS @
14	074315	103	117	116	EMS14: .ASCIZ @CONTROL @
15	074326	102	125	123	EMS15: .ASCIZ @BUS PARITY ERROR @
16	074350	042	115	103	EMS16: .ASCIZ @'MCPE' (RMCS1, BIT 13) @
17	074400	124	122	101	EMS17: .ASCIZ @TRANSFER ERROR (RMCS1, BIT 14) @
18	074440	123	110	117	EMS20: .ASCIZ @SHOULD NOT BE SET @
19	074463	127	117	122	EMS21: .ASCIZ @WORD COUNT (RMWC) @
20	074506	102	125	123	EMS22: .ASCIZ @BUS (RMB) @
21	074522	042	114	102	EMS23: .ASCIZ @'LBT' (RMDS, BIT 10) @
22	074550	042	101	117	EMS24: .ASCIZ @'AOE' (RMER1, BIT 09) @
23	074577	104	111	123	EMS25: .ASCIZ @DISK (RMDA) @
24	074614	103	131	114	EMS26: .ASCIZ @CYLINDER (RMDC) @
25	074635	101	104	104	FMS27: .ASCIZ @ADDRESS @
26	074646	123	124	101	EMS30: .ASCIZ @STATUS @
27	074656	042	127	114	EMS31: .ASCIZ @'WLE' (RMER1, BIT 11) @
28	074705	042	125	120	EMS32: .ASCIZ @'UPE' (RMCS2, BIT 13) @
29	074734	042	127	103	EMS33: .ASCIZ @'WCF' (RMER1, BIT 5) @
30	074762	127	122	111	EMS34: .ASCIZ @WRITE CHECK ERROR-'WCE' (RMCS2, BIT 14) @
31	075033	042	115	104	EMS35: .ASCIZ @'MDPE' (RMCS2, BIT 1) @
32	075062	042	104	103	EMS36: .ASCIZ @'DCK' (RMER1, BIT 15) @
33	075111	042	105	103	EMS37: .ASCIZ @'ECH' (RMER1, BIT 6) @
34	075137	042	104	114	EMS40: .ASCIZ @'DLT' (RMCS2, BIT 15) @
35	075166	042	115	130	EMS41: .ASCIZ @'MXF' (RMCS2, BIT 9) @
36	075214	042	104	124	EMS42: .ASCIZ @'DTE' (RMER1, BIT 12) @
37	075243	042	110	103	EMS43: .ASCIZ @'HCRC' (RMER1, BIT 8) @
38	075272	110	105	101	EMS44: .ASCIZ @HEADER COMPARE ERROR 'HCE' (RMER1, BIT 7) @
39	075345	106	117	122	EMS45: .ASCIZ @FORMAT ERROR 'FER' (RMER1, BIT 4) @
40	075410	042	111	101	EMS46: .ASCIZ @'IAE' (RMER1, BIT 10) @
41	075437	042	117	120	EMS47: .ASCIZ @'OPI' (RMER1, BIT 13) @
42	075466	042	123	113	EMS50: .ASCIZ @'SKI' (RMER2, BIT 14) @
43	075515	042	120	111	EMS51: .ASCIZ @'PIP' (RMDS, BIT 13) @
44	075543	124	110	105	EMS52: .ASCIZ @THE RM @
45	075553	104	105	124	EMS53: .ASCIZ @DETECTED @
46	075565	101	124	040	EMS54: .ASCIZ @AT AN UNEXPECTED @
47	075607	111	116	103	EMS55: .ASCIZ @INCORRECT DATA DURING @
48	075636	111	116	126	EMS56: .ASCIZ @INVALID COMMAND ERROR 'IVC' (RMER2, BIT 12) @
49	075713	104	125	122	EMS57: .ASCIZ @DURING DATA TRANSFER @
50	075741	104	101	124	EMS60: .ASCIZ @DATA READ @
51	075754	104	117	105	EMS61: .ASCIZ @DOES NOT COMPARE WITH @
52	076003	104	101	124	EMS62: .ASCIZ @DATA WRITTEN @
53	076021	042	120	101	EMS63: .ASCIZ @'PAR' (RMER1, BIT 3) @
54	076047	111	123	040	EMS64: .ASCIZ @IS INCORRECT @
55	076065	103	117	115	EMS65: .ASCIZ @COMPOSITE ERROR 'ERR' (RMDS, BIT 14) @
56	076133	104	101	124	EMS66: .ASCIZ @DATA PARITY ERROR 'DPE' (RMER2, BIT 3) @
57	076203	104	125	122	EMS67: .ASCIZ @DURING SEEK COMMAND @

58	076230	111	123	040	EMS70:	.ASCIZ	@IS RESET @
59	076242	105	122	122	EMS71:	.ASCIZ	@ERRONEOUS @
60	076255	111	123	040	EMS72:	.ASCIZ	@IS SET @
61	076265	104	111	104	EMS73:	.ASCIZ	@DID NOT SET @
62	076302	104	111	104	EMS74:	.ASCIZ	@DID NOT RESET @
63	076321	114	117	123	EMS75:	.ASCIZ	@LOST @
64	076327	104	125	122	EMS76:	.ASCIZ	@DURING PACK ACK COMMAND @
65	076360	042	122	115	EMS77:	.ASCIZ	@'RMR' (RMER, BIT 2) @
66	076406	042	111	114	EMS100:	.ASCIZ	@'ILR' (RMER1, BIT 1) @
67	076434	042	111	114	EMS101:	.ASCIZ	@'ILF' (RMER1, BIT 0) @
68	076462	104	125	122	EMS102:	.ASCIZ	@DURING SEARCH COMMAND @
69	076511	105	122	122	EMS103:	.ASCIZ	@ERROR @
70	076520	042	114	102	EMS104:	.ASCIZ	@'LBC' (RMER2, BIT 10) @
71	076547	042	114	123	EMS105:	.ASCIZ	@'LSC' (RMER2, BIT 11) @
72	076576	110	105	101	EMS106:	.ASCIZ	@HEADER ERRORS @
73	076615	102	125	123	EMS110:	.ASCIZ	@BUS TIMEOUT (04 TRAP) @
74	076644	101	104	104	EMS111:	.ASCIZ	@ADDRESS @
75	076655	127	110	105	EMS112:	.ASCII	@WHEN READING/WRITING RH REGISTERS @
76	076717	101	124	040		.ASCIZ	@AT THE FOLLOWING @
77	076741	124	110	105	EMS113:	.ASCIZ	@THE SELECTED DEVICE IS @
78	076771	116	117	116	EMS114:	.ASCIZ	@NONEXISTENT DEVICE @
79	077015	015	012	000	EMS115:	.ASCIZ	<CR><LF>
80	077020	124	110	105	EMS116:	.ASCIZ	@THE MASSBUS CONTROLLER @
81	077050	106	101	111	EMS117:	.ASCIZ	@FAILED TO DETECT @
82	077072	116	117	124	EMS120:	.ASCIZ	@NOT AN RM05/3/2 @
83	077113	103	117	116	EMS121:	.ASCIZ	@CONTROL STATUS REGISTER 1, RMCS1, @
84	077156	102	125	123	EMS122:	.ASCIZ	@BUS ADDRESS REGISTER, RMBA, @
85	077213	103	117	116	EMS123:	.ASCIZ	@CONTROL STATUS REGISTER 2, RMCS2, @
86	077256	105	122	122	EMS124:	.ASCIZ	@ERROR REGISTER 1, RMER1, @
87	077310	101	124	124	EMS125:	.ASCIZ	@ATTENTION SUMMARY REGISTER, RMAS, @
88	077353	115	101	111	EMS126:	.ASCIZ	@MAINTENANCE REGISTER #1, RMMR #1, @
89	077416	105	103	103	EMS127:	.ASCIZ	@ECC POSITION REGISTER, RMEC1, @
90	077455	105	103	103	EMS130:	.ASCIZ	@ECC PATTERN REGISTER, RMEC2, @
91	077513	115	101	111	EMS131:	.ASCIZ	@MAINTENANCE REGISTER 2, RMMR2, @
92	077553	116	117	124	EMS132:	.ASCIZ	@NOT INITIALIZED BY @
93	077577	125	116	111	EMS133:	.ASCIZ	@UNIBUS INITIALIZE @
94	077622	103	117	116	EMS134:	.ASCIZ	@CONTROLLER CLEAR, I.E. BIT 5 OF @
95	077664	122	110	057	EMS135:	.ASCIZ	@RH/RM ERROR CLEAR (RMCS1, BIT 14) @
96	077727	104	122	111	EMS136:	.ASCIZ	@DRIVE CLEAR COMMAND @
97	077754	104	122	111	EMS137:	.ASCIZ	@DRIVE STATUS REGISTER, RMDS @
98	100011	115	105	104	EMS140:	.ASCIZ	@MEDIUM OFF LINE @
99	100032	042	115	117	EMS141:	.ASCIZ	@'MOL' (RMDS, BIT 12) @
100	100060	104	122	111	EMS142:	.ASCIZ	@DRIVE FAULT @
101	100075	042	104	126	EMS143:	.ASCIZ	@'DVC' (RMER2, BIT 7) @
102	100123	125	116	123	EMS144:	.ASCIZ	@UNSAFE @
103	100133	042	125	116	EMS145:	.ASCIZ	@'UNS' (RMER1, BIT 14) @
104	100162	123	110	117	EMS146:	.ASCIZ	@SHOULD BE SET @
105	100201	117	106	106	EMS147:	.ASCIZ	@OFFSET MODE @
106	100216	040	126	117	EMS150:	.ASCIZ	@ VOLUME VALID @
107	100235	042	117	115	EMS151:	.ASCIZ	@'OM' (RMDS, BIT 0) @
108	100261	042	126	126	EMS152:	.ASCIZ	@'VV' (RMDS, BIT 6) @
109	100305	120	101	103	EMS153:	.ASCIZ	@PACK ACK COMMAND @
110	100327	116	117	124	EMS154:	.ASCIZ	@NOT SET BY @
111	100343	117	106	106	EMS155:	.ASCIZ	@OFFSET COMMAND @
112	100363	116	117	124	EMS156:	.ASCIZ	@NOT RESET BY @
113	100401	122	124	103	EMS157:	.ASCIZ	@RTC COMMAND @
114	100416	122	111	120	EMS160:	.ASCIZ	@RIP COMMAND @

115	100433	117	106	106	EMS161: .ASCIZ @OFFSET REGISTER (RMOF) @
116	100463				EMS162: ;<UNUSED>
117	100463	104	125	122	EMS163: .ASCIZ @DURING RECALIBRATE @
118	100507	111	123	040	EMS164: .ASCII @IS INTERMITTENT OR DRIVE DIDNT DROP ON @
119	100556	103	131	114	.ASCIZ @CYLINDER @
120	100570	125	116	105	EMS165: .ASCIZ @UNEXPECTED @
121	100604	122	105	103	EMS166: .ASCIZ @RECALIBRATE COMMAND @
122	100631	042	101	124	EMS167: .ASCIZ @'ATA' (RMDS, BIT15) @
123	100656	127	110	105	EMS170: .ASCIZ @WHEN READING REGISTER @
124	100705	105	122	122	EMS171: .ASCIZ @ERROR REGISTER #2, RMER2, @
125	100740	116	117	116	EMS172: .ASCIZ @NONRECOVERABLE @
126	100760	122	105	103	EMS173: .ASCIZ @RECOVERABLE @
127	100775	104	101	124	EMS174: .ASCIZ @DATA @
128	101003	104	125	122	EMS175: .ASCIZ @DURING WRITE COMMAND @
129	101031	042	117	120	EMS176: .ASCIZ @'DPE' (RMER2, BIT 13) @
130	101060	111	116	040	EMS177: .ASCIZ @IN WRITE PROTECT @
131	101102	103	101	116	EMS200: .ASCIZ @CAN NOT SET @
132	101117	104	111	101	EMS201: .ASCIZ @DIAGNOSTIC MODE 'DMD' (RMMR1, BIT 0) @
133	101165	104	125	122	EMS202: .ASCIZ @DURING DIAGNOSTIC MODE @
134	101215	111	116	103	EMS203: .ASCIZ @INCORRECT @
135	101230	042	127	122	EMS204: .ASCIZ @'WRL' (RMDS, BIT 11) @
136	101256	105	130	105	EMS205: .ASCIZ @EXECUTED @
137	101270	127	111	124	EMS206: .ASCIZ @WITH COMP ERROR SET @
138	101315	042	107	117	EMS207: .ASCIZ @'GO' (RMCS1, BIT 0) @
139	101342	127	122	111	EMS210: .ASCIZ @WRITING @
140	101353	127	101	123	EMS211: .ASCIZ @WAS RESET BY @
141	101371	120	122	117	EMS212: .ASCIZ @PROGRAM INTERRUPT @
142	101414	127	101	123	EMS213: .ASCIZ @WAS NOT GENERATED @
143	101437	123	105	105	EMS214: .ASCIZ @SEEK COMMAND @
144	101455	120	122	117	EMS215: .ASCIZ @PROGRAM TIMEOUT @
145	101476	104	125	122	EMS216: .ASCIZ @DURING LOOK AHEAD TEST @
146	101526	114	117	117	EMS217: .ASCIZ @LOOK AHEAD REGISTER, RMLA, @
147	101561	123	105	101	EMS220: .ASCIZ @SEARCH COMMAND @
148	101601	102	101	104	EMS221: .ASCIZ @BAD SECTOR ERROR 'BSE' (RMER2, BIT 15) @
149	101651	101	040	104	EMS222: .ASCIZ @A DATA TRANSFER COMMAND @
150	101702	110	105	101	EMS223: .ASCIZ @HEADER COMPARE INHIBIT 'HCI' (RMOF, BIT 10) @
151	101757	116	117	116	EMS224: .ASCIZ @NONEXISTENT MEMORY 'NEM' (RMCS2, BIT 11) @

ERROR MESSAGE STRINGS

1	102031	105	130	120	EH1:	.ASCIZ	@EXPCTD	RECEVD@				
2	102050	102	125	123	EH110:	.ASCIZ	@BUSADR@					
3	102057	040	122	115	EH111:	.ASCIZ	@ RMCS2	RMCS1@				
4												
5	102076	122	105	103	EH114:	.ASCIZ	@RECEVD	SNGPRT	DULPRT@			
6	102125	105	130	120	EH223:	.ASCIZ	@EXPCTD	RECEVD	DATA@			
7	102153	105	130	120	EH256:	.ASCIZ	@EXPCTD	RECEVD	RGSTR@<CRLF>			
8	102201	123	124	101		.ASCIZ	@STATUS	STATUS	INDEX@			
9												
10	102227	107	104	101	FH336:	.ASCIZ	@GDADRS	GDDATA	BDADRS	BDDATA@		
11	102266	122	115	103	EH337:	.ASCII	@RMCS2	STATUS	FAILING	DATA@<CRLF>		
12	102325	137	137	137		.ASCII	@			@<CRLF>		
13	102364	105	130	120		.ASCIZ	@EXPCTD	RECEVD	BIT	ADRESS@		
14												
15	102423	122	115	105	EH344:	.ASCII	@RMER1	STATUS	HEADER	FAILING@<CRLF>		
16	102463	137	137	137		.ASCII	@		WORD	BIT@<CRLF>		
17	102521	105	130	120		.ASCIZ	@EXPCTD	RECEVD	NUMBER	POSITION@		
18	102561	105	130	120	EH353:	.ASCII	@EXPCTD	RECEVD@<CRLF>				
19	102600	040	122	115		.ASCIZ	@ RMLA	RMLA	RMOF @			
20												
21	102627	040	122	115	STSH1:	.ASCII	@ RMCS1	RMCS2	RMDS	RMER1	RMER2@	
22	102674	040	040	040		.ASCIZ	@ RMAS@					
23	102704	040	122	115	STSH2:	.ASCII	@ RMWC	RMBA	RMDA	RMOF	RMDC@	
24	102751	040	040	040		.ASCIZ	@ RMEC1	RMEC2@				
25	102773	040	122	115	STSH3:	.ASCIZ	@ RMDA	RMDC	RMOF	RMLA@		
26	103031	040	122	115	STSH4:	.ASCIZ	@ RMMR1	RMMR2	RMDT	RMSN@		
27					.EVEN							

1	103070	001140	001142	000000	ED1:	.WORD	\$GDDAT,\$BDDAT,0
2	103076	001276	000000		ED110:	.WORD	\$BASE,0
3	103102	001174	001176	000000	ED111:	.WORD	\$TMP0,\$TMP1,0
4							
5	103110	001362	001176	001200	ED114:	.WORD	RMDTI,\$TMP1,\$TMP2,0
6	103120	001140	001142	001174	ED223:	.WORD	\$GDDAT,\$BDDAT,\$TMP0,0
7							
8	103130	001134	001140	001136	ED336:	.WORD	\$GDADR,\$GDDAT,\$BDADR,\$BDDAT,0
9							
10	103142	001140	001142	001174	ED337:	.WORD	\$GDDAT,\$BDDAT,\$TMP0,\$TMP1,0
11	103154	001140	001142	001442	ED353:	.WORD	\$GDDAT,\$BDDAT,RMOFO,0
12							
13	103164	001334	001344	001346	STSD1:	.WORD	RMCS1I, RMCS2I, RMDSI, RMER1I, RMER2I, RMASI, 0
14	103202	001336	001340	001342	STSD2:	.WORD	RMWCI, RMBAI, RMDAI, RMOFI, RMDCI, RMECI
15	103216	001402	000000			.WORD	RMEC2I, 0
16	103222	001342	001370	001366	STSD3:	.WORD	RMDAI, RMDCI, RMOFI, RMLAI, 0
17	103234	001360	001374	001362	STSD4:	.WORD	RMMR1I, RMMR2I, RMDTI, RMSNI, 0

1	103246	000			EF110:	.BYTE	0
2	103247	000	000		EF111:	.BYTE	0,0
3	103251	000	000	000	EF114:	.BYTE	0,0,0
4	103254	000	000	000	EF336:	.BYTE	0,0,0,0
5	103260	000	000	000	EF337:	.BYTE	0,0,0,0,0
6							
7	103265	000	000	000	STSF:	.BYTE	0,0,0,0,0,0,0

1 103274
2 103274
3 104300
4
5 103274
6
7 103274
8 103274 200
9 103275 200
10 103276 114 111 123
11 103314 055 055 055
12 103332 124 061 011
13 103364 124 062 011
14 103416 124 063 011
15 103447 124 064 011
16 103473 124 065 011
17 103520 124 065 011
18 103560 124 067 011
19 103610 124 061 060
20 103642 124 061 061
21 103667 124 061 062
22 103722 124 061 063
23 103753 124 061 064
24 104004 124 061 065
25 104036 124 061 066
26 104063 124 061 067
27 104100 124 062 060
28 104120 124 062 061
29 104140 124 062 062
30 104163 124 062 063
31 104206 124 062 064
32 104241 124 062 065
33 104274 124 062 066
34 104332 124 062 067
35 104362 124 063 060
36 104415 124 063 061
37 104447 124 063 062
38 104500 124 063 063
39 104537 124 063 064
40 104571 124 063 065
41 104617 124 063 066
42 104634 124 063 067
43 104662 124 064 060
44 104714 124 064 061
45 104744 124 064 062
46 104773 124 064 063
47 105024 124 064 064
48 105055 124 064 065
49 105121 124 064 066
50 105153 124 064 067
51 105205 124 065 060
52 105235 124 065 061
53 105264 124 065 062
54 105316 124 065 063
55 105340 124 065 064
56 105364 124 065 065
57 105407 124 065 066

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

.=BUFFER

HELP:

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

```

58 105433      124      065      067  .ASCII @T57  SEARCH ON CYLINDER@<CRLF>
59 105462      124      066      060  .ASCII @T60  SEARCH OFF CYLINDER@<CRLF>
60 105512      124      066      061  .ASCII @T61  SEARCH INVALID SECTOR@<CRLF>
61 105544      124      066      062  .ASCII @T62  SEARCH INVALID TRACK@<CRLF>
62 105575      124      066      063  .ASCII @T63  SEARCH INVALID CYLINDER@<CRLF>
63 105631      124      066      064  .ASCII @T64  IVC SEARCH TEST@<CRLF>
64 105655      124      066      065  .ASCII @T65  ABORT SEARCH TEST@<CRLF>
65 105703      124      066      066  .ASCII @T66  SEARCH AT OFFSET@<CRLF>
66 105730      124      066      067  .ASCII @T67  HEAD ALIGNMENT SEEK@<CRLF>
67 105760      200
68 105761      117      120      105  .ASCII @OPERATIONAL SWITCH SETTINGS@<CRLF>
69 106015      055      055      055  .ASCII @-----@<CRLF>
70 106051      123      127      111  .ASCII @SWITCH USE@<CRLF>
71 106066      055      055      055  .ASCII @-----@<CRLF>
72 106123      040      040      061  .ASCII @ 15  HALT ON ERROR@<CRLF>
73 106147      040      040      061  .ASCII @ 14  LOOP ON TEST@<CRLF>
74 106172      040      040      061  .ASCII @ 13  INHIBIT ERROR TYPEOUTS@<CRLF>
75 106227      040      040      061  .ASCII @ 12  @<CRLF>
76 106236      040      040      061  .ASCII @ 11  INHIBIT ITERATIONS@<CRLF>
77 106267      040      040      061  .ASCII @ 10  BELL ON ERROR@<CRLF>
78 106313      040      040      040  .ASCII @ 9  LOOP ON ERROR@<CRLF>
79 106337      040      040      040  .ASCII @ 8  LOOP ON TEST IN SWR<7:0>@<CRLF>
80 106376      040      040      040  .ASCII @ 7  TN128@<CRLF>
81 106412      040      040      040  .ASCII @ 6  TN64@<CRLF>
82 106425      040      040      040  .ASCII @ 5  TN32@<CRLF>
83 106440      040      040      040  .ASCII @ 4  TN16@<CRLF>
84 106453      040      040      040  .ASCII @ 3  TN8@<CRLF>
85 106465      040      040      040  .ASCII @ 2  TN4@<CRLF>
86 106477      040      040      040  .ASCII @ 1  TN2@<CRLF>
87 106511      040      040      040  .ASCII @ 0  TN1@<CRLF>
88
89              000200      .END      200
  
```


ABASE = 176700	ALUNIT = 000000	CH = 002000	EDT114 073500	EMS103 076511
ACDW1 = 000000	AUSWR = 000000	CHGADR 001326	EDT2 073466	EMS104 076520
ACDW2 = 000000	AVECT1= 120254	CHRCNT 064007	EDT223 073502	EMS105 076547
ACKSTS 053262	AVECT2= 000000	CKSWR = 104410	EDT336 073512	EMS106 076576
ACPUOP= 000000	A16 = 000400	CLKADR 001510	EDT337 073522	EMS11 074232
ADDW0 = 000000	A17 = 001000	CLKVCT 001512	EDT344 073532	EMS110 076615
ADDW1 = 000000	BACK = 000001	CLR = 000040	EDT353 073544	EMS111 076644
ADDW10= 000000	BADTMO 005332	CLRSTS 052402	ED1 103070	EMS112 076655
ADDW11= 000000	BAI = 000010	CMNSTA 007466	ED110 103076	EMS113 076741
ADDW12= 000000	BB00 = 000001	CMPEER 050400	ED111 103102	EMS114 076771
ADDW13= 000000	BB01 = 000002	CNSL01 066375	ED114 103110	EMS115 077015
ADDW14= 000000	BB02 = 000004	CNSL02 066405	ED223 103120	EMS116 077020
ADDW15= 000000	BB03 = 000010	CNSL03 066447	ED336 103130	EMS117 077050
ADDW2 = 000000	BB04 = 000020	CNSL04 066456	ED337 103142	EMS12 074243
ADDW3 = 000000	BB05 = 000040	CNSL07 066512	ED353 103154	EMS120 077072
ADDW4 = 000000	BB06 = 000100	CNSL08 066654	EECC = 000020	EMS121 077113
ADDW5 = 000000	BB07 = 000200	CNSL09 066655	EFT1 073546	EMS122 077156
ADDW6 = 000000	BB08 = 000400	CNTCLR 052264	EFT110 073564	EMS123 077213
ADDW7 = 000000	BB09 = 001000	COMMA 066341	EFT111 073566	EMS124 077256
ADDW8 = 000000	BIT0 = 000001	CONT = 000100	EFT114 073570	EMS125 077310
ADDW9 = 000000	BIT00 = 000001	CPSAVE 063130	EFT2 073556	EMS126 077353
ADEVCT= 000000	BIT01 = 000002	CR = 000015	EFT223 073572	EMS127 077416
ADEVN = 000000	BIT02 = 000004	CRLF = 000200	EFT336 073602	EMS13 074304
ADR = 000001	BIT03 = 000010	CYLMSK= 001777	EFT337 073612	EMS130 077455
AENV = 000000	BIT04 = 000020	DBCK = 100000	EFT344 073622	EMS131 077513
AENVN = 000000	BIT05 = 000040	DBEN = 040000	EFT353 073632	EMS132 077553
AFATAL= 000000	BIT06 = 000100	DBL = 002000	EF110 103246	EMS133 077577
ALL 066330	BIT07 = 000200	DCK = 100000	EF111 103247	EMS134 077622
AMADR1= 000000	BIT08 = 000400	DDISP = 177570	EF114 103251	EMS135 077664
AMADR2= 000000	BIT09 = 001000	DEBL = 020000	EF336 103254	EMS136 077727
AMADR3= 000000	BIT1 = 000002	DEVSEL 050612	EF337 103260	EMS137 077754
AMADR4= 000000	BIT10 = 002000	DISPLA 001156	EHT1 073532	EMS14 074315
AMAMS1= 000000	BIT11 = 004000	DISPRE 000174	EHT110 073354	EMS140 100011
AMAMS2= 000000	BIT12 = 010000	DLT = 100000	EHT111 073360	EMS141 100032
AMAMS3= 000000	BIT13 = 020000	DMD = 000001	EHT114 073364	EMS142 100060
AMAMS4= 000000	BIT14 = 040000	DPE = 000010	EHT2 073344	EMS143 100075
AMSGAD= 000000	BIT15 = 100000	DPEHI = 040000	EHT223 073370	EMS144 100123
AMSGLG= 000000	BIT2 = 000004	DPELO = 020000	EHT256 073402	EMS145 100133
AMSGTY= 000000	BIT3 = 000010	DPR = 000400	EHT336 073414	EMS146 100162
AMTYP1= 000000	BIT4 = 000020	DRIVES 067070	EHT337 073426	EMS147 100201
AMTYP2= 000000	BIT5 = 000040	DRQ = 004000	EHT344 073440	EMS15 074326
AMTYP3= 000000	BIT6 = 000100	DRVCLR= 000010	EHT353 073452	EMS150 100216
AMTYP4= 000000	BIT7 = 000200	DRVSTS 055620	EH1 102031	EMS151 100235
AOE = 001000	BIT8 = 000400	DRY = 000200	EH110 102050	EMS152 100261
APASS = 000000	BIT9 = 001000	DSWR = 177570	EH111 102057	EMS153 100305
APE = 100000	BLNKS1 067133	DTE = 010000	EH114 102076	EMS154 100327
APRIOR= 000000	BLNKS2 067132	DTO = 010000	EH223 102125	EMS155 100343
APTCSU= 000040	BLNKS3 067131	DULPRT= 024024	EH256 102153	EMS156 100363
APTENV= 000001	BLNKS4 067130	DVA = 004000	EH336 102227	EMS157 100401
APTSIZ= 000200	BOTADR 064004	DVC = 000200	EH337 102266	EMS16 074350
APTSPO= 000100	BOTFLG 064006	EBL = 020000	EH344 102423	EMS160 100416
ARGS = 000002	BPTVEC= 000014	ECH = 000100	EH353 102561	EMS161 100433
ASWREG= 000000	BSE = 100000	ECL = 004000	EMS1 073634	EMS162 100463
ATA = 100000	BUFFER 103274	ECRC = 001000	EMS10 074170	EMS163 100463
ATESTN= 000000	BUFONE 103274	EDT1 073456	EMS100 076406	EMS164 100507
ATNMSK= 000377	BUFTWO 104300	EDT110 073474	EMS101 076434	EMS165 100570
ATNTBL 067236	CC = 004000	EDT111 073476	EMS102 076462	EMS166 100604

EMS167	100631	EMS47	075437	EMT13	067334	EMT211	071534	EMT274	072334
EMS17	074400	EMS5	074026	EMT130	070654	EMT212	071550	EMT275	072352
EMS170	100656	EMS50	075466	EMT131	070666	EMT213	071564	EMT276	072364
EMS171	100705	EMS51	075515	EMT132	070700	EMT214	071574	EMT277	072400
EMS172	100740	EMS52	075543	EMT133	070712	EMT215	071614	EMT3	067260
EMS173	100760	EMS53	075553	EMT134	070724	EMT216	071626	EMT30	067502
EMS174	100775	EMS54	075565	EMT135	070736	EMT217	071636	EMT300	072416
EMS175	101003	EMS55	075607	EMT136	070750	EMT22	067422	EMT301	072436
EMS176	101031	EMS56	075636	EMT137	070762	EMT220	071646	EMT302	072460
EMS177	101060	EMS57	075713	EMT14	067346	EMT221	071656	EMT303	072472
EMS2	073703	EMS6	074056	EMT140	070772	EMT222	071666	EMT304	072504
EMS20	074440	EMS60	075741	EMT141	071002	EMT223	071676	EMT305	072516
EMS200	101102	EMS61	075754	EMT142	071012	EMT224	071706	EMT306	072530
EMS201	101117	EMS62	076003	EMT143	071022	EMT225	071710	EMT307	072540
EMS202	101165	EMS63	076021	EMT144	071032	EMT226	071712	EMT31	067512
EMS203	101215	EMS64	076047	EMT145	071042	EMT227	071714	EMT310	072560
EMS204	101230	EMS65	076065	EMT146	071052	EMT23	067432	EMT311	072572
EMS205	101256	EMS66	076133	EMT147	071062	EMT230	071716	EMT312	072604
EMS206	101270	EMS67	076203	EMT15	067354	EMT231	071720	EMT313	072616
EMS207	101315	EMS7	074123	EMT150	071072	EMT232	071722	EMT314	072636
EMS21	074463	EMS70	076230	EMT151	071102	EMT233	071724	EMT315	072650
EMS210	101342	EMS71	076242	EMT152	071114	EMT234	071726	EMT316	072662
EMS211	101353	EMS72	076255	EMT153	071126	EMT235	071730	EMT317	072674
EMS212	101371	EMS73	076265	EMT154	071144	EMT236	071732	EMT32	067522
EMS213	101414	EMS74	076302	EMT155	071162	EMT237	071734	EMT320	072704
EMS214	101437	EMS75	076321	EMT156	071174	EMT24	067442	EMT321	072714
EMS215	101455	EMS76	076327	EMT157	071206	EMT240	071736	EMT322	072724
EMS216	101476	EMS77	076360	EMT16	067362	EMT241	071740	EMT323	072732
EMS217	101526	EMTVEC=	000030	EMT160	071220	EMT242	071742	EMT324	072742
EMS22	074506	EMT1	067246	EMT161	071230	EMT243	071744	EMT325	072754
EMS220	101561	EMT10	067316	EMT162	071242	EMT244	071746	EMT326	072766
EMS221	101601	EMT100	070314	EMT163	071254	EMT245	071750	EMT327	073004
EMS222	101651	EMT101	070332	EMT164	071264	EMT246	071752	EMT33	067532
EMS223	101702	EMT102	070350	EMT165	071272	EMT247	071762	EMT330	073016
EMS224	101757	EMT103	070360	EMT166	071300	EMT25	067452	EMT331	073026
EMS23	074522	EMT104	070372	EMT167	071306	EMT250	071774	EMT332	073040
EMS24	074550	EMT105	070410	EMT17	067372	EMT251	072010	EMT333	073052
EMS25	074577	EMT106	070420	EMT170	071314	EMT252	072016	EMT334	073070
EMS26	074614	EMT107	070430	EMT171	071326	EMT253	072024	EMT335	073106
EMS27	074635	EMT11	067322	EMT172	071336	EMT254	072042	EMT336	073126
EMS3	073720	EMT110	070450	EMT173	071346	EMT255	072052	EMT337	073136
EMS30	074646	EMT111	070462	EMT174	071356	EMT256	072062	EMT34	067542
EMS31	074656	EMT112	070470	EMT175	071370	EMT257	072072	EMT340	073146
EMS32	074705	EMT113	070476	EMT176	071376	EMT26	067462	EMT341	073160
EMS33	074734	EMT114	070512	EMT177	071404	EMT260	072104	EMT342	073166
EMS34	074762	EMT115	070520	EMT2	067252	EMT261	072116	EMT343	073200
EMS35	075033	EMT116	070530	EMT20	067402	EMT262	072136	EMT344	073212
EMS36	075062	EMT117	070540	EMT200	071406	EMT263	072146	EMT345	073224
EMS37	075111	EMT12	067326	EMT201	071420	EMT264	072156	EMT346	073234
EMS4	073763	EMT120	070550	EMT202	071432	EMT265	072176	EMT347	073250
EMS40	075137	EMT121	070560	EMT203	071444	EMT266	072216	EMT35	067552
EMS41	075166	EMT122	070570	EMT204	071456	EMT267	072230	EMT350	073262
EMS42	075214	EMT123	070600	EMT205	071474	EMT27	067472	EMT351	073300
EMS43	075243	EMT124	070610	EMT206	071504	EMT270	072246	EMT352	073310
EMS44	075272	EMT125	070620	EMT207	071514	EMT271	072262	EMT353	073314
EMS45	075345	EMT126	070630	EMT21	067412	EMT272	072300	EMT354	073324
EMS46	075410	EMT127	070642	EMT210	071526	EMT273	072316	EMT36	067562

EMT37	067572	F2	= 000010	LBT	= 002000	PIP	= 020000	RMDB	= 000022
EMT4	067266	F3	= 000020	LF	= 000012	PIRQ	= 177772	RMDBI	001356
EMT40	067602	F4	= 000040	LODEV	066776	PIRQVE	= 000240	RMDBO	001432
EMT41	067610	GET	042710	LS	= 000004	PLFS	= 002000	RMDC	= 000034
EMT42	067620	GETBUF	001334	LSC	= 004000	PRIERR	043706	RMDCI	001370
EMT43	067632	GETINX	001514	LST	= 000002	PRO	= 000000	RMDCO	001444
EMT44	067642	GETSTS	042624	LSTRK	001332	PR1	= 000040	RMDS	= 000012
EMT45	067652	GO	= 000001	MCLK	= 004000	PR2	= 000100	RMDSI	001346
EMT46	067662	GTSWR	= 104407	MCPE	= 020000	PR3	= 000140	RMDSO	001422
EMT47	067672	HCE	= 000200	MDF	= 000100	PR4	= 000200	RMDT	= 000026
EMT5	067274	HCI	= 002000	MDPE	= 000400	PR5	= 000240	RMDTI	001362
EMT50	067700	HCRC	= 000400	MI	= 000004	PR6	= 000300	RMDTO	001436
EMT51	067710	HELP	103274	MOC	= 000400	PR7	= 000340	RMEC1	= 000044
EMT52	067722	HT	= 000011	MOH	= 020000	PS	= 177776	RMEC1I	001400
EMT53	067740	IAE	= 002000	MOL	= 010000	PSEL	= 002000	RMEC10	001454
EMT54	067756	IBSAVE	063132	MRD	= 002000	PSW	= 177776	RMEC2	= 000046
EMT55	067766	IDXMSK	= 000077	MS	= 000040	PUT	043160	RMEC2I	001402
EMT56	070000	IE	= 000100	MSC	= 000002	PUTBUF	001410	RMEC20	001456
EMT57	070016	ILF	= 000001	MSDRVS	066676	PUTINX	001543	RMER1	= 000014
EMT6	067302	ILF02	= 000002	MSE	= 100000	PWRVEC	= 000024	RMER1I	001350
EMT60	070026	ILF24	= 000024	MSER	= 000200	QUES	066335	RMER10	001424
EMT61	070044	ILF26	= 000026	MSGDRV	066712	RCLSTS	054056	RMER2	= 000042
EMT62	070064	ILF30	= 000030	MSHELP	066344	RD	= 000070	RMER2I	001376
EMT63	070100	ILF32	= 000032	MUR	= 001000	RDCHR	= 104411	RMER20	001452
EMT64	070102	ILF34	= 000034	MWD	= 000010	RDLIN	= 104412	RMHR	= 000036
EMT65	070122	ILF36	= 000036	MWP	= 000010	RDOCT	= 104413	RMHRI	001372
EMT66	070142	ILF40	= 000040	MXF	= 001000	RDY	= 000200	RMHRO	001446
EMT67	070154	ILF42	= 000042	N	067124	READY	007660	RMLA	= 000020
EMT7	067310	ILF44	= 000044	NDTMSK	= 115760	RECAL	= 000006	RMLAI	001354
EMT70	070166	ILF46	= 000046	NED	= 010000	RESREG	= 104415	RMLAO	001430
EMT71	070176	ILF54	= 000054	NEM	= 004000	RESVEC	= 000010	RMMR1	= 000024
EMT72	070206	ILF56	= 000056	NONE	067117	REX	= 010000	RMMR1I	001360
EMT73	070224	ILF64	= 000064	NOP	= 000000	RG	= 040000	RMMR10	001434
EMT74	070242	ILF66	= 000066	NOTAVL	067030	RH	= 000072	RMMR2	= 000040
EMT75	070252	ILF74	= 000074	NOTPRS	067013	RIP	= 000020	RMMR2I	001374
EMT76	070272	ILF76	= 000076	NOTRM	066755	RELEASE	= 000012	RMMR20	001450
EMT77	070302	ILR	= 000002	NSA	= 100000	RMAS	= 000016	RMOF	= 000032
QUALS	066326	ILRG50	= 000050	OCC	= 100000	RMASI	001352	RMOFI	001366
ERR	= 040000	ILRG52	= 000052	OFD	= 000200	RMASO	001426	RMOFO	001442
ERRNMB	064002	ILRG54	= 000054	OFFSET	= 000014	RMB	= 000004	RMR	= 000004
ERROR	= 104000	ILRG56	= 000056	OM	= 000001	RMBAE	= 000050	RMSN	= 000030
ERRTYP	063134	ILRG60	= 000060	OPE	= 020000	RMBAEI	001404	RMSNI	001364
ERRVEC	= 000004	ILRG62	= 000062	OPI	= 020000	RMBAEO	001460	RMSNO	001440
ERTY00	064010	ILRG64	= 000064	OR	= 000200	RMBAI	001340	RMWC	= 000002
ERTY01	064015	ILRG66	= 000066	PACACK	= 000022	RMBAO	001414	RMWCI	001336
ERTY02	064025	ILRG70	= 000070	PAKACK	= 000022	RMCS1	= 000000	RMWCO	001412
ERTY03	064034	ILRG72	= 000072	PAR	= 000010	RMCS1I	001334	RQA	= 100000
ERTY04	064042	ILRG74	= 000074	PAT	= 000020	RMCS10	001410	RQB	= 040000
ERTY05	064045	ILRG76	= 000076	PDA	= 000400	RMCS2	= 000010	RTC	= 000016
ESRC	= 004000	IOTVEC	= 000020	PFECH	064052	RMCS2I	001344	R6	= 000006
FER	= 000020	IPCK0	= 000001	PFECH1	064062	RMCS20	001420	R7	= 000007
FIND	= 000001	IPCK1	= 000002	PFECH2	064150	RMCS3	= 000052	SADMSK	= 000377
FMT16	= 010000	IPCK2	= 000004	PFECH3	064166	RMCS3I	001406	SAVREG	= 104414
FNCDTB	067136	IPCK3	= 000010	PFECH4	064174	RMCS30	001462	SA1	= 000001
FNCMSK	= 000077	IR	= 000100	PGE	= 002000	RMDA	= 000006	SA16	= 000020
FO	= 000002	IVC	= 010000	PGM	= 001000	RMDAI	001342	SA2	= 000002
F1	= 000004	LBC	= 002000	PHA	= 000200	RMDAO	001416	SA4	= 000004

SAB = 000010	SW4 = 000020	TST42 = 027306	\$ATY1 = 066060	\$GET42 = 041620
SC = 100000	SW5 = 000040	TST43 = 027574	\$ATY3 = 066066	\$GTSWR = 064600
SCHSTS = 056422	SW6 = 000100	TST44 = 030106	\$ATY4 = 066076	\$GT42P = 041614
SCOPE = 000004	SW7 = 000200	TST45 = 030406	\$AUTOB = 001150	\$HD = 000000
SCTMSK = 003700	SW8 = 000400	TST46 = 030666	\$BASE = 001276	\$HIBTS = 001100
SCO = 000100	SW9 = 001000	TST47 = 031170	\$BDADR = 001136	\$HIOCT = 065572
SC1 = 000200	SYSTAT = 066720	TST5 = 011660	\$BDDAT = 001142	\$ICNT = 001120
SC2 = 000400	TADMASK = 177400	TST50 = 031474	\$BELL = 001212	\$ILLUP = 066042
SC3 = 001000	TAG = 020000	TST51 = 032112	\$BIN = 060640	\$INTAG = 001151
SC4 = 002000	TAGADR = 001114	TST52 = 032540	\$CDW1 = 001302	\$ITEMB = 001130
SEARCH = 000030	TAP = 040000	TST53 = 033156	\$CDW2 = 001304	\$LF = 001220
SECERR = 044540	TA1 = 000400	TST54 = 033516	\$CHARC = 061644	\$LFLG = 066323
SEEK = 000004	TA16 = 010000	TST55 = 034060	\$CKSWR = 064510	\$LPADR = 001122
SEKSTS = 051024	TA2 = 001000	TST56 = 034404	\$CMTAG = 001114	\$LPERR = 001124
SHUT = 065450	TA4 = 002000	TST57 = 035032	\$CM3 = 000000	\$MADR1 = 001254
SIZCLK = 043400	TA8 = 004000	TST6 = 012174	\$CM4 = 000005	\$MADR2 = 001260
SKI = 040000	TBITVE = 000014	TST60 = 035510	\$CNTLC = 065406	\$MADR3 = 001264
SNGPRT = 020024	TIMOUT = 043522	TST61 = 036214	\$CNTLG = 065420	\$MADR4 = 001270
STACK = 001100	TKVEC = 000060	TST62 = 036614	\$CNTLU = 065413	\$MAIL = 001222
STANDA = 007010	TFVEC = 000064	TST63 = 037224	\$CPUOP = 001250	\$MAMS1 = 001252
START = 005422	TRAPVE = 000034	TST64 = 037624	\$CRLF = 001217	\$MAMS2 = 001256
START1 = 005412	TRE = 040000	TST65 = 040204	\$DBLK = 061056	\$MAMS3 = 001262
START2 = 005426	TRTVEC = 000014	TST66 = 040550	\$DDW0 = 001306	\$MAMS4 = 001266
STCDRV = 057766	TST = 010000	TST67 = 041106	\$DDW1 = 001310	\$MBADR = 001102
STKLMT = 177774	TSTNMB = 064000	TST7 = 012354	\$DDW2 = 001312	\$MFLG = 066322
STOP = 062512	TSTPRP = 041650	TYPBN = 104406	\$DDW3 = 001314	\$MNEW = 065436
STSD1 = 103164	TSTQUE = 001464	TYPDS = 104405	\$DDW4 = 001316	\$MSGAD = 001236
STSD2 = 103202	TST1 = 010046	TYPE = 104401	\$DDW5 = 001320	\$MSGLG = 001240
STSD3 = 103222	TST10 = 013426	TYPOC = 104402	\$DDW6 = 001322	\$MSGTY = 001222
STSD4 = 103234	TST11 = 013726	TYPON = 104404	\$DDW7 = 001324	\$MSWR = 065425
STSF = 103265	TST12 = 014144	TYPOS = 104403	\$DEVCT = 001232	\$MTYP1 = 001253
STSH1 = 102627	TST13 = 014452	UNS = 040000	\$DEVM = 001300	\$MTYP2 = 001257
STSH2 = 102704	TST14 = 014762	UNTMSK = 000007	\$DOAGN = 041640	\$MTYP3 = 001263
STSH3 = 102773	TST15 = 015224	UNTOFF = 067047	\$DTBL = 061046	\$MTYP4 = 001267
STSH4 = 103031	TST16 = 015502	UNTON = 067060	\$ENDAD = 041630	\$MXCNT = 062332
SWR = 001154	TST17 = 015752	UPE = 020000	\$ENDCT = 041666	\$NULL = 001170
SWREG = 000176	TST2 = 010246	USE = 040000	\$ENULL = 041644	\$NWTST = 000001
SW0 = 000001	TST20 = 016202	U0 = 000001	\$ENV = 001242	\$OCNT = 061310
SW00 = 000001	TST21 = 016526	U1 = 000002	\$ENVM = 001243	\$OMODE = 061312
SW01 = 000002	TST22 = 017146	U2 = 000004	\$EOP = 041432	\$OVER = 062316
SW02 = 000004	TST23 = 017512	VV = 000100	\$EOPCT = 041460	\$PASS = 001230
SW03 = 000010	TST24 = 020104	WC = 000040	\$EOSP = 041374	\$PASTM = 001106
SW04 = 000020	TST25 = 020574	WCD = 000050	\$ERFLG = 001117	\$POWER = 066050
SW05 = 000040	TST26 = 021142	WCE = 040000	\$ERMAX = 001131	\$PWRDN = 065702
SW06 = 000100	TST27 = 021464	WCEHI = 010000	\$ERROR = 062540	\$PWRMG = 066036
SW07 = 000200	TST3 = 011116	WCELO = 004000	\$ERRPC = 001132	\$PWRUP = 065754
SW08 = 000400	TST30 = 022046	WCF = 000040	\$ERRTB = 001572	\$QUES = 001216
SW09 = 001000	TST31 = 022364	WCH = 000052	\$ERTTL = 001126	\$RDCHR = 065052
SW1 = 000002	TST32 = 023150	WD = 000060	\$ESCAP = 001210	\$RDLIN = 065142
SW10 = 002000	TST33 = 023674	WH = 000062	\$ETABL = 001242	\$RDOCT = 065472
SW11 = 004000	TST34 = 024464	WLE = 004000	\$ETEND = 001326	\$RDSZ = 000010
SW12 = 010000	TST35 = 025250	WRL = 004000	\$FATAL = 001224	\$RESRE = 060530
SW13 = 020000	TST36 = 025602	XSIZ = 006454	\$FFLG = 066324	\$RM02 = 066736
SW14 = 040000	TST37 = 026242	XXDP = 001330	\$FILLC = 001172	\$RM03 = 066743
SW15 = 100000	TST4 = 011454	Y = 067126	\$FILLS = 001171	\$RM05 = 066750
SW2 = 000004	TST40 = 026462	\$APTHD = 001100	\$GDADR = 001134	\$RTNAD = 041642
SW3 = 000010	TST41 = 027020	\$ATYC = 066104	\$GDDAT = 001140	\$SAVRE = 060472

\$SAVR6 066046	\$TKB 001162	\$TMP3 001202	\$STNM 001116	\$UNITM 001110
\$SCOPE 061650	\$TKCNT 064200	\$TMP4 001204	\$TTYIN 065376	\$USWR 001246
\$SETUP= 000137	\$TKINT 064210	\$TN = 000070	\$TYPBN 060566	\$VECT1 001272
\$STUP = 177777	\$TKQEN= 064207	\$TPB 001166	\$TYPDS 060642	\$VECT2 001274
\$SVLAD 062262	\$TKQIN 064202	\$TPFLG 001173	\$TYPE 061314	\$XOFF = 000023
\$SVPC = 000210	\$TKQJ 064204	\$TPS 001164	\$TYPEC 061526	\$XON = 000021
\$SWR = 167400	\$TKQJR 064206	\$TRAP 065574	\$TYPEX 061646	\$XTSTR 061672
\$SWREG 001244	\$TKS 001160	\$TRAP2 065634	\$TYPOC 061112	\$\$GET4= 000000
\$SWRMK= 000000	\$TKSRV 064260	\$TRP = 000016	\$TYPON 061126	\$\$\$W08= 000070
\$SWOBT 062334	\$TMP0 001174	\$TRPAD 065646	\$TYPOS 061066	\$OFILL 061311
\$TESTN 001226	\$TMP1 001176	\$TSTM 001104	\$UNIT 001234	.\$X = 001100
\$TIMES 001206	\$TMP2 001200			

. ABS. 106524 000
000000 001

ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 61952 WORDS (242 PAGES)
DYNAMIC MEMORY AVAILABLE FOR 70 PAGES
CZRMMB.BIC,CZRMMB/C=CZRMMB.DOC,CZRMMB,SYSMAC/M

\$CM3	6-0	6-0#												
\$CM4	6-0	6-0	6-0	6-0	6-0	6-0	6-0	6-0	6-0	6-0	6-0#	6-0#	6-0#	6-0#
\$CMTAG	6-0#	6-0#												
\$CNTLC	42-1	42-1	10-23	10-23	10-23	10-23	10-23	10-23	10-23					
\$CNTLG	42-1	42-1#												
\$CNTLU	42-1	42-1	42-1#											
\$CPUOP	6-0#													
\$CRLF	6-0#	10-48	10-80	10-140	11-20	11-28	11-62	11-100	12-25	12-66	14-20	38-1	38-1	38-1
	40-1	40-1	40-1	41-20	41-61	41-104	41-114	41-126	41-134	41-154	42-1	42-1	42-1	42-1
\$DBLK	36-1	36-1	36-1#											
\$DDW0	6-0#													
\$DDW1	6-0#													
\$DDW2	6-0#													
\$DDW3	6-0#													
\$DDW4	6-0#													
\$DDW5	6-0#													
\$DDW6	6-0#													
\$DDW7	6-0#													
\$DEVCT	6-0#													
\$DEVVM	6-0#	10-67*	10-78	10-117*	10-121*	11-63*	11-70*	11-97*	11-98	12-4	12-38	13-28*	42-6*	
\$DOAGN	14-20	14-20	14-20#											
\$DTBL	36-1	36-1#												
\$ENDAD	5-5	10-28	14-20#	40-1										
\$ENDCT	10-23	14-20#												
\$ENULL	14-20#													
\$ENV	6-0#	10-28	38-1	40-1	46-1	46-1								
\$ENVM	6-0#	10-23	10-71	38-1	38-1	46-1								
\$EOP	13-29	14-20#	42-7											
\$EOPCT	10-23*	14-20	14-20#											
\$EOSP	13-239	14-9#												
\$ERFLG	6-0#	39-1	39-1	39-1	39-1	39-1	39-1*	40-1	40-1	40-1*				
\$ERMAX	6-0#	10-23*	39-1	39-1	39-1	39-1*								
\$ERROR	10-23	40-1#												
\$ERRPC	6-0#	40-1	40-1	40-1	40-1*	40-1*	41-54							
\$ERRTB	8-0#	41-72												
\$ERTTL	6-0#	14-20	14-20	14-20*	40-1	40-1	40-1*							
\$ESCAP	6-0#	10-23*	39-1*	40-1	40-1	40-1	40-1							
\$ETABL	6-0#													
\$ETEND	5-8	6-0#												
\$FATAL	6-0#	46-1*												
\$FFLG	46-1	46-1#	46-1*	46-1*	46-1*									
\$FILLC	6-0#	38-1	38-1	38-1										
\$FILLS	6-0#	38-1	38-1											
\$GDADR	6-0#	56-8												
\$GDDAT	6-0#	13-70*	13-71	13-75*	13-84*	13-85	13-89*	13-107*	13-108	13-112*	13-119*	13-120	13-124*	13-131*
	13-132	13-192*	13-214*	13-215*	13-216*	13-224*	13-230*	13-237*	13-243*	13-244*	13-251*	13-283*	13-291*	13-297*
	13-303*	13-309*	13-315*	13-323*	13-331*	13-337*	13-343*	13-349*	13-355*	13-361*	13-370*	13-573*	13-579*	13-599*
	13-615*	13-637*	13-654*	13-672*	13-682*	13-824*	13-832*	13-866*	13-871*	13-876*	13-900*	13-911*	13-956*	13-957*
	13-960	13-964*	13-967*	13-968*	13-971	13-:23*	13-:24*	13-:27	13-:31*	13-:34*	13-:35*	13-:38	13-:89*	13-:90*
	13-:93	13-:97*	13-:00*	13-:01*	13-:04	13-:55*	13-:56*	13-:59	13-:63*	13-:66*	13-:67*	13-:70	13-<49*	13-<78*
	13-:08*	13-:11	13-A48*	13-A80*	13-B91*	13-E86*	13-F17*	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*	26-36*	26-47*	26-63*	26-68	26-87*	26-101*	26-102*
	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*	56-1
	56-6	56-8	56-10	56-11										
\$GET42	12-34	12-43	14-20#											
\$GT42P	14-20	14-20#												
\$GTSWR	42-1#	44-1	44-1											
\$HD	4-471	4-471	4-471											
\$HIBTS	5-8#													
\$HIOCT	43-1#	43-1*												
\$ICNT	6-0#	39-1	39-1	39-1	39-1*	39-1*								
\$ILLUP	45-1	45-1	45-1#											
\$INTAG	6-0#	42-1	42-1	42-1	42-1	42-1*								
\$ITEMB	6-0#	40-1	40-1	40-1	40-1	40-1	40-1*	40-1*	41-49					
\$LF	6-0#	38-1	38-1	40-1	40-1	42-1	42-1	42-1						
\$LFLG	46-1#	46-1*												
\$LPADR	6-0#	10-23*	39-1	39-1	39-1	39-1*	39-1*	39-1*						
\$LPERR	6-0#	10-23*	39-1	39-1	39-1	39-1*	40-1							
\$MADR1	6-0#													
\$MADR2	6-0#													
\$MADR3	6-0#													
\$MADR4	6-0#													
\$MAIL	5-8	5-8	6-0#	10-23	10-28	13-1	13-32	13-137	13-179	13-204	13-255	13-273	13-375	13-400
	13-417	13-441	13-463	13-488	13-510	13-540	13-558	13-589	13-625	13-660	13-692	13-757	13-811	13-842
	13-886	13-915	13-981	13-:48	13-:14	13-:80	13-<22	13-<54	13-<87	13-=56	13-=84	13->12	13->46	13->80
	13-?08	13-?42	13-?76	13-@25	13-@76	13-A24	13-A58	13-A95	13-B31	13-C08	13-C59	13-D19	13-D64	13-E12
	13-E58	13-E96	13-F35	13-F79	38-1	39-1	40-1							
\$MAMS1	6-0#													
\$MAMS2	6-0#													
\$MAMS3	6-0#													
\$MAMS4	6-0#													
\$MBADR	5-8#													
\$MFLG	46-1	46-1#	46-1*	46-1*										
\$MNEW	42-1	42-1#												
\$MSGAD	6-0#	46-1	46-1*											
\$MSGLG	6-0#	46-1*												
\$MSGTY	6-0#	46-1	46-1	46-1*	46-1*									
\$MSWR	42-1	42-1#												
\$MTYP1	6-0#													
\$MTYP2	6-0#													
\$MTYP3	6-0#													
\$MTYP4	6-0#													
\$MXCNT	39-1	39-1	39-1	39-1#										
\$NULL	6-0#	38-1	38-1	38-1										
\$NWTST	13-1	13-1#	13-1#	13-32	13-32#	13-32#	13-137	13-137#	13-137#	13-179	13-179#	13-179#	13-204	13-204#
	13-204#	13-255	13-255#	13-255#	13-273	13-273#	13-273#	13-375	13-375#	13-375#	13-400	13-400#	13-400#	13-417
	13-417#	13-417#	13-441	13-441#	13-441#	13-463	13-463#	13-463#	13-488	13-488#	13-488#	13-510	13-510#	13-510#
	13-540	13-540#	13-540#	13-558	13-558#	13-558#	13-589	13-589#	13-589#	13-625	13-625#	13-625#	13-660	13-660#
	13-660#	13-692	13-692#	13-692#	13-757	13-757#	13-757#	13-811	13-811#	13-811#	13-811#	13-842	13-842#	13-842#
	13-886#	13-886#	13-915	13-915#	13-915#	13-981	13-981#	13-981#	13-:48	13-:48#	13-:48#	13-:14	13-:14#	13-:14#
	13-:80	13-:80#	13-:80#	13-<22	13-<22#	13-<22#	13-<54	13-<54#	13-<87	13-<87#	13-<87#	13-<87#	13->46	13->80
	13-=56#	13-=84	13-=84#	13-=84#	13->12	13->12#	13->12#	13->46	13->46#	13->46#	13->80	13->80#	13->80#	13-?08
	13-?08#	13-?08#	13-?42	13-?42#	13-?42#	13-?76	13-?76#	13-?76#	13-@25	13-@25#	13-@25#	13-@76	13-@76#	13-@76#

AMTYP3	6-0	6-0												
AMTYP4	6-0	6-0												
AOE	4-563#	4-574	22-311	22-326	22-341	22-356	22-360	23-117	23-121	48-77	48-78	48-81	48-82	48-85
APASS	48-86													
APE	6-0	6-0												
APRIOR	4-741#													
APTCSU	6-0													
APTENV	38-1	46-1#												
APTSIZ	38-1	40-1	46-1	46-1#										
APTSPO	10-23	46-1#												
ARGS	38-1	46-1	46-1#											
	13-41#	13-59#	13-65#	13-146#	13-163#	13-170#	13-172#	13-176#	13-189#	13-210#	13-264#	13-267#	13-270#	13-280#
	13-320#	13-366#	13-387#	13-390#	13-393#	13-397#	13-402	13-402#	13-408#	13-410#	13-412#	13-414#	13-419	13-419#
	13-426#	13-432#	13-434#	13-436#	13-438#	13-443	13-443#	13-453#	13-455#	13-460#	13-465	13-465#	13-475#	13-479#
	13-481#	13-483#	13-485#	13-490	13-490#	13-497#	13-499#	13-501#	13-503#	13-505#	13-507#	13-512	13-512#	13-525#
	13-530#	13-533#	13-535#	13-537#	13-542	13-542#	13-547#	13-549#	13-551	13-551#	13-553#	13-555#	13-560	13-560#
	13-566#	13-568#	13-582	13-582#	13-584#	13-586#	13-591	13-591#	13-596#	13-604	13-604#	13-608#	13-610#	13-618
	13-618#	13-620#	13-622#	13-627	13-627#	13-632#	13-645#	13-647#	13-649#	13-657	13-657#	13-662	13-662#	13-667#
	13-679#	13-687#	13-689#	13-694	13-694#	13-718#	13-733#	13-735#	13-743#	13-745#	13-747	13-747#	13-749#	13-751#
	13-759	13-759#	13-804#	13-813	13-813#	13-819#	13-821#	13-835	13-835#	13-837#	13-839#	13-844	13-844#	13-856#
	13-860#	13-862#	13-879	13-879#	13-881#	13-883#	13-888	13-888#	13-893#	13-895#	13-906#	13-917	13-917#	13-950#
	13-953#	13-955#	13-983	13-983#	13-:17#	13-:20#	13-:22#	13-:50	13-:50#	13-:83#	13-:86#	13-:88#	13-:16	13-:16#
	13-:49#	13-:52#	13-:54#	13-:82	13-:82#	13-:85#	13-:88#	13-<07#	13-<10#	13-<12#	13-<14#	13-<24	13-<24#	13-<27#
	13-<35#	13-<44#	13-<56	13-<56#	13-<60#	13-<72#	13-<74#	13-<89	13-<89#	13-=03#	13-=05#	13-=07#	13-=59	13-=59#
	13-=69#	13-=73#	13-=75#	13-=77#	13-=79	13-=79#	13-=81#	13-=87	13-=87#	13-=97#	13->01#	13->03#	13->05#	13->07
	13->07#	13->09#	13->16	13->16#	13->25#	13->30#	13->32#	13->34#	13->36	13->36#	13->38#	13->49	13->49#	13->61#
	13->65#	13->67#	13->69#	13->71	13->71#	13->73#	13->83	13->83#	13->93#	13->97#	13->99#	13-?01#	13-?03	13-?03#
	13-?05#	13-?11	13-?11#	13-?22#	13-?26#	13-?28#	13-?30#	13-?32	13-?32#	13-?34#	13-?45	13-?45#	13-?56#	13-?60#
	13-?62#	13-?64#	13-?66	13-?66#	13-?68#	13-?79	13-?79#	13-?95#	13-?97#	13-?99#	13-a03#	13-a05#	13-a07#	13-a10
	13-a10#	13-a12#	13-a27	13-a27#	13-a44#	13-a46#	13-a48#	13-a52#	13-a54#	13-a56#	13-a59	13-a59#	13-a61#	13-a78
	13-a78#	13-a94#	13-a96#	13-a98#	13-A02#	13-A04#	13-A06#	13-A08	13-A08#	13-A10#	13-A41#	13-A43#	13-A45#	13-A53
	13-A53#	13-A55#	13-A60	13-A60#	13-A73#	13-A75#	13-A84#	13-A86#	13-A88#	13-A90	13-A90#	13-A92#	13-A98	13-A98#
	13-B08#	13-B15#	13-B18#	13-B20#	13-B22#	13-B24#	13-B26	13-B26#	13-B28#	13-B48#	13-B51#	13-B53#	13-C10	13-C10#
	13-C24#	13-C27#	13-C29#	13-C31#	13-C33#	13-C37#	13-C39#	13-C41#	13-C43	13-C43#	13-C45#	13-C61	13-C61#	13-C77#
	13-C81#	13-C83#	13-C85#	13-C89#	13-C93#	13-C95#	13-C97#	13-C99	13-C99#	13-D01#	13-D21	13-D21#	13-D37#	13-D41#
	13-D43#	13-D45#	13-D47#	13-D49	13-D49#	13-D51#	13-D66	13-D66#	13-D83#	13-D87#	13-D89#	13-D91#	13-D93#	13-D95
	13-D95#	13-D97#	13-E14	13-E14#	13-E30#	13-E34#	13-E36#	13-E38#	13-E40#	13-E42	13-E42#	13-E44#	13-E60	13-E60#
	13-E76#	13-E80#	13-E82#	13-E91	13-E91#	13-E93#	13-E98	13-E98#	13-F11#	13-F13#	13-F24#	13-F26#	13-F28#	13-F30
	13-F30#	13-F32#	13-F37	13-F37#	13-F49#	13-F57#	13-F61#	13-F63#	13-F65#	13-F67	13-F67#	13-F69#	13-F82	13-F82#
	13-F92#	13-F96#	13-F98#	13-G00#	13-G02	13-G02#	13-G04#							
ASWREG	6-0	6-0												
ATA	4-543#	13-215	13-575	13-578	13-579	13-597	13-599	13-601	13-611	13-614	13-633	13-636	13-637	13-650
	13-653	13-668	13-671	13-672	13-680	13-682	13-684	13-828	13-831	13-832	13-967	13-968	13-970	13-:34
	13-:35	13-:37	13-:00	13-:01	13-:03	13-:66	13-:67	13-:69	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	48-58	48-59	48-60	48-63	48-64
	48-67	48-68	48-69	48-70	48-71	48-72	48-73	48-74	48-75	48-76	48-79	48-80	48-83	48-84
	48-87	48-88												
ATESTN	6-0	6-0												
ATNMSK	4-580#	13-640	31-55											
ATNTBL	10-78	10-117	10-121	11-97	12-17	49-3#								
AUNIT	6-0	6-0												
AUSWR	6-0	6-0												
AVECT1	4-764#	6-0	6-0											
AVECT2	6-0	6-0												
BACK	13-41	13-41#	13-59	13-59#	13-65	13-65#	13-146	13-146#	13-163	13-163#	13-170	13-170#	13-172	13-172#
	13-172#	13-176	13-176#	13-176#	13-189	13-189#	13-210	13-210#	13-264	13-264#	13-267	13-267#	13-267#	13-270
	13-270#	13-270#	13-280	13-280#	13-320	13-320#	13-366	13-366#	13-387	13-387#	13-390	13-390#	13-390#	13-393

CLKVCT	7-0#	19-7*	19-13*											
CLR	4-728#	10-84	12-56	27-16										
CLRSTS	13-172	13-?97	13-@46	13-@96	15-98	28-24#								
CMNSTA	10-141	12-2#												
CMPEER	13-551	13-582	13-604	13-618	13-657	13-747	13-835	13-879	13-=:79	13->07	13->36	13->71	13-?03	13-?32
	13-?66	13-@10	13-@59	13-A08	13-A53	13-A90	13-B26	13-C43	13-C99	13-D49	13-D95	13-E42	13-E91	13-F30
	13-F67	13-G02	24-21#											
CNSLO1	11-32	47-8#												
CNSLO2	11-40	47-9#												
CNSLO3	11-44	47-10#												
CNSLO4	11-54	47-11#												
CNSLO7	11-61	47-12#												
CNSLO8	11-92	47-14#												
CNSLO9	47-15#													
CNTCLR	13-34	13-139	13-167	13-181	13-206	13-257	13-275	13-377	13-922	13-988	13-:55	13-:21	13-:88	13-<60
	13-<99	13-?93	13-@42	13-@92	13-A26	13-B33	13-D35	13-D81	13-E28	15-76	27-12#			
COMMA	11-86	12-6	12-14	47-6#										
CONT	4-621#													
CPSAVE	39-1	39-1	39-1	39-1*	39-1*	40-1	40-1	40-1	40-1	40-1	40-1#	40-1*	40-1*	41-180
CR	4-475#	11-73	11-84	38-1	38-1	41-84	54-79							
CRLF	4-475#	10-6	10-28	10-28	10-42	10-53	10-59	10-60	12-30	38-1	38-1	47-4	47-7	47-8
	47-9	47-11	47-12	47-13	47-14	47-15	47-16	47-18	47-28	55-7	55-11	55-12	55-15	55-16
	55-18	58-8	58-9	58-10	58-11	58-12	58-13	58-14	58-15	58-16	58-17	58-18	58-19	58-20
	58-21	58-22	58-23	58-24	58-25	58-26	58-27	58-28	58-29	58-30	58-31	58-32	58-33	58-34
	58-35	58-36	58-37	58-38	58-39	58-40	58-41	58-42	58-43	58-44	58-45	58-46	58-47	58-48
	58-49	58-50	58-51	58-52	58-53	58-54	58-55	58-56	58-57	58-58	58-59	58-60	58-61	58-62
	58-63	58-64	58-65	58-66	58-67	58-68	58-69	58-70	58-71	58-72	58-73	58-74	58-75	58-76
	58-77	58-78	58-79	58-80	58-81	58-82	58-83	58-84	58-85	58-86	58-87			
CYLMSK	4-648#													
DBCK	4-595#													
DBEN	4-596#	13-47	13-152	13-<38										
DBL	4-746#													
DCK	4-557#	4-574	22-469	22-489										
DDISP	4-475#	6-0	10-23											
DEBL	4-597#													
DEVSEL	13-=:03	15-62	25-12#											
DISPLA	6-0#	10-23*	10-23*	39-1*	40-1*									
DISPRE	5-1#	10-23												
DLT	4-718#	22-313												
DMD	4-609#	4-627#	13-47	13-152	13-277	13-281	13-283	13-318	13-321	13-323	13-379	13-445	13-927	13-929
	13-993	13-995	13-:60	13-:62	13-:26	13-:28	13-<29	13-<31	13-<38	13-A28	13-E62			
DPE	4-672#	13-551	13-582	13-618	13-747	13-835	13-879	13-=:79	13->07	13->36	13->71	13-?03	13-?32	13-?66
	13-@10	13-@59	13-A08	13-A53	13-A90	13-B26	13-C43	13-C99	13-D49	13-D95	13-E42	13-E91	13-F30	13-F67
	13-G02	21-139	22-255	26-32	30-31	32-52								
DPEHI	4-742#													
DPELO	4-743#													
DPR	4-550#	13-130	13-131	13-244	26-195	28-127	28-:28	31-30						
DRIVES	12-3	47-28#												
DRQ	4-634#													
DRVCLR	4-495#	13-364	13-515	31-18										
DRVSTS	13-535	31-9#												
DRY	4-551#	13-130	13-131	13-244	21-106	22-38	22-41	22-42	26-195	28-127	28-128	31-30		
DSWR	4-475#	6-0	10-23											
DTE	4-560#	4-574												
DTO	4-598#													
DULPRT	4-637#													
DVA	4-481#	10-89	13-70	17-43	18-37	21-67	21-70	25-37	28-33	31-17	31-18			

DVC	4-671#	13-126	13-221	13-311	13-314	13-351	13-354	13-355	26-127	28-117	30-92	30-130	30-133	30-266
	32-125	32-166	32-169	33-89	33-94	33-97								
FBL	4-614#													
ECH	4-566#	4-574	22-452	22-467	22-485	22-491	48-77	48-78	48-85	48-86				
ECI	4-642#	13-854	13-864	22-487										
ECRC	4-618#													
ED1	52-1	56-1#												
ED110	52-4	56-2#												
ED111	52-5	56-3#												
ED114	52-7	56-5#												
ED223	52-8	56-6#												
ED336	52-10	56-8#												
ED337	52-11	52-12	56-10#											
ED353	52-14	56-11#												
EDT1	8-3	8-6	8-9	8-12	8-15	8-18	8-24	8-27	8-30	8-33	8-36	8-39	8-42	8-45
	8-48	8-51	8-54	8-57	8-60	8-63	8-66	8-69	8-72	8-75	8-78	8-81	8-84	8-87
	8-90	8-93	8-96	8-99	8-102	8-105	8-108	8-111	8-114	8-117	8-120	8-123	8-126	8-130
	8-133	8-136	8-139	8-142	8-146	8-150	8-154	8-160	8-163	8-166	8-169	8-172	8-175	8-178
	8-182	8-185	8-188	8-191	8-194	8-197	8-200	8-203	8-206	8-209	8-212	8-215	8-218	8-236
	8-239	8-242	8-245	8-248	8-251	8-254	8-257	8-260	8-263	8-266	8-269	8-272	8-275	8-278
	8-281	8-284	8-287	8-290	8-293	8-296	8-299	8-302	8-305	8-308	8-311	8-314	8-317	8-320
	8-323	8-326	8-329	8-332	8-335	8-338	8-341	8-344	8-347	8-354	8-357	8-360	8-363	8-366
	8-369	8-372	8-375	8-378	8-381	8-384	8-390	8-393	8-396	8-399	8-402	8-405	8-408	8-411
	8-414	8-417	8-420	8-423	8-432	8-435	8-438	8-441	8-444	8-504	8-507	8-510	8-519	8-522
	8-525	8-531	8-534	8-537	8-540	8-543	8-546	8-549	8-552	8-556	8-559	8-563	8-566	8-569
	8-573	8-577	8-581	8-586	8-590	8-594	8-597	8-600	8-603	8-606	8-609	8-612	8-615	8-618
	8-621	8-624	8-627	8-630	8-633	8-636	8-639	8-642	8-645	8-648	8-651	8-654	8-657	8-660
	8-663	8-666	8-669	8-672	8-675	8-678	8-693	8-696	8-702	8-705	8-708	8-711	8-714	8-723
	52-1#													
ED110	8-221	52-4#												
ED111	8-224	8-227	52-5#											
ED114	8-233	52-7#												
EDT2	8-426	8-429	8-513	8-516	52-2#									
EDT223	8-447	8-528	52-8#											
EDT336	8-351	8-681	8-687	8-690	52-10#									
EDT337	8-684	52-11#												
EDT344	8-699	52-12#												
EDT353	8-720	52-14#												
EECC	4-623#													
EF110	53-4	57-1#												
EF111	53-1	53-5	57-2#											
EF114	53-7	53-8	53-14	57-3#										
EF336	53-10	53-11	53-12	57-4#										
EF337	57-5#													
EFT1	8-3	8-6	8-9	8-12	8-15	8-18	8-24	8-27	8-30	8-33	8-36	8-39	8-42	8-45
	8-48	8-51	8-54	8-57	8-60	8-63	8-66	8-69	8-72	8-75	8-78	8-81	8-84	8-87
	8-90	8-93	8-96	8-99	8-102	8-105	8-108	8-111	8-114	8-117	8-120	8-123	8-126	8-130
	8-133	8-136	8-139	8-142	8-146	8-150	8-154	8-160	8-163	8-166	8-169	8-172	8-175	8-178
	8-182	8-185	8-188	8-191	8-194	8-197	8-200	8-203	8-206	8-209	8-212	8-215	8-218	8-236
	8-239	8-242	8-245	8-248	8-251	8-254	8-257	8-260	8-263	8-266	8-269	8-272	8-275	8-278
	8-281	8-284	8-287	8-290	8-293	8-296	8-299	8-302	8-305	8-308	8-311	8-314	8-317	8-320
	8-323	8-326	8-329	8-332	8-335	8-338	8-341	8-344	8-347	8-354	8-357	8-360	8-363	8-366
	8-369	8-372	8-375	8-378	8-381	8-384	8-390	8-393	8-396	8-399	8-402	8-405	8-408	8-411
	8-414	8-417	8-420	8-423	8-432	8-435	8-438	8-441	8-444	8-504	8-507	8-510	8-519	8-522
	8-525	8-531	8-534	8-537	8-540	8-543	8-546	8-549	8-552	8-556	8-559	8-563	8-566	8-569
	8-573	8-577	8-581	8-586	8-590	8-594	8-597	8-600	8-603	8-606	8-609	8-612	8-615	8-618
	8-621	8-624	8-627	8-630	8-633	8-636	8-639	8-642	8-645	8-648	8-651	8-654	8-657	8-660

	8-663	8-666	8-669	8-672	8-675	8-678	8-693	8-696	8-702	8-705	8-708	8-711	8-714	8-723
EFT110	53-1#													
EFT111	8-221	53-4#												
EFT114	8-224	8-227	53-5#											
EFT2	8-233	53-7#												
EFT223	8-426	8-429	8-513	8-516	53-2#									
EFT336	8-447	8-528	53-8#											
EFT337	8-351	8-681	8-687	8-690	53-10#									
EFT344	8-684	53-11#												
EFT353	8-699	53-12#												
EH1	8-720	53-14#												
EH110	51-1	55-1#												
EH111	51-4	55-2#												
EH114	51-5	55-3#												
EH223	51-7	55-5#												
EH256	51-8	55-6#												
EH336	51-9	55-7#												
EH337	51-11	55-10#												
EH344	51-12	55-11#												
EH353	51-13	55-15#												
EHT1	51-15	55-18#												
	8-3	8-6	8-9	8-12	8-15	8-18	8-24	8-27	8-30	8-33	8-36	8-39	8-42	8-45
	8-48	8-51	8-54	8-57	8-60	8-63	8-66	8-69	8-72	8-75	8-78	8-81	8-84	8-87
	8-90	8-93	8-96	8-99	8-102	8-105	8-108	8-111	8-114	8-117	8-120	8-123	8-126	8-130
	8-133	8-136	8-139	8-142	8-146	8-150	8-154	8-160	8-163	8-166	8-169	8-172	8-175	8-178
	8-182	8-185	8-188	8-191	8-194	8-197	8-200	8-203	8-206	8-209	8-212	8-215	8-218	8-236
	8-239	8-242	8-245	8-248	8-251	8-254	8-257	8-260	8-263	8-266	8-269	8-272	8-275	8-278
	8-281	8-284	8-287	8-290	8-293	8-296	8-299	8-302	8-305	8-308	8-311	8-314	8-317	8-320
	8-323	8-326	8-329	8-332	8-335	8-338	8-341	8-344	8-347	8-354	8-357	8-360	8-363	8-366
	8-369	8-372	8-375	8-378	8-381	8-384	8-390	8-393	8-396	8-399	8-402	8-405	8-408	8-411
	8-414	8-417	8-420	8-423	8-432	8-435	8-438	8-441	8-444	8-504	8-507	8-510	8-519	8-522
	8-525	8-531	8-534	8-537	8-540	8-543	8-546	8-549	8-552	8-556	8-559	8-563	8-566	8-569
	8-573	8-577	8-581	8-586	8-590	8-594	8-597	8-600	8-603	8-606	8-609	8-612	8-615	8-618
	8-621	8-624	8-627	8-630	8-633	8-636	8-639	8-642	8-645	8-648	8-651	8-654	8-657	8-660
	8-663	8-666	8-669	8-672	8-675	8-678	8-693	8-696	8-702	8-705	8-708	8-711	8-714	8-723
EHT110	51-1#													
EHT111	8-221	51-4#												
EHT114	8-224	8-227	51-5#											
EHT2	8-233	51-7#												
EHT223	8-426	8-429	8-513	8-516	51-2#									
EHT256	8-528	51-8#												
EHT336	8-351	8-681	8-687	8-690	51-11#									
EHT337	8-684	51-12#												
EHT344	8-699	51-13#												
EHT353	8-720	51-15#												
EMS1	50-3	54-3#												
EMS10	50-7	50-11	54-10#											
EMS100	50-39	50-71	50-176	50-196	54-66#									
EMS101	50-40	50-59	50-174	50-197	54-67#									
EMS102	50-177	50-178	50-181	50-182	50-183	50-184	50-186	50-187	50-188	50-189	50-234	50-235	54-68#	
EMS103	50-56	50-57	50-135	50-141	50-196	50-197	50-198	50-203	50-204	50-205	50-206	50-207	50-208	50-209
	50-210	50-211	50-213	50-214	50-215	50-219	50-220	54-69#						
EMS104	50-136	50-180	50-203	54-70#										
EMS105	50-204	54-71#												
EMS106	50-212	50-232	54-72#											
EMS11	50-12	50-15	50-16	50-17	50-18	50-19	50-20	50-21	50-22	50-23	50-24	50-25	50-26	50-27

EMT113	8-230	50-77#
EMT114	8-233	50-78#
EMT115	8-236	50-79#
EMT116	8-239	50-80#
EMT117	8-242	50-81#
EMT12	8-30	50-12#
EMT120	8-245	50-82#
EMT121	8-248	50-83#
EMT122	8-251	50-84#
EMT123	8-254	50-85#
EMT124	8-257	50-86#
EMT125	8-260	50-87#
EMT126	8-263	50-88#
EMT127	8-266	50-89#
EMT13	8-33	50-13#
EMT130	8-269	50-90#
EMT131	8-272	50-91#
EMT132	8-275	50-92#
EMT133	8-278	50-93#
EMT134	8-281	50-94#
EMT135	8-284	50-95#
EMT136	8-287	50-96#
EMT137	8-290	50-97#
EMT14	8-36	50-14#
EMT140	8-293	50-98#
EMT141	8-296	50-99#
EMT142	8-299	50-100#
EMT143	8-302	50-101#
EMT144	8-305	50-102#
EMT145	8-308	50-103#
EMT146	8-311	50-104#
EMT147	8-314	50-105#
EMT15	8-39	50-15#
EMT150	8-317	50-106#
EMT151	8-320	50-107#
EMT152	8-323	50-108#
EMT153	8-326	50-109#
EMT154	8-329	50-110#
EMT155	8-332	50-111#
EMT156	8-335	50-112#
EMT157	8-338	50-113#
EMT16	8-42	50-16#
EMT160	8-341	50-114#
EMT161	8-344	50-115#
EMT162	8-347	50-116#
EMT163	50-117#	
EMT164	8-354	50-118#
EMT165	8-357	50-119#
EMT166	8-360	50-120#
EMT167	8-363	50-121#
EMT17	8-45	50-17#
EMT170	8-366	50-122#
EMT171	8-369	50-123#
EMT172	8-372	50-124#
EMT173	8-375	50-125#
EMT174	8-378	50-126#
EMT175	8-381	50-127#

EMT176	8-384	50-128#
EMT177	50-129#	
EMT2	8-6	50-4#
EMT20	8-48	50-18#
EMT200	8-390	50-130#
EMT201	8-393	50-131#
EMT202	8-396	50-132#
EMT203	8-399	50-133#
EMT204	8-402	50-134#
EMT205	8-405	50-135#
EMT206	8-408	50-136#
EMT207	8-411	50-137#
EMT21	8-51	50-19#
EMT210	8-414	50-138#
EMT211	8-417	50-139#
EMT212	8-420	50-140#
EMT213	8-423	50-141#
EMT214	8-426	50-142#
EMT215	8-429	50-143#
EMT216	8-432	50-144#
EMT217	8-435	50-145#
EMT22	8-54	50-20#
EMT220	8-438	50-146#
EMT221	8-441	50-147#
EMT222	8-444	50-148#
EMT223	8-447	50-149#
EMT224	50-150#	
EMT225	50-151#	
EMT226	50-152#	
EMT227	50-153#	
EMT23	8-57	50-21#
EMT230	50-154#	
EMT231	50-155#	
EMT232	50-156#	
EMT233	50-157#	
EMT234	50-158#	
EMT235	50-159#	
EMT236	50-160#	
EMT237	50-161#	
EMT24	8-60	50-22#
EMT240	50-162#	
EMT241	50-163#	
EMT242	50-164#	
EMT243	50-165#	
EMT244	50-166#	
EMT245	50-167#	
EMT246	8-504	50-168#
EMT247	8-507	50-169#
EMT25	8-63	50-23#
EMT250	8-510	50-170#
EMT251	8-513	50-171#
EMT252	8-516	50-172#
EMT253	8-519	50-173#
EMT254	8-522	50-174#
EMT255	8-525	50-175#
EMT256	8-528	50-176#
EMT257	8-531	50-177#

EMT26 8-66 50-24#
EMT260 8-534 50-178#
EMT261 8-537 50-179#
EMT262 8-540 50-180#
EMT263 8-543 50-181#
EMT264 8-546 50-182#
EMT265 8-549 50-183#
EMT266 8-552 50-184#
EMT267 8-556 50-185#
EMT27 8-69 50-25#
EMT270 8-559 50-186#
EMT271 8-563 50-187#
EMT272 8-566 50-188#
EMT273 8-569 50-189#
EMT274 8-573 50-190#
EMT275 8-577 50-191#
EMT276 8-581 50-192#
EMT277 8-586 50-193#
EMT3 8-9 50-5#
EMT30 8-72 50-26#
EMT300 8-590 50-194#
EMT301 8-594 50-195#
EMT302 8-597 50-196#
EMT303 8-600 50-197#
EMT304 8-603 50-198#
EMT305 8-606 50-199#
EMT306 8-609 50-200#
EMT307 8-612 50-201#
EMT31 8-75 50-27#
EMT310 8-615 50-202#
EMT311 8-618 50-203#
EMT312 8-621 50-204#
EMT313 8-624 50-205#
EMT314 8-627 50-206#
EMT315 8-630 50-207#
EMT316 8-633 50-208#
EMT317 8-636 50-209#
EMT32 8-78 50-28#
EMT320 8-639 50-210#
EMT321 8-642 50-211#
EMT322 8-645 50-212#
EMT323 8-648 50-213#
EMT324 8-651 50-214#
EMT325 8-654 50-215#
EMT326 8-657 50-216#
EMT327 8-660 50-217#
EMT33 8-81 50-29#
EMT330 8-663 50-218#
EMT331 8-666 50-219#
EMT332 8-669 50-220#
EMT333 8-672 50-221#
EMT334 8-675 50-222#
EMT335 8-678 50-223#
EMT336 8-351 8-681 50-224#
EMT337 8-684 50-225#
EMT34 8-84 50-30#
EMT340 8-687 50-226#

50-224#

ERRVEC	4-475#	10-23	10-23*	10-23*	10-25*	10-26*	13-4	13-4	13-5*	13-6*	13-19*	13-19*	13-23*	13-23*
	13-<93	13-<94	13-<95*	13-<96*	13-=53*	13-=54*	13-C04*	13-C05*	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*	39-1	39-1	39-1*	39-1*	39-1*	39-1*	39-1*	40-1	40-1*	40-1*	
ERTY00	41-21	41-165#												
ERTY01	41-46	41-166#												
ERTY02	41-51	41-167#												
ERTY03	41-53	41-168#												
ERTY04	41-152	41-169#												
ERTY05	41-39	41-170#												
ESRC	4-616#													
FO	4-486#	22-29												
F1	4-485#	22-29												
F2	4-484#	22-29	23-113											
F3	4-483#	22-29												
F4	4-482#	22-29												
FER	4-568#	4-574	22-416											
FIND	13-41	13-41#	13-41#	13-59	13-59#	13-59#	13-65	13-65#	13-65#	13-146	13-146#	13-146#	13-163	13-163#
	13-163#	13-170	13-170#	13-170#	13-172	13-172#	13-172#	13-176	13-176#	13-176#	13-189	13-189#	13-189#	13-210
	13-210#	13-210#	13-264	13-264#	13-264#	13-267	13-267#	13-267#	13-270	13-270#	13-270#	13-280	13-280#	13-280#
	13-320	13-320#	13-320#	13-366	13-366#	13-366#	13-387	13-387#	13-387#	13-390	13-390#	13-390#	13-393	13-393#
	13-393#	13-397	13-397#	13-397#	13-402	13-402#	13-402#	13-408	13-408#	13-408#	13-410	13-410#	13-410#	13-412
	13-412#	13-412#	13-414	13-414#	13-414#	13-419	13-419#	13-419#	13-426	13-426#	13-426#	13-432	13-432#	13-432#
	13-434	13-434#	13-434#	13-436	13-436#	13-436#	13-438	13-438#	13-438#	13-443	13-443#	13-443#	13-453	13-453#
	13-453#	13-455	13-455#	13-455#	13-460	13-460#	13-460#	13-465	13-465#	13-465#	13-475	13-475#	13-475#	13-479
	13-479#	13-479#	13-481	13-481#	13-481#	13-483	13-483#	13-483#	13-485	13-485#	13-485#	13-490	13-490#	13-490#
	13-497	13-497#	13-499	13-499#	13-501	13-501#	13-501#	13-503	13-503#	13-503#	13-505	13-505#	13-505#	13-507
	13-507#	13-507#	13-512	13-512#	13-512#	13-525	13-525#	13-525#	13-530	13-530#	13-530#	13-533	13-533#	13-533#
	13-535	13-535#	13-535#	13-537	13-537#	13-537#	13-542	13-542#	13-542#	13-547	13-547#	13-547#	13-549	13-549#
	13-549#	13-551	13-551#	13-551#	13-553	13-553#	13-553#	13-555	13-555#	13-555#	13-560	13-560#	13-560#	13-566
	13-566#	13-566#	13-568	13-568#	13-568#	13-582	13-582#	13-582#	13-584	13-584#	13-584#	13-586	13-586#	13-586#
	13-591	13-591#	13-591#	13-596	13-596#	13-596#	13-604	13-604#	13-604#	13-608	13-608#	13-608#	13-610	13-610#
	13-610#	13-618	13-618#	13-618#	13-620	13-620#	13-620#	13-622	13-622#	13-622#	13-627	13-627#	13-627#	13-632
	13-632#	13-632#	13-645	13-645#	13-645#	13-647	13-647#	13-647#	13-649	13-649#	13-649#	13-657	13-657#	13-657#
	13-662	13-662#	13-662#	13-667	13-667#	13-667#	13-679	13-679#	13-679#	13-687	13-687#	13-687#	13-689	13-689#
	13-689#	13-694	13-694#	13-694#	13-718	13-718#	13-718#	13-733	13-733#	13-733#	13-735	13-735#	13-735#	13-743
	13-743#	13-743#	13-745	13-745#	13-745#	13-747	13-747#	13-747#	13-749	13-749#	13-749#	13-751	13-751#	13-751#
	13-759	13-759#	13-759#	13-804	13-804#	13-804#	13-813	13-813#	13-813#	13-819	13-819#	13-819#	13-821	13-821#
	13-821#	13-835	13-835#	13-835#	13-837	13-837#	13-837#	13-839	13-839#	13-839#	13-844	13-844#	13-844#	13-856
	13-856#	13-856#	13-860	13-860#	13-860#	13-862	13-862#	13-862#	13-879	13-879#	13-879#	13-881	13-881#	13-881#
	13-883	13-883#	13-883#	13-888	13-888#	13-888#	13-893	13-893#	13-893#	13-895	13-895#	13-895#	13-906	13-906#
	13-906#	13-917	13-917#	13-917#	13-950	13-950#	13-950#	13-953	13-953#	13-953#	13-955	13-955#	13-955#	13-983
	13-983#	13-983#	13-:17	13-:17#	13-:17#	13-:20	13-:20#	13-:20#	13-:22	13-:22#	13-:22#	13-:50	13-:50#	13-:50#
	13-:83	13-:83#	13-:83#	13-:86	13-:86#	13-:86#	13-:88	13-:88#	13-:88#	13-:16	13-:16#	13-:16#	13-:49	13-:49#
	13-:49#	13-:52	13-:52#	13-:52#	13-:54	13-:54#	13-:54#	13-:82	13-:82#	13-:82#	13-:85	13-:85#	13-:88	13-:88#
	13-:88#	13-<07	13-<07#	13-<07#	13-<10	13-<10#	13-<12	13-<12#	13-<12#	13-<14	13-<14#	13-<14#	13-<24	13-<24#
	13-<24#	13-<27	13-<27#	13-<35	13-<35#	13-<35#	13-<44	13-<44#	13-<44#	13-<56	13-<56#	13-<56#	13-<60	13-<60#
	13-<60#	13-<72	13-<72#	13-<72#	13-<74	13-<74#	13-<89	13-<89#	13-<89#	13-=03	13-=03#	13-=03#	13-=05	13-=05#
	13-=05#	13-=05#	13-=07	13-=07#	13-=07#	13-=59	13-=59#	13-=59#	13-=69	13-=69#	13-=69#	13-=73	13-=73#	13-=73#
	13-=75	13-=75#	13-=75#	13-=77	13-=77#	13-=77#	13-=79	13-=79#	13-=79#	13-=81	13-=81#	13-=81#	13-=87	13-=87#
	13-=87#	13-=97	13-=97#	13-=97#	13->01	13->01#	13->01#	13->03	13->03#	13->03#	13->05	13->05#	13->05#	13->07
	13->07#	13->07#	13->09	13->09#	13->09#	13->16	13->16#	13->16#	13->25	13->25#	13->25#	13->30	13->30#	13->30#
	13->32	13->32#	13->32#	13->34	13->34#	13->34#	13->36	13->36#	13->36#	13->38	13->38#	13->38#	13->49	13->49#
	13->49#	13->61	13->61#	13->61#	13->65	13->65#	13->65#	13->67	13->67#	13->67#	13->69	13->69#	13->69#	13->71
	13->71#	13->71#	13->73	13->73#	13->73#	13->83	13->83#	13->83#	13->93	13->93#	13->93#	13->97	13->97#	13->97#
	13->99	13->99#	13->99#	13-?01	13-?01#	13-?01#	13-?03	13-?03#	13-?03#	13-?05	13-?05#	13-?05#	13-?11	13-?11#

	13-?11#	13-?22	13-?22#	13-?22#	13-?26	13-?26#	13-?26#	13-?28	13-?28#	13-?28#	13-?30	13-?30#	13-?30#	13-?32
	13-?32#	13-?32#	13-?34	13-?34#	13-?34#	13-?45	13-?45#	13-?45#	13-?56	13-?56#	13-?56#	13-?60	13-?60#	13-?60#
	13-?62	13-?62#	13-?62#	13-?64	13-?64#	13-?64#	13-?66	13-?66#	13-?66#	13-?68	13-?68#	13-?68#	13-?79	13-?79#
	13-?79#	13-?95	13-?95#	13-?95#	13-?97	13-?97#	13-?97#	13-?99	13-?99#	13-?99#	13-a03	13-a03#	13-a03#	13-a05
	13-a05#	13-a05#	13-a07	13-a07#	13-a07#	13-a10	13-a10#	13-a10#	13-a12	13-a12#	13-a12#	13-a27	13-a27#	13-a27#
	13-a44	13-a44#	13-a44#	13-a46	13-a46#	13-a46#	13-a48	13-a48#	13-a48#	13-a52	13-a52#	13-a52#	13-a54	13-a54#
	13-a54#	13-a56	13-a56#	13-a56#	13-a59	13-a59#	13-a59#	13-a61	13-a61#	13-a61#	13-a78	13-a78#	13-a78#	13-a94
	13-a94#	13-a94#	13-a96	13-a96#	13-a96#	13-a98	13-a98#	13-a98#	13-A02	13-A02#	13-A02#	13-A04	13-A04#	13-A04#
	13-A06	13-A06#	13-A06#	13-A08	13-A08#	13-A08#	13-A10	13-A10#	13-A10#	13-A10#	13-A41	13-A41#	13-A41#	13-A43
	13-A43#	13-A45	13-A45#	13-A45#	13-A53	13-A53#	13-A53#	13-A55	13-A55#	13-A55#	13-A60	13-A60#	13-A60#	13-A73
	13-A73#	13-A73#	13-A75	13-A75#	13-A75#	13-A84	13-A84#	13-A84#	13-A86	13-A86#	13-A86#	13-A88	13-A88#	13-A88#
	13-A90	13-A90#	13-A90#	13-A92	13-A92#	13-A92#	13-A98	13-A98#	13-A98#	13-B08	13-B08#	13-B08#	13-B15	13-B15#
	13-B15#	13-B18	13-B18#	13-B20	13-B20#	13-B20#	13-B22	13-B22#	13-B22#	13-B24	13-B24#	13-B24#	13-B26	13-B26#
	13-B26#	13-B28	13-B28#	13-B28#	13-B48	13-B48#	13-B48#	13-B51	13-B51#	13-B51#	13-B53	13-B53#	13-B53#	13-C10
	13-C10#	13-C10#	13-C24	13-C24#	13-C24#	13-C27	13-C27#	13-C27#	13-C29	13-C29#	13-C29#	13-C31	13-C31#	13-C31#
	13-C33	13-C33#	13-C33#	13-C37	13-C37#	13-C37#	13-C39	13-C39#	13-C39#	13-C41	13-C41#	13-C41#	13-C43	13-C43#
	13-C43#	13-C45	13-C45#	13-C45#	13-C61	13-C61#	13-C61#	13-C77	13-C77#	13-C77#	13-C81	13-C81#	13-C81#	13-C83
	13-C83#	13-C83#	13-C85	13-C85#	13-C85#	13-C89	13-C89#	13-C89#	13-C93	13-C93#	13-C93#	13-C95	13-C95#	13-C95#
	13-C97	13-C97#	13-C97#	13-C99	13-C99#	13-C99#	13-D01	13-D01#	13-D01#	13-D21	13-D21#	13-D21#	13-D37	13-D37#
	13-D37#	13-D41	13-D41#	13-D41#	13-D43	13-D43#	13-D43#	13-D45	13-D45#	13-D45#	13-D47	13-D47#	13-D47#	13-D49
	13-D49#	13-D49#	13-D51	13-D51#	13-D51#	13-D66	13-D66#	13-D66#	13-D83	13-D83#	13-D83#	13-D87	13-D87#	13-D87#
	13-D89	13-D89#	13-D89#	13-D91	13-D91#	13-D91#	13-D93	13-D93#	13-D93#	13-D95	13-D95#	13-D95#	13-D97	13-D97#
	13-D97#	13-E14	13-E14#	13-E14#	13-E30	13-E30#	13-E30#	13-E34	13-E34#	13-E34#	13-E36	13-E36#	13-E36#	13-E38
	13-E38#	13-E38#	13-E40	13-E40#	13-E40#	13-E42	13-E42#	13-E42#	13-E44	13-E44#	13-E44#	13-E60	13-E60#	13-E60#
	13-E76	13-E76#	13-E76#	13-E80	13-E80#	13-E80#	13-E82	13-E82#	13-E82#	13-E91	13-E91#	13-E91#	13-E93	13-E93#
	13-E93#	13-E98	13-E98#	13-E98#	13-F11	13-F11#	13-F11#	13-F13	13-F13#	13-F13#	13-F24	13-F24#	13-F24#	13-F26
	13-F26#	13-F26#	13-F28	13-F28#	13-F28#	13-F30	13-F30#	13-F30#	13-F32	13-F32#	13-F32#	13-F37	13-F37#	13-F37#
	13-F49	13-F49#	13-F49#	13-F57	13-F57#	13-F57#	13-F61	13-F61#	13-F61#	13-F63	13-F63#	13-F63#	13-F65	13-F65#
	13-F65#	13-F67	13-F67#	13-F67#	13-F69	13-F69#	13-F69#	13-F82	13-F82#	13-F82#	13-F92	13-F92#	13-F92#	13-F96
FMT16	13-F96#	13-F96#	13-F98	13-F98#	13-F98#	13-G00	13-G00#	13-G00#	13-G02	13-G02#	13-G02#	13-G04	13-G04#	13-G04#
	4-641#	13-854	13-864	13-941	13-:08	13-:74	13-:40	13-:98	13-a18	13-a20	13-a67	13-a69	13-A16	13-A18
	13-B78	13-B98	13-C00	13-C51	13-C53	13-D09	13-D11	13-D57	13-D59	13-E03	13-E05	13-E50	13-E52	13-F41
	26-58	32-99												
FNCDTB	13-956	13-964	13-:23	13-:31	13-:89	13-:97	13-:55	13-:63	22-130	48-55#				
FNCMSK	4-488#	31-17												
GET	13-65	13-170	13-189	13-210	13-264	13-280	13-320	13-366	13-387	13-408	13-426	13-432	13-453	13-479
	13-501	13-530	13-547	13-566	13-596	13-608	13-632	13-647	13-667	13-679	13-733	13-743	13-804	13-819
	13-860	13-893	13-906	13-953	13-:20	13-:86	13-:52	13-<12	13-<35	13-<44	13-<74	13-=05	13-=73	13->01
	13->30	13->65	13->97	13-?26	13-?60	13-?95	13-a03	13-a44	13-a52	13-a94	13-A02	13-A43	13-A75	13-A84
	13-B20	13-B51	13-C27	13-C37	13-C81	13-C93	13-D41	13-D87	13-E34	13-E80	13-F13	13-F24	13-F61	13-F96
	15-47	15-90	15-114	15-143	15-170	15-205	17-31#							
GETBUF	7-0#	17-37												
GETINX	7-0#	13-185*	13-186*	13-187*	16-13	17-38								
GETSTS	13-63	13-168	13-208	13-261	13-278	13-384	13-405	13-424	13-450	13-476	13-497	13-527	13-545	13-563
	13-594	13-630	13-665	13-723	13-779	13-816	13-857	13-891	13-919	13-985	13-:52	13-:18	13-:85	13-<27
	13-<92	13-=70	13-=98	13->27	13->62	13->94	13-?23	13-?57	13-?91	13-a40	13-a90	13-A40	13-A72	13-B09
	13-B46	13-C23	13-C76	13-D33	13-D79	13-E26	13-E75	13-F10	13-F51	13-F93	15-46	16-10#		
GNS	5-1	5-1	10-6	10-28	10-42	10-53	10-59	10-60	12-30	14-20	14-20	44-1	44-1	44-1
	44-1	44-1	44-1	44-1	44-1	44-1	44-1	44-1	44-1	44-1	44-1	44-1	44-1	44-1
	44-1	44-1	44-1	44-1	44-1	44-1	44-1	44-1	44-1	44-1	44-1	44-1	44-1	44-1
GO	4-487#	13-259	13-364	13-383	13-404	13-423	13-449	13-469	13-492	13-494	13-515	13-544	13-562	13-593
	13-606	13-629	13-664	13-677	13-720	13-781	13-815	13-855	13-890	13-933	13-935	13-:02	13-:66	13-:68
	13-:32	13-:34	13-:92	13-<33	13-<40	13-<42	13-=63	13-=91	13->19	13->54	13->87	13-?15	13-?49	13-?81
	13-a29	13-a80	13-A30	13-A65	13-B02	13-B12	13-B39	13-C16	13-C32	13-C68	13-C88	13-D26	13-D68	13-E16
	13-E67	13-F01	13-F42	13-F56	13-F86	15-125	15-185	20-20	21-92	21-104	21-109	21-125	22-51	22-54
GTSWR	10-28	44-1#												
HCE	4-565#	4-574	22-379	22-384	22-399	22-401	22-414	22-429	48-77	48-78	48-81	48-85	48-86	

RMDSI	7-0#	13-129	13-212	13-214	13-217	13-241	13-243	13-245	13-287	13-289	13-293	13-295	13-299	13-301
	13-327	13-329	13-333	13-335	13-339	13-341	13-357	13-359	13-427	13-569	13-571	13-575	13-577	13-597
	13-600	13-611	13-613	13-633	13-635	13-650	13-652	13-668	13-670	13-680	13-683	13-822	13-825	13-828
	13-830	13-896	13-898	13-907	13-909	13-965	13-969	13-:32	13-:36	13-:98	13-:02	13-:64	13-:68	13-A76
	13-A78	13-F15	13-F18	15-122	15-180	21-106	22-38	22-40	22-85	22-102	22-136	22-140	22-196	22-223
	22-358	23-92	26-91	26-146	26-155	26-162	26-166	26-168	26-177	26-181	26-183	26-192	26-194	26-196
	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	56-13
RMDSO	7-0#													
RMDT	4-688#	10-91	12-58	41-26										
RMDTI	7-0#	56-5	56-17											
RMDTO	7-0#													
RMEC1	4-695#													
RMEC1I	7-0#	56-14												
RMEC1O	7-0#													
RMEC2	4-696#	16-15												
RMEC2I	7-0#	13-113	16-14	28-94	31-94	56-15								
RMEC2O	7-0#													
RMER1	4-684#	13-53	13-157	13-421	13-522	13-675	13-:90	13-A69	13-F07	21-145				
RMER1I	7-0#	13-90	13-227	13-229	13-234	13-236	13-958	13-:91	13-:57	13-<47	13-<50	13-<76	13-<79	13-=09
	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	56-13					
RMER1O	7-0#	13-45*	13-150*	13-421*	13-516*	13-675*	13-:90*	13-A62*	13-F00*	21-141				
RMER2	4-694#	13-54	13-158	13-521										
RMER2I	7-0#	13-125	13-221	13-223	13-248	13-250	13-305	13-307	13-311	13-313	13-345	13-347	13-351	13-353
	13-367	13-369	13-456	13-:25	13-A46	13-A49	13-E84	13-E87	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	56-13										
RMER2O	7-0#	13-46*	13-151*	13-517*	13-518*									
RMHR	4-692#													
RMHRI	7-0#													
RMHRO	7-0#													
RMLA	4-686#	13-B57	13-B58	13-B67										
RMLAI	7-0#	56-16												
RMLAO	7-0#													
RMMR1	4-687#	13-55	13-159	13-277	13-318	13-379	13-381	13-445	13-447	13-927	13-929	13-993	13-995	13-:60
	13-:62	13-:26	13-:28	13-<29	13-<31	13-<38	13-A28	13-A35	13-E62	13-E69				
RMMR1I	7-0#	13-105	13-281	13-284	13-321	13-324	28-81	31-69	56-17					
RMMR1O	7-0#	13-47*	13-152*	13-277*	13-318*	13-379*	13-381*	13-445*	13-447*	13-927*	13-929*	13-993*	13-995*	13-:60*
	13-:62*	13-:26*	13-:28*	13-<29*	13-<31*	13-<38*	13-A28*	13-A31*	13-E62*	13-E63*				
RMMR2	4-693#													
RMMR2I	7-0#	13-117	28-103	31-81	56-17									
RMMR2O	7-0#													
RMOF	4-690#	13-849	13-945	13-:12	13-:78	13-:44	13-<02	13-?86	13-a35	13-a87	13-B43	13-C20	13-C71	13-D29
	13-D76	13-E21	13-E72	13-F46										
RMOFI	7-0#	13-863	22-377	22-487	56-14	56-16								
RMOFO	7-0#	13-854*	13-941*	13-:08*	13-:74*	13-:40*	13-:98*	13-?82*	13-a18	13-a20*	13-a33*	13-a67	13-a69*	13-a83*
	13-A16	13-A18*	13-B36*	13-B78	13-B98	13-C00*	13-C15*	13-C51	13-C53*	13-C64*	13-D09	13-D11*	13-D23*	13-D57
	13-D59*	13-D72*	13-E03	13-E05*	13-E19*	13-E50	13-E52*	13-E66*	13-F41*	26-58	32-99	56-11		

RMR	4-570#	13-<47	13-<49	29-51	29-83	29-87	30-232	30-250	30-253					
RMSN	4-689#													
RMSNI	7-0#	56-17												
RMSNO	7-0#													
RMWC	4-756#	13-9*	13-10	13-944	13-:11	13-:77	13-;43	13-<01						
RMWCI	7-0#	23-10	23-23	23-44	23-115	56-14								
RMWCO	7-0#	13-938*	13-:05*	13-:71*	13-;37*	13-;95*	23-24	23-45						
RQA	4-653#	13-118	28-104	31-82										
RQB	4-654#	13-118	28-104	31-82										
RTC	4-498#	13-815												
SA1	4-535#													
SA16	4-531#													
SA2	4-534#													
SA4	4-533#													
SA8	4-532#													
SADMSK	4-539#	23-68												
SAVREG	41-13	44-1#												
SC	4-707#	13-69	21-92	21-109	21-125	28-32								
SCO	4-588#													
SC1	4-587#													
SC2	4-586#													
SC3	4-585#													
SC4	4-584#													
SCHSTS	13-C41	13-C97	13-D45	13-D91	13-E38	13-F65	32-40#							
SCOPE	4-475#	13-1	13-32	13-137	13-179	13-204	13-255	13-273	13-375	13-400	13-417	13-441	13-463	13-488
	13-510	13-540	13-558	13-589	13-625	13-660	13-692	13-757	13-811	13-842	13-886	13-915	13-981	13-:48
	13-:14	13-:80	13-<22	13-<54	13-<87	13-=56	13-=84	13->12	13->46	13->80	13-?08	13-?42	13-?76	13-a25
	13-a76	13-A24	13-A58	13-A95	13-B31	13-C08	13-C59	13-D19	13-D64	13-E12	13-E58	13-E96	13-F35	13-F79
	14-9													
SCTMSK	4-590#													
SEARCH	4-504#	13-B39	13-C32	13-C88	13-D26	13-D68	13-E16	13-E67	13-F01	13-F56	22-104	23-5		
SECERR	13-176	13-270	13-397	13-414	13-438	13-460	13-485	13-507	13-537	13-555	13-586	13-622	13-689	13-751
	13-839	13-883	13-=81	13->09	13->38	13->73	13-?05	13-?34	13-?68	13-a12	13-a61	13-A10	13-A55	13-A92
	13-B28	13-C45	13-D01	13-D51	13-D97	13-E44	13-E93	13-F32	13-F69	13-G04	22-24#			
SEEK	4-493#	13-=63	13-=91	13->19	13->54	13->87	13-?15	13-?49	13-?81	13-a29	13-a80	13-A30	13-A65	13-B12
	13-C16	13-C68	13-F86											
SEKSTS	13-=77	13->05	13->34	13->69	13-?01	13-?30	13-?64	13-a07	13-a56	13-A06	13-B24	13-C31	13-C85	13-G00
	26-19#													
SHUT	42-1	42-1	42-3#											
SIZCLK	12-28	19-3#												
SKI	4-666#	13-126	13-248	13-305	13-308	13-345	13-348	13-349	15-178	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	4-636#													
STACK	4-475#	10-23	13-1	13-32	13-137	13-179	13-204	13-255	13-273	13-375	13-400	13-417	13-441	13-463
	13-488	13-510	13-540	13-558	13-589	13-625	13-660	13-692	13-757	13-811	13-842	13-886	13-915	13-981
	13-:48	13-:14	13-:80	13-<22	13-<54	13-<87	13-=56	13-=84	13->12	13->46	13->80	13-?08	13-?42	13-?76
	13-a25	13-a76	13-A24	13-A58	13-A95	13-B31	13-C08	13-C59	13-D19	13-D64	13-E12	13-E58	13-E96	13-F35
	13-F79													
STANDA	10-66	11-3#												
START	5-1	10-17#	12-33	12-42	42-5									
START1	5-3	10-14#	13-27											
START2	10-15	10-18#												
STCDRV	13-436	13-553	13-584	13-620	13-687	13-749	13-837	13-881	13-A88	13-D47	13-D93	13-E40	13-F28	33-27#
STKLMT	4-475#													
STOP	39-1	39-5#												
STSD1	52-1	52-2	52-8	52-10	52-11	52-12	56-13#							

STSD2	52-1	52-2	52-8	52-10	52-11	52-12	56-14#							
STSD3	56-16#													
STSD4	52-1	52-2	52-8	52-10	52-11	52-12	56-17#							
STSF	53-1	53-1	53-1	53-2	53-2	53-2	53-8	53-8	53-8	53-10	53-10	53-10	53-11	53-11
	53-11	53-12	53-12	53-12	57-7#									
STSH1	51-1	51-2	51-8	51-9	51-11	51-12	51-13	55-21#						
STSH2	51-1	51-2	51-8	51-9	51-11	51-12	51-13	55-23#						
STSH3	55-25#													
STSH4	51-1	51-2	51-8	51-9	51-11	51-12	51-13	55-26#						
SW0	4-475#													
SW00	4-475	4-475#												
SW01	4-475	4-475#												
SW02	4-475	4-475#												
SW03	4-475	4-475#												
SW04	4-475	4-475#												
SW05	4-475	4-475#												
SW06	4-475	4-475#												
SW07	4-475	4-475#												
SW08	4-475	4-475#												
SW09	4-475	4-475#												
SW1	4-475#													
SW10	4-475#													
SW11	4-475#													
SW12	4-475#													
SW13	4-475#	41-14												
SW14	4-475#													
SW15	4-475#													
SW2	4-475#													
SW3	4-475#													
SW4	4-475#													
SW5	4-475#													
SW6	4-475#													
SW7	4-475#													
SW8	4-475#													
SW9	4-475#													
SWR	6-0#	10-23	10-23	10-23*	10-23*	10-23*	10-28	39-1	39-1	39-1	39-1	39-1	39-1	39-1*
	39-1*	39-1*	39-1*	40-1	40-1	40-1	40-1	40-1	41-14	42-1	42-1	42-1*	45-1	45-1*
SWREG	5-1#	10-23	10-28	42-1	42-1	42-1								
SYSTAT	10-76	47-18#												
TA1	4-528#													
TA16	4-524#	12-62												
TA2	4-527#	12-62												
TA4	4-526#	12-54												
TA8	4-525#													
TADMSK	4-538#	23-66												
TAG	4-655#													
TAGADR	5-9#	6-0												
TAP	4-632#													
TBITVE	4-475#													
TIMOUT	13-262	13-385	13-406	13-430	13-451	13-477	13-499	13-528	13-564	13-722	13-783	13-817	13-858	13-951
	13-:18	13-:84	13-:50	13-<10	13-=71	13-=99	13->28	13->63	13->95	13-?24	13-?58	13-@01	13-@50	13-A00
	13-A82	13-B10	13-B18	13-B49	13-C25	13-C35	13-C79	13-C91	13-D39	13-D85	13-E32	13-E78	13-F22	13-F52
	13-F59	13-F94	15-136	15-198	20-8#									
TKVEC	4-475#	42-1*	42-1*											
TPVEC	4-475#													
TRAPVE	4-475#	10-23*	10-23*											
TRE	4-708#	13-183	13-193	21-92	21-109	21-125	22-106	22-108	23-12					

CROSS REFERENCE PAGE (CONT.)

TRTVEC	4-475#			
TST	4-656#	13-119	28-105	31-83
TST1	13-1#	39-1		
TST10	13-375#	39-1		
TST11	13-400#	39-1		
TST12	13-417#	39-1		
TST13	13-441#	39-1		
TST14	13-463#	39-1		
TST15	13-488#	39-1		
TST16	13-510#	39-1		
TST17	13-540#	39-1		
TST2	13-32#	39-1		
TST20	13-558#	39-1		
TST21	13-589#	39-1		
TST22	13-625#	39-1		
TST23	13-660#	39-1		
TST24	13-692#	39-1		
TST25	13-757#	39-1		
TST26	13-811#	39-1		
TST27	13-842#	39-1		
TST3	13-137#	39-1		
TST30	13-886#	39-1		
TST31	13-915#	39-1		
TST32	13-981#	39-1		
TST33	13-:48#	39-1		
TST34	13-:14#	39-1		
TST35	13-:80#	39-1		
TST36	13-<22#	39-1		
TST37	13-<54#	39-1		
TST4	13-179#	39-1		
TST40	13-<87#	39-1		
TST41	13-=56#	39-1		
TST42	13-=84#	39-1		
TST43	13->12#	39-1		
TST44	13->46#	39-1		
TST45	13->80#	39-1		
TST46	13-?08#	39-1		
TST47	13-?42#	39-1		
TST5	13-204#	39-1		
TST50	13-?76#	39-1		
TST51	13-a25#	39-1		
TST52	13-a76#	39-1		
TST53	13-A24#	39-1		
TST54	13-A58#	39-1		
TST55	13-A95#	39-1		
TST56	13-B31#	39-1		
TST57	13-C08#	39-1		
TST6	13-255#	39-1		
TST60	13-C59#	39-1		
TST61	13-D19#	39-1		
TST62	13-D64#	39-1		
TST63	13-E12#	39-1		
TST64	13-E58#	39-1		
TST65	13-E96#	39-1		
TST66	13-F35#	39-1		
TST67	13-F79#	39-1		
TST7	13-273#	39-1		

PUSH	4-475#	13-4	13-13	13-710	13-711	13-712	13-713	13-769	13-770	13-771	13-776	13-777	13-778	13-<93
	13-<94	16-10	16-11	16-12	17-35	18-29	19-3	20-8	21-143	22-128	25-20	27-12	28-54	31-51
	31-52	34-1	36-1	43-1	45-1	45-1	46-1	46-1	46-1					
PUTREG	4-433#	13-183	13-259	13-277	13-318	13-364	13-379	13-381	13-383	13-404	13-421	13-423	13-445	13-447
	13-449	13-492	13-494	13-544	13-562	13-593	13-606	13-629	13-664	13-675	13-677	13-720	13-781	13-790
	13-815	13-890	13-904	13-927	13-929	13-933	13-993	13-995	13-:60	13-:62	13-:66	13-:26	13-:28	13-:32
	13-:90	13-<29	13-<31	13-<33	13-<38	13-<40	13-<42	13-A28	13-E62					
REPORT	4-475#													
RGBFMC	4-69#	7-0	7-0											
SETPRI	4-475#	12-48	39-5	39-9	42-1									
SETTRA	44-1	44-1	44-1	44-1	44-1	44-1	44-1	44-1	44-1	44-1	44-1	44-1	44-1	44-1#
SETUP	4-475#	10-23												
SKIP	4-475#													
SLASH	4-475#													
STARS	4-475#	5-5	5-8	5-8	5-8	6-0	6-0	6-0	13-1	13-1	13-32	13-32	13-137	13-137
	13-179	13-179	13-204	13-204	13-255	13-255	13-273	13-273	13-375	13-375	13-400	13-400	13-417	13-417
	13-441	13-441	13-463	13-463	13-488	13-488	13-510	13-510	13-540	13-540	13-558	13-558	13-589	13-589
	13-625	13-625	13-660	13-660	13-692	13-692	13-757	13-757	13-811	13-811	13-842	13-842	13-886	13-886
	13-915	13-915	13-981	13-981	13-:48	13-:48	13-:14	13-:14	13-:80	13-:80	13-<22	13-<22	13-<54	13-<54
	13-<37	13-<87	13-=56	13-=56	13-=84	13-=84	13->12	13->12	13->46	13->46	13->80	13->80	13-?08	13-?08
	13-?42	13-?42	13-?76	13-?76	13-a25	13-a25	13-a76	13-a76	13-A24	13-A24	13-A58	13-A58	13-A95	13-A95
	13-B31	13-B31	13-C08	13-C08	13-C59	13-C59	13-D19	13-D19	13-D64	13-D64	13-E12	13-E12	13-E58	13-E58
	13-E96	13-E96	13-F35	13-F35	13-F79	13-F79	14-20	15-57	15-70	15-84	15-107	15-138	15-160	15-200
	21-2	22-26	22-34	22-119	23-1	23-3	28-2	34-1	35-1	36-1	37-1	38-1	39-1	40-1
	42-1	42-1	42-1	42-1	42-1	43-1	44-1	45-1	45-1	46-1				
SWRSU	4-475#	10-23	10-23#											
TAGS	4-103#	6-0												
TRMTRP	44-1#													
TYPBIN	4-475#													
TYPDEC	4-475#	14-20	14-20											
TYPNAM	4-463#	4-475#	10-28											
TYPNUM	4-475#													
TYPOCS	4-475#	10-81	12-16	12-68	41-22	41-47	41-52	41-54						
TYPOCT	4-475#	11-33	42-1											
TYPTXT	4-475#	10-6	10-42	10-53	10-59	10-60	12-30	14-20	14-20					
XPER	4-12#	8-3	8-6	8-9	8-12	8-15	8-18	8-21	8-24	8-27	8-30	8-33	8-36	8-39
	8-42	8-45	8-48	8-51	8-54	8-57	8-60	8-63	8-66	8-69	8-72	8-75	8-78	8-81
	8-84	8-87	8-90	8-93	8-96	8-99	8-102	8-105	8-108	8-111	8-114	8-117	8-120	8-123
	8-126	8-130	8-133	8-136	8-139	8-142	8-146	8-150	8-154	8-157	8-160	8-163	8-166	8-169
	8-172	8-175	8-178	8-182	8-185	8-188	8-191	8-194	8-197	8-200	8-203	8-206	8-209	8-212
	8-215	8-218	8-221	8-224	8-227	8-230	8-233	8-236	8-239	8-242	8-245	8-248	8-251	8-254
	8-257	8-260	8-263	8-266	8-269	8-272	8-275	8-278	8-281	8-284	8-287	8-290	8-293	8-296
	8-299	8-302	8-305	8-308	8-311	8-314	8-317	8-320	8-323	8-326	8-329	8-332	8-335	8-338
	8-341	8-344	8-347	8-351	8-354	8-357	8-360	8-363	8-366	8-369	8-372	8-375	8-378	8-381
	8-384	8-387	8-390	8-393	8-396	8-399	8-402	8-405	8-408	8-411	8-414	8-417	8-420	8-423
	8-426	8-429	8-432	8-435	8-438	8-441	8-444	8-447	8-450	8-453	8-456	8-459	8-462	8-465
	8-468	8-471	8-474	8-477	8-480	8-483	8-486	8-489	8-492	8-495	8-498	8-501	8-504	8-507
	8-510	8-513	8-516	8-519	8-522	8-525	8-528	8-531	8-534	8-537	8-540	8-543	8-546	8-549
	8-552	8-556	8-559	8-563	8-566	8-569	8-573	8-577	8-581	8-586	8-590	8-594	8-597	8-600
	8-603	8-606	8-609	8-612	8-615	8-618	8-621	8-624	8-627	8-630	8-633	8-636	8-639	8-642
	8-645	8-648	8-651	8-654	8-657	8-660	8-663	8-666	8-669	8-672	8-675	8-678	8-681	8-684
	8-687	8-690	8-693	8-696	8-699	8-702	8-705	8-708	8-711	8-714	8-717	8-720	8-723	